Package 'EnvStats'

May 7, 2026

Type Package

Title Package for Environmental Statistics, Including US EPA Guidance

Version 3.1.0

Date 2025-04-17

Depends R (>= 3.5.0)

Imports MASS, ggplot2, nortest

Suggests lattice, qcc, sp, boot, tinytest, covr, Hmisc

Description Graphical and statistical analyses of environmental data, with focus on analyzing chemical concentrations and physical parameters, usually in the context of mandated environmental monitoring. Major environmental statistical methods found in the literature and regulatory guidance documents, with extensive help that explains what these methods do, how to use them, and where to find them in the literature. Numerous built-in data sets from regulatory guidance documents and environmental statistics literature. Includes scripts reproducing analyses presented in the book ``EnvStats: An R Package for Environmental Statistics'' (Millard, 2013, Springer, ISBN 978-1-4614-8455-4, <doi:10.1007/978-1-4614-8456-1>).

License GPL (>= 3)

URL https://github.com/alexkowa/EnvStats, https://alexkowa.github.io/EnvStats/

LazyLoad yes

LazyData yes

NeedsCompilation no

Author Steven P. Millard [aut], Alexander Kowarik [ctb, cre] (ORCID: <https://orcid.org/0000-0001-8598-4130>)

Maintainer Alexander Kowarik <alexander.kowarik@statistik.gv.at>

Repository CRAN

Date/Publication 2025-04-24 12:00:07 UTC

Contents

ACE.13.TCE.df . . . 10
anovaPE . . . 10
aovN . . . 13
aovPower . . . 16
base . . . 20
Beal.2010.Pb.df . . . 22
Benthic.df . . . 22
BJC.2000.df . . . 25
boxcox . . . 26
boxcox.object . . . 35
boxcoxCensored . . . 38
boxcoxCensored.object . . . 47
boxcoxLm.object . . . 51
boxcoxTransform . . . 53
calibrate . . . 57
calibrate.object . . . 60
CastilloAndHadi1994 . . . 62
cdfCompare . . . 64
cdfCompareCensored . . . 69
cdfPlot . . . 78
chenTTest . . . 81
Chi . . . 87
ciBinomHalfWidth . . . 89
ciBinomN . . . 95
ciNormHalfWidth . . . 103
ciNormN . . . 107
ciNparConfLevel . . . 111
ciNparN . . . 115
ciTableMean . . . 117
ciTableProp . . . 124
cv . . . 128
detectionLimitCalibrate . . . 131
distChoose . . . 135
distChoose.object . . . 147
distChooseCensored . . . 150
distChooseCensored.object . . . 165
Distribution.df . . . 168
ebeta . . . 176
ebinom . . . 179
ecdfPlot . . . 186
ecdfPlotCensored . . . 190
eevd . . . 195
eexp . . . 201
egamma . . . 204
egammaAltCensored . . . 214
egammaCensored . . . 223

egeom . . . 232
egevd . . . 234
ehyper . . . 241
elnorm . . . 244
elnorm3 . . . 246
elnormAlt . . . 259
elnormAltCensored . . . 271
elnormCensored . . . 289
elogis . . . 298
Empirical . . . 302
enbinom . . . 307
enorm . . . 310
enormCensored . . . 314
enpar . . . 335
enparCensored . . . 341
Environmental . . . 355
EPA.02d.Ex.2.ug.per.L.vec . . . 357
EPA.02d.Ex.4.mg.per.kg.vec . . . 357
EPA.02d.Ex.6.mg.per.kg.vec . . . 358
EPA.02d.Ex.9.mg.per.L.vec . . . 358
EPA.09.Ex.10.1.nickel.df . . . 359
EPA.09.Ex.11.1.arsenic.df . . . 359
EPA.09.Ex.12.1.ccl4.df . . . 360
EPA.09.Ex.12.4.naphthalene.df . . . 361
EPA.09.Ex.13.1.iron.df . . . 361
EPA.09.Ex.14.1.manganese.df . . . 362
EPA.09.Ex.14.3.alkalinity.df . . . 363
EPA.09.Ex.14.4.arsenic.df . . . 363
EPA.09.Ex.14.8.df . . . 364
EPA.09.Ex.15.1.manganese.df . . . 365
EPA.09.Ex.16.1.sulfate.df . . . 366
EPA.09.Ex.16.2.benzene.df . . . 366
EPA.09.Ex.16.4.copper.df . . . 367
EPA.09.Ex.16.5.PCE.df . . . 368
EPA.09.Ex.17.1.loglead.df . . . 369
EPA.09.Ex.17.2.toluene.df . . . 369
EPA.09.Ex.17.3.chrysene.df . . . 370
EPA.09.Ex.17.3.log.chrysene.df . . . 371
EPA.09.Ex.17.4.copper.df . . . 372
EPA.09.Ex.17.5.chloride.df . . . 372
EPA.09.Ex.17.6.sulfate.df . . . 373
EPA.09.Ex.17.7.sodium.df . . . 374
EPA.09.Ex.18.1.arsenic.df . . . 375
EPA.09.Ex.18.2.chrysene.df . . . 375
EPA.09.Ex.18.3.TCE.df . . . 376
EPA.09.Ex.18.4.xylene.df . . . 377
EPA.09.Ex.19.1.sulfate.df . . . 378
EPA.09.Ex.19.2.chloride.df . . . 379

EPA.09.Ex.19.5.mercury.df . . . 379
EPA.09.Ex.20.1.nickel.df . . . 380
EPA.09.Ex.21.1.aldicarb.df . . . 381
EPA.09.Ex.21.2.benzene.df . . . 382
EPA.09.Ex.21.5.beryllium.df . . . 382
EPA.09.Ex.21.6.nitrate.df . . . 383
EPA.09.Ex.21.7.TCE.df . . . 384
EPA.09.Ex.22.1.VC.df . . . 385
EPA.09.Ex.22.2.Specific.Conductance.df . . . 385
EPA.09.Ex.6.3.sulfate.df . . . 386
EPA.09.Ex.7.1.arsenic.df . . . 387
EPA.09.Table.9.1.TCE.df . . . 388
EPA.09.Table.9.3.df . . . 388
EPA.09.Table.9.4.nickel.vec . . . 389
EPA.89b.aldicarb1.df . . . 390
EPA.89b.aldicarb2.df . . . 391
EPA.89b.benzene.df . . . 391
EPA.89b.cadmium.df . . . 392
EPA.89b.chlordane1.df . . . 392
EPA.89b.chlordane2.df . . . 393
EPA.89b.edb.df . . . 394
EPA.89b.lead.df . . . 394
EPA.89b.loglead.df . . . 395
EPA.89b.manganese.df . . . 395
EPA.89b.sulfate.df . . . 396
EPA.89b.t29.df . . . 397
EPA.89b.toc.vec . . . 397
EPA.92c.arsenic1.df . . . 398
EPA.92c.arsenic2.df . . . 398
EPA.92c.arsenic3.df . . . 399
EPA.92c.benzene1.df . . . 400
EPA.92c.benzene2.df . . . 400
EPA.92c.ccl4.df . . . 401
EPA.92c.chrysene.df . . . 402
EPA.92c.copper1.df . . . 402
EPA.92c.copper2.df . . . 403
EPA.92c.lognickel1.df . . . 404
EPA.92c.nickel1.df . . . 404
EPA.92c.nickel2.df . . . 405
EPA.92c.toluene.df . . . 405
EPA.92c.zinc.df . . . 406
EPA.92d.chromium.df . . . 407
EPA.92d.chromium.vec . . . 407
EPA.94b.lead.df . . . 408
EPA.94b.tccb.df . . . 408
EPA.97.cadmium.111.df . . . 409
epareto . . . 410
epdfPlot . . . 412

epois . . . 415
epoisCensored . . . 419
eqbeta . . . 427
eqbinom . . . 429
eqevd . . . 432
eqexp . . . 435
eqgamma . . . 437
eqgeom . . . 444
eqgevd . . . 446
eqhyper . . . 449
eqlnorm . . . 452
eqlnorm3 . . . 456
eqlnormCensored . . . 460
eqlogis . . . 467
eqnbinom . . . 469
eqnorm . . . 472
eqnormCensored . . . 478
eqnpar . . . 487
eqpareto . . . 503
eqpois . . . 505
equnif . . . 509
eqweibull . . . 511
eqzmlnorm . . . 514
eqzmnorm . . . 517
errorBar . . . 519
estimate.object . . . 522
estimateCensored.object . . . 527
EulersConstant . . . 531
eunif . . . 532
EVD . . . 535
evNormOrdStats . . . 538
eweibull . . . 542
ezmlnorm . . . 545
ezmnorm . . . 550
FcnsByCat . . . 555
FcnsByCatCalibration . . . 556
FcnsByCatCensoredData . . . 556
FcnsByCatDataTrans . . . 559
FcnsByCatEstDistParams . . . 560
FcnsByCatEstDistQuants . . . 561
FcnsByCatGOFTests . . . 562
FcnsByCatHypothTests . . . 563
FcnsByCatMCandRisk . . . 564
FcnsByCatPlotProbDists . . . 564
FcnsByCatPlotUsingggplot2 . . . 565
FcnsByCatPower . . . 566
FcnsByCatPredInts . . . 570
FcnsByCatPrintPlot . . . 571

FcnsByCatProbDists . . . 572
FcnsByCatSumStats . . . 573
FcnsByCatTolInts . . . 574
FcnsByCatTrend . . . 575
GammaAlt . . . 575
geoMean . . . 577
geom_stripchart . . . 579
geoSD . . . 590
GEVD . . . 592
Gibbons.et.al.09.Alkilinity.vec . . . 594
Gibbons.et.al.09.Vinyl.Chloride.vec . . . 595
gof.object . . . 595
gofCensored.object . . . 599
gofGroup.object . . . 602
gofGroupTest . . . 605
gofOutlier.object . . . 612
gofTest . . . 615
gofTestCensored . . . 640
gofTwoSample.object . . . 657
gpqCiNormCensored . . . 659
gpqTolIntNormCensored . . . 663
Graham.et.al.75.etu.df . . . 666
Grice.Bain.80.mat . . . 667
Helsel.Cohn.88.app.b.df . . . 668
Helsel.Cohn.88.silver.df . . . 669
Helsel.Hirsch.02.Mayfly.df . . . 669
HoskingEtAl1985 . . . 670
htest.htestEnvStats.object . . . 675
htestCensored.object . . . 679
inversePredictCalibrate . . . 682
iqr . . . 686
kendallSeasonalTrendTest . . . 688
kendallTrendTest . . . 703
kurtosis . . . 711
Lin.Evans.80.df . . . 715
linearTrendTestN . . . 716
linearTrendTestPower . . . 718
linearTrendTestScaledMds . . . 726
lMoment . . . 729
Lognormal3 . . . 734
LognormalAlt . . . 738
LognormalMix . . . 741
LognormalMixAlt . . . 743
LognormalTrunc . . . 745
LognormalTruncAlt . . . 748
longToWide . . . 750
Millard.Deverel.88.df . . . 752
Modified.TcCB.df . . . 753

newsEnvStats . . . 754
NIOSH.89.air.lead.vec . . . 755
NormalMix . . . 755
NormalTrunc . . . 757
Olympic.NH4.df . . . 760
oneSamplePermutationTest . . . 761
Ozone.NE.df . . . 766
Pareto . . . 767
pdfPlot . . . 770
permutationTest.object . . . 773
plot.boxcox . . . 776
plot.boxcoxCensored . . . 779
plot.boxcoxLm . . . 783
plot.gof . . . 787
plot.gofCensored . . . 792
plot.gofGroup . . . 798
plot.gofTwoSample . . . 802
plot.permutationTest . . . 808
plotAovDesign . . . 811
plotCiBinomDesign . . . 815
plotCiNormDesign . . . 821
plotCiNparDesign . . . 826
plotLinearTrendTestDesign . . . 829
plotPredIntLnormAltSimultaneousTestPowerCurve . . . 832
plotPredIntLnormAltTestPowerCurve . . . 837
plotPredIntNormDesign . . . 841
plotPredIntNormSimultaneousTestPowerCurve . . . 845
plotPredIntNormTestPowerCurve . . . 850
plotPredIntNparDesign . . . 853
plotPredIntNparSimultaneousDesign . . . 857
plotPredIntNparSimultaneousTestPowerCurve . . . 860
plotPropTestDesign . . . 865
plotTolIntNormDesign . . . 870
plotTolIntNparDesign . . . 874
plotTTestDesign . . . 878
plotTTestLnormAltDesign . . . 882
pointwise . . . 887
ppointsCensored . . . 891
predict . . . 903
predict.lm . . . 906
predIntGamma . . . 908
predIntGammaSimultaneous . . . 915
predIntLnorm . . . 925
predIntLnormAltSimultaneousTestPower . . . 934
predIntLnormAltTestPower . . . 938
predIntLnormSimultaneous . . . 940
predIntNorm . . . 947
predIntNormHalfWidth . . . 953

predIntNormK . . . 957
predIntNormN . . . 963
predIntNormSimultaneous . . . 967
predIntNormSimultaneousK . . . 978
predIntNormSimultaneousTestPower . . . 987
predIntNormTestPower . . . 995
predIntNpar . . . 999
predIntNparConfLevel . . . 1008
predIntNparN . . . 1011
predIntNparSimultaneous . . . 1014
predIntNparSimultaneousConfLevel . . . 1027
predIntNparSimultaneousN . . . 1031
predIntNparSimultaneousTestPower . . . 1036
predIntPois . . . 1043
print . . . 1052
print.boxcox . . . 1054
print.boxcoxCensored . . . 1055
print.boxcoxLm . . . 1056
print.distChoose . . . 1057
print.distChooseCensored . . . 1058
print.estimate . . . 1059
print.estimateCensored . . . 1060
print.gof . . . 1061
print.gofCensored . . . 1062
print.gofGroup . . . 1063
print.gofOutlier . . . 1064
print.gofTwoSample . . . 1065
print.htest.htestEnvStats . . . 1066
print.htestCensored . . . 1069
print.permutationTest . . . 1070
print.summaryStats . . . 1071
propTestMdd . . . 1072
propTestN . . . 1078
propTestPower . . . 1085
ProUCL.5.2.TRS.df . . . 1091
ProUCL.Crit.Vals.for.AD.Test.for.Gamma.array . . . 1092
ProUCL.Crit.Vals.for.KS.Test.for.Gamma.array . . . 1093
pwMoment . . . 1094
qqPlot . . . 1098
qqPlotCensored . . . 1107
qqPlotGestalt . . . 1116
quantileTest . . . 1120
quantileTestPValue . . . 1125
Refinery.CO.df . . . 1127
rosnerTest . . . 1128
serialCorrelationTest . . . 1141
signTest . . . 1152
simulateMvMatrix . . . 1160

simulateVector . . . 1168
Skagit.NH3_N.df . . . 1172
skewness . . . 1173
stat_mean_sd_text . . . 1176
stat_median_iqr_text . . . 1180
stat_n_text . . . 1184
stat_test_text . . . 1188
stripChart . . . 1193
summaryFull . . . 1205
summaryStats . . . 1211
summaryStats.object . . . 1222
tolIntGamma . . . 1224
tolIntLnorm . . . 1232
tolIntLnormCensored . . . 1237
tolIntNorm . . . 1243
tolIntNormCensored . . . 1248
tolIntNormHalfWidth . . . 1254
tolIntNormK . . . 1257
tolIntNormN . . . 1263
tolIntNpar . . . 1267
tolIntNparConfLevel . . . 1273
tolIntNparCoverage . . . 1276
tolIntNparN . . . 1278
tolIntPois . . . 1281
Total.P.df . . . 1285
Triangular . . . 1286
tTestAlpha . . . 1289
tTestLnormAltN . . . 1291
tTestLnormAltPower . . . 1296
tTestLnormAltRatioOfMeans . . . 1303
tTestN . . . 1307
tTestPower . . . 1311
tTestScaledMdd . . . 1318
twoSampleLinearRankTest . . . 1322
twoSampleLinearRankTestCensored . . . 1331
twoSamplePermutationTestLocation . . . 1347
twoSamplePermutationTestProportion . . . 1354
varGroupTest . . . 1360
varTest . . . 1364
ZeroModifiedLognormal . . . 1367
ZeroModifiedLognormalAlt . . . 1370
ZeroModifiedNormal . . . 1373
zTestGevdShape . . . 1376

Index **1380**

ACE.13.TCE.df *Trichloroethylene Concentrations Before and After Remedation*

Description

Trichloroethylene (TCE) concentrations (mg/L) at 10 groundwater monitoring wells before and after remediation.

Usage

```
data(ACE.13.TCE.df)
```

Format

A data frame with 20 observations on the following 3 variables.

`TCE.mg.per.L` TCE concentrations

`Well` a factor indicating the well number

`Period` a factor indicating the period (before vs. after remediation)

Source

USACE. (2013). *Environmental Quality - Environmental Statistics*. Engineer Manual EM 200-1-16, 31 May 2013. Department of the Army, U.S. Army Corps of Engineers, Washington, D.C. 20314-1000, p. M-10. https://www.publications.usace.army.mil/Portals/76/Publications/EngineerManuals/EM_200-1-16.pdf.

anovaPE *Compute Lack-of-Fit and Pure Error Anova Table for a Linear Model*

Description

Compute a lack-of-fit and pure error anova table for either a linear model with one predictor variable or else a linear model for which all predictor variables in the model are functions of a single variable (for example, x, x^2, etc.). There must be replicate observations for at least one value of the predictor variable(s).

Usage

```
anovaPE(object)
```

Arguments

object	an object of class "lm". The object must have only one predictor variable in the formula, or else all predictor variables in the model must be functions of a single variable (for example, x, x^2, etc.). Also, the predictor variable(s) must have replicate observations for at least one value of the predictor variable(s).

Finally, the total number of observations must be such that the degrees of freedom associated with the residual sums of squares is greater than the number of observations minus the number of unique observations.

Details

Produces an anova table with the the sums of squares partitioned by "Lack of Fit" and "Pure Error". See Draper and Smith (1998, pp.47-53) for details. This function is called by the function calibrate.

Value

An object of class "anova" inheriting from class "data.frame".

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, pp.47-53.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

See Also

anova.lm, lm, calibrate.

Examples

```
# The data frame EPA.97.cadmium.111.df contains calibration data for
# cadmium at mass 111 (ng/L) that appeared in Gibbons et al. (1997b)
# and were provided to them by the U.S. EPA.
#
# First, display a plot of these data along with the fitted calibration line
# and 99% non-simultaneous prediction limits.  See
# Millard and Neerchal (2001, pp.566-569) for more details on this
# example.

EPA.97.cadmium.111.df
#   Cadmium Spike
#1     0.88     0
#2     1.57     0
#3     0.70     0
```

```
#...
#33   99.20   100
#34   93.71   100
#35  100.43   100

Cadmium <- EPA.97.cadmium.111.df$Cadmium

Spike <- EPA.97.cadmium.111.df$Spike

calibrate.list <- calibrate(Cadmium ~ Spike,
  data = EPA.97.cadmium.111.df)

newdata <- data.frame(Spike = seq(min(Spike), max(Spike), length.out = 100))

pred.list <- predict(calibrate.list, newdata = newdata, se.fit = TRUE)

pointwise.list <- pointwise(pred.list, coverage = 0.99, individual = TRUE)

plot(Spike, Cadmium, ylim = c(min(pointwise.list$lower),
  max(pointwise.list$upper)), xlab = "True Concentration (ng/L)",
  ylab = "Observed Concentration (ng/L)")

abline(calibrate.list, lwd = 2)

lines(newdata$Spike, pointwise.list$lower, lty = 8, lwd = 2)

lines(newdata$Spike, pointwise.list$upper, lty = 8, lwd = 2)

title(paste("Calibration Line and 99% Prediction Limits",
  "for US EPA Cadmium 111 Data", sep="\n"))

rm(Cadmium, Spike, newdata, calibrate.list, pred.list,
  pointwise.list)

#----------

# Now fit the linear model and produce the anova table to check for
# lack of fit.  There is no evidence for lack of fit (p = 0.41).

fit <- lm(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)


anova(fit)
#Analysis of Variance Table
#
#Response: Cadmium
#          Df Sum Sq Mean Sq F value    Pr(>F)
#Spike      1  43220   43220  9356.9 < 2.2e-16 ***
#Residuals 33    152       5
#---
#Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#Analysis of Variance Table
#
```

```
#Response: Cadmium
#
#Terms added sequentially (first to last)
#          Df Sum of Sq  Mean Sq  F Value Pr(F)
#    Spike  1  43220.27 43220.27 9356.879     0
#Residuals 33    152.43     4.62


anovaPE(fit)
#                 Df Sum Sq Mean Sq  F value Pr(>F)
#Spike             1  43220   43220 9341.559 <2e-16 ***
#Lack of Fit       3     14       5    0.982 0.4144
#Pure Error       30    139       5
#---
#Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


rm(fit)
```

---

aovN *Compute Sample Size Necessary to Achieve Specified Power for One-Way Fixed-Effects Analysis of Variance*

Description

Compute the sample sizes necessary to achieve a specified power for a one-way fixed-effects analysis of variance test, given the population means, population standard deviation, and significance level.

Usage

```
aovN(mu.vec, sigma = 1, alpha = 0.05, power = 0.95,
  round.up = TRUE, n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

`mu.vec`	required numeric vector of population means. The length of `mu.vec` must be at least 2. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`sigma`	optional numeric scalar specifying the population standard deviation (σ) for each group. The default value is `sigma=1`.
`alpha`	optional numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.
`power`	optional numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.
`round.up`	optional logical scalar indicating whether to round up the value of the computed sample size to the next smallest integer. The default value is `round.up=TRUE`.

n.max	positive integer greater then 1 indicating the maximum sample size per group. The default value is n.max=5000.
tol	optional numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.
maxiter	optional positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.

Details

The F-statistic to test the equality of k population means assuming each population has a normal distribution with the same standard deviation σ is presented in most basic statistics texts, including Zar (2010, Chapter 10), Berthouex and Brown (2002, Chapter 24), and Helsel and Hirsh (1992, pp.164-169). The formula for the power of this test is given in Scheffe (1959, pp.38-39,62-65). The power of the one-way fixed-effects ANOVA depends on the sample sizes for each of the k groups, the value of the population means for each of the k groups, the population standard deviation σ, and the significance level α. See the help file for aovPower.

The function aovN assumes equal sample sizes for each of the k groups and uses a search algorithm to determine the sample size n required to attain a specified power, given the values of the population means and the significance level.

Value

numeric scalar indicating the required sample size for each group. (The number of groups is equal to the length of the argument mu.vec.)

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. Sometimes it is necessary to compare several means to determine whether any are significantly different from each other (e.g., USEPA, 2009, p.6-38). In this case, assuming normally distributed data, you perform a one-way parametric analysis of variance.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, Type I error level, power, and differences in means if one of the objectives of the sampling program is to determine whether a particular mean differs from a group of means. The functions aovPower, aovN, and plotAovDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York, Chapters 27, 29, 30.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Scheffe, H. (1959). *The Analysis of Variance*. John Wiley and Sons, New York, 477pp.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.6-38.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 10.

See Also

aovPower, plotAovDesign, Normal, aov.

Examples

```
# Look at how the required sample size for a one-way ANOVA
# increases with increasing power:

aovN(mu.vec = c(10, 12, 15), sigma = 5, power = 0.8)
#[1] 21

aovN(mu.vec = c(10, 12, 15), sigma = 5, power = 0.9)
#[1] 27

aovN(mu.vec = c(10, 12, 15), sigma = 5, power = 0.95)
#[1] 33

#----------------------------------------------------------------

# Look at how the required sample size for a one-way ANOVA,
# given a fixed power, decreases with increasing variability
# in the population means:

aovN(mu.vec = c(10, 10, 11), sigma=5)
#[1] 581

aovN(mu.vec = c(10, 10, 15), sigma = 5)
#[1] 25

aovN(mu.vec = c(10, 13, 15), sigma = 5)
#[1] 33

aovN(mu.vec = c(10, 15, 20), sigma = 5)
#[1] 10

#----------------------------------------------------------------
```

```
# Look at how the required sample size for a one-way ANOVA,
# given a fixed power, decreases with increasing values of
# Type I error:

aovN(mu.vec = c(10, 12, 14), sigma = 5, alpha = 0.001)
#[1] 89

aovN(mu.vec = c(10, 12, 14), sigma = 5, alpha = 0.01)
#[1] 67

aovN(mu.vec = c(10, 12, 14), sigma = 5, alpha = 0.05)
#[1] 50

aovN(mu.vec = c(10, 12, 14), sigma = 5, alpha = 0.1)
#[1] 42
```

---

aovPower *Compute the Power of a One-Way Fixed-Effects Analysis of Variance*

Description

Compute the power of a one-way fixed-effects analysis of variance, given the sample sizes, population means, population standard deviation, and significance level.

Usage

```
aovPower(n.vec, mu.vec = rep(0, length(n.vec)), sigma = 1, alpha = 0.05)
```

Arguments

n.vec	numeric vector of sample sizes for each group. The i^{th} element of n.vec denotes the sample size for group i. The length of n.vec must be at least 2, and all elements of n.vec must be greater than or equal to 2. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
mu.vec	numeric vector of population means. The length of mu.vec must be the same as the length of n.vec. The default value is a vector of zeros. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
sigma	numeric scalar specifying the population standard deviation (σ) for each group. The default value is sigma=1.
alpha	numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

Details

Consider k normally distributed populations with common standard deviation σ. Let μ_i denote the mean of the i'th group ($i = 1, 2, \ldots, k$), and let $\underline{x}_i = x_{i1}, x_{i2}, \ldots, x_{in_i}$ denote a vector of n_i

observations from the i'th group. The statistical method of analysis of variance (ANOVA) tests the null hypothesis:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_k \qquad (1)$$

against the alternative hypothesis that at least one of the means is different from the rest by using the F-statistic given by:

$$F = \frac{[\sum_{i=1}^{k} n_i(\bar{x}_{i.} - \bar{x}_{..})^2]/(k-1)}{[\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_{i.})^2]/(N-k)} \qquad (2)$$

where

$$\bar{x}_{i.} = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij} \qquad (3)$$

$$\bar{x}_{..} = \frac{1}{N}\sum_{i=1}^{k} n_i \bar{x}_{i.} = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{n_i} x_{ij} \qquad (4)$$

$$N = \sum_{i=1}^{k} n_i \qquad (5)$$

Under the null hypothesis (1), the F-statistic in (2) follows an F-distribution with $k-1$ and $N-k$ degrees of freedom. Analysis of variance rejects the null hypothesis (1) at significance level α when

$$F > F_{k-1,N-k}(1-\alpha) \qquad (6)$$

where $F_{\nu_1,\nu_2}(p)$ denotes the p'th quantile of the F-distribution with ν_1 and ν_2 degrees of freedom (Zar, 2010, Chapter 10; Berthouex and Brown, 2002, Chapter 24; Helsel and Hirsh, 1992, pp. 164–169).

The power of this test, denoted by $1-\beta$, where β denotes the probability of a Type II error, is given by:

$$1 - \beta = Pr[F_{k-1,N-k,\Delta} > F_{k-1,N-k}(1-\alpha)] \qquad (7)$$

where

$$\Delta = \frac{\sum_{i=1}^{k} n_i(\mu_i - \bar{\mu}_.)^2}{\sigma^2} \qquad (8)$$

$$\bar{\mu}_. = \frac{1}{k}\sum_{i=1}^{k} \mu_i \qquad (9)$$

and $F_{\nu_1,\nu_2,\Delta}$ denotes a non-central F random variable with ν_1 and ν_2 degrees of freedom and non-centrality parameter Δ. Equation (7) can be re-written as:

$$1 - \beta = 1 - H[F_{k-1,N-k}(1-\alpha), k-1, N-k, \Delta] \qquad (10)$$

where $H(x, \nu_1, \nu_2, \Delta)$ denotes the cumulative distribution function of this random variable evaluated at x (Scheffe, 1959, pp.38–39, 62–65).

The power of the one-way fixed-effects ANOVA depends on the sample sizes for each of the k groups, the value of the population means for each of the k groups, the population standard deviation σ, and the significance level α.

Value

a numeric scalar indicating the power of the one-way fixed-effects ANOVA for the given sample sizes, population means, population standard deviation, and significance level.

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. Sometimes it is necessary to compare several means to determine whether any are significantly different from each other (e.g., USEPA, 2009, p.6-38). In this case, assuming normally distributed data, you perform a one-way parametric analysis of variance.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, Type I error level, power, and differences in means if one of the objectives of the sampling program is to determine whether a particular mean differs from a group of means. The functions `aovPower`, `aovN`, and `plotAovDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York, Chapters 27, 29, 30.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Scheffe, H. (1959). *The Analysis of Variance*. John Wiley and Sons, New York, 477pp.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.6-38.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 10.

See Also

`aovN`, `plotAovDesign`, `Normal`, `aov`.

Examples

```
# Look at how the power of a one-way ANOVA increases
# with increasing sample size:
```

```
aovPower(n.vec = rep(5, 3), mu.vec = c(10, 15, 20), sigma = 5)
#[1] 0.7015083

aovPower(n.vec = rep(10, 3), mu.vec = c(10, 15, 20), sigma = 5)
#[1] 0.9732551

#--------------------------------------------------------------------

# Look at how the power of a one-way ANOVA increases
# with increasing variability in the population means:

aovPower(n.vec = rep(5,3), mu.vec = c(10, 10, 11), sigma=5)
#[1] 0.05795739

aovPower(n.vec = rep(5, 3), mu.vec = c(10, 10, 15), sigma = 5)
#[1] 0.2831863

aovPower(n.vec = rep(5, 3), mu.vec = c(10, 13, 15), sigma = 5)
#[1] 0.2236093

aovPower(n.vec = rep(5, 3), mu.vec = c(10, 15, 20), sigma = 5)
#[1] 0.7015083

#--------------------------------------------------------------------

# Look at how the power of a one-way ANOVA increases
# with increasing values of Type I error:

aovPower(n.vec = rep(10,3), mu.vec = c(10, 12, 14),
  sigma = 5, alpha = 0.001)
#[1] 0.02655785

aovPower(n.vec = rep(10,3), mu.vec = c(10, 12, 14),
  sigma = 5, alpha = 0.01)
#[1] 0.1223527

aovPower(n.vec = rep(10,3), mu.vec = c(10, 12, 14),
  sigma = 5, alpha = 0.05)
#[1] 0.3085313

aovPower(n.vec = rep(10,3), mu.vec = c(10, 12, 14),
  sigma = 5, alpha = 0.1)
#[1] 0.4373292

#==========

# The example on pages 5-11 to 5-14 of USEPA (1989b) shows
# log-transformed concentrations of lead (mg/L) at two
# background wells and four compliance wells, where observations
# were taken once per month over four months (the data are
# stored in EPA.89b.loglead.df.)  Assume the true mean levels
# at each well are 3.9, 3.9, 4.5, 4.5, 4.5, and 5, respectively.
# Compute the power of a one-way ANOVA to test for mean
```

```
# differences between wells.  Use alpha=0.05, and assume the
# true standard deviation is equal to the one estimated from
# the data in this example.

# First look at the data
names(EPA.89b.loglead.df)
#[1] "LogLead"   "Month"     "Well"      "Well.type"

dev.new()
stripChart(LogLead ~ Well, data = EPA.89b.loglead.df,
  show.ci = FALSE, xlab = "Well Number",
  ylab="Log [ Lead (ug/L) ]",
  main="Lead Concentrations at Six Wells")

# Note: The assumption of a constant variance across
# all wells is suspect.


# Now perform the ANOVA and get the estimated sd
aov.list <- aov(LogLead ~ Well, data=EPA.89b.loglead.df)

summary(aov.list)
#            Df Sum Sq Mean Sq F value  Pr(>F)
#Well         5 5.7447 1.14895  3.3469 0.02599 *
#Residuals   18 6.1791 0.34328
#---
#Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Now call the function aovPower
aovPower(n.vec = rep(4, 6),
  mu.vec = c(3.9,3.9,4.5,4.5,4.5,5), sigma=sqrt(0.34))
#[1] 0.5523148

# Clean up
rm(aov.list)
```

---

base *Base b Representation of a Number*

Description

For any number represented in base 10, compute the representation in any user-specified base.

Usage

```
base(n, base = 10, num.digits = max(0, floor(log(n, base))) + 1)
```

Arguments

n	a non-negative integer (base 10).
base	a positive integer greater than 1 indicating what base to represent n in.
num.digits	a positive integer indicating how many digits to use to represent n in base base. By default, num.digits is equal to just the number of required digits (i.e., max(0, floor(log(n, base))) + 1). Setting num.digits to a larger number than this will result in 0's padding the left.

Details

If b is a positive integer greater than 1, and n is a positive integer, then n can be expressed uniquely in the form

$$n = a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 b + a0$$

where k is a non-negative integer, the coefficients $a_0, a_1, \ldots, a_k$ are non-negative integers less than b, and $a_k > 0$ (Rosen, 1988, p.105). The function base computes the coefficients $a_0, a_1, \ldots, a_k$.

Value

A numeric vector of length num.digits showing the representation of n in base base.

Note

The function base is included in **EnvStats** because it is called by the function oneSamplePermutationTest.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Rosen, K.H. (1988). *Discrete Mathematics and Its Applications*. Random House, New York, pp.105-107.

See Also

oneSamplePermutationTest.

Examples

```
# Compute the value of 7 in base 2.

base(7, 2)
#[1] 1 1 1

base(7, 2, num.digits=5)
#[1] 0 0 1 1 1
```

Beal.2010.Pb.df *Lead concentration in soil samples.*

Description

Lead (Pb) concentrations (mg/kg) in 29 discrete environmental soil samples from a site suspected to be contaminated with lead.

Usage

```
Beal.2010.Pb.df
data(Beal.2010.Pb.df)
```

Format

A data frame with 29 observations on the following 3 variables.

`Pb.char` Character vector indicating lead concentrations. Nondetects indicated with the less-than sign (e.g., <1)

`Pb` numeric vector indicating lead concentration.

`Censored` logical vector indicating censoring status.

Source

Beal, D. (2010). *A Macro for Calculating Summary Statistics on Left Censored Environmental Data Using the Kaplan-Meier Method*. Paper SDA-09, presented at Southeast SAS Users Group 2010, September 26-28, Savannah, GA. https://analytics.ncsu.edu/sesug/2010/SDA09.Beal.pdf.

Benthic.df *Benthic Data from Monitoring Program in Chesapeake Bay*

Description

Benthic data from a monitoring program in the Chesapeake Bay, Maryland, covering July 1994 - December 1991.

Usage

```
Benthic.df
```

Format

A data frame with 585 observations on the following 7 variables.

`Site.ID` Site ID

`Stratum` Stratum Number (101-131)

`Latitude` Latitude (degrees North)

`Longitude` Longitude (negative values; degrees West)

`Index` Benthic Index (between 1 and 5)

`Salinity` Salinity (ppt)

`Silt` Silt Content (% clay in soil)

Details

Data from the Long Term Benthic Monitoring Program of the Chesapeake Bay. The data consist of measurements of benthic characteristics and a computed index of benthic health for several locations in the bay. Sampling methods and designs of the program are discussed in Ranasinghe et al. (1992).

The data represent observations collected at 585 separate point locations (sites). The sites are divided into 31 different strata, numbered 101 through 131, each strata consisting of geographically close sites of similar degradation conditions. The benthic index values range from 1 to 5 on a continuous scale, where high values correspond to healthier benthos. Salinity was measured in parts per thousand (ppt), and silt content is expressed as a percentage of clay in the soil with high numbers corresponding to muddy areas.

The United States Environmental Protection Agency (USEPA) established an initiative for the Chesapeake Bay in partnership with the states bordering the bay in 1984. The goal of the initiative is the restoration (abundance, health, and diversity) of living resources to the bay by reducing nutrient loadings, reducing toxic chemical impacts, and enhancing habitats. USEPA's Chesapeake Bay Program Office is responsible for implementing this initiative and has established an extensive monitoring program that includes traditional water chemistry sampling, as well as collecting data on living resources to measure progress towards meeting the restoration goals.

Sampling benthic invertebrate assemblages has been an integral part of the Chesapeake Bay monitoring program due to their ecological importance and their value as biological indicators. The condition of benthic assemblages is a measure of the ecological health of the bay, including the effects of multiple types of environmental stresses. Nevertheless, regional-scale assessment of ecological status and trends using benthic assemblages are limited by the fact that benthic assemblages are strongly influenced by naturally variable habitat elements, such as salinity, sediment type, and depth. Also, different state agencies and USEPA programs use different sampling methodologies, limiting the ability to integrate data into a unified assessment. To circumvent these limitations, USEPA has standardized benthic data from several different monitoring programs into a single database, and from that database developed a Restoration Goals Benthic Index that identifies whether benthic restoration goals are being met.

Source

Ranasinghe, J.A., L.C. Scott, and R. Newport. (1992). *Long-term Benthic Monitoring and Assessment Program for the Maryland Portion of the Bay*, Jul 1984-Dec 1991. Report prepared for the Maryland Department of the Environment and the Maryland Department of Natural Resources by Versar, Inc., Columbia, MD.

Examples

```
attach(Benthic.df)

# Show station locations
#----------------------
dev.new()
plot(Longitude, Latitude,
    xlab = "-Longitude (Degrees West)",
    ylab = "Latitude",
    main = "Sampling Station Locations")


# Scatterplot matrix of benthic index, salinity, and silt
#--------------------------------------------------------
dev.new()
pairs(~ Index + Salinity + Silt, data = Benthic.df)


# Contour and perspective plots based on loess fit
# showing only predicted values within the convex hull
# of station locations
#-------------------------------------------------------
library(sp)

loess.fit <- loess(Index ~ Longitude * Latitude,
    data=Benthic.df, normalize=FALSE, span=0.25)
lat <- Benthic.df$Latitude
lon <- Benthic.df$Longitude
Latitude <- seq(min(lat), max(lat), length=50)
Longitude <- seq(min(lon), max(lon), length=50)
predict.list <- list(Longitude=Longitude,
    Latitude=Latitude)
predict.grid <- expand.grid(predict.list)
predict.fit <- predict(loess.fit, predict.grid)
index.chull <- chull(lon, lat)
inside <- point.in.polygon(point.x = predict.grid$Longitude,
    point.y = predict.grid$Latitude,
    pol.x = lon[index.chull],
    pol.y = lat[index.chull])
predict.fit[inside == 0] <- NA

dev.new()
contour(Longitude, Latitude, predict.fit,
    levels=seq(1, 5, by=0.5), labcex=0.75,
    xlab="-Longitude (degrees West)",
    ylab="Latitude (degrees North)")
title(main=paste("Contour Plot of Benthic Index",
    "Based on Loess Smooth", sep="\n"))

dev.new()
persp(Longitude, Latitude, predict.fit,
    xlim = c(-77.3, -75.9), ylim = c(38.1, 39.5), zlim = c(0, 6),
```

```
    theta = -45, phi = 30, d = 0.5,
    xlab="-Longitude (degrees West)",
    ylab="Latitude (degrees North)",
    zlab="Benthic Index", ticktype = "detailed")
title(main=paste("Surface Plot of Benthic Index",
    "Based on Loess Smooth", sep="\n"))

detach("Benthic.df")

rm(loess.fit, lat, lon, Latitude, Longitude, predict.list,
    predict.grid, predict.fit, index.chull, inside)
```

BJC.2000.df *Randomly sampled measurements of an analyte in soil samples.*

Description

Analyte concentrations (μg/g) in 11 discrete environmental soil samples.

Usage

```
BJC.2000.df
data(BJC.2000.df)
```

Format

A data frame with 11 observations on the following 4 variables.

`Analyte.char` Character vector indicating lead concentrations. Nondetects indicated with the letter U after the measure (e.g., 0.10U)

`Analyte` numeric vector indicating analyte concentration.

`Censored` logical vector indicating censoring status.

`Detect` numeric vector of 0s (nondetects) and 1s (detects) indicating censoring status.

Source

BJC. (2000). *Improved Methods for Calculating Concentrations Used in Exposure Assessments*. BJC/OR-416, Prepared by the Lockheed Martin Energy Research Corporation. Prepared for the U.S. Department of Energy Office of Environmental Management. Bechtel Jacobs Company, LLC. January, 2000. https://rais.ornl.gov/documents/bjc_or416.pdf.

boxcox *Boxcox Power Transformation*

Description

boxcox is a generic function used to compute the value(s) of an objective for one or more Box-Cox power transformations, or to compute an optimal power transformation based on a specified objective. The function invokes particular methods which depend on the class of the first argument.

Currently, there is a default method and a method for objects of class "lm".

Usage

```
boxcox(x, ...)

## Default S3 method:
boxcox(x,
    lambda = {if (optimize) c(-2, 2) else seq(-2, 2, by = 0.5)},
    optimize = FALSE, objective.name = "PPCC",
    eps = .Machine$double.eps, include.x = TRUE, ...)

## S3 method for class 'lm'
boxcox(x,
    lambda = {if (optimize) c(-2, 2) else seq(-2, 2, by = 0.5)},
    optimize = FALSE, objective.name = "PPCC",
    eps = .Machine$double.eps, include.x = TRUE, ...)
```

Arguments

x	an object of class "lm" for which the response variable is all positive numbers, or else a numeric vector of positive numbers. When x is an object of class "lm", the object must have been created with a call to the function lm that includes the data argument. When x is a numeric vector of positive observations, missing (NA), undefined (NaN), and infinite (-Inf, Inf) values are allowed but will be removed.
lambda	numeric vector of finite values indicating what powers to use for the Box-Cox transformation. When optimize=FALSE, the default value is lambda=seq(-2, 2, by=0.5). When optimize=TRUE, lambda must be a vector with two values indicating the range over which the optimization will occur and the range of these two values must include 1. In this case, the default value is lambda=c(-2, 2).
optimize	logical scalar indicating whether to simply evalute the objective function at the given values of lambda (optimize=FALSE; the default), or to compute the optimal power transformation within the bounds specified by lambda (optimize=TRUE).
objective.name	character string indicating what objective to use. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).

eps	finite, positive numeric scalar. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation. The default value is eps=.Machine$double.eps.
include.x	logical scalar indicating whether to include the finite, non-missing values of the argument x with the returned object. The default value is include.x=TRUE.
...	optional arguments for possible future methods. Currently not used.

Details

Two common assumptions for several standard parametric hypothesis tests are:

1. The observations all come from a normal distribution.
2. The observations all come from distributions with the same variance.

For example, the standard one-sample t-test assumes all the observations come from the same normal distribution, and the standard two-sample t-test assumes that all the observations come from a normal distribution with the same variance, although the mean may differ between the two groups.

When the original data do not satisfy the above assumptions, data transformations are often used to attempt to satisfy these assumptions. The rest of this section is divided into two parts: one that discusses Box-Cox transformations in the context of the original observations, and one that discusses Box-Cox transformations in the context of linear models.

Box-Cox Transformations Based on the Original Observations

Box and Cox (1964) presented a formalized method for deciding on a data transformation. Given a random variable X from some distribution with only positive values, the Box-Cox family of power transformations is defined as:

$$Y = \begin{cases} \frac{X^\lambda - 1}{\lambda} & \lambda \neq 0 \\ log(X) & \lambda = 0 \end{cases} \quad (1)$$

where Y is assumed to come from a normal distribution. This transformation is continuous in λ. Note that this transformation also preserves ordering. See the help file for boxcoxTransform for more information on data transformations.

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a random sample of n observations from some distribution and assume that there exists some value of λ such that the transformed observations

$$y_i = \begin{cases} \frac{x_i^\lambda - 1}{\lambda} & \lambda \neq 0 \\ log(x_i) & \lambda = 0 \end{cases} \quad (2)$$

$(i = 1, 2, \ldots, n)$ form a random sample from a normal distribution.

Box and Cox (1964) proposed choosing the appropriate value of λ based on maximizing the likelihood function. Alternatively, an appropriate value of λ can be chosen based on another objective, such as maximizing the probability plot correlation coefficient or the Shapiro-Wilk goodness-of-fit statistic.

In the case when optimize=TRUE, the function boxcox calls the R function nlminb to minimize the negative value of the objective (i.e., maximize the objective) over the range of possible values of λ specified in the argument lambda. The starting value for the optimization is always $\lambda = 1$ (i.e., no transformation).

The rest of this sub-section explains how the objective is computed for the various options for objective.name.

Objective Based on Probability Plot Correlation Coefficient (objective.name="PPCC")
When objective.name="PPCC", the objective is computed as the value of the normal probability plot correlation coefficient based on the transformed data (see the description of the Probability Plot Correlation Coefficient (PPCC) goodness-of-fit test in the help file for gofTest). That is, the objective is the correlation coefficient for the normal quantile-quantile plot for the transformed data. Large values of the PPCC tend to indicate a good fit to a normal distribution.

Objective Based on Shapiro-Wilk Goodness-of-Fit Statistic (objective.name="Shapiro-Wilk")
When objective.name="Shapiro-Wilk", the objective is computed as the value of the Shapiro-Wilk goodness-of-fit statistic based on the transformed data (see the description of the Shapiro-Wilk test in the help file for gofTest). Large values of the Shapiro-Wilk statistic tend to indicate a good fit to a normal distribution.

Objective Based on Log-Likelihood Function (objective.name="Log-Likelihood")
When objective.name="Log-Likelihood", the objective is computed as the value of the log-likelihood function. Assuming the transformed observations in Equation (2) above come from a normal distribution with mean μ and standard deviation σ, we can use the change of variable formula to write the log-likelihood function as:

$$log[L(\lambda, \mu, \sigma)] = \frac{-n}{2} log(2\pi) - \frac{n}{2} log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2 + (\lambda - 1) \sum_{i=1}^{n} log(x_i) \qquad (3)$$

where y_i is defined in Equation (2) above (Box and Cox, 1964). For a fixed value of λ, the log-likelihood function is maximized by replacing μ and σ with their maximum likelihood estimators:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} y_i \qquad (4)$$

$$\hat{\sigma} = [\frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2]^{1/2} \qquad (5)$$

Thus, when optimize=TRUE, Equation (3) is maximized by iteratively solving for λ using the values for μ and σ given in Equations (4) and (5). When optimize=FALSE, the value of the objective is computed by using Equation (3), using the values of λ specified in the argument lambda, and using the values for μ and σ given in Equations (4) and (5).

Box-Cox Transformation for Linear Models
In the case of a standard linear regression model with n observations and p predictors:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \epsilon_i, \; i = 1, 2, \ldots, n \qquad (6)$$

the standard assumptions are:

1. The error terms ϵ_i come from a normal distribution with mean 0.
2. The variance is the same for all of the error terms and does not depend on the predictor variables.

Assuming Y is a random variable from some distribution that may depend on the predictor variables and Y takes on only positive values, the Box-Cox family of power transformations is defined as:

$$Y^* = \begin{cases} \frac{Y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ log(Y) & \lambda = 0 \end{cases} \quad (7)$$

where Y^* becomes the new response variable and the errors are now assumed to come from a normal distribution with a mean of 0 and a constant variance.

In this case, the objective is computed as described above, but it is based on the residuals from the fitted linear model in which the response variable is now Y^* instead of Y.

Value

When `x` is an object of class `"lm"`, `boxcox` returns a list of class `"boxcoxLm"` containing the results. See the help file for `boxcoxLm.object` for details.

When `x` is simply a numeric vector of positive numbers, `boxcox` returns a list of class `"boxcox"` containing the results. See the help file for `boxcox.object` for details.

Note

Data transformations are often used to induce normality, homoscedasticity, and/or linearity, common assumptions of parametric statistical tests and estimation procedures. Transformations are not "tricks" used by the data analyst to hide what is going on, but rather useful tools for understanding and dealing with data (Berthouex and Brown, 2002, p.61). Hoaglin (1988) discusses "hidden" transformations that are used everyday, such as the pH scale for measuring acidity. Johnson and Wichern (2007, p.192) note that "Transformations are nothing more than a reexpression of the data in different units."

In the case of a linear model, there are at least two approaches to improving a model fit: transform the Y and/or X variable(s), and/or use more predictor variables. Often in environmental data analysis, we assume the observations come from a lognormal distribution and automatically take logarithms of the data. For a simple linear regression (i.e., one predictor variable), if regression diagnostic plots indicate that a straight line fit is not adequate, but that the variance of the errors appears to be fairly constant, you may only need to transform the predictor variable X or perhaps use a quadratic or cubic model in X. On the other hand, if the diagnostic plots indicate that the constant variance and/or normality assumptions are suspect, you probably need to consider transforming the response variable Y. Data transformations for linear regression models are discussed in Draper and Smith (1998, Chapter 13) and Helsel and Hirsch (1992, pp. 228-229).

One problem with data transformations is that translating results on the transformed scale back to the original scale is not always straightforward. Estimating quantities such as means, variances, and confidence limits in the transformed scale and then transforming them back to the original scale usually leads to biased and inconsistent estimates (Gilbert, 1987, p.149; van Belle et al., 2004, p.400). For example, exponentiating the confidence limits for a mean based on log-transformed

data does not yield a confidence interval for the mean on the original scale. Instead, this yields a confidence interval for the median (see the help file for `elnormAlt`). It should be noted, however, that quantiles (percentiles) and rank-based procedures are invariant to monotonic transformations (Helsel and Hirsch, 1992, p.12).

Finally, there is no guarantee that a Box-Cox tranformation based on the "optimal" value of λ will provide an adequate transformation to allow the assumption of approximate normality and constant variance. Any set of transformed data should be inspected relative to the assumptions you want to make about it (Johnson and Wichern, 2007, p.194).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Box, G.E.P., and D.R. Cox. (1964). An Analysis of Transformations (with Discussion). *Journal of the Royal Statistical Society, Series B* **26**(2), 211–252.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, pp.47-53.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY.

Hinkley, D.V., and G. Runger. (1984). The Analysis of Transformed Data (with Discussion). *Journal of the American Statistical Association* **79**, 302–320.

Hoaglin, D.C., F.M. Mosteller, and J.W. Tukey, eds. (1983). *Understanding Robust and Exploratory Data Analysis*. John Wiley and Sons, New York, Chapter 4.

Hoaglin, D.C. (1988). Transformations in Everyday Experience. *Chance* **1**, 40–45.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions, Second Edition*. John Wiley and Sons, New York, p.163.

Johnson, R.A., and D.W. Wichern. (2007). *Applied Multivariate Statistical Analysis, Sixth Edition*. Pearson Prentice Hall, Upper Saddle River, NJ, pp.192–195.

Shumway, R.H., A.S. Azari, and P. Johnson. (1989). Estimating Mean Concentrations Under Transformations for Environmental Data With Detection Limits. *Technometrics* **31**(3), 347–356.

Stoline, M.R. (1991). An Examination of the Lognormal and Box and Cox Family of Transformations in Fitting Environmental Data. *Environmetrics* **2**(1), 85–106.

van Belle, G., L.D. Fisher, Heagerty, P.J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences, 2nd Edition*. John Wiley & Sons, New York.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 13.

See Also

boxcox.object, plot.boxcox, print.boxcox, boxcoxLm.object, plot.boxcoxLm, print.boxcoxLm, boxcoxTransform, Data Transformations, Goodness-of-Fit Tests.

Examples

```
# Generate 30 observations from a lognormal distribution with
# mean=10 and cv=2.  Look at some values of various objectives
# for various transformations.  Note that for both the PPCC and
# the Log-Likelihood objective, the optimal value of lambda is
# about 0, indicating that a log transformation is appropriate.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

dev.new()
hist(x, col = "cyan")

# Using the PPCC objective:
#--------------------------

boxcox(x)
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Data:                            x
#
#Sample Size:                     30
#
# lambda      PPCC
#   -2.0 0.5423739
#   -1.5 0.6402782
#   -1.0 0.7818160
#   -0.5 0.9272219
#    0.0 0.9921702
#    0.5 0.9581178
#    1.0 0.8749611
#    1.5 0.7827009
#    2.0 0.7004547

boxcox(x, optimize = TRUE)
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Data:                            x
#
#Sample Size:                     30
```

```
#
#Bounds for Optimization:         lower = -2
#                                 upper =  2
#
#Optimal Value:                   lambda = 0.04530789
#
#Value of Objective:              PPCC = 0.9925919


# Using the Log-Likelihodd objective
#-----------------------------------

boxcox(x, objective.name = "Log-Likelihood")
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  Log-Likelihood
#
#Data:                            x
#
#Sample Size:                     30
#
# lambda Log-Likelihood
#   -2.0     -154.94255
#   -1.5     -128.59988
#   -1.0     -106.23882
#   -0.5      -90.84800
#    0.0      -85.10204
#    0.5      -88.69825
#    1.0      -99.42630
#    1.5     -115.23701
#    2.0     -134.54125

boxcox(x, objective.name = "Log-Likelihood", optimize = TRUE)
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  Log-Likelihood
#
#Data:                            x
#
#Sample Size:                     30
#
#Bounds for Optimization:         lower = -2
#                                 upper =  2
#
#Optimal Value:                   lambda = 0.0405156
#
#Value of Objective:              Log-Likelihood = -85.07123

#----------

# Plot the results based on the PPCC objective
```

```
#-------------------------------------------
boxcox.list <- boxcox(x)
dev.new()
plot(boxcox.list)

#Look at QQ-Plots for the candidate values of lambda
#---------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

#==========

# The data frame Environmental.df contains daily measurements of
# ozone concentration, wind speed, temperature, and solar radiation
# in New York City for 153 consecutive days between May 1 and
# September 30, 1973.  In this example, we'll plot ozone vs.
# temperature and look at the Q-Q plot of the residuals.  Then
# we'll look at possible Box-Cox transformations.  The "optimal" one
# based on the PPCC looks close to a log-transformation
# (i.e., lambda=0).  The power that produces the largest PPCC is
# about 0.2, so a cube root (lambda=1/3) transformation might work too.

head(Environmental.df)
#           ozone radiation temperature wind
#05/01/1973    41       190          67  7.4
#05/02/1973    36       118          72  8.0
#05/03/1973    12       149          74 12.6
#05/04/1973    18       313          62 11.5
#05/05/1973    NA        NA          56 14.3
#05/06/1973    28        NA          66 14.9

tail(Environmental.df)
#           ozone radiation temperature wind
#09/25/1973    14        20          63 16.6
#09/26/1973    30       193          70  6.9
#09/27/1973    NA       145          77 13.2
#09/28/1973    14       191          75 14.3
#09/29/1973    18       131          76  8.0
#09/30/1973    20       223          68 11.5


# Fit the model with the raw Ozone data
#--------------------------------------
ozone.fit <- lm(ozone ~ temperature, data = Environmental.df)

# Plot Ozone vs. Temperature, with fitted line
#---------------------------------------------
dev.new()
with(Environmental.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
    ylab = "Ozone (ppb)", main = "Ozone vs. Temperature"))
abline(ozone.fit)

# Look at the Q-Q Plot for the residuals
```

```
#--------------------------------------
dev.new()
qqPlot(ozone.fit$residuals, add.line = TRUE)

# Look at Box-Cox transformations of Ozone
#-----------------------------------------
boxcox.list <- boxcox(ozone.fit)
boxcox.list
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Linear Model:                    ozone.fit
#
#Sample Size:                     116
#
# lambda      PPCC
#   -2.0 0.4286781
#   -1.5 0.4673544
#   -1.0 0.5896132
#   -0.5 0.8301458
#    0.0 0.9871519
#    0.5 0.9819825
#    1.0 0.9408694
#    1.5 0.8840770
#    2.0 0.8213675

# Plot PPCC vs. lambda based on Q-Q plots of residuals
#-----------------------------------------------------
dev.new()
plot(boxcox.list)

# Look at Q-Q plots of residuals for the various transformation
#--------------------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

# Compute the "optimal" transformation
#-------------------------------------
boxcox(ozone.fit, optimize = TRUE)
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Linear Model:                    ozone.fit
#
#Sample Size:                     116
#
#Bounds for Optimization:         lower = -2
#                                 upper =  2
#
#Optimal Value:                   lambda = 0.2004305
```

```
#
#Value of Objective:              PPCC = 0.9940222

#==========

# Clean up
#---------
rm(x, boxcox.list, ozone.fit)
graphics.off()
```

---

boxcox.object *S3 Class "boxcox"*

Description

Objects of S3 class "boxcox" are returned by the **EnvStats** function boxcox, which computes objective values for user-specified powers, or computes the optimal power for the specified objective.

Details

Objects of class "boxcox" are lists that contain information about the powers that were used, the objective that was used, the values of the objective for the given powers, and whether an optimization was specified.

Value

Required Components

The following components must be included in a legitimate list of class "boxcox".

lambda	Numeric vector containing the powers used in the Box-Cox transformations. If the value of the optimize component is FALSE, then lambda contains the values of all of the powers at which the objective was evaluated. If the value of the optimize component is TRUE, then lambda is a scalar containing the value of the power that maximizes the objective.
objective	Numeric vector containing the value(s) of the objective for the given value(s) of λ that are stored in the component lambda.
objective.name	character string indicating the objective that was used. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).
optimize	logical scalar indicating whether the objective was simply evaluted at the given values of lambda (optimize=FALSE), or instead the optimal power transformation was computed within the bounds specified by lambda (optimize=TRUE).
optimize.bounds	Numeric vector of length 2 with a names attribute indicating the bounds within which the optimization took place. When optimize=FALSE, this contains missing values.

eps	finite, positive numeric scalar indicating what value of eps was used. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation.
sample.size	Numeric scalar indicating the number of finite, non-missing observations.
data.name	The name of the data object used for the Box-Cox computations.
bad.obs	The number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the Box-Cox computations.

Optional Component

The following component may optionally be included in a legitimate list of class "boxcox". It must be included if you want to call the function plot.boxcox and specify Q-Q plots or Tukey Mean-Difference Q-Q plots.

data	Numeric vector containing the data actually used for the Box-Cox computations (i.e., the original data without any missing or infinite values).

Methods

Generic functions that have methods for objects of class "boxcox" include:
plot, print.

Note

Since objects of class "boxcox" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

boxcox, plot.boxcox, print.boxcox, boxcoxLm.object.

Examples

```
# Create an object of class "boxcox", then print it out.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

dev.new()
hist(x, col = "cyan")

boxcox.list <- boxcox(x)

data.class(boxcox.list)
```

```
#[1] "boxcox"

names(boxcox.list)
# [1] "lambda"          "objective"       "objective.name"
# [4] "optimize"        "optimize.bounds" "eps"
# [7] "data"            "sample.size"     "data.name"
#[10] "bad.obs"

boxcox.list
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Data:                            x
#
#Sample Size:                     30
#
# lambda      PPCC
#   -2.0 0.5423739
#   -1.5 0.6402782
#   -1.0 0.7818160
#   -0.5 0.9272219
#    0.0 0.9921702
#    0.5 0.9581178
#    1.0 0.8749611
#    1.5 0.7827009
#    2.0 0.7004547

boxcox(x, optimize = TRUE)
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Data:                            x
#
#Sample Size:                     30
#
#Bounds for Optimization:         lower = -2
#                                 upper =  2
#
#Optimal Value:                   lambda = 0.04530789
#
#Value of Objective:              PPCC = 0.9925919

#----------

# Clean up
#---------
rm(x, boxcox.list)
```

boxcoxCensored *Boxcox Power Transformation for Type I Censored Data*

Description

Compute the value(s) of an objective for one or more Box-Cox power transformations, or to compute an optimal power transformation based on a specified objective, based on Type I censored data.

Usage

```
boxcoxCensored(x, censored, censoring.side = "left",
 lambda = {if (optimize) c(-2, 2) else seq(-2, 2, by = 0.5)}, optimize = FALSE,
  objective.name = "PPCC", eps = .Machine$double.eps,
  include.x.and.censored = TRUE, prob.method = "michael-schucany",
  plot.pos.con = 0.375)
```

Arguments

x	a numeric vector of positive numbers. Missing (NA), undefined (NaN), and infinite (-Inf, Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
lambda	numeric vector of finite values indicating what powers to use for the Box-Cox transformation. When optimize=FALSE, the default value is lambda=seq(-2, 2, by=0.5). When optimize=TRUE, lambda must be a vector with two values indicating the range over which the optimization will occur and the range of these two values must include 1. In this case, the default value is lambda=c(-2, 2).
optimize	logical scalar indicating whether to simply evalute the objective function at the given values of lambda (optimize=FALSE; the default), or to compute the optimal power transformation within the bounds specified by lambda (optimize=TRUE).
objective.name	character string indicating what objective to use. The possible values are "PPCC" (probability plot correlation coefficient; the default), "Shapiro-Wilk" (the Shapiro-Wilk goodness-of-fit statistic), and "Log-Likelihood" (the log-likelihood function).
eps	finite, positive numeric scalar. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation. The default value is eps=.Machine$double.eps.

include.x.and.censored
: logical scalar indicating whether to include the finite, non-missing values of the argument x and the corresponding values of censored with the returned object. The default value is include.x.and.censored=TRUE.

prob.method
: for multiply censored data, character string indicating what method to use to compute the plotting positions (empirical probabilities) when objective.name="PPCC". Possible values are:

  "kaplan-meier" (product-limit method of Kaplan and Meier (1958)),
  "modified kaplan-meier" (same as "kaplan-meier" with the maximum value included),
  "nelson" (hazard plotting method of Nelson (1972)),
  "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and
  "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)).

  The default value is prob.method="michael-schucany".

  The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" is only available for censoring.side="left". See the DETAILS section for more explanation.

  This argument is ignored if objective.name is not equal to "PPCC" and/or the data are singly censored.

plot.pos.con
: for multiply censored data, numeric scalar between 0 and 1 containing the value of the plotting position constant when objective.name="PPCC". The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

  This argument is ignored if objective.name is not equal to "PPCC" and/or the data are singly censored.

Details

Two common assumptions for several standard parametric hypothesis tests are:

1. The observations all come from a normal distribution.
2. The observations all come from distributions with the same variance.

For example, the standard one-sample t-test assumes all the observations come from the same normal distribution, and the standard two-sample t-test assumes that all the observations come from a normal distribution with the same variance, although the mean may differ between the two groups.

When the original data do not satisfy the above assumptions, data transformations are often used to attempt to satisfy these assumptions. Box and Cox (1964) presented a formalized method for deciding on a data transformation. Given a random variable X from some distribution with only positive values, the Box-Cox family of power transformations is defined as:

$$Y = \begin{cases} \frac{X^{\lambda}-1}{\lambda} & \lambda \neq 0 \\ log(X) & \lambda = 0 \end{cases} \quad (1)$$

where Y is assumed to come from a normal distribution. This transformation is continuous in λ. Note that this transformation also preserves ordering. See the help file for `boxcoxTransform` for more information on data transformations.

Box and Cox (1964) proposed choosing the appropriate value of λ based on maximizing the likelihood function. Alternatively, an appropriate value of λ can be chosen based on another objective, such as maximizing the probability plot correlation coefficient or the Shapiro-Wilk goodness-of-fit statistic.

Shumway et al. (1989) investigated extending the method of Box and Cox (1964) to the case of Type I censored data, motivated by the desire to produce estimated means and confidence intervals for air monitoring data that included censored values.

In the case when `optimize=TRUE`, the function `boxcoxCensored` calls the R function `nlminb` to minimize the negative value of the objective (i.e., maximize the objective) over the range of possible values of λ specified in the argument `lambda`. The starting value for the optimization is always $\lambda = 1$ (i.e., no transformation).

The next section explains assumptions and notation, and the section after that explains how the objective is computed for the various options for `objective.name`.

Assumptions and Notation

Let $\underline{x}$ denote a random sample of N observations from some continuous distribution. Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_K; \; K \geq 1 \quad (2)$$

For the case when $K \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $K = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j = 1, 2, \ldots, K$, so that

$$\sum_{i=1}^{K} c_j = c \quad (3)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations, and let Ω_j denote the set of c_j subscripts in the "ordered" sample that correspond to the censored observations censored at censoring level T_j for $j = 1, 2, \ldots, k$.

We assume that there exists some value of λ such that the transformed observations

$$y_i = \begin{cases} \frac{x_i^\lambda - 1}{\lambda} & \lambda \neq 0 \\ log(x_i) & \lambda = 0 \end{cases} \quad (4)$$

$(i = 1, 2, \ldots, n)$ form a random sample of Type I censored data from a normal distribution.

Note that for the censored observations, Equation (4) becomes:

$$y_{(i)} = T_j^* = \begin{cases} \frac{T_j^\lambda - 1}{\lambda} & \lambda \neq 0 \\ log(T_j) & \lambda = 0 \end{cases} \quad (5)$$

where $i \in \Omega_j$.

Computing the Objective

Objective Based on Probability Plot Correlation Coefficient (`objective.name="PPCC"`)
When `objective.name="PPCC"`, the objective is computed as the value of the normal probability plot correlation coefficient based on the transformed data (see the description of the Probability Plot Correlation Coefficient (PPCC) goodness-of-fit test in the help file for `gofTestCensored`). That is, the objective is the correlation coefficient for the normal quantile-quantile plot for the transformed data. Large values of the PPCC tend to indicate a good fit to a normal distribution.

Objective Based on Shapiro-Wilk Goodness-of-Fit Statistic (`objective.name="Shapiro-Wilk"`)
When `objective.name="Shapiro-Wilk"`, the objective is computed as the value of the Shapiro-Wilk goodness-of-fit statistic based on the transformed data (see the description of the Shapiro-Wilk test in the help file for `gofTestCensored`). Large values of the Shapiro-Wilk statistic tend to indicate a good fit to a normal distribution.

Objective Based on Log-Likelihood Function (`objective.name="Log-Likelihood"`)
When `objective.name="Log-Likelihood"`, the objective is computed as the value of the log-likelihood function. Assuming the transformed observations in Equation (4) above come from a normal distribution with mean μ and standard deviation σ, we can use the change of variable formula to write the log-likelihood function as follows.

For Type I left censored data, the likelihood function is given by:

$$log[L(\lambda, \mu, \sigma)] = log[\binom{N}{c_1 c_2 \ldots c_k n}] + \sum_{j=1}^{k} c_j log[F(T_j^*)] + \sum_{i \in \Omega} log\{f[y_{(i)}]\} + (\lambda - 1) \sum_{i \in \Omega} log[x_{(i)}] \quad (6)$$

where f and F denote the probability density function (pdf) and cumulative distribution function (cdf) of the population. That is,

$$f(t) = \phi(\frac{t - \mu}{\sigma}) \quad (7)$$

$$F(t) = \Phi(\frac{t-\mu}{\sigma}) \qquad (8)$$

where ϕ and Φ denote the pdf and cdf of the standard normal distribution, respectively (Shumway et al., 1989). For left singly censored data, Equation (6) simplifies to:

$$log[L(\lambda,\mu,\sigma)] = log[\binom{N}{c}] + clog[F(T^*)] + \sum_{i=c+1}^{N} log\{f[y_{(i)}]\} + (\lambda-1)\sum_{i=c+1}^{N} log[x_{(i)}] \qquad (9)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$log[L(\lambda,\mu,\sigma)] = log[\binom{N}{c_1 c_2 \ldots c_k n}] + \sum_{j=1}^{k} c_j log[1-F(T_j^*)] + \sum_{i\in\Omega} log\{f[y_{(i)}]\} + (\lambda-1)\sum_{i\in\Omega} log[x_{(i)}] \qquad (10)$$

and for right singly censored data this simplifies to:

$$log[L(\lambda,\mu,\sigma)] = log[\binom{N}{c}] + clog[1-F(T^*)] + \sum_{i=1}^{n} log\{f[y_{(i)}]\} + (\lambda-1)\sum_{i=1}^{n} log[x_{(i)}] \qquad (11)$$

For a fixed value of λ, the log-likelihood function is maximized by replacing μ and σ with their maximum likelihood estimators (see the section *Maximum Likelihood Estimation* in the help file for `enormCensored`).

Thus, when `optimize=TRUE`, Equation (6) or (10) is maximized by iteratively solving for λ using the MLEs for μ and σ. When `optimize=FALSE`, the value of the objective is computed by using Equation (6) or (10), using the values of λ specified in the argument `lambda`, and using the MLEs of μ and σ.

Value

`boxcoxCensored` returns a list of class `"boxcoxCensored"` containing the results. See the help file for `boxcoxCensored.object` for details.

Note

Data transformations are often used to induce normality, homoscedasticity, and/or linearity, common assumptions of parametric statistical tests and estimation procedures. Transformations are not "tricks" used by the data analyst to hide what is going on, but rather useful tools for understanding and dealing with data (Berthouex and Brown, 2002, p.61). Hoaglin (1988) discusses "hidden" transformations that are used everyday, such as the pH scale for measuring acidity. Johnson and Wichern (2007, p.192) note that "Transformations are nothing more than a reexpression of the data in different units."

Shumway et al. (1989) investigated extending the method of Box and Cox (1964) to the case of Type I censored data, motivated by the desire to produce estimated means and confidence intervals for air monitoring data that included censored values.

Stoline (1991) compared the goodness-of-fit of Box-Cox transformed data (based on using the "optimal" power transformation from a finite set of values between -1.5 and 1.5) with log-transformed data for 17 groundwater chemistry variables. Using the Probability Plot Correlation Coefficient statistic for censored data as a measure of goodness-of-fit (see `gofTest`), Stoline (1991) found that

only 6 of the variables were adequately modeled by a Box-Cox transformation (p >0.10 for these 6 variables). Of these variables, five were adequately modeled by a a log transformation. Ten of variables were "marginally" fit by an optimal Box-Cox transformation, and of these 10 only 6 were marginally fit by a log transformation. Based on these results, Stoline (1991) recommends checking the assumption of lognormality before automatically assuming environmental data fit a lognormal distribution.

One problem with data transformations is that translating results on the transformed scale back to the original scale is not always straightforward. Estimating quantities such as means, variances, and confidence limits in the transformed scale and then transforming them back to the original scale usually leads to biased and inconsistent estimates (Gilbert, 1987, p.149; van Belle et al., 2004, p.400). For example, exponentiating the confidence limits for a mean based on log-transformed data does not yield a confidence interval for the mean on the original scale. Instead, this yields a confidence interval for the median (see the help file for `elnormAltCensored`). It should be noted, however, that quantiles (percentiles) and rank-based procedures are invariant to monotonic transformations (Helsel and Hirsch, 1992, p.12).

Finally, there is no guarantee that a Box-Cox tranformation based on the "optimal" value of λ will provide an adequate transformation to allow the assumption of approximate normality and constant variance. Any set of transformed data should be inspected relative to the assumptions you want to make about it (Johnson and Wichern, 2007, p.194).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Box, G.E.P., and D.R. Cox. (1964). An Analysis of Transformations (with Discussion). *Journal of the Royal Statistical Society, Series B* **26**(2), 211–252.

Cohen, A.C. (1991). Truncated and Censored Samples. *Marcel Dekker*, New York, New York, pp.50–59.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, pp.47-53.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY.

Hinkley, D.V., and G. Runger. (1984). The Analysis of Transformed Data (with Discussion). *Journal of the American Statistical Association* **79**, 302–320.

Hoaglin, D.C., F.M. Mosteller, and J.W. Tukey, eds. (1983). *Understanding Robust and Exploratory Data Analysis*. John Wiley and Sons, New York, Chapter 4.

Hoaglin, D.C. (1988). Transformations in Everyday Experience. *Chance* **1**, 40–45.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions, Second Edition*. John Wiley and Sons, New York, p.163.

Johnson, R.A., and D.W. Wichern. (2007). *Applied Multivariate Statistical Analysis, Sixth Edition*. Pearson Prentice Hall, Upper Saddle River, NJ, pp.192–195.

Shumway, R.H., A.S. Azari, and P. Johnson. (1989). Estimating Mean Concentrations Under Transformations for Environmental Data With Detection Limits. *Technometrics* **31**(3), 347–356.

Stoline, M.R. (1991). An Examination of the Lognormal and Box and Cox Family of Transformations in Fitting Environmental Data. *Environmetrics* **2**(1), 85–106.

van Belle, G., L.D. Fisher, Heagerty, P.J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences, 2nd Edition*. John Wiley & Sons, New York.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 13.

See Also

boxcoxCensored.object, plot.boxcoxCensored, print.boxcoxCensored, boxcox, Data Transformations, Goodness-of-Fit Tests.

Examples

```
# Generate 15 observations from a lognormal distribution with
# mean=10 and cv=2 and censor the observations less than 2.
# Then generate 15 more observations from this distribution and
# censor the observations less than 4.
# Then Look at some values of various objectives for various transformations.
# Note that for both the PPCC objective the optimal value is about -0.3,
# whereas for the Log-Likelihood objective it is about 0.3.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

x.1 <- rlnormAlt(15, mean = 10, cv = 2)
censored.1 <- x.1 < 2
x.1[censored.1] <- 2

x.2 <- rlnormAlt(15, mean = 10, cv = 2)
censored.2 <- x.2 < 4
x.2[censored.2] <- 4

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)

#--------------------------
# Using the PPCC objective:
#--------------------------

boxcoxCensored(x, censored)

#Results of Box-Cox Transformation
#Based on Type I Censored Data
#---------------------------------
#
```

```
#Objective Name:                  PPCC
#
#Data:                            x
#
#Censoring Variable:              censored
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 4
#
#Sample Size:                     30
#
#Percent Censored:                26.7%
#
# lambda      PPCC
#   -2.0 0.8954683
#   -1.5 0.9338467
#   -1.0 0.9643680
#   -0.5 0.9812969
#    0.0 0.9776834
#    0.5 0.9471025
#    1.0 0.8901990
#    1.5 0.8187488
#    2.0 0.7480494


boxcoxCensored(x, censored, optimize = TRUE)

#Results of Box-Cox Transformation
#Based on Type I Censored Data
#---------------------------------
#
#Objective Name:                  PPCC
#
#Data:                            x
#
#Censoring Variable:              censored
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 4
#
#Sample Size:                     30
#
#Percent Censored:                26.7%
#
#Bounds for Optimization:         lower = -2
#                                 upper =  2
#
#Optimal Value:                   lambda = -0.3194799
#
#Value of Objective:              PPCC = 0.9827546
```

```
#----------------------------------
# Using the Log-Likelihodd objective
#----------------------------------

boxcoxCensored(x, censored, objective.name = "Log-Likelihood")

#Results of Box-Cox Transformation
#Based on Type I Censored Data
#---------------------------------
#
#Objective Name:                  Log-Likelihood
#
#Data:                            x
#
#Censoring Variable:              censored
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 4
#
#Sample Size:                     30
#
#Percent Censored:                26.7%
#
# lambda Log-Likelihood
#   -2.0      -95.38785
#   -1.5      -84.76697
#   -1.0      -75.36204
#   -0.5      -68.12058
#    0.0      -63.98902
#    0.5      -63.56701
#    1.0      -66.92599
#    1.5      -73.61638
#    2.0      -82.87970


boxcoxCensored(x, censored, objective.name = "Log-Likelihood",
  optimize = TRUE)

#Results of Box-Cox Transformation
#Based on Type I Censored Data
#---------------------------------
#
#Objective Name:                  Log-Likelihood
#
#Data:                            x
#
#Censoring Variable:              censored
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 4
```

```
#
#Sample Size:                     30
#
#Percent Censored:                26.7%
#
#Bounds for Optimization:         lower = -2
#                                 upper =  2
#
#Optimal Value:                   lambda = 0.3049744
#
#Value of Objective:              Log-Likelihood = -63.2733

#----------

# Plot the results based on the PPCC objective
#---------------------------------------------
boxcox.list <- boxcoxCensored(x, censored)
dev.new()
plot(boxcox.list)

#Look at QQ-Plots for the candidate values of lambda
#----------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)

#==========

# Clean up
#---------
rm(x.1, censored.1, x.2, censored.2, x, censored, boxcox.list)
graphics.off()
```

`boxcoxCensored.object` *S3 Class "boxcoxCensored"*

Description

Objects of S3 class `"boxcoxCensored"` are returned by the **EnvStats** function `boxcoxCensored`, which computes objective values for user-specified powers, or computes the optimal power for the specified objective, based on Type I censored data.

Details

Objects of class `"boxcoxCensored"` are lists that contain information about the powers that were used, the objective that was used, the values of the objective for the given powers, and whether an optimization was specified.

Value

Required Components

The following components must be included in a legitimate list of class `"boxcoxCensored"`.

`lambda` Numeric vector containing the powers used in the Box-Cox transformations. If the value of the `optimize` component is `FALSE`, then `lambda` contains the values of all of the powers at which the objective was evaluated. If the value of the `optimize` component is `TRUE`, then `lambda` is a scalar containing the value of the power that maximizes the objective.

`objective` Numeric vector containing the value(s) of the objective for the given value(s) of λ that are stored in the component `lambda`.

`objective.name` Character string indicating the objective that was used. The possible values are `"PPCC"` (probability plot correlation coefficient; the default), `"Shapiro-Wilk"` (the Shapiro-Wilk goodness-of-fit statistic), and `"Log-Likelihood"` (the log-likelihood function).

`optimize` Logical scalar indicating whether the objective was simply evaluted at the given values of `lambda` (`optimize=FALSE`), or instead the optimal power transformation was computed within the bounds specified by `lambda` (`optimize=TRUE`).

`optimize.bounds` Numeric vector of length 2 with a names attribute indicating the bounds within which the optimization took place. When `optimize=FALSE`, this contains missing values.

`eps` Finite, positive numeric scalar indicating what value of `eps` was used. When the absolute value of `lambda` is less than `eps`, lambda is assumed to be 0 for the Box-Cox transformation.

`sample.size` Numeric scalar indicating the number of finite, non-missing observations.

`censoring.side` Character string indicating the censoring side. Possible values are `"left"` and `"right"`.

`censoring.levels` Numeric vector containing the censoring levels.

`percent.censored` Numeric scalar indicating the percent of observations that are censored.

`data.name` The name of the data object used for the Box-Cox computations.

`censoring.name` The name of the data object indicating which observations are censored.

`bad.obs` The number of missing (`NA`), undefined (`NaN`) and/or infinite (`Inf`, `-Inf`) values that were removed from the data object prior to performing the Box-Cox computations.

Optional Component

The following components may optionally be included in a legitimate list of class `"boxcoxCensored"`. They must be included if you want to call the function `plot.boxcoxCensored` and specify Q-Q plots or Tukey Mean-Difference Q-Q plots.

`data` Numeric vector containing the data actually used for the Box-Cox computations (i.e., the original data without any missing or infinite values).

`censored` Logical vector indicating which of the vales in the component `data` are censored.

Methods

Generic functions that have methods for objects of class "boxcoxCensored" include:
plot, print.

Note

Since objects of class "boxcoxCensored" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

boxcoxCensored, plot.boxcoxCensored, print.boxcoxCensored.

Examples

```
# Create an object of class "boxcoxCensored", then print it out.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

x.1 <- rlnormAlt(15, mean = 10, cv = 2)
censored.1 <- x.1 < 2
x.1[censored.1] <- 2

x.2 <- rlnormAlt(15, mean = 10, cv = 2)
censored.2 <- x.2 < 4
x.2[censored.2] <- 4

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)

boxcox.list <- boxcoxCensored(x, censored)

data.class(boxcox.list)
#[1] "boxcoxCensored"

names(boxcox.list)
# [1] "lambda"           "objective"        "objective.name"
# [4] "optimize"         "optimize.bounds"  "eps"
# [7] "data"             "censored"         "sample.size"
#[10] "censoring.side"   "censoring.levels" "percent.censored"
#[13] "data.name"        "censoring.name"   "bad.obs"

boxcox.list

#Results of Box-Cox Transformation
#Based on Type I Censored Data
#---------------------------------
```

```
#
#Objective Name:                  PPCC
#
#Data:                            x
#
#Censoring Variable:              censored
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 4
#
#Sample Size:                     30
#
#Percent Censored:                26.7%
#
# lambda      PPCC
#   -2.0 0.8954683
#   -1.5 0.9338467
#   -1.0 0.9643680
#   -0.5 0.9812969
#    0.0 0.9776834
#    0.5 0.9471025
#    1.0 0.8901990
#    1.5 0.8187488
#    2.0 0.7480494

boxcox.list2 <- boxcox(x, optimize = TRUE)
names(boxcox.list2)
# [1] "lambda"          "objective"       "objective.name"
# [4] "optimize"        "optimize.bounds" "eps"
# [7] "data"            "sample.size"     "data.name"
#[10] "bad.obs"

boxcox.list2
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Data:                            x
#
#Sample Size:                     30
#
#Bounds for Optimization:         lower = -2
#                                 upper =  2
#
#Optimal Value:                   lambda = -0.5826431
#
#Value of Objective:              PPCC = 0.9755402

#==========

# Clean up
```

```
#---------
rm(x.1, censored.1, x.2, censored.2, x, censored, boxcox.list, boxcox.list2)
```

`boxcoxLm.object` *S3 Class "boxcoxLm"*

Description

Objects of S3 class `"boxcoxLm"` are returned by the **EnvStats** function `boxcox` when the argument `x` is an object of class `"lm"`. In this case, `boxcox` computes values of an objective function for user-specified powers, or computes the optimal power for the specified objective, based on residuals from the linear model.

Details

Objects of class `"boxcoxLm"` are lists that contain information about the `"lm"` object that was suplied, the powers that were used, the objective that was used, the values of the objective for the given powers, and whether an optimization was specified.

Value

The following components must be included in a legitimate list of class `"boxcoxLm"`.

`lambda`	Numeric vector containing the powers used in the Box-Cox transformations. If the value of the `optimize` component is `FALSE`, then `lambda` contains the values of all of the powers at which the objective was evaluated. If the value of the `optimize` component is `TRUE`, then `lambda` is a scalar containing the value of the power that maximizes the objective.
`objective`	Numeric vector containing the value(s) of the objective for the given value(s) of λ that are stored in the component `lambda`.
`objective.name`	character string indicating the objective that was used. The possible values are `"PPCC"` (probability plot correlation coefficient; the default), `"Shapiro-Wilk"` (the Shapiro-Wilk goodness-of-fit statistic), and `"Log-Likelihood"` (the log-likelihood function).
`optimize`	logical scalar indicating whether the objective was simply evaluted at the given values of `lambda` (`optimize=FALSE`), or instead the optimal power transformation was computed within the bounds specified by `lambda` (`optimize=TRUE`).
`optimize.bounds`	Numeric vector of length 2 with a names attribute indicating the bounds within which the optimization took place. When `optimize=FALSE`, this contains missing values.
`eps`	finite, positive numeric scalar indicating what value of `eps` was used. When the absolute value of `lambda` is less than `eps`, lambda is assumed to be 0 for the Box-Cox transformation.
`lm.obj`	the value of the argument `x` provided to `boxcox` (an object that must inherit from class `"lm"`).
`sample.size`	Numeric scalar indicating the number of finite, non-missing observations.
`data.name`	The name of the data object used for the Box-Cox computations.

Methods

Generic functions that have methods for objects of class "boxcoxLm" include:
plot, print.

Note

Since objects of class "boxcoxLm" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

boxcox, plot.boxcoxLm, print.boxcoxLm, boxcox.object.

Examples

```
# Create an object of class "boxcoxLm", then print it out.

# The data frame Environmental.df contains daily measurements of
# ozone concentration, wind speed, temperature, and solar radiation
# in New York City for 153 consecutive days between May 1 and
# September 30, 1973.  In this example, we'll plot ozone vs.
# temperature and look at the Q-Q plot of the residuals.  Then
# we'll look at possible Box-Cox transformations.  The "optimal" one
# based on the PPCC looks close to a log-transformation
# (i.e., lambda=0).  The power that produces the largest PPCC is
# about 0.2, so a cube root (lambda=1/3) transformation might work too.

# Fit the model with the raw Ozone data
#--------------------------------------
ozone.fit <- lm(ozone ~ temperature, data = Environmental.df)

# Plot Ozone vs. Temperature, with fitted line
#---------------------------------------------
dev.new()
with(Environmental.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
    ylab = "Ozone (ppb)", main = "Ozone vs. Temperature"))
abline(ozone.fit)

# Look at the Q-Q Plot for the residuals
#---------------------------------------
dev.new()
qqPlot(ozone.fit$residuals, add.line = TRUE)

# Look at Box-Cox transformations of Ozone
#-----------------------------------------
boxcox.list <- boxcox(ozone.fit)
boxcox.list
```

```
#Results of Box-Cox Transformation
#---------------------------------
#
#Objective Name:                  PPCC
#
#Linear Model:                    ozone.fit
#
#Sample Size:                     116
#
# lambda      PPCC
#   -2.0 0.4286781
#   -1.5 0.4673544
#   -1.0 0.5896132
#   -0.5 0.8301458
#    0.0 0.9871519
#    0.5 0.9819825
#    1.0 0.9408694
#    1.5 0.8840770
#    2.0 0.8213675

#----------

# Clean up
#---------
rm(ozone.fit, boxcox.list)
```

---

boxcoxTransform *Apply a Box-Cox Power Transformation to a Set of Data*

Description

Apply a Box-Cox power transformation to a set of data to attempt to induce normality and homogeneity of variance.

Usage

```
boxcoxTransform(x, lambda, eps = .Machine$double.eps)
```

Arguments

x	a numeric vector of positive numbers.
lambda	finite numeric scalar indicating what power to use for the Box-Cox transformation.
eps	finite, positive numeric scalar. When the absolute value of lambda is less than eps, lambda is assumed to be 0 for the Box-Cox transformation. The default value is eps=.Machine$double.eps.

Details

Two common assumptions for several standard parametric hypothesis tests are:

1. The observations all come from a normal distribution.
2. The observations all come from distributions with the same variance.

For example, the standard one-sample t-test assumes all the observations come from the same normal distribution, and the standard two-sample t-test assumes that all the observations come from a normal distribution with the same variance, although the mean may differ between the two groups. For standard linear regression models, these assumptions can be stated as: the error terms all come from a normal distribution with mean 0 and and a constant variance.

Often, especially with environmental data, the above assumptions do not hold because the original data are skewed and/or they follow a distribution that is not really shaped like a normal distribution. It is sometimes possible, however, to transform the original data so that the transformed observations in fact come from a normal distribution or close to a normal distribution. The transformation may also induce homogeneity of variance and, for the case of a linear regression model, a linear relationship between the response and predictor variable(s).

Sometimes, theoretical considerations indicate an appropriate transformation. For example, count data often follow a Poisson distribution, and it can be shown that taking the square root of observations from a Poisson distribution tends to make these data look more bell-shaped (Johnson et al., 1992, p.163; Johnson and Wichern, 2007, p.192; Zar, 2010, p.291). A common example in the environmental field is that chemical concentration data often appear to come from a lognormal distribution or some other positively-skewed distribution (e.g., gamma). In this case, taking the logarithm of the observations often appears to yield normally distributed data.

Ideally, a data transformation is chosen based on knowledge of the process generating the data, as well as graphical tools such as quantile-quantile plots and histograms.

Box and Cox (1964) presented a formalized method for deciding on a data transformation. Given a random variable X from some distribution with only positive values, the Box-Cox family of power transformations is defined as:

$$Y = \begin{cases} \frac{X^{\lambda}-1}{\lambda} & \lambda \neq 0 \\ log(X) & \lambda = 0 \end{cases} \quad (1)$$

where Y is assumed to come from a normal distribution. This transformation is continuous in λ. Note that this transformation also preserves ordering; that is, if $X_1 < X_2$ then $Y_1 < Y_2$.

Box and Cox (1964) proposed choosing the appropriate value of λ based on maximizing a likelihood function. See the help file for `boxcox` for details.

Note that for non-zero values of λ, instead of using the formula of Box and Cox in Equation (1), you may simply use the power transformation:

$$Y = X^{\lambda} \quad (2)$$

since these two equations differ only by a scale difference and origin shift, and the essential character of the transformed distribution remains unchanged.

The value $\lambda = 1$ corresponds to no transformation. Values of λ less than 1 shrink large values of X, and are therefore useful for transforming positively-skewed (right-skewed) data. Values of λ larger

than 1 inflate large values of X, and are therefore useful for transforming negatively-skewed (left-skewed) data (Helsel and Hirsch, 1992, pp.13-14; Johnson and Wichern, 2007, p.193). Commonly used values of λ include 0 (log transformation), 0.5 (square-root transformation), -1 (reciprocal), and -0.5 (reciprocal root).

It is often recommend that when dealing with several similar data sets, it is best to find a common transformation that works reasonably well for all the data sets, rather than using slightly different transformations for each data set (Helsel and Hirsch, 1992, p.14; Shumway et al., 1989).

Value

numeric vector of transformed observations.

Note

Data transformations are often used to induce normality, homoscedasticity, and/or linearity, common assumptions of parametric statistical tests and estimation procedures. Transformations are not "tricks" used by the data analyst to hide what is going on, but rather useful tools for understanding and dealing with data (Berthouex and Brown, 2002, p.61). Hoaglin (1988) discusses "hidden" transformations that are used everyday, such as the pH scale for measuring acidity.

In the case of a linear model, there are at least two approaches to improving a model fit: transform the Y and/or X variable(s), and/or use more predictor variables. Often in environmental data analysis, we assume the observations come from a lognormal distribution and automatically take logarithms of the data. For a simple linear regression (i.e., one predictor variable), if regression diagnostic plots indicate that a straight line fit is not adequate, but that the variance of the errors appears to be fairly constant, you may only need to transform the predictor variable X or perhaps use a quadratic or cubic model in X. On the other hand, if the diagnostic plots indicate that the constant variance and/or normality assumptions are suspect, you probably need to consider transforming the response variable Y. Data transformations for linear regression models are discussed in Draper and Smith (1998, Chapter 13) and Helsel and Hirsch (1992, pp. 228-229).

One problem with data transformations is that translating results on the transformed scale back to the original scale is not always straightforward. Estimating quantities such as means, variances, and confidence limits in the transformed scale and then transforming them back to the original scale usually leads to biased and inconsistent estimates (Gilbert, 1987, p.149; van Belle et al., 2004, p.400). For example, exponentiating the confidence limits for a mean based on log-transformed data does not yield a confidence interval for the mean on the original scale. Instead, this yields a confidence interval for the median (see the help file for `elnormAlt`). It should be noted, however, that quantiles (percentiles) and rank-based procedures are invariant to monotonic transformations (Helsel and Hirsch, 1992, p.12).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Box, G.E.P., and D.R. Cox. (1964). An Analysis of Transformations (with Discussion). *Journal of the Royal Statistical Society, Series B* **26**(2), 211–252.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, pp.47-53.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY.

Hinkley, D.V., and G. Runger. (1984). The Analysis of Transformed Data (with Discussion). *Journal of the American Statistical Association* **79**, 302–320.

Hoaglin, D.C., F.M. Mosteller, and J.W. Tukey, eds. (1983). *Understanding Robust and Exploratory Data Analysis*. John Wiley and Sons, New York, Chapter 4.

Hoaglin, D.C. (1988). Transformations in Everyday Experience. *Chance* **1**, 40–45.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions, Second Edition*. John Wiley and Sons, New York, p.163.

Johnson, R.A., and D.W. Wichern. (2007). *Applied Multivariate Statistical Analysis, Sixth Edition*. Pearson Prentice Hall, Upper Saddle River, NJ, pp.192–195.

Shumway, R.H., A.S. Azari, and P. Johnson. (1989). Estimating Mean Concentrations Under Transformations for Environmental Data With Detection Limits. *Technometrics* **31**(3), 347–356.

Stoline, M.R. (1991). An Examination of the Lognormal and Box and Cox Family of Transformations in Fitting Environmental Data. *Environmetrics* **2**(1), 85–106.

van Belle, G., L.D. Fisher, Heagerty, P.J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences, 2nd Edition*. John Wiley & Sons, New York.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 13.

See Also

boxcox, Data Transformations, Goodness-of-Fit Tests.

Examples

```
# Generate 30 observations from a lognormal distribution with
# mean=10 and cv=2, then look at some normal quantile-quantile
# plots for various transformations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

dev.new()
qqPlot(x, add.line = TRUE)

dev.new()
qqPlot(boxcoxTransform(x, lambda = 0.5), add.line = TRUE)
```

```
dev.new()
qqPlot(boxcoxTransform(x, lambda = 0), add.line = TRUE)


# Clean up
#---------
rm(x)
```

`calibrate` *Fit a Calibration Line or Curve*

Description

Fit a calibration line or curve based on linear regression.

Usage

```
calibrate(formula, data, test.higher.orders = TRUE, max.order = 4, p.crit = 0.05,
  F.test = "partial", weights, subset, na.action, method = "qr", model = FALSE,
   x = FALSE, y = FALSE, contrasts = NULL, warn = TRUE, ...)
```

Arguments

`formula`	a `formula` object, with the response on the left of a ~ operator, and the single predictor variable on the right. For example, `Cadmium ~ Spike`.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `calibrate` is called.
`test.higher.orders`	logical scalar indicating whether to start with a model that contains a single predictor variable and test the fit of higher order polynomials to consider for the calibration curve (`test.higher.orders=TRUE`; the default), or to simply use the model suppled and add the model matrix to the fit if it was not already indicated by the argument `x=TRUE` in the call to `calibrate`.
`max.order`	integer indicating the maximum order of the polynomial to consider for the calibration curve. The default value is `max.order=4`, however, the final value of `max.order` is the minimum of `max.order` and value of the number of unique predictor values minus 1. So, for example, if there are only 4 unique values of the single predictor variable, then the final value of `max.order` is the minimum of what the user supplies and 3; thus, in this case, the highest order polynomial that will be potentially tested is a cubic. See also the explanation below for the argument `warn`.
`p.crit`	numeric scaler between 0 and 1 indicating the p-value to use for the stepwise regression when determining which polynomial model to use. The default value is `p.crit=0.05`.

F.test	character string indicating whether to perform the stepwise regression using the standard partial F-test (F.test="partial"; the default) or using the lack-of-fit F-test (F.test="lof").
weights	optional vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the squared residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the subset argument.
subset	optional expression saying which subset of the rows of the data should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
na.action	optional function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
method	optional method to be used; for fitting, currently only method = "qr" is supported; method = "model.frame" returns the model frame (the same as with model = TRUE, see below).
model, x, y, qr	optional logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
contrasts	an optional list. See the argument contrasts.arg of model.matrix.
warn	logical scalar indicating whether to issue a warning (warn=TRUE; the default) when the value of max.order has been decreased from what the user supplied. See also the explanation above for the argument max.order.
...	additional arguments to be passed to the low level regression fitting functions (see lm).

Details

A simple and frequently used calibration model is a straight line where the response variable S denotes the signal of the machine and the predictor variable C denotes the true concentration in the physical sample. The error term is assumed to follow a normal distribution with mean 0. Note that the average value of the signal for a blank ($C = 0$) is the intercept. Other possible calibration models include higher order polynomial models such as a quadratic or cubic model.

In a typical setup, a small number of samples (e.g., n = 6) with known concentrations are measured and the signal is recorded. A sample with no chemical in it, called a blank, is also measured. (You have to be careful to define exactly what you mean by a "blank." A blank could mean a container from the lab that has nothing in it but is prepared in a similar fashion to containers with actual samples in them. Or it could mean a field blank: the container was taken out to the field and subjected to the same process that all other containers were subjected to, except a physical sample of soil or water was not placed in the container.) Usually, replicate measures at the same known concentrations are taken. (The term "replicate" must be well defined to distinguish between for example the same physical samples that are measured more than once vs. two different physical samples of the same known concentration.)

The function `calibrate` initially fits a linear calibration model. If the argument `max.order` is greater than 1, `calibrate` then performs forward stepwise linear regression to determine the "best" polynomial model.

In the case where replicates are not availble, `calibrate` uses standard stepwise ANOVA to compare models (Draper and Smith, 1998, p.335). In this case, if the p-value for the partial F-test to compare models is greater than or equal to `p.crit`, then the model with fewer terms is used as the final model.

In the case where replicates are available, if `F.test="lof"`, then for each model `calibrate` computes the p-value of the ANOVA for lack-of-fit vs. pure error (Draper and Smith, 1998, Chapters 2; see `anovaPE`). If the p-value is greater than or equal to `p.crit`, then this is the final model; otherwise the next higher-order term is added to the polynomial and the model is re-fit. If, during the stepwise procedure, the degrees of freedom associated with the residual sums of squares of a model to be tested is less than or equal to the number of observations minus the number of unique observations, `calibrate` uses the partial F-test instead of the lack-of-fit F-test.

The stepwise algorithm terminates when either the p-value is greater than or equal to `p.crit`, or the currently selected model in the algorithm is of order `max.order`. The algorithm will terminate earlier than this if the next model to be fit includes singularities so that not all coefficients can be estimted.

Value

An object of `class` `"calibrate"` that inherits from `class` `"lm"` and includes a component called `x` that stores the model matrix (the values of the predictor variables for the final calibration model).

Note

Almost always the process of determining the concentration of a chemical in a soil, water, or air sample involves using some kind of machine that produces a signal, and this signal is related to the concentration of the chemical in the physical sample. The process of relating the machine signal to the concentration of the chemical is called **calibration**. Once calibration has been performed, estimated concentrations in physical samples with unknown concentrations are computed using inverse regression (see `inversePredictCalibrate`). The uncertainty in the process used to estimate the concentration may be quantified with decision, detection, and quantitation limits.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, Chapter 3 and p.335.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley & Sons, Hoboken. Chapter 6, p. 111.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey. Chapter 3, p. 22.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.562-575.

See Also

calibrate.object, anovaPE, inversePredictCalibrate, detectionLimitCalibrate, lm.

Examples

```
# The data frame EPA.97.cadmium.111.df contains calibration data for
# cadmium at mass 111 (ng/L) that appeared in Gibbons et al. (1997b)
# and were provided to them by the U.S. EPA.
# Display a plot of these data along with the fitted calibration line
# and 99% non-simultaneous prediction limits.  See
# Millard and Neerchal (2001, pp.566-569) for more details on this
# example.

Cadmium <- EPA.97.cadmium.111.df$Cadmium

Spike <- EPA.97.cadmium.111.df$Spike

calibrate.list <- calibrate(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)

newdata <- data.frame(Spike = seq(min(Spike), max(Spike), len = 100))

pred.list <- predict(calibrate.list, newdata = newdata, se.fit = TRUE)

pointwise.list <- pointwise(pred.list, coverage = 0.99, individual = TRUE)

dev.new()
plot(Spike, Cadmium, ylim = c(min(pointwise.list$lower),
  max(pointwise.list$upper)), xlab = "True Concentration (ng/L)",
  ylab = "Observed Concentration (ng/L)")

abline(calibrate.list, lwd = 2)

lines(newdata$Spike, pointwise.list$lower, lty = 8, lwd = 2)

lines(newdata$Spike, pointwise.list$upper, lty = 8, lwd = 2)

title(paste("Calibration Line and 99% Prediction Limits",
  "for US EPA Cadmium 111 Data", sep = "\n"))

#----------

# Clean up
#---------
rm(Cadmium, Spike, newdata, calibrate.list, pred.list, pointwise.list)
graphics.off()
```

---

calibrate.object *S3 Class "calibrate"*

Description

Objects of S3 class "calibrate" are returned by the **EnvStats** function calibrate, which fits a calibration line or curve based on linear regression.

Details

Objects of class "calibrate" are lists that inherit from class "lm" and include a component called x that stores the model matrix (the values of the predictor variables for the final calibration model).

Value

See the help file for lm.

Required Components

Besides the usual components in the list returned by the function lm, the following components must be included in a legitimate list of class "calibrate".

x	the model matrix from the linear model fit.

Methods

Generic functions that have methods for objects of class "calibrate" include:
NONE AT PRESENT.

Note

Since objects of class "calibrate" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

calibrate, inversePredictCalibrate, detectionLimitCalibrate.

Examples

```
# Create an object of class "calibrate", then print it out.

# The data frame EPA.97.cadmium.111.df contains calibration data for
# cadmium at mass 111 (ng/L) that appeared in Gibbons et al. (1997b)
# and were provided to them by the U.S. EPA.

calibrate.list <- calibrate(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)

names(calibrate.list)

calibrate.list
```

```
#----------

# Clean up
#---------
rm(calibrate.list)
```

`CastilloAndHadi1994` *Abstract: Castillo and Hadi (1994)*

Description

Detailed abstract of the manuscript:

Castillo, E., and A. Hadi. (1994). Parameter and Quantile Estimation for the Generalized Extreme-Value Distribution. *Environmetrics* **5**, 417–432.

Details

Abstract

Castillo and Hadi (1994) introduce a new way to estimate the parameters and quantiles of the generalized extreme value distribution (GEVD) with parameters `location`=η, `scale`=θ, and `shape`=κ. The estimator is based on a two-stage procedure using order statistics, denoted here by "TSOE", which stands for two-stage order-statistics estimator. Castillo and Hadi (1994) compare the TSOE to the maximum likelihood estimator (MLE; Jenkinson, 1969; Prescott and Walden, 1983) and probability-weighted moments estimator (PWME; Hosking et al., 1985).

Castillo and Hadi (1994) note that for some samples the likelihood may not have a local maximum, and also when $\kappa > 1$ the likelihood can be made infinite so the MLE does not exist. They also note, as do Hosking et al., 1985), that when $\kappa \leq -1$, the moments and probability-weighed moments of the GEVD do not exist, hence neither does the PWME. (Hosking et al., however, claim that in practice the shape parameter usually lies between -1/2 and 1/2.) On the other hand, the TSOE exists for all values of κ.

Based on computer simulations, Castillo and Hadi (1994) found that the performance (bias and root mean squared error) of the TSOE is comparable to the PWME for values of κ in the range $-1/2 \leq \kappa \leq 1/2$. They also found that the TSOE is superior to the PWME for large values of κ. Their results, however, are based on using the PWME computed using the approximation given in equation (14) of Hosking et al. (1985, p.253). The true PWME is computed using equation (12) of Hosking et al. (1985, p.253). Hosking et al. (1985) introduced the approximation as a matter of computational convenience, and noted that it is valid in the range $-1/2 \leq \kappa \leq 1/2$. If Castillo and Hadi (1994) had used the true PWME for values of κ larger than 1/2, they probably would have gotten very different results for the PWME. (Note: the function `egevd` with `method="pwme"` uses the exact equation (12) of Hosking et al. (1985), not the approximation (14)).

Castillo and Hadi (1994) suggest using the bootstrap or jackknife to obtain variance estimates and confidence intervals for the distribution parameters based on the TSOE.

More Details Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a generalized extreme value distribution with parameters `location`=η, `scale`=θ, and `shape`=κ with cumulative distribution function F. Also, let $x(1), x(2), \ldots, x(n)$ denote the ordered values of $\underline{x}$.

First Stage

Castillo and Hadi (1994) propose as initial estimates of the distribution parameters the solutions to the following set of simultaneous equations based on just three observations from the total sample of size n:

$$F[x(1); \eta, \theta, \kappa] = p_{1,n}$$

$$F[x(j); \eta, \theta, \kappa] = p_{j,n}$$

$$F[x(n); \eta, \theta, \kappa] = p_{n,n} \quad (1)$$

where $2 \leq j \leq n-1$, and

$$p_{i,n} = \hat{F}[x(i); \eta, \theta, \kappa]$$

denotes the i'th plotting position for a sample of size n; that is, a nonparametric estimate of the value of F at $x(i)$. Typically, plotting positions have the form:

$$p_{i,n} = \frac{i-a}{n+b} \quad (2)$$

where $b > -a > -1$. In their simulation studies, Castillo and Hadi (1994) used a=0.35, b=0.

Since j is arbitrary in the above set of equations (1), denote the solutions to these equations by:

$$\hat{\eta}_j, \hat{\theta}_j, \hat{\kappa}_j$$

There are thus $n-2$ sets of estimates.

Castillo and Hadi (1994) show that the estimate of the shape parameter, κ, is the solution to the equation:

$$\frac{x(j) - x(n)}{x(1) - x(n)} = \frac{1 - A_{jn}^{\kappa}}{1 - A_{1n}^{\kappa}} \quad (3)$$

where

$$A_{ik} = C_i / C_k \quad (4)$$

$$C_i = -log(p_{i,n}) \quad (5)$$

Castillo and Hadi (1994) show how to easily solve equation (3) using the method of bisection.

Once the estimate of the shape parameter is obtained, the other estimates are given by:

$$\hat{\theta}_j = \frac{\hat{\kappa}_j [x(1) - x(n)]}{(C_n)^{\hat{\kappa}_j} - (C_1)^{\hat{\kappa}_j}} \quad (6)$$

$$\hat{\eta}_j = x(1) - \frac{\hat{\theta}_j [1 - (C_1)^{\hat{\kappa}_j}]}{\hat{\kappa}_j} \quad (7)$$

Second Stage

Apply a robust function to the $n-2$ sets of estimates obtained in the first stage. Castillo and Hadi (1994) suggest using either the median or the least median of squares (using a column of 1's as

the predictor variable; see the help file for lmsreg in the package **MASS**). Using the median, for example, the final distribution parameter estimates are given by:

$$\hat{\eta} = Median(\hat{\eta}_2, \hat{\eta}_3, \ldots, \hat{\eta}_{n-1})$$

$$\hat{\theta} = Median(\hat{\theta}_2, \hat{\theta}_3, \ldots, \hat{\theta}_{n-1})$$

$$\hat{\kappa} = Median(\hat{\kappa}_2, \hat{\kappa}_3, \ldots, \hat{\kappa}_{n-1})$$

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Hosking, J.R.M. (1985). Algorithm AS 215: Maximum-Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Applied Statistics* **34**(3), 301–310.

Jenkinson, A.F. (1969). Statistics of Extremes. *Technical Note 98*, World Meteorological Office, Geneva.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

Prescott, P., and A.T. Walden. (1983). Maximum Likelihood Estimation of the Three-Parameter Generalized Extreme-Value Distribution from Censored Samples. *Journal of Statistical Computing and Simulation* **16**, 241–250.

See Also

Generalized Extreme Value Distribution, egevd, Hosking et al., 1985).

---

cdfCompare *Plot Two Cumulative Distribution Functions*

Description

For one sample, plots the empirical cumulative distribution function (ecdf) along with a theoretical cumulative distribution function (cdf). For two samples, plots the two ecdf's. These plots are used to graphically assess goodness of fit.

Usage

```
cdfCompare(x, y = NULL, discrete = FALSE,
 prob.method = ifelse(discrete, "emp.probs", "plot.pos"), plot.pos.con = NULL,
  distribution = "norm", param.list = NULL,
  estimate.params = is.null(param.list), est.arg.list = NULL,
  x.col = "blue", y.or.fitted.col = "black",
  x.lwd = 3 * par("cex"), y.or.fitted.lwd = 3 * par("cex"),
  x.lty = 1, y.or.fitted.lty = 2, digits = .Options$digits, ...,
  type = ifelse(discrete, "s", "l"), main = NULL, xlab = NULL, ylab = NULL,
  xlim = NULL, ylim = NULL)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
y	a numeric vector (not necessarily of the same length as x). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The default value is y=NULL, in which case the empirical cdf of x will be plotted along with the theoretical cdf specified by the argument distribution.
discrete	logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).
prob.method	character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are plot.pos (plotting positions, the default if discrete=FALSE) and emp.probs (empirical probabilities, the default if discrete=TRUE). See the help file for ecdfPlot for more explanation.
plot.pos.con	numeric scalar between 0 and 1 containing the value of the plotting position constant. When y is supplied, the default value is plot.pos.con=0.375. When y is not supplied, for the normal, lognormal, three-parameter lognormal, zero-modified normal, and zero-modified lognormal distributions, the default value is plot.pos.con=0.375. For the Type I extreme value (Gumbel) distribution (distribution="evd"), the default value is plot.pos.con=0.44. For all other distributions, the default value is plot.pos.con=0.4. See the help files for ecdfPlot and qqPlot for more information. This argument is ignored if prob.method="emp.probs".
distribution	when y is not supplied, a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations. This argument is ignored if y is supplied.
param.list	when y is not supplied, a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. This argument is ignored if y is supplied or estimate.params=TRUE.
estimate.params	when y is not supplied, a logical scalar indicating whether to compute the cdf for x based on estimating the distribution parameters (estimate.params=TRUE)

or using the known distribution parameters specified in `param.list` (`estimate.params=FALSE`). The default value is `TRUE` unless the argument `param.list` is supplied. The argument `estimate.params` is ignored if `y` is supplied.

`est.arg.list` when `y` is not supplied and `estimate.params=TRUE`, a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the help file Estimating Distribution Parameters). For example, all functions used to estimate distribution parameters have an optional argument called `method` that specifies the method to use to estimate the parameters. (See the help file for `Distribution.df` for a list of available estimation methods for each distribution.) To override the default estimation method, supply the argument `est.arg.list` with a component called `method`; for example `est.arg.list=list(method="mle")`. The default value is `est.arg.list=NULL` so that all default values for the estimating function are used. This argument is ignored if `estimate.params=FALSE` or `y` is supplied.

`x.col` a numeric scalar or character string determining the color of the empirical cdf (based on `x`) line or points. The default value is `x.col="blue"`. See the entry for `col` in the help file for `par` for more information.

`y.or.fitted.col`
a numeric scalar or character string determining the color of the empirical cdf (based on `y`) or the theoretical cdf line or points. The default value is `y.or.fitted.col="black"`. See the entry for `col` in the help file for `par` for more information.

`x.lwd` a numeric scalar determining the width of the empirical cdf (based on `x`) line. The default value is `x.lwd=3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`y.or.fitted.lwd`
a numeric scalar determining the width of the empirical cdf (based on `y`) or theoretical cdf line. The default value is `y.or.fitted.lwd=3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`x.lty` a numeric scalar determining the line type of the empirical cdf (based on `x`) line. The default value is `x.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`y.or.fitted.lty`
a numeric scalar determining the line type of the empirical cdf (based on `y`) or theoretical cdf line. The default value is `y.or.fitted.lty=2`. See the entry for `lty` in the help file for `par` for more information.

`digits` when `y` is not supplied, a scalar indicating how many significant digits to print for the distribution parameters. The default value is `digits=.Options$digits`.

`type, main, xlab, ylab, xlim, ylim, ...`
additional graphical parameters (see `lines` and `par`). In particular, the argument `type` specifies the kind of line type. By default, the function `cdfCompare` plots a step function (`type="s"`) when `discrete=TRUE`, and plots a straight line between points (`type="l"`) when `discrete=FALSE`. The user may override these defaults by supplying the graphics parameter `type` (`type="s"` for a step function, `type="l"` for linear interpolation, `type="p"` for points only, etc.).

Details

When both x and y are supplied, the function cdfCompare creates the empirical cdf plot of x and y on the same plot by calling the function ecdfPlot.

When y is not supplied, the function cdfCompare creates the emprical cdf plot of x (by calling ecdfPlot) and the theoretical cdf plot (by calling cdfPlot and using the argument distribution) on the same plot.

Value

When y is supplied, cdfCompare invisibly returns a list with components:

x.ecdf.list — a list with components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the x values.

y.ecdf.list — a list with components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the y values.

When y is not supplied, cdfCompare invisibly returns a list with components:

x.ecdf.list — a list with components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the x values.

fitted.cdf.list — a list with components Quantiles and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the fitted cdf.

Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the characteristics of a set of data. It is easy to determine quartiles and the minimum and maximum values from such a plot. Also, ecdf plots allow you to assess local density: a higher density of observations occurs where the slope is steep.

Chambers et al. (1983, pp.11-16) plot the observed order statistics on the y-axis vs. the ecdf on the x-axis and call this a quantile plot.

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots (see cdfPlot and cdfCompare) to graphically assess whether a sample of observations comes from a particular distribution. The Kolmogorov-Smirnov goodness-of-fit test (see gofTest) is the statistical companion of this kind of comparison; it is based on the maximum vertical distance between the empirical cdf plot and the theoretical cdf plot. More often, however, quantile-quantile (Q-Q) plots are used instead of ecdf plots to graphically assess departures from an assumed distribution (see qqPlot).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11-16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7-62.

See Also

cdfPlot, ecdfPlot, qqPlot.

Examples

```
# Generate 20 observations from a normal (Gaussian) distribution
# with mean=10 and sd=2 and compare the empirical cdf with a
# theoretical normal cdf that is based on estimating the parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rnorm(20, mean = 10, sd = 2)
dev.new()
cdfCompare(x)

#----------

# Generate 30 observations from an exponential distribution with parameter
# rate=0.1 (see the R help file for Exponential) and compare the empirical
# cdf with the empirical cdf of the normal observations generated in the
# previous example:

set.seed(432)
y <- rexp(30, rate = 0.1)
dev.new()
cdfCompare(x, y)

#==========

# Generate 20 observations from a Poisson distribution with parameter lambda=10
# (see the R help file for Poisson) and compare the empirical cdf with a
# theoretical Poisson cdf based on estimating the distribution parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rpois(20, lambda = 10)
dev.new()
cdfCompare(x, dist = "pois")

#==========

# Clean up
#---------
```

```
rm(x, y)
graphics.off()
```

---

cdfCompareCensored *Plot Two Cumulative Distribution Functions Based on Censored Data*

Description

For one sample, plots the empirical cumulative distribution function (ecdf) along with a theoretical cumulative distribution function (cdf). For two samples, plots the two ecdf's. These plots are used to graphically assess goodness of fit.

Usage

```
cdfCompareCensored(x, censored, censoring.side = "left",
  y = NULL, y.censored = NULL, y.censoring.side = censoring.side,
  discrete = FALSE, prob.method = "michael-schucany",
  plot.pos.con = NULL, distribution = "norm", param.list = NULL,
  estimate.params = is.null(param.list), est.arg.list = NULL,
  x.col = "blue", y.or.fitted.col = "black", x.lwd = 3 * par("cex"),
  y.or.fitted.lwd = 3 * par("cex"), x.lty = 1, y.or.fitted.lty = 2,
  include.x.cen = FALSE, x.cen.pch = ifelse(censoring.side == "left", 6, 2),
  x.cen.cex = par("cex"), x.cen.col = "red",
  include.y.cen = FALSE, y.cen.pch = ifelse(y.censoring.side == "left", 6, 2),
  y.cen.cex = par("cex"), y.cen.col = "black", digits = .Options$digits, ...,
  type = ifelse(discrete, "s", "l"), main = NULL, xlab = NULL, ylab = NULL,
  xlim = NULL, ylim = NULL)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
y	a numeric vector (not necessarily of the same length as x). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The default value is y=NULL, in which case the empirical cdf of x will be plotted along with the theoretical cdf specified by the argument distribution.

y.censored numeric or logical vector indicating which values of y are censored. This must be the same length as y. If the mode of censored is "logical", TRUE values correspond to elements of y that are censored, and FALSE values correspond to elements of y that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.

This argument is ignored when y is not supplied. The default value is y.censored=NULL since the default value of y is y=NULL.

y.censoring.side character string indicating on which side the censoring occurs for the values of y. The possible values are "left" (the default) and "right". This argument is ignored when y is not supplied. The default value is y.censoring.side=censoring.side.

discrete logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).

prob.method character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany".

The "nelson" method is only available for censoring.side="right". See the help file for ecdfPlotCensored for more explanation.

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant. When y is supplied, the default value is plot.pos.con=0.375. When y is not supplied, for the normal, lognormal, three-parameter lognormal, zero-modified normal, and zero-modified lognormal distributions, the default value is plot.pos.con=0.375. For the Type I extreme value (Gumbel) distribution (distribution="evd"), the default value is plot.pos.con=0.44. For all other distributions, the default value is plot.pos.con=0.4. See the help files for ecdfPlot and qqPlot for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

distribution when y is not supplied, a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations. This argument is ignored if y is supplied.

param.list when y is not supplied, a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. This argument is ignored if y is supplied or estimate.params=TRUE.

estimate.params when y is not supplied, a logical scalar indicating whether to compute the cdf for x based on estimating the distribution parameters (estimate.params=TRUE) or using the known distribution parameters specified in param.list

(estimate.params=FALSE). The default value is TRUE unless the argument param.list is supplied. The argument estimate.params is ignored if y is supplied.

est.arg.list when y is not supplied and estimate.params=TRUE, a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the Section **Estimating Distribution Parameters** in the help file Censored Data). For example, all functions used to estimate distribution parameters have an optional argument called method that specifies the method to use to estimate the parameters. (See the help file for Distribution.df for a list of available estimation methods for each distribution.) To override the default estimation method, supply the argument est.arg.list with a component called method; for example est.arg.list=list(method="mle"). The default value is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE or y is supplied.

x.col a numeric scalar or character string determining the color of the empirical cdf (based on x) line or points. The default value is x.col="blue". See the entry for col in the help file for par for more information.

y.or.fitted.col
a numeric scalar or character string determining the color of the empirical cdf (based on y) or the theoretical cdf line or points. The default value is y.or.fitted.col="black". See the entry for col in the help file for par for more information.

x.lwd a numeric scalar determining the width of the empirical cdf (based on x) line. The default value is x.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.

y.or.fitted.lwd
a numeric scalar determining the width of the empirical cdf (based on y) or theoretical cdf line. The default value is y.or.fitted.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.

x.lty a numeric scalar determining the line type of the empirical cdf (based on x) line. The default value is x.lty=1. See the entry for lty in the help file for par for more information.

y.or.fitted.lty
a numeric scalar determining the line type of the empirical cdf (based on y) or theoretical cdf line. The default value is y.or.fitted.lty=2. See the entry for lty in the help file for par for more information.

include.x.cen logical scalar indicating whether to include censored values in x in the plot. The default value is include.x.cen=FALSE. If include.x.cen=TRUE, censored values in x are plotted using the plotting character indicated by the argument x.cen.pch (see below). This argument is ignored if there are no censored values in x.

x.cen.pch numeric scalar or character string indicating the plotting character to use to plot censored values in x. The default value is x.cen.pch=2 (hollow triangle pointing up) when x.censoring.side="right", and x.cen.pch=6 (hollow triangle pointing down) when x.censoring.side="left". See the R help file for

points for an explanation of how plotting symbols are specified. This argument is ignored if include.x.cen=FALSE.

x.cen.cex numeric scalar that determines the size of the plotting character used to plot censored values in x. The default value is the current value of the cex graphics parameter. See the entry for cex in the R help file for par for more information. This argument is ignored if include.x.cen=FALSE.

x.cen.col numeric scalar or character string that determines the color of the plotting character used to plot censored values in x. The default value is x.cen.col="red". See the entry for col in the R help file for par for more information. This argument is ignored if include.x.cen=FALSE.

include.y.cen logical scalar indicating whether to include censored values in y in the plot. The default value is include.y.cen=FALSE. If include.y.cen=TRUE, censored values in y are plotted using the plotting character indicated by the argument y.cen.pch (see below). This argument is ignored if y is not supplied and/or there are no censored values in y.

y.cen.pch numeric scalar or character string indicating the plotting character to use to plot censored values in y. The default value is y.cen.pch=2 (hollow triangle pointing up) when y.censoring.side="right", and y.cen.pch=6 (hollow triangle pointing down) when y.censoring.side="left". See the R help file for points for an explanation of how plotting symbols are specified. This argument is ignored if include.y.cen=FALSE.

y.cen.cex numeric scalar that determines the size of the plotting character used to plot censored values in y. The default value is the current value of the cex graphics parameter. See the entry for cex in the R help file for par for more information. This argument is ignored if include.y.cen=FALSE.

y.cen.col numeric scalar or character string that determines the color of the plotting character used to plot censored values in y. The default value is y.cen.col="black". See the entry for col in the R help file for par for more information. This argument is ignored if include.y.cen=FALSE.

digits when y is not supplied, a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

type, main, xlab, ylab, xlim, ylim, ... additional graphical parameters (see lines and par). In particular, the argument type specifies the kind of line type. By default, the function cdfCompareCensored plots a step function (type="s") when discrete=TRUE, and plots a straight line between points (type="l") when discrete=FALSE. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).

Details

When both x and y are supplied, the function cdfCompareCensored creates the empirical cdf plot of x and y on the same plot by calling the function ecdfPlotCensored.

When y is not supplied, the function cdfCompareCensored creates the emprical cdf plot of x (by calling ecdfPlotCensored) and the theoretical cdf plot (by calling cdfPlot and using the argument distribution) on the same plot.

Value

When y is supplied, cdfCompareCensored invisibly returns a list with components:

x.ecdf.list — a list with components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the x values.

y.ecdf.list — a list with components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the y values.

When y is not supplied, cdfCompareCensored invisibly returns a list with components:

x.ecdf.list — a list with components Order.Statistics and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the x values.

fitted.cdf.list — a list with components Quantiles and Cumulative.Probabilities, giving coordinates of the points that have been plotted for the fitted cdf.

Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the characteristics of a set of data. It is easy to determine quartiles and the minimum and maximum values from such a plot. Also, ecdf plots allow you to assess local density: a higher density of observations occurs where the slope is steep.

Chambers et al. (1983, pp.11-16) plot the observed order statistics on the y-axis vs. the ecdf on the x-axis and call this a quantile plot.

Censored observations complicate the procedures used to graphically explore data. Techniques from survival analysis and life testing have been developed to generalize the procedures for constructing plotting positions, empirical cdf plots, and q-q plots to data sets with censored observations (see ppointsCensored).

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots to graphically assess whether a sample of observations comes from a particular distribution. More often, however, quantile-quantile (Q-Q) plots are used instead of ecdf plots to graphically assess departures from an assumed distribution (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11-16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7-62.

Gillespie, B.W., Q. Chen, H. Reichert, A. Franzblau, E. Hedgeman, J. Lepkowski, P. Adriaens, A. Demond, W. Luksemburg, and D.H. Garabrant. (2010). Estimating Population Distributions

When Some Data Are Below a Limit of Detection by Using a Reverse Kaplan-Meier Estimator. *Epidemiology* **21**(4), S64–S70.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997-2004.

Hirsch, R.M., and J.R. Stedinger. (1987). Plotting Positions for Historical Floods and Their Precision. *Water Resources Research* **23**(4), 715-727.

Kaplan, E.L., and P. Meier. (1958). Nonparametric Estimation From Incomplete Observations. *Journal of the American Statistical Association* **53**, 457-481.

Lee, E.T., and J.W. Wang. (2003). *Statistical Methods for Survival Data Analysis, Third Edition*. John Wiley & Sons, Hoboken, New Jersey, 513pp.

Michael, J.R., and W.R. Schucany. (1986). Analysis of Data from Censored Samples. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, 560pp, Chapter 11, 461-496.

Nelson, W. (1972). Theory and Applications of Hazard Plotting for Censored Failure Data. *Technometrics* **14**, 945-966.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. Chapter 15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

cdfPlot, ecdfPlotCensored, qqPlotCensored.

Examples

```
# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then compare the empirical cdf with a
# theoretical normal cdf that is based on estimating the parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(333)
x <- sort(rnorm(20, mean=20, sd=5))
x
# [1]  9.743551 12.370197 14.375499 15.628482 15.883507 17.080124
# [7] 17.197588 18.097714 18.654182 19.585942 20.219308 20.268505
#[13] 20.552964 21.388695 21.763587 21.823639 23.168039 26.165269
#[19] 26.843362 29.673405

censored <- x < 18
x[censored] <- 18
```

```
sum(censored)
#[1] 7

dev.new()
cdfCompareCensored(x, censored)

# Clean up
#---------
rm(x, censored)

#==========

# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# computing plotting positions based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring
# wells.  The data for this example are stored in
# EPA.09.Ex.15.1.manganese.df.  Here we will compare the empirical
# cdf based on Kaplan-Meier plotting positions or Michael-Schucany
# plotting positions with various assumed distributions
# (based on estimating the parameters of these distributions):
# 1) normal distribution
# 2) lognormal distribution
# 3) gamma distribution

# First look at the data:
#------------------------

EPA.09.Ex.15.1.manganese.df
#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#4       4 Well.1               21.6          21.6    FALSE
#5       5 Well.1                 <2           2.0     TRUE
#...
#21      1 Well.5               17.9          17.9    FALSE
#22      2 Well.5               22.7          22.7    FALSE
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2
```

```
# Assume a normal distribution
#-----------------------------

# Michael-Schucany plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  cdfCompareCensored(Manganese.ppb, Censored))

# Kaplan-Meier plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  cdfCompareCensored(Manganese.ppb, Censored,
    prob.method = "kaplan-meier"))


# Assume a lognormal distribution
#--------------------------------

# Michael-Schucany plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  cdfCompareCensored(Manganese.ppb, Censored, dist = "lnorm"))

# Kaplan-Meier plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  cdfCompareCensored(Manganese.ppb, Censored, dist = "lnorm",
    prob.method = "kaplan-meier"))


# Assume a gamma distribution
#----------------------------

# Michael-Schucany plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  cdfCompareCensored(Manganese.ppb, Censored, dist = "gamma"))

# Kaplan-Meier plotting positions:
dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  cdfCompareCensored(Manganese.ppb, Censored, dist = "gamma",
    prob.method = "kaplan-meier"))

# Clean up
#---------
graphics.off()

#==========

# Compare the distributions of copper and zinc between the Alluvial Fan Zone
# and the Basin-Trough Zone using the data of Millard and Deverel (1988).
# The data are stored in Millard.Deverel.88.df.
```

```
Millard.Deverel.88.df
#    Cu.orig Cu Cu.censored Zn.orig  Zn Zn.censored         Zone Location
#1       < 1  1        TRUE     <10  10        TRUE Alluvial.Fan        1
#2       < 1  1        TRUE       9   9       FALSE Alluvial.Fan        2
#3         3  3       FALSE      NA  NA       FALSE Alluvial.Fan        3
#.
#.
#.
#116       5  5       FALSE      50  50       FALSE Basin.Trough       48
#117      14 14       FALSE      90  90       FALSE Basin.Trough       49
#118       4  4       FALSE      20  20       FALSE Basin.Trough       50

Cu.AF <- with(Millard.Deverel.88.df,
  Cu[Zone == "Alluvial.Fan"])

Cu.AF.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Alluvial.Fan"])

Cu.BT <- with(Millard.Deverel.88.df,
  Cu[Zone == "Basin.Trough"])

Cu.BT.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Basin.Trough"])

Zn.AF <- with(Millard.Deverel.88.df,
  Zn[Zone == "Alluvial.Fan"])

Zn.AF.cen <- with(Millard.Deverel.88.df,
  Zn.censored[Zone == "Alluvial.Fan"])

Zn.BT <- with(Millard.Deverel.88.df,
  Zn[Zone == "Basin.Trough"])

Zn.BT.cen <- with(Millard.Deverel.88.df,
  Zn.censored[Zone == "Basin.Trough"])


# First compare the copper concentrations
#----------------------------------------
dev.new()
cdfCompareCensored(x = Cu.AF, censored = Cu.AF.cen,
  y = Cu.BT, y.censored = Cu.BT.cen)


# Now compare the zinc concentrations
#------------------------------------
dev.new()
cdfCompareCensored(x = Zn.AF, censored = Zn.AF.cen,
  y = Zn.BT, y.censored = Zn.BT.cen)


# Compare the Zinc concentrations again, but delete
```

```
# the one "outlier".
#-------------------------------------------------

summaryStats(Zn.AF)
#       N    Mean      SD Median Min Max NA's N.Total
#Zn.AF 67 23.5075 74.4192     10   3 620    1      68

summaryStats(Zn.BT)
#       N  Mean      SD Median Min Max
#Zn.BT 50 21.94 18.7044   18.5   3  90

which(Zn.AF == 620)
#[1] 38

summaryStats(Zn.AF[-38])
#            N    Mean     SD Median Min Max NA's N.Total
#Zn.AF[-38] 66 14.4697 8.1604     10   3  50    1      67


dev.new()
cdfCompareCensored(x = Zn.AF[-38], censored = Zn.AF.cen[-38],
  y = Zn.BT, y.censored = Zn.BT.cen)

#----------

# Clean up
#---------

rm(Cu.AF, Cu.AF.cen, Cu.BT, Cu.BT.cen,
   Zn.AF, Zn.AF.cen, Zn.BT, Zn.BT.cen)
graphics.off()
```

---

cdfPlot *Plot Cumulative Distribution Function*

Description

Produce a cumulative distribution function (cdf) plot for a user-specified distribution.

Usage

```
cdfPlot(distribution = "norm", param.list = list(mean = 0, sd = 1),
  left.tail.cutoff = ifelse(is.finite(supp.min), 0, 0.001),
  right.tail.cutoff = ifelse(is.finite(supp.max), 0, 0.001), plot.it = TRUE,
  add = FALSE, n.points = 1000, cdf.col = "black", cdf.lwd = 3 * par("cex"),
  cdf.lty = 1, curve.fill = FALSE, curve.fill.col = "cyan",
  digits = .Options$digits, ..., type = ifelse(discrete, "s", "l"),
  main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL)
```

Arguments

distribution	a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations.
param.list	a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution.
left.tail.cutoff	a numeric scalar indicating what proportion of the left-tail of the probability distribution to omit from the plot. For densities with a finite support minimum (e.g., Lognormal) the default value is 0; for all other densities the default value is 0.001.
right.tail.cutoff	a scalar indicating what proportion of the right-tail of the probability distribution to omit from the plot. For densities with a finite support maximum (e.g., Binomial) the default value is 0; for all other densities the default value is 0.001.
plot.it	a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x, y) values is returned (see the section VALUE below). The default value is plot.it=TRUE.
add	a logical scalar indicating whether to add the cumulative distribution function curve to the existing plot (add=TRUE), or to create a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.
n.points	a numeric scalar specifying at how many evenly-spaced points the cumulative distribution function will be evaluated. The default value is n.points=1000.
cdf.col	a numeric scalar or character string determining the color of the cdf line in the plot. The default value is pdf.col="black". See the entry for col in the help file for par for more information.
cdf.lwd	a numeric scalar determining the width of the cdf line in the plot. The default value is pdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.
cdf.lty	a numeric scalar determining the line type of the cdf line in the plot. The default value is pdf.lty=1. See the entry for lty in the help file for par for more information.
curve.fill	a logical value indicating whether to fill in the area below the cumulative distribution function curve with the color specified by curve.fill.col. The default value is curve.fill=FALSE.
curve.fill.col	when curve.fill=TRUE, a numeric scalar or character string indicating what color to use to fill in the area below the cumulative distribution function curve. The default value is curve.fill.col="cyan". See the entry for col in the help file for par for more information.
digits	a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

`type`, `main`, `xlab`, `ylab`, `xlim`, `ylim`, `...`
: additional graphical parameters (see `lines` and `par`). In particular, the argument `type` specifies the kind of line type. By default, the function `cdfPlot` plots a step function (`type="s"`) for discrete distributions, and plots a straight line between points (`type="l"`) otherwise. The user may override these defaults by supplying the graphics parameter `type` (`type="s"` for a step function, `type="l"` for linear interpolation, `type="p"` for points only, etc.).

Details

The ***cumulative distribution function (cdf)*** of a random variable X, usually denoted F, is defined as:

$$F(x) = Pr(X \le x) \qquad (1)$$

That is, $F(x)$ is the probability that X is less than or equal to x. This is the probability that the random variable X takes on a value in the interval $(-\infty, x]$ and is simply the (Lebesgue) integral of the pdf evaluated between $-\infty$ and x. That is,

$$F(x) = Pr(X \le x) = \int_{-\infty}^{x} f(t)dt \qquad (2)$$

where $f(t)$ denotes the probability density function of X evaluated at t. For discrete distributions, Equation (2) translates to summing up the probabilities of all values in this interval:

$$F(x) = Pr(X \le x) = \sum_{t \in (-\infty, x]} f(t) = \sum_{t \in (-\infty, x]} Pr(X = t) \qquad (3)$$

A ***cumulative distribution function (cdf) plot*** plots the values of the cdf against quantiles of the specified distribution. Theoretical cdf plots are sometimes plotted along with empirical cdf plots to visually assess whether data have a particular distribution.

Value

`cdfPlot` invisibly returns a list giving coordinates of the points that have been or would have been plotted:

`Quantiles`
: The quantiles used for the plot.

`Cumulative.Probabilities`
: The values of the cdf associated with the quantiles.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions, Second Edition*. John Wiley and Sons, New York.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

Distribution.df, ecdfPlot, cdfCompare, pdfPlot.

Examples

```
# Plot the cdf of the standard normal distribution
#-------------------------------------------------
dev.new()
cdfPlot()

#==========

# Plot the cdf of the standard normal distribution
# and a N(2, 2) distribution on the sample plot.
#-------------------------------------------------
dev.new()
cdfPlot(param.list = list(mean=2, sd=2), main = "")

cdfPlot(add = TRUE, cdf.col = "red")

legend("topleft", legend = c("N(2,2)", "N(0,1)"),
  col = c("black", "red"), lwd = 3 * par("cex"))

title("CDF Plots for Two Normal Distributions")

#==========

# Clean up
#---------
graphics.off()
```

---

chenTTest *Chen's Modified One-Sided t-test for Skewed Distributions*

Description

For a skewed distribution, estimate the mean, standard deviation, and skew; test the null hypothesis that the mean is equal to a user-specified value vs. a one-sided alternative; and create a one-sided confidence interval for the mean.

Usage

```
chenTTest(x, y = NULL, alternative = "greater", mu = 0, paired = !is.null(y),
  conf.level = 0.95, ci.method = "z")
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
y	optional numeric vector of observations that are paired with the observations in x. The length of y must be the same as the length of x. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. This argument is ignored if paired=FALSE, and must be supplied if paired=TRUE. The default value is y=NULL.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "greater" (the default) and "less". The value "greater" should be used for positively-skewed distributions, and the value "less" should be used for negatively-skewed distributions.
mu	numeric scalar indicating the hypothesized value of the mean. The default value is mu=0.
paired	character string indicating whether to perform a paired or one-sample t-test. The possible values are paired=FALSE (the default; indicates a one-sample t-test) and paired=TRUE.
conf.level	numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the population mean. The default value is conf.level=0.95.
ci.method	character string indicating which critical value to use to construct the confidence interval for the mean. The possible values are "z" (the default), "t", and "Avg. of z and t". See the DETAILS section below for more information.

Details

One-Sample Case (paired=FALSE)
Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n independent and identically distributed (i.i.d.) observations from some distribution with mean μ and standard deviation σ.

Background: The Conventional Student's t-Test
Assume that the n observations come from a normal (Gaussian) distribution, and consider the test of the null hypothesis:

$$H_0 : \mu = \mu_0 \qquad (1)$$

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

$$H_a : \mu > \mu_0 \qquad (2)$$

the lower one-sided alternative (alternative="less"):

$$H_a : \mu < \mu_0 \qquad (3)$$

and the two-sided alternative:

$$H_a : \mu \neq \mu_0 \qquad (4)$$

The test of the null hypothesis (1) versus any of the three alternatives (2)-(4) is usually based on the Student t-statistic:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} \qquad (5)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (6)$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (7)$$

(see the R help file for `t.test`). Under the null hypothesis (1), the t-statistic in (5) follows a Student's t-distribution with $n-1$ degrees of freedom (Zar, 2010, p.99; Johnson et al., 1995, pp.362-363). The t-statistic is fairly robust to departures from normality in terms of maintaining Type I error and power, provided that the sample size is sufficiently large.

Chen's Modified t-Test for Skewed Distributions

In the case when the underlying distribution of the n observations is positively skewed and the sample size is small, the sampling distribution of the t-statistic under the null hypothesis (1) does not follow a Student's t-distribution, but is instead negatively skewed. For the test against the upper alternative in (2) above, this leads to a Type I error smaller than the one assumed and a loss of power (Chen, 1995b, p.767).

Similarly, in the case when the underlying distribution of the n observations is negatively skewed and the sample size is small, the sampling distribution of the t-statistic is positively skewed. For the test against the lower alternative in (3) above, this also leads to a Type I error smaller than the one assumed and a loss of power.

In order to overcome these problems, Chen (1995b) proposed the following modified t-statistic that takes into account the skew of the underlying distribution:

$$t_2 = t + a(1 + 2t^2) + 4a^2(t + 2t^3) \qquad (8)$$

where

$$a = \frac{\sqrt{\hat{\beta}_1}}{6n} \qquad (9)$$

$$\hat{\beta}_1 = \frac{\hat{\mu}_3}{\hat{\sigma}^3} \qquad (10)$$

$$\hat{\mu}_3 = \frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}(x_i - \bar{x})^3 \qquad (11)$$

$$\hat{\sigma}^3 = s^3 = [\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2]^{3/2} \qquad (12)$$

Note that the quantity $\sqrt{\hat{\beta}_1}$ in (9) is an estimate of the skew of the underlying distribution and is based on unbiased estimators of central moments (see the help file for `skewness`).

For a positively-skewed distribution, Chen's modified t-test rejects the null hypothesis (1) in favor of the upper one-sided alternative (2) if the t-statistic in (8) is too large. For a negatively-skewed distribution, Chen's modified t-test rejects the null hypothesis (1) in favor of the lower one-sided alternative (3) if the t-statistic in (8) is too small.

Chen's modified t-test is **not** applicable to testing the two-sided alternative (4). It should also **not** be used to test the upper one-sided alternative (2) based on negatively-skewed data, nor should it be

used to test the lower one-sided alternative (3) based on positively-skewed data.

Determination of Critical Values and p-Values
Chen (1995b) performed a simulation study in which the modified t-statistic in (8) was compared to a critical value based on the normal distribution (z-value), a critical value based on Student's t-distribution (t-value), and the average of the critical z-value and t-value. Based on the simulation study, Chen (1995b) suggests using either the z-value or average of the z-value and t-value when n (the sample size) is small (e.g., $n \leq 10$) or α (the Type I error) is small (e.g. $\alpha \leq 0.01$), and using either the t-value or the average of the z-value and t-value when $n \geq 20$ or $\alpha \geq 0.05$.

The function `chenTTest` returns three different p-values: one based on the normal distribution, one based on Student's t-distribution, and one based on the average of these two p-values. This last p-value should roughly correspond to a p-value based on the distribution of the average of a normal and Student's t random variable.

Computing Confidence Intervals
The function `chenTTest` computes a one-sided confidence interval for the true mean μ based on finding all possible values of μ for which the null hypothesis (1) will not be rejected, with the confidence level determined by the argument `conf.level`. The argument `ci.method` determines which p-value is used in the algorithm to determine the bounds on μ. When `ci.method="z"`, the p-value is based on the normal distribution, when `ci.method="t"`, the p-value is based on Student's t-distribution, and when `ci.method="Avg. of z and t"` the p-value is based on the average of the p-values based on the normal and Student's t-distribution.

Paired-Sample Case (`paired=TRUE`)
When the argument `paired=TRUE`, the arguments `x` and `y` are assumed to have the same length, and the n differences

$$d_i = x_i - y_i, \quad i = 1, 2, \ldots, n$$

are assumed to be i.i.d. observations from some distribution with mean μ and standard deviation σ. Chen's modified t-test can then be applied to the differences.

Value

a list of class `"htestEnvStats"` containing the results of the hypothesis test. See the help file for `htestEnvStats.object` for details.

Note

The presentation of Chen's (1995b) method in USEPA (2002d) and Singh et al. (2010b, p. 52) is incorrect for two reasons: it is based on an intermediate formula instead of the actual statistic that Chen proposes, and it uses the intermediate formula to compute an *upper* confidence limit for the mean when the sample data are positively skewed. As explained above, for the case of positively skewed data, Chen's method is appropriate to test the upper one-sided alternative hypothesis that the population mean is greater than some specified value, and a one-sided upper alternative corresponds to creating a one-sided *lower* confidence limit, not an upper confidence limit (see, for example, Millard and Neerchal, 2001, p. 371).

A frequent question in environmental statistics is "Is the concentration of chemical X greater than Y units?" For example, in groundwater assessment (compliance) monitoring at hazardous and solid

waste sites, the concentration of a chemical in the groundwater at a downgradient may be compared to a groundwater protection standard (GWPS). If the concentration is "above" the GWPS, then the site enters corrective action monitoring. As another example, soil screening at a Superfund site involves comparing the concentration of a chemical in the soil with a pre-determined soil screening level (SSL). If the concentration is "above" the SSL, then further investigation and possible remedial action is required. Determining what it means for the chemical concentration to be "above" a GWPS or an SSL is a policy decision: the average of the distribution of the chemical concentration must be above the GWPS or SSL, or the median must be above the GWPS or SSL, or the 95'th percentile must be above the GWPS or SSL, or something else. Often, the first interpretation is used.

The regulatory guidance document *Soil Screening Guidance: Technical Background Document* (USEPA, 1996c, Part 4) recommends using Chen's t-test as one possible method to compare chemical concentrations in soil samples to a soil screening level (SSL). The document notes that the distribution of chemical concentrations will almost always be positively-skewed, but not necessarily fit a lognormal distribution well (USEPA, 1996c, pp.107, 117-119). It also notes that using a confidence interval based on Land's (1971) method is extremely sensitive to the assumption of a lognormal distribution, while Chen's test is robust with respect to maintaining Type I and Type II errors for a variety of positively-skewed distributions (USEPA, 1996c, pp.99, 117-119, 123-125).

Hypothesis tests you can use to perform tests of location include: Student's t-test, Fisher's randomization test, the Wilcoxon signed rank test, Chen's modified t-test, the sign test, and a test based on a bootstrap confidence interval. For a discussion comparing the performance of these tests, see Millard and Neerchal (2001, pp.408–409).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chen, L. (1995b). Testing the Mean of Skewed Distributions. *Journal of the American Statistical Association* **90**(430), 767–772.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York, Chapters 28, 31.

Land, C.E. (1971). Confidence Intervals for Linear Functions of the Normal Mean and Variance. *The Annals of Mathematical Statistics* **42**(4), 1187–1205.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.402–404.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (1996c). *Soil Screening Guidance: Technical Background Document*. EPA/540/R-95/128, PB96963502. Office of Emergency and Remedial Response, U.S. Environmental Protection Agency, Washington, D.C., May, 1996.

USEPA. (2002d). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

t.test, elnorm, elnormAlt.

Examples

```
# The guidance document "Calculating Upper Confidence Limits for
# Exposure Point Concentrations at Hazardous Waste Sites"
# (USEPA, 2002d, Exhibit 9, p. 16) contains an example of 60 observations
# from an exposure unit.  Here we will use Chen's modified t-test to test
# the null hypothesis that the average concentration is less than 30 mg/L
# versus the alternative that it is greater than 30 mg/L.
# In EnvStats these data are stored in the vector EPA.02d.Ex.9.mg.per.L.vec.

sort(EPA.02d.Ex.9.mg.per.L.vec)
# [1]  16  17  17  17  18  18  20  20  20  21  21  21  21  21  21  22
#[17]  22  22  23  23  23  23  24  24  24  25  25  25  25  25  25  26
#[33]  26  26  26  27  27  28  28  28  28  29  29  30  30  31  32  32
#[49]  32  33  33  35  35  97  98 105 107 111 117 119

dev.new()
hist(EPA.02d.Ex.9.mg.per.L.vec, col = "cyan", xlab = "Concentration (mg/L)")

# The Shapiro-Wilk goodness-of-fit test rejects the null hypothesis of a
# normal, lognormal, and gamma distribution:

gofTest(EPA.02d.Ex.9.mg.per.L.vec)$p.value
#[1] 2.496781e-12

gofTest(EPA.02d.Ex.9.mg.per.L.vec, dist = "lnorm")$p.value
#[1] 3.349035e-09

gofTest(EPA.02d.Ex.9.mg.per.L.vec, dist = "gamma")$p.value
#[1] 1.564341e-10


# Use Chen's modified t-test to test the null hypothesis that
# the average concentration is less than 30 mg/L versus the
# alternative that it is greater than 30 mg/L.

chenTTest(EPA.02d.Ex.9.mg.per.L.vec, mu = 30)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 mean = 30
#
#Alternative Hypothesis:          True mean is greater than 30
#
#Test Name:                       One-sample t-Test
#                                 Modified for
#                                 Positively-Skewed Distributions
#                                 (Chen, 1995)
```

```
#
#Estimated Parameter(s):          mean = 34.566667
#                                 sd   = 27.330598
#                                 skew =  2.365778
#
#Data:                            EPA.02d.Ex.9.mg.per.L.vec
#
#Sample Size:                     60
#
#Test Statistic:                  t = 1.574075
#
#Test Statistic Parameter:        df = 59
#
#P-values:                        z               = 0.05773508
#                                 t               = 0.06040889
#                                 Avg. of z and t = 0.05907199
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Based on z
#
#Confidence Interval Type:        Lower
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 29.82
#                                 UCL =   Inf

# The estimated mean, standard deviation, and skew are 35, 27, and 2.4,
# respectively.  The p-value is 0.06, and the lower 95% confidence interval
# is [29.8, Inf).  Depending on what you use for your Type I error rate, you
# may or may not want to reject the null hypothesis.
```

Chi *The Chi Distribution*

Description

Density, distribution function, quantile function, and random generation for the chi distribution.

Usage

```
dchi(x, df)
pchi(q, df)
qchi(p, df)
rchi(n, df)
```

Arguments

x	vector of (positive) quantiles.
q	vector of (positive) quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
df	vector of (positive) degrees of freedom (> 0). Non-integer values are allowed.

Details

Elements of x, q, p, or df that are missing will cause the corresponding elements of the result to be missing.

The chi distribution with n degrees of freedom is the distribution of the positive square root of a random variable having a chi-squared distribution with n degrees of freedom.

The chi density function is given by:

$$f(x, \nu) = g(x^2, \nu)2x, x > 0$$

where $g(x, \nu)$ denotes the density function of a chi-square random variable with n degrees of freedom.

Value

density (dchi), probability (pchi), quantile (qchi), or random sample (rchi) for the chi distribution with df degrees of freedom.

Note

The chi distribution takes on positive real values. It is important because for a sample of n observations from a normal distribution, the sample standard deviation multiplied by the square root of the degrees of freedom ν and divided by the true standard deviation follows a chi distribution with ν degrees of freedom. The chi distribution is also used in computing exact prediction intervals for the next k observations from a normal distribution (see predIntNorm).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

Chisquare, Normal, predIntNorm, Probability Distributions and Random Numbers.

Examples

```
# Density of a chi distribution with 4 degrees of freedom, evaluated at 3:

dchi(3, 4)
#[1] 0.1499715

#----------

# The 95'th percentile of a chi distribution with 10 degrees of freedom:

qchi(.95, 10)
#[1] 4.278672

#----------

# The cumulative distribution function of a chi distribution with
# 5 degrees of freedom evaluated at 3:

pchi(3, 5)
#[1] 0.8909358

#----------

# A random sample of 2 numbers from a chi distribution with 7 degrees of freedom.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rchi(2, 7)
#[1] 3.271632 2.035179
```

---

ciBinomHalfWidth *Half-Width of Confidence Interval for Binomial Proportion or Difference Between Two Proportions*

Description

Compute the half-width of a confidence interval for a binomial proportion or the difference between two proportions, given the sample size(s), estimated proportion(s), and confidence level.

Usage

```
ciBinomHalfWidth(n.or.n1, p.hat.or.p1.hat = 0.5,
  n2 = n.or.n1, p2.hat = 0.4, conf.level = 0.95,
  sample.type = "one.sample", ci.method = "score",
  correct = TRUE, warn = TRUE)
```

Arguments

n.or.n1	numeric vector of sample sizes. When sample.type="one.sample", n.or.n1 denotes n, the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes n_1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
p.hat.or.p1.hat	numeric vector of estimated proportions. When sample.type="one.sample", p.hat.or.p1.hat denotes the estimated value of p, the probability of "success". When sample.type="two.sample", p.hat.or.p1.hat denotes the estimated value of p_1, the probability of "success" in group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
n2	numeric vector of sample sizes for group 2. The default value is the value of n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
p2.hat	numeric vector of estimated proportions for group 2. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
conf.level	numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is conf.level=0.95.
sample.type	character string indicating whether this is a one-sample or two-sample confidence interval. When sample.type="one.sample", the computed half-width is based on a confidence interval for a single proportion. When sample.type="two.sample", the computed half-width is based on a confidence interval for the difference between two proportions. The default value is sample.type="one.sample" unless the argument n2 or p2.hat is supplied.
ci.method	character string indicating which method to use to construct the confidence interval. Possible values are "score" (the default), "exact", "adjusted Wald", and "Wald" (the "Wald" method is **never** recommended but is included for historical purposes). The exact method is only available for the one-sample case, i.e., when sample.type="one.sample".
correct	logical scalar indicating whether to use the continuity correction when ci.method="score" or ci.method="Wald". The default value is correct=TRUE.
warn	logical scalar indicating whether to issue a warning when ci.method="Wald" for cases when the normal approximation to the binomial distribution probably is not accurate. The default value is warn=TRUE.

Details

If the arguments n.or.n1, p.hat.or.p1.hat, n2, p2.hat, and conf.level are not all the same length, they are replicated to be the same length as the length of the longest argument.

The values of p.hat.or.p1.hat and p2.hat are automatically adjusted to the closest legitimate values, given the user-supplied values of n.or.n1 and n2. For example, if n.or.n1=5, legitimate values for p.hat.or.p1.hat are 0, 0.2, 0.4, 0.6, 0.8 and 1. In this case, if the user supplies p.hat.or.p1.hat=0.45, then p.hat.or.p1.hat is reset to p.hat.or.p1.hat=0.4, and if the user supplies p.hat.or.p1.hat=0.55, then p.hat.or.p1.hat is reset to p.hat.or.p1.hat=0.6. In cases where the two closest legitimate values are equal distance from the user-suppled value of p.hat.or.p1.hat or p2.hat, the value closest to 0.5 is chosen since that will tend to yield the wider confidence interval.

One-Sample Case (sample.type="one.sample").

ci.method="score" The confidence interval for p based on the score method was developed by Wilson (1927) and is discussed by Newcombe (1998a), Agresti and Coull (1998), and Agresti and Caffo (2000). When ci=TRUE and ci.method="score", the function ebinom calls the R function prop.test to compute the confidence interval. This method has been shown to provide the best performance (in terms of actual coverage matching assumed coverage) of all the methods provided here, although unlike the exact method, the actual coverage can fall below the assumed coverage.

ci.method="exact" The confidence interval for p based on the exact (Clopper-Pearson) method is discussed by Newcombe (1998a), Agresti and Coull (1998), and Zar (2010, pp.543-547). This is the method used in the R function binom.test. This method ensures the actual coverage is greater than or equal to the assumed coverage.

ci.method="Wald" The confidence interval for p based on the Wald method (with or without a correction for continuity) is the usual "normal approximation" method and is discussed by Newcombe (1998a), Agresti and Coull (1998), Agresti and Caffo (2000), and Zar (2010, pp.543-547). This method is **never** recommended but is included for historical purposes.

ci.method="adjusted Wald" The confidence interval for p based on the adjusted Wald method is discussed by Agresti and Coull (1998), Agresti and Caffo (2000), and Zar (2010, pp.543-547). This is a simple modification of the Wald method and performs surpringly well.

Two-Sample Case (sample.type="two.sample").

ci.method="score" This method is presented in Newcombe (1998b) and is based on the score method developed by Wilson (1927) for the one-sample case. This is the method used by the R function prop.test. In a comparison of 11 methods, Newcombe (1998b) showed this method performs remarkably well.

ci.method="Wald" The confidence interval for the difference between two proportions based on the Wald method (with or without a correction for continuity) is the usual "normal approximation" method and is discussed by Newcombe (1998b), Agresti and Caffo (2000), and Zar (2010, pp.549-552). This method is **not** recommended but is included for historical purposes.

ci.method="adjusted Wald" This method is discussed by Agresti and Caffo (2000), and Zar (2010, pp.549-552). This is a simple modification of the Wald method and performs surpringly well.

Value

a list with information about the half-widths, sample sizes, and estimated proportions.

One-Sample Case (sample.type="one.sample").
When sample.type="one.sample", the function ciBinomHalfWidth returns a list with these components:

half.width	the half-width(s) of the confidence interval(s)
n	the sample size(s) associated with the confidence interval(s)
p.hat	the estimated proportion(s)
method	the method used to construct the confidence interval(s)

Two-Sample Case (sample.type="two.sample").
When sample.type="two.sample", the function ciBinomHalfWidth returns a list with these components:

half.width	the half-width(s) of the confidence interval(s)
n1	the sample size(s) for group 1 associated with the confidence interval(s)
p1.hat	the estimated proportion(s) for group 1
n2	the sample size(s) for group 2 associated with the confidence interval(s)
p2.hat	the estimated proportion(s) for group 2
method	the method used to construct the confidence interval(s)

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", p, is the same on each trial. A binomial discrete random variable X is the number of "successes" in n independent trials. A special case of the binomial distribution occurs when $n = 1$, in which case X is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7). (However, USEPA 2009, p.8-27 recommends using the Wilcoxon rank sum test (wilcox.test) instead of comparing proportions.)

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions ciBinomHalfWidth, ciBinomN, and plotCiBinomDesign can be used to investigate these relationships for the case of binomial proportions.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Agresti, A., and B.A. Coull. (1998). Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions. *The American Statistician*, **52**(2), 119–126.

Agresti, A., and B. Caffo. (2000). Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures. *The American Statistician*, **54**(4), 280–288.

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapters 2 and 15.

Cochran, W.G. (1977). *Sampling Techniques*. John Wiley and Sons, New York, Chapter 3.

Fisher, R.A., and F. Yates. (1963). *Statistical Tables for Biological, Agricultural, and Medical Research*. 6th edition. Hafner, New York, 146pp.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. Second Edition. John Wiley and Sons, New York, Chapters 1-2.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, Chapter 11.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Newcombe, R.G. (1998a). Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods. *Statistics in Medicine*, **17**, 857–872.

Newcombe, R.G. (1998b). Interval Estimation for the Difference Between Independent Proportions: Comparison of Eleven Methods. *Statistics in Medicine*, **17**, 873–890.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL, Chapter 4.

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.6-38.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 24.

See Also

ciBinomN, plotCiBinomDesign, ebinom, binom.test, prop.test.

Examples

```
# Look at how the half-width of a one-sample confidence interval
# decreases with sample size:

ciBinomHalfWidth(n.or.n1 = c(10, 50, 100, 500))
#$half.width
#[1] 0.26340691 0.13355486 0.09616847 0.04365873
#
#$n
#[1]  10  50 100 500
#
#$p.hat
#[1] 0.5 0.5 0.5 0.5
#
#$method
```

```
#[1] "Score normal approximation, with continuity correction"

#----------------------------------------------------------------

# Look at how the half-width of a one-sample confidence interval
# tends to decrease as the estimated value of p decreases below
# 0.5 or increases above 0.5:

seq(0.2, 0.8, by = 0.1)
#[1] 0.2 0.3 0.4 0.5 0.6 0.7 0.8

ciBinomHalfWidth(n.or.n1 = 30, p.hat = seq(0.2, 0.8, by = 0.1))
#$half.width
#[1] 0.1536299 0.1707256 0.1801322 0.1684587 0.1801322 0.1707256
#[7] 0.1536299
#
#$n
#[1] 30 30 30 30 30 30 30
#
#$p.hat
#[1] 0.2 0.3 0.4 0.5 0.6 0.7 0.8
#
#$method
#[1] "Score normal approximation, with continuity correction"

#----------------------------------------------------------------

# Look at how the half-width of a one-sample confidence interval
# increases with increasing confidence level:

ciBinomHalfWidth(n.or.n1 = 20, conf.level = c(0.8, 0.9, 0.95, 0.99))
#$half.width
#[1] 0.1377380 0.1725962 0.2007020 0.2495523
#
#$n
#[1] 20 20 20 20
#
#$p.hat
#[1] 0.5 0.5 0.5 0.5
#
#$method
#[1] "Score normal approximation, with continuity correction"

#----------------------------------------------------------------

# Compare the half-widths for a one-sample
# confidence interval based on the different methods:

ciBinomHalfWidth(n.or.n1 = 30, ci.method = "score")$half.width
#[1] 0.1684587

ciBinomHalfWidth(n.or.n1 = 30, ci.method = "exact")$half.width
#[1] 0.1870297
```

```
ciBinomHalfWidth(n.or.n1 = 30, ci.method = "adjusted Wald")$half.width
#[1] 0.1684587

ciBinomHalfWidth(n.or.n1 = 30, ci.method = "Wald")$half.width
#[1] 0.1955861

#----------------------------------------------------------------

# Look at how the half-width of a two-sample
# confidence interval decreases with increasing
# sample sizes:

ciBinomHalfWidth(n.or.n1 = c(10, 50, 100, 500), sample.type = "two")
#$half.width
#[1] 0.53385652 0.21402654 0.14719748 0.06335658
#
#$n1
#[1]  10  50 100 500
#
#$p1.hat
#[1] 0.5 0.5 0.5 0.5
#
#$n2
#[1]  10  50 100 500
#
#$p2.hat
#[1] 0.4 0.4 0.4 0.4
#
#$method
#[1] "Score normal approximation, with continuity correction"
```

`ciBinomN` *Sample Size for Specified Half-Width of Confidence Interval for Binomial Proportion or Difference Between Two Proportions*

Description

Compute the sample size necessary to achieve a specified half-width of a confidence interval for a binomial proportion or the difference between two proportions, given the estimated proportion(s), and confidence level.

Usage

```
ciBinomN(half.width, p.hat.or.p1.hat = 0.5, p2.hat = 0.4,
  conf.level = 0.95, sample.type = "one.sample", ratio = 1,
  ci.method = "score", correct = TRUE, warn = TRUE,
  n.or.n1.min = 2, n.or.n1.max = 10000,
  tol.half.width = 5e-04, tol.p.hat = 5e-04,
  tol = 1e-7, maxiter = 1000)
```

Arguments

half.width	numeric vector of (positive) half-widths. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
p.hat.or.p1.hat	numeric vector of estimated proportions. When sample.type="one.sample", p.hat.or.p1.hat denotes the estimated value of p, the probability of "success". When sample.type="two.sample", p.hat.or.p1.hat denotes the estimated value of p_1, the probability of "success" in group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
p2.hat	numeric vector of estimated proportions for group 2. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
conf.level	numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is conf.level=0.95.
sample.type	character string indicating whether this is a one-sample or two-sample confidence interval. When sample.type="one.sample", the computed half-width is based on a confidence interval for a single proportion. When sample.type="two.sample", the computed half-width is based on a confidence interval for the difference between two proportions. The default value is sample.type="one.sample" unless the argument p2.hat or ratio is supplied.
ratio	numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 (n_2/n_1). The default value is ratio=1. All values of ratio must be greater than or equal to 1. This argument is ignored if sample.type="one.sample".
ci.method	character string indicating which method to use to construct the confidence interval. Possible values are: • "score" (the default), • "exact", • "adjusted Wald" and, • "Wald" (the "Wald" method is **never** recommended but is included for historical purposes). The exact method is only available for the one-sample case, i.e., when sample.type="one.sample".
correct	logical scalar indicating whether to use the continuity correction when ci.method="score" or ci.method="Wald". The default value is correct=TRUE.
warn	logical scalar indicating whether to issue a warning when ci.method="Wald" for cases when the normal approximation to the binomial distribution probably is not accurate. The default value is warn=TRUE.
n.or.n1.min	integer indicating the minimum allowed value for n (sample.type="one.sample") or

n_1 (sample.type="two.sample").
The default value is n.or.n1.min=2.

n.or.n1.max integer indicating the maximum allowed value for n (sample.type="one.sample") or n_1 (sample.type="two.sample"). The default value is n.or.n1.max=10000.

tol.half.width numeric scalar indicating the tolerance to use for the half width for the search algorithm. The sample sizes are computed so that the actual half width is less than or equal to half.width + tol.half.width. The default value is tol.half.width=5e-04.

tol.p.hat numeric scalar indicating the tolerance to use for the estimated proportion(s) for the search algorithm. For the one-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of p.hat.or.p1.hat and the actual estimated proportion is less than or equal to tol.p.hat. For the two-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of p.hat.or.p1.hat and the actual estimated proportion for group 1 is less than or equal to tol.p.hat, and the absolute value of the difference between the user supplied value of p2.hat and the actual estimated proportion for group 2 is less than or equal to tol.p.hat. The default value is tol.p.hat=0.005.

tol positive scalar indicating the tolerance to use for the search algorithm (passed to uniroot). The default value is tol=1e-7.

maxiter integer indicating the maximum number of iterations to use for the search algorithm (passed to uniroot). The default value is maxiter=1000.

Details

If the arguments half.width, p.hat.or.p1.hat, p2.hat, conf.level and ratio are not all the same length, they are replicated to be the same length as the length of the longest argument.

For the one-sample case, the arguments p.hat.or.p1.hat, tol.p.hat, half.width, and tol.half.width must satisfy:
(p.hat.or.p1.hat + tol.p.hat + half.width + tol.half.width) <= 1,
and
(p.hat.or.p1.hat - tol.p.hat - half.width - tol.half.width) >= 0.

For the two-sample case, the arguments p.hat.or.p1.hat, p2.hat, tol.p.hat, half.width, and tol.half.width must satisfy:
((p.hat.or.p1.hat + tol.p.hat) - (p2.hat - tol.p.hat) + half.width + tol.half.width) <= 1, and
((p.hat.or.p1.hat - tol.p.hat) - (p2.hat + tol.p.hat) - half.width - tol.half.width) >= -1.

The function ciBinomN uses the search algorithm in the function uniroot to call the function ciBinomHalfWidth to find the values of n (sample.type="one.sample") or n_1 and n_2 (sample.type="two.sample") that satisfy the requirements for the half-width, estimated proportions, and confidence level. See the Details section of the help file for ciBinomHalfWidth for more information.

Value

a list with information about the sample sizes, estimated proportions, and half-widths.

One-Sample Case (`sample.type="one.sample"`).
When `sample.type="one.sample"`, the function `ciBinomN` returns a list with these components:

`n`	the sample size(s) associated with the confidence interval(s)
`p.hat`	the estimated proportion(s)
`half.width`	the half-width(s) of the confidence interval(s)
`method`	the method used to construct the confidence interval(s)

Two-Sample Case (`sample.type="two.sample"`).
When `sample.type="two.sample"`, the function `ciBinomN` returns a list with these components:

`n1`	the sample size(s) for group 1 associated with the confidence interval(s)
`n2`	the sample size(s) for group 2 associated with the confidence interval(s)
`p1.hat`	the estimated proportion(s) for group 1
`p2.hat`	the estimated proportion(s) for group 2
`half.width`	the half-width(s) of the confidence interval(s)
`method`	the method used to construct the confidence interval(s)

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", p, is the same on each trial. A binomial discrete random variable X is the number of "successes" in n independent trials. A special case of the binomial distribution occurs when $n = 1$, in which case X is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7). (However, USEPA 2009, p.8-27 recommends using the Wilcoxon rank sum test (`wilcox.test`) instead of comparing proportions.)

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions `ciBinomHalfWidth`, `ciBinomN`, and `plotCiBinomDesign` can be used to investigate these relationships for the case of binomial proportions.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Agresti, A., and B.A. Coull. (1998). Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions. *The American Statistician*, **52**(2), 119–126.

Agresti, A., and B. Caffo. (2000). Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures. *The American Statistician*, **54**(4), 280–288.

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapters 2 and 15.

Cochran, W.G. (1977). *Sampling Techniques*. John Wiley and Sons, New York, Chapter 3.

Fisher, R.A., and F. Yates. (1963). *Statistical Tables for Biological, Agricultural, and Medical Research*. 6th edition. Hafner, New York, 146pp.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. Second Edition. John Wiley and Sons, New York, Chapters 1-2.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, Chapter 11.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Newcombe, R.G. (1998a). Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods. *Statistics in Medicine*, **17**, 857–872.

Newcombe, R.G. (1998b). Interval Estimation for the Difference Between Independent Proportions: Comparison of Eleven Methods. *Statistics in Medicine*, **17**, 873–890.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL, Chapter 4.

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.6-38.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 24.

See Also

`ciBinomHalfWidth`, `uniroot`, `plotCiBinomDesign`, `ebinom`, `binom.test`, `prop.test`.

Examples

```
# Look at how the required sample size of a one-sample
# confidence interval increases with decreasing
# required half-width:

ciBinomN(half.width = c(0.1, 0.05, 0.03))
```

```
#$n
#[1]   92  374 1030
#
#$p.hat
#[1] 0.5 0.5 0.5
#
#$half.width
#[1] 0.10010168 0.05041541 0.03047833
#
#$method
#[1] "Score normal approximation, with continuity correction"

#----------

# Note that the required sample size decreases if we are less
# stringent about how much the confidence interval width can
# deviate from the supplied value of the 'half.width' argument:

ciBinomN(half.width = c(0.1, 0.05, 0.03), tol.half.width = 0.005)
#$n
#[1]  84 314 782
#
#$p.hat
#[1] 0.5 0.5 0.5
#
#$half.width
#[1] 0.10456066 0.05496837 0.03495833
#
#$method
#[1] "Score normal approximation, with continuity correction"

#-----------------------------------------------------------------------

# Look at how the required sample size for a one-sample
# confidence interval tends to decrease as the estimated
# value of p decreases below 0.5 or increases above 0.5:

seq(0.2, 0.8, by = 0.1)
#[1] 0.2 0.3 0.4 0.5 0.6 0.7 0.8

ciBinomN(half.width = 0.1, p.hat = seq(0.2, 0.8, by = 0.1))
#$n
#[1]  70  90 100  92 100  90  70
#
#$p.hat
#[1] 0.2 0.3 0.4 0.5 0.6 0.7 0.8
#
#$half.width
#[1] 0.09931015 0.09839843 0.09910818 0.10010168 0.09910818 0.09839843
#[7] 0.09931015
#
#$method
#[1] "Score normal approximation, with continuity correction"
```

```
#-------------------------------------------------------------------

# Look at how the required sample size for a one-sample
# confidence interval increases with increasing confidence level:

ciBinomN(half.width = 0.05, conf.level = c(0.8, 0.9, 0.95, 0.99))
#$n
#[1] 160 264 374 644
#
#$p.hat
#[1] 0.5 0.5 0.5 0.5
#
#$half.width
#[1] 0.05039976 0.05035948 0.05041541 0.05049152
#
#$method
#[1] "Score normal approximation, with continuity correction"

#-------------------------------------------------------------------

# Compare required sample size for a one-sample
# confidence interval based on the different methods:

ciBinomN(half.width = 0.05, ci.method = "score")
#$n
#[1] 374
#
#$p.hat
#[1] 0.5
#
#$half.width
#[1] 0.05041541
#
#$method
#[1] "Score normal approximation, with continuity correction"

ciBinomN(half.width = 0.05, ci.method = "exact")
#$n
#[1] 394
#
#$p.hat
#[1] 0.5
#
#$half.width
#[1] 0.05047916
#
#$method
#[1] "Exact"

ciBinomN(half.width = 0.05, ci.method = "adjusted Wald")
#$n
#[1] 374
```

```
#
#$p.hat
#[1] 0.5
#
#$half.width
#[1] 0.05041541
#
#$method
#[1] "Adjusted Wald normal approximation"

ciBinomN(half.width = 0.05, ci.method = "Wald")
#$n
#[1] 398
#
#$p.hat
#[1] 0.5
#
#$half.width
#[1] 0.05037834
#
#$method
#[1] "Wald normal approximation, with continuity correction"

#---------------------------------------------------------------

## Not run:
# Look at how the required sample size of a two-sample
# confidence interval increases with decreasing
# required half-width:

ciBinomN(half.width = c(0.1, 0.05, 0.03), sample.type = "two")
#$n1
#[1]  210  778 2089
#
#$n2
#[1]  210  778 2089
#
#$p1.hat
#[1] 0.5000000 0.5000000 0.4997607
#
#$p2.hat
#[1] 0.4000000 0.3997429 0.4001915
#
#$half.width
#[1] 0.09943716 0.05047044 0.03049753
#
#$method
#[1] "Score normal approximation, with continuity correction"

## End(Not run)
```

`ciNormHalfWidth` *Half-Width of Confidence Interval for Normal Distribution Mean or Difference Between Two Means*

Description

Compute the half-width of a confidence interval for the mean of a normal distribution or the difference between two means, given the sample size(s), estimated standard deviation, and confidence level.

Usage

```
ciNormHalfWidth(n.or.n1, n2 = n.or.n1,
  sigma.hat = 1, conf.level = 0.95,
  sample.type = ifelse(missing(n2), "one.sample", "two.sample"))
```

Arguments

`n.or.n1`	numeric vector of sample sizes. When `sample.type="one.sample"`, this argument denotes n, the number of observations in the single sample. When `sample.type="two.sample"`, this argument denotes n_1, the number of observations from group 1. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`n2`	numeric vector of sample sizes for group 2. The default value is the value of `n.or.n1`. This argument is ignored when `sample.type="one.sample"`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`sigma.hat`	numeric vector specifying the value(s) of the estimated standard deviation(s).
`conf.level`	numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is `conf.level=0.95`.
`sample.type`	character string indicating whether this is a one-sample (`sample.type="one.sample"`) or two-sample (`sample.type="two.sample"`) confidence interval. When `sample.type="one.sample"`, the computed half-width is based on a confidence interval for a single mean. When `sample.type="two.sample"`, the computed half-width is based on a confidence interval for the difference between two means. The default value is `sample.type="one.sample"` unless the argument `n2` is supplied.

Details

If the arguments `n.or.n1`, `n2`, `sigma.hat`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

One-Sample Case (`sample.type="one.sample"`)
Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with mean μ

and standard deviation σ. A two-sided $(1-\alpha)100\%$ confidence interval for μ is given by:

$$[\hat{\mu} - t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}},\ \hat{\mu} + t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}}] \qquad (1)$$

where

$$\hat{\mu} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and $t(\nu, p)$ is the p'th quantile of Student's t-distribution with ν degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992). Thus, the half-width of this confidence interval is given by:

$$HW = t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}} \qquad (4)$$

Two-Sample Case (`sample.type="two.sample"`)
Let $\underline{x}_1 = x_{11}, x_{12}, \ldots, x_{1n_1}$ denote a vector of n_1 observations from a normal distribution with mean μ_1 and standard deviation σ, and let $\underline{x}_2 = x_{21}, x_{22}, \ldots, x_{2n_2}$ denote a vector of n_2 observations from a normal distribution with mean μ_2 and standard deviation σ. A two-sided $(1-\alpha)100\%$ confidence interval for $\mu_1 - \mu_2$ is given by:

$$[(\hat{\mu}_1 - \hat{\mu}_2) - t(n_1+n_2-2, 1-\alpha/2)\hat{\sigma}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}},\ (\hat{\mu}_1 - \hat{\mu}_2) + t(n_1+n_2-2, 1-\alpha/2)\hat{\sigma}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}] \qquad (5)$$

where

$$\hat{\mu}_1 = \bar{x}_1 = \frac{1}{n_1}\sum_{i=1}^{n_1} x_{1i} \qquad (6)$$

$$\hat{\mu}_2 = \bar{x}_2 = \frac{1}{n_2}\sum_{i=1}^{n_2} x_{2i} \qquad (7)$$

$$\hat{\sigma}^2 = s_p^2 = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2} \qquad (8)$$

$$s_1^2 = \frac{1}{n_1-1}\sum_{i=1}^{n_1}(x_{1i} - \bar{x}_1)^2 \qquad (9)$$

$$s_2^2 = \frac{1}{n_2-1}\sum_{i=1}^{n_2}(x_{2i} - \bar{x}_2)^2 \qquad (10)$$

(Zar, 2010, p.142; Helsel and Hirsch, 1992, p.135, Berthouex and Brown, 2002, pp.157–158). Thus, the half-width of this confidence interval is given by:

$$HW = t(n_1+n_2-2, 1-\alpha/2)\hat{\sigma}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}} \qquad (11)$$

Note that for the two-sample case, the function `ciNormHalfWidth` assumes the two populations have the same standard deviation.

Value

a numeric vector of half-widths.

Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with confidence intervals.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions `ciNormHalfWidth`, `ciNormN`, and `plotCiNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Millard, S.P., and N. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-3.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapters 7 and 8.

See Also

`ciNormN`, `plotCiNormDesign`, `Normal`, `enorm`, `t.test`
Estimating Distribution Parameters.

Examples

```
# Look at how the half-width of a one-sample confidence interval
# decreases with increasing sample size:

seq(5, 30, by = 5)
#[1] 5 10 15 20 25 30

hw <- ciNormHalfWidth(n.or.n1 = seq(5, 30, by = 5))

round(hw, 2)
#[1] 1.24 0.72 0.55 0.47 0.41 0.37

#----------

# Look at how the half-width of a one-sample confidence interval
# increases with increasing estimated standard deviation:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

hw <- ciNormHalfWidth(n.or.n1 = 20, sigma.hat = seq(0.5, 2, by = 0.5))

round(hw, 2)
#[1] 0.23 0.47 0.70 0.94

#----------

# Look at how the half-width of a one-sample confidence interval
# increases with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

hw <- ciNormHalfWidth(n.or.n1 = 20, conf.level = seq(0.5, 0.9, by = 0.1))

round(hw, 2)
#[1] 0.15 0.19 0.24 0.30 0.39

#==========

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
# determine how adding another four months of observations to
# increase the sample size from 4 to 8 will affect the half-width
# of a two-sided 95% confidence interval for the Aldicarb level at
# the first compliance well.
#
# Use the estimated standard deviation from the first four months
# of data.  (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
# Note that the half-width changes from 34% of the observed mean to
# 18% of the observed mean by increasing the sample size from
# 4 to 8.
```

```
EPA.09.Ex.21.1.aldicarb.df
#   Month   Well Aldicarb.ppb
#1      1 Well.1         19.9
#2      2 Well.1         29.6
#3      3 Well.1         18.7
#4      4 Well.1         24.2
#...

mu.hat <- with(EPA.09.Ex.21.1.aldicarb.df,
  mean(Aldicarb.ppb[Well=="Well.1"]))

mu.hat
#[1] 23.1

sigma.hat <- with(EPA.09.Ex.21.1.aldicarb.df,
  sd(Aldicarb.ppb[Well=="Well.1"]))

sigma.hat
#[1] 4.93491

hw.4 <- ciNormHalfWidth(n.or.n1 = 4, sigma.hat = sigma.hat)

hw.4
#[1] 7.852543

hw.8 <- ciNormHalfWidth(n.or.n1 = 8, sigma.hat = sigma.hat)

hw.8
#[1] 4.125688

100 * hw.4/mu.hat
#[1] 33.99369

100 * hw.8/mu.hat
#[1] 17.86012

#==========

# Clean up
#---------
rm(hw, mu.hat, sigma.hat, hw.4, hw.8)
```

---

ciNormN *Sample Size for Specified Half-Width of Confidence Interval for Normal Distribution Mean or Difference Between Two Means*

Description

Compute the sample size necessary to achieve a specified half-width of a confidence interval for the mean of a normal distribution or the difference between two means, given the estimated standard deviation and confidence level.

Usage

```
ciNormN(half.width, sigma.hat = 1, conf.level = 0.95,
  sample.type = ifelse(is.null(n2), "one.sample", "two.sample"),
  n2 = NULL, round.up = TRUE, n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

half.width	numeric vector of (positive) half-widths. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
sigma.hat	numeric vector specifying the value(s) of the estimated standard deviation(s).
conf.level	numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is conf.level=0.95.
sample.type	character string indicating whether this is a one-sample (sample.type="one.sample") or two-sample (sample.type="two.sample") confidence interval. When sample.type="one.sample", the computed sample size is based on a confidence interval for a single mean. When sample.type="two.sample", the computed sample size is based on a confidence interval for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.
n2	numeric vector of sample sizes for group 2. The default value is NULL, in which case it is assumed that the sample sizes for groups 1 and 2 are equal. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
round.up	logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=TRUE.
n.max	positive integer greater than 1 specifying the maximum sample size for the single group when sample.type="one.sample" or for group 1 when sample.type="two.sample". The default value is n.max=5000.
tol	numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.
maxiter	positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.

Details

If the arguments half.width, n2, sigma.hat, and conf.level are not all the same length, they are replicated to be the same length as the length of the longest argument.

The function ciNormN uses the formulas given in the help file for ciNormHalfWidth for the half-width of the confidence interval to iteratively solve for the sample size. For the two-sample case, the default is to assume equal sample sizes for each group unless the argument n2 is supplied.

Value

When `sample.type="one.sample"`, or `sample.type="two.sample"` and `n2` is not supplied (so equal sample sizes for each group is assumed), the function `ciNormN` returns a numeric vector of sample sizes. When `sample.type="two.sample"` and `n2` is supplied, the function `ciNormN` returns a list with two components called `n1` and `n2`, specifying the sample sizes for each group.

Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with confidence intervals.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions `ciNormHalfWidth`, `ciNormN`, and `plotCiNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Millard, S.P., and N. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-3.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapters 7 and 8.

See Also

`ciNormHalfWidth`, `plotCiNormDesign`, `Normal`, `enorm`, `t.test`,
Estimating Distribution Parameters.

Examples

```
# Look at how the required sample size for a one-sample
# confidence interval decreases with increasing half-width:

seq(0.25, 1, by = 0.25)
#[1] 0.25 0.50 0.75 1.00

ciNormN(half.width = seq(0.25, 1, by = 0.25))
#[1] 64 18 10 7

ciNormN(seq(0.25, 1, by=0.25), round = FALSE)
#[1] 63.897899 17.832337  9.325967  6.352717

#--------------------------------------------------------------------

# Look at how the required sample size for a one-sample
# confidence interval increases with increasing estimated
# standard deviation for a fixed half-width:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

ciNormN(half.width = 0.5, sigma.hat = seq(0.5, 2, by = 0.5))
#[1] 7 18 38 64

#--------------------------------------------------------------------

# Look at how the required sample size for a one-sample
# confidence interval increases with increasing confidence
# level for a fixed half-width:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

ciNormN(half.width = 0.25, conf.level = seq(0.5, 0.9, by = 0.1))
#[1] 9 13 19 28 46

#--------------------------------------------------------------------

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
# determine the required sample size in order to achieve a
# half-width that is 10% of the observed mean (based on the first
# four months of observations) for the Aldicarb level at the first
# compliance well.  Assume a 95% confidence level and use the
# estimated standard deviation from the first four months of data.
# (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
#
# The required sample size is 20, so almost two years of data are
# required assuming observations are taken once per month.

EPA.09.Ex.21.1.aldicarb.df
#   Month   Well Aldicarb.ppb
```

```
#1      1 Well.1         19.9
#2      2 Well.1         29.6
#3      3 Well.1         18.7
#4      4 Well.1         24.2
#...

mu.hat <- with(EPA.09.Ex.21.1.aldicarb.df,
  mean(Aldicarb.ppb[Well=="Well.1"]))

mu.hat
#[1] 23.1

sigma.hat <- with(EPA.09.Ex.21.1.aldicarb.df,
  sd(Aldicarb.ppb[Well=="Well.1"]))

sigma.hat
#[1] 4.93491

ciNormN(half.width = 0.1 * mu.hat, sigma.hat = sigma.hat)
#[1] 20

#----------
# Clean up
rm(mu.hat, sigma.hat)
```

`ciNparConfLevel` *Compute Confidence Level Associated with a Nonparametric Confidence Interval for a Quantile*

Description

Compute the confidence level associated with a nonparametric confidence interval for a quantile, given the sample size and order statistics associated with the lower and upper bounds.

Usage

```
ciNparConfLevel(n, p = 0.5, lcl.rank = ifelse(ci.type == "upper", 0, 1),
  n.plus.one.minus.ucl.rank = ifelse(ci.type == "lower", 0, 1),
  ci.type = "two.sided")
```

Arguments

`n`	numeric vector of sample sizes. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`p`	numeric vector of probabilities specifying which quantiles to consider for the sample size calculation. All values of `p` must be between 0 and 1. The default value is `p=0.5`.

`lcl.rank`, `n.plus.one.minus.ucl.rank`
: numeric vectors of non-negative integers indicating the ranks of the order statistics that are used for the lower and upper bounds of the confidence interval for the specified quantile(s). When `lcl.rank=1` that means use the smallest value as the lower bound, when `lcl.rank=2` that means use the second to smallest value as the lower bound, etc. When `n.plus.one.minus.ucl.rank=1` that means use the largest value as the upper bound, when `n.plus.one.minus.ucl.rank=2` that means use the second to largest value as the upper bound, etc. A value of `0` for `lcl.rank` indicates no lower bound (i.e., -Inf) and a value of `0` for `n.plus.one.minus.ucl.rank` indicates no upper bound (i.e., `Inf`). When `ci.type="upper"` then `lcl.rank` is set to `0` by default, otherwise it is set to `1` by default. When `ci.type="lower"` then `n.plus.one.minus.ucl.rank` is set to `0` by default, otherwise it is set to `1` by default.

`ci.type`
: character string indicating what kind of confidence interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

Details

If the arguments `n`, `p`, `lcl.rank`, and `n.plus.one.minus.ucl.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for `eqnpar` explains how nonparametric confidence intervals for quantiles are constructed and how the confidence level associated with the confidence interval is computed based on specified values for the sample size and the ranks of the order statistics used for the bounds of the confidence interval.

Value

A numeric vector of confidence levels.

Note

See the help file for `eqnpar`.

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

See the help file for `eqnpar`.

See Also

`eqnpar`, `ciNparN`, `plotCiNparDesign`.

Examples

```
# Look at how the confidence level of a nonparametric confidence interval
# increases with increasing sample size for a fixed quantile:

seq(5, 25, by = 5)
#[1] 5 10 15 20 25

round(ciNparConfLevel(n = seq(5, 25, by = 5), p = 0.9), 2)
#[1] 0.41 0.65 0.79 0.88 0.93

#---------

# Look at how the confidence level of a nonparametric confidence interval
# decreases as the quantile moves away from 0.5:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

round(ciNparConfLevel(n = 10, p = seq(0.5, 0.9, by = 0.1)), 2)
#[1] 1.00 0.99 0.97 0.89 0.65

#==========

# Reproduce Example 21-6 on pages 21-21 to 21-22 of USEPA (2009).
# Use 12 measurements of nitrate (mg/L) at a well used for drinking water
# to determine with 95% confidence whether or not the infant-based, acute
# risk standard of 10 mg/L has been violated.  Assume that the risk
# standard represents an upper 95'th percentile limit on nitrate
# concentrations.  So what we need to do is construct a one-sided
# lower nonparametric confidence interval for the 95'th percentile
# that has associated confidence level of no more than 95%, and we will
# compare the lower confidence limit with the MCL of 10 mg/L.
#
# The data for this example are stored in EPA.09.Ex.21.6.nitrate.df.

# Look at the data:
#------------------

EPA.09.Ex.21.6.nitrate.df
#   Sampling.Date       Date Nitrate.mg.per.l.orig Nitrate.mg.per.l Censored
#1      7/28/1999 1999-07-28                  <5.0              5.0     TRUE
#2       9/3/1999 1999-09-03                  12.3             12.3    FALSE
#3     11/24/1999 1999-11-24                  <5.0              5.0     TRUE
#4       5/3/2000 2000-05-03                  <5.0              5.0     TRUE
#5      7/14/2000 2000-07-14                   8.1              8.1    FALSE
#6     10/31/2000 2000-10-31                  <5.0              5.0     TRUE
#7     12/14/2000 2000-12-14                    11             11.0    FALSE
#8      3/27/2001 2001-03-27                  35.1             35.1    FALSE
#9      6/13/2001 2001-06-13                  <5.0              5.0     TRUE
#10     9/16/2001 2001-09-16                  <5.0              5.0     TRUE
#11    11/26/2001 2001-11-26                   9.3              9.3    FALSE
#12      3/2/2002 2002-03-02                  10.3             10.3    FALSE
```

```
# Determine what order statistic to use for the lower confidence limit
# in order to achieve no more than 95% confidence.
#-------------------------------------------------------------------

conf.levels <- ciNparConfLevel(n = 12, p = 0.95, lcl.rank = 1:12,
  ci.type = "lower")
names(conf.levels) <- 1:12

round(conf.levels, 2)
#   1    2    3    4    5    6    7    8    9   10   11   12
#1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.98 0.88 0.54

# Using the 11'th largest observation for the lower confidence limit
# yields a confidence level of 88%.  Using the 10'th largest
# observation yields a confidence level of 98%.  The example in
# USEPA (2009) uses the 10'th largest observation.
#
# The 10'th largest observation is 11 mg/L which exceeds the
# MCL of 10 mg/L, so there is evidence of contamination.
#-------------------------------------------------------------------

with(EPA.09.Ex.21.6.nitrate.df,
  eqnpar(Nitrate.mg.per.l, p = 0.95, ci = TRUE,
    ci.type = "lower", lcl.rank = 10))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           95'th %ile = 22.56
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            Nitrate.mg.per.l
#
#Sample Size:                     12
#
#Confidence Interval for:         95'th %ile
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        lower
#
#Confidence Level:                98.04317%
#
#Confidence Limit Rank(s):        10
#
#Confidence Interval:             LCL =  11
#                                 UCL = Inf

#==========
```

```
# Clean up
#---------
rm(conf.levels)
```

---

ciNparN *Sample Size for Nonparametric Confidence Interval for a Quantile*

Description

Compute the sample size necessary to achieve a specified confidence level for a nonparametric confidence interval for a quantile.

Usage

```
ciNparN(p = 0.5, lcl.rank = ifelse(ci.type == "upper", 0, 1),
  n.plus.one.minus.ucl.rank = ifelse(ci.type == "lower", 0, 1),
  ci.type = "two.sided", conf.level = 0.95)
```

Arguments

p	numeric vector of probabilities specifying the quantiles. All values of p must be between 0 and 1. The default value is p=0.5.
lcl.rank, n.plus.one.minus.ucl.rank	numeric vectors of non-negative integers indicating the ranks of the order statistics that are used for the lower and upper bounds of the confidence interval for the specified quantile(s). When lcl.rank=1 that means use the smallest value as the lower bound, when lcl.rank=2 that means use the second to smallest value as the lower bound, etc. When n.plus.one.minus.ucl.rank=1 that means use the largest value as the upper bound, when n.plus.one.minus.ucl.rank=2 that means use the second to largest value as the upper bound, etc. A value of 0 for lcl.rank indicates no lower bound (i.e., -Inf) and a value of 0 for n.plus.one.minus.ucl.rank indicates no upper bound (i.e., Inf). When ci.type="upper" then lcl.rank is set to 0 by default, otherwise it is set to 1 by default. When ci.type="lower" then n.plus.one.minus.ucl.rank is set to 0 by default, otherwise it is set to 1 by default.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level	numeric vector of numbers between 0 and 1 indicating the confidence level associated with the confidence interval(s). The default value is conf=0.95.

Details

If the arguments p, lcl.rank, n.plus.one.minus.ucl.rank and conf.level are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for eqnpar explains how nonparametric confidence intervals for quantiles are constructed and how the confidence level associated with the confidence interval is computed based on

specified values for the sample size and the ranks of the order statistics used for the bounds of the confidence interval.

The function ciNparN determines the required the sample size via a nonlinear optimization.

Value

numeric vector of sample sizes.

Note

See the help file for eqnpar.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for eqnpar.

See Also

eqnpar, ciNparConfLevel, plotCiNparDesign.

Examples

```
# Look at how the required sample size for a confidence interval
# increases with increasing confidence level for a fixed quantile:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

ciNparN(p = 0.9, conf.level=seq(0.5, 0.9, by = 0.1))
#[1]  7  9 12 16 22

#----------

# Look at how the required sample size for a confidence interval increases
# as the quantile moves away from 0.5:

ciNparN(p = seq(0.5, 0.9, by = 0.1))
#[1]  6  7  9 14 29
```

ciTableMean *Table of Confidence Intervals for Mean or Difference Between Two Means*

Description

Create a table of confidence intervals for the mean of a normal distribution or the difference between two means following Bacchetti (2010), by varying the estimated standard deviation and the estimated mean or differene between the two estimated means given the sample size(s).

Usage

```
ciTableMean(n1 = 10, n2 = n1, diff.or.mean = 2:0, SD = 1:3,
  sample.type = "two.sample", ci.type = "two.sided", conf.level = 0.95,
  digits = 1)
```

Arguments

n1	positive integer greater than 1 specifying the sample size when sample.type="one.sample" or the sample size for group 1 when sample.type="two.sample". The default value is n1=10.
n2	positive integer greater than 1 specifying the sample size for group 2 when sample.type="two.sample". The default value is n2=n1, i.e., equal sample sizes. This argument is ignored when sample.type="one.sample".
diff.or.mean	numeric vector indicating either the assumed difference between the two sample means when sample.type="two.sample" or the value of the sample mean when sample.type="one.sample". The default value is diff.or.mean=2:0. Missing (NA), undefined (NaN), an infinite (-Inf, Inf) values are not allowed.
SD	numeric vector of positive values specifying the assumed estimated standard deviation. The default value is SD=1:3. Missing (NA), undefined (NaN), an infinite (-Inf, Inf) values are not allowed.
sample.type	character string specifying whether to create confidence intervals for the difference between two means (sample.type="two.sample"; the default) or confidence intervals for a single mean (sample.type="one.sample").
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95.
digits	positive integer indicating how many decimal places to display in the table. The default value is digits=1.

Details

Following Bacchetti (2010) (see NOTE below), the function `ciTableMean` allows you to perform sensitivity analyses while planning future studies by producing a table of confidence intervals for the mean or the difference between two means by varying the estimated standard deviation and the estimated mean or differene between the two estimated means given the sample size(s).

One Sample Case (`sample.type="one.sample"`)
Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from an normal (Gaussian) distribution with parameters `mean`=μ and `sd`=σ.

The usual confidence interval for μ is constructed as follows. If `ci.type="two-sided"`, the $(1-\alpha)100\%$ confidence interval for μ is given by:

$$[\hat{\mu} - t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}},\ \hat{\mu} + t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}}] \qquad (1)$$

where

$$\hat{\mu} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and $t(\nu, p)$ is the p'th quantile of Student's t-distribution with ν degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992).

If `ci.type="lower"`, the $(1-\alpha)100\%$ confidence interval for μ is given by:

$$[\hat{\mu} - t(n-1, 1-\alpha)\frac{\hat{\sigma}}{\sqrt{n}},\ \infty] \qquad (4)$$

and if `ci.type="upper"`, the confidence interval is given by:

$$[-\infty,\ \hat{\mu} + t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}}] \qquad (5)$$

For the one-sample case, the argument `n1` corresponds to n in Equation (1), the argument `diff.or.mean` corresponds to $\hat{\mu} = \bar{x}$ in Equation (2), and the argument `SD` corresponds to $\hat{\sigma} = s$ in Equation (3).

Two Sample Case (`sample.type="two.sample"`)
Let $\underline{x}_1 = (x_{11}, x_{21}, \ldots, x_{n_1 1})$ be a vector of n_1 observations from an normal (Gaussian) distribution with parameters `mean`=μ_1 and `sd`=σ, and let $\underline{x}_2 = (x_{12}, x_{22}, \ldots, x_{n_2 2})$ be a vector of n_2 observations from an normal (Gaussian) distribution with parameters `mean`=μ_2 and `sd`=σ.

The usual confidence interval for the difference between the two population means $\mu_1 - \mu_2$ is constructed as follows. If `ci.type="two-sided"`, the $(1-\alpha)100\%$ confidence interval for $\mu_1 - \mu_2$ is given by:

$$[(\hat{\mu}_1 - \hat{\mu}_2) - t(n_1+n_2-2, 1-\alpha/2)\hat{\sigma}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}},\ (\hat{\mu}_1 - \hat{\mu}_2) + t(n_1+n_2-2, 1-\alpha/2)\hat{\sigma}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}] \qquad (6)$$

where

$$\hat{\mu}_1 = \bar{x}_1 = \frac{1}{n_1}\sum_{i=1}^{n_1} x_{i1} \qquad (7)$$

$$\hat{\mu}_2 = \bar{x}_2 = \frac{1}{n_2}\sum_{i=1}^{n_2} x_{i2} \qquad (8)$$

$$\hat{\sigma}^2 = s_p^2 = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2} \qquad (9)$$

$$s_1^2 = \frac{1}{n_1-1}\sum_{i=1}^{n_1}(x_{i1}-\bar{x}_1)^2 \qquad (10)$$

$$s_2^2 = \frac{1}{n_2-1}\sum_{i=1}^{n_2}(x_{i2}-\bar{x}_2)^2 \qquad (11)$$

and $t(\nu, p)$ is the p'th quantile of Student's t-distribution with ν degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992).

If `ci.type="lower"`, the $(1-\alpha)100\%$ confidence interval for $\mu_1 - \mu_2$ is given by:

$$[(\hat{\mu}_1 - \hat{\mu}_2) - t(n_1+n_2-2, 1-\alpha)\hat{\sigma}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}},\ \infty] \qquad (12)$$

and if `ci.type="upper"`, the confidence interval is given by:

$$[-\infty,\ (\hat{\mu}_1 - \hat{\mu}_2) - t(n_1+n_2-2, 1-\alpha)\hat{\sigma}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}] \qquad (13)$$

For the two-sample case, the arguments `n1` and `n2` correspond to n_1 and n_2 in Equation (6), the argument `diff.or.mean` corresponds to $\hat{\mu_1} - \hat{\mu_2} = \bar{x}_1 - \bar{x}_2$ in Equations (7) and (8), and the argument `SD` corresponds to $\hat{\sigma} = s_p$ in Equation (9).

Value

a data frame with the rows varying the standard deviation and the columns varying the estimated mean or difference between the means. Elements of the data frame are character strings indicating the confidence intervals.

Note

Bacchetti (2010) presents strong arguments against the current convention in scientific research for computing sample size that is based on formulas that use a fixed Type I error (usually 5%) and a fixed minimal power (often 80%) without regard to costs. He notes that a key input to these formulas is a measure of variability (usually a standard deviation) that is difficult to measure accurately "unless there is so much preliminary data that the study isn't really needed." Also, study designers often avoid defining what a scientifically meaningful difference is by presenting sample size results in terms of the effect size (i.e., the difference of interest divided by the elusive standard deviation). Bacchetti (2010) encourages study designers to use simple tables in a sensitivity analysis to see what results of a study may look like for low, moderate, and high rates of variability and large, intermediate, and no underlying differences in the populations or processes being studied.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bacchetti, P. (2010). Current sample size conventions: Flaws, Harms, and Alternatives. *BMC Medicine* **8**, 17–23.

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

enorm, t.test, ciTableProp, ciNormHalfWidth, ciNormN, plotCiNormDesign.

Examples

```
# Show how potential confidence intervals for the difference between two means
# will look assuming standard deviations of 1, 2, or 3, differences between
# the two means of 2, 1, or 0, and a sample size of 10 in each group.

ciTableMean()
#          Diff=2      Diff=1      Diff=0
#SD=1 [ 1.1, 2.9] [ 0.1, 1.9] [-0.9, 0.9]
#SD=2 [ 0.1, 3.9] [-0.9, 2.9] [-1.9, 1.9]
#SD=3 [-0.8, 4.8] [-1.8, 3.8] [-2.8, 2.8]

#==========

# Show how a potential confidence interval for a mean will look assuming
# standard deviations of 1, 2, or 5, a sample mean of 5, 3, or 1, and
# a sample size of 15.

ciTableMean(n1 = 15, diff.or.mean = c(5, 3, 1), SD = c(1, 2, 5), sample.type = "one")
#          Mean=5      Mean=3      Mean=1
#SD=1 [ 4.4, 5.6] [ 2.4, 3.6] [ 0.4, 1.6]
#SD=2 [ 3.9, 6.1] [ 1.9, 4.1] [-0.1, 2.1]
#SD=5 [ 2.2, 7.8] [ 0.2, 5.8] [-1.8, 3.8]

#==========

# The data frame EPA.09.Ex.16.1.sulfate.df contains sulfate concentrations
```

```
# (ppm) at one background and one downgradient well. The estimated
# mean and standard deviation for the background well are 536 and 27 ppm,
# respectively, based on a sample size of n = 8 quarterly samples taken over
# 2 years.  A two-sided 95% confidence interval for this mean is [514, 559],
# which has a  half-width of 23 ppm.
#
# The estimated mean and standard deviation for the downgradient well are
# 608 and 18 ppm, respectively, based on a sample size of n = 6 quarterly
# samples.  A two-sided 95% confidence interval for the difference between
# this mean and the background mean is [44, 100] ppm.
#
# Suppose we want to design a future sampling program and are interested in
# the size of the confidence interval for the difference between the two means.
# We will use ciTableMean to generate a table of possible confidence intervals
# by varying the assumed standard deviation and assumed differences between
# the means.


# Look at the data
#-----------------

EPA.09.Ex.16.1.sulfate.df
#   Month Year    Well.type Sulfate.ppm
#1    Jan 1995   Background         560
#2    Apr 1995   Background         530
#3    Jul 1995   Background         570
#4    Oct 1995   Background         490
#5    Jan 1996   Background         510
#6    Apr 1996   Background         550
#7    Jul 1996   Background         550
#8    Oct 1996   Background         530
#9    Jan 1995 Downgradient          NA
#10   Apr 1995 Downgradient          NA
#11   Jul 1995 Downgradient         600
#12   Oct 1995 Downgradient         590
#13   Jan 1996 Downgradient         590
#14   Apr 1996 Downgradient         630
#15   Jul 1996 Downgradient         610
#16   Oct 1996 Downgradient         630


# Compute the estimated mean and standard deviation for the
# background well.
#-------------------------------------------------------------

Sulfate.back <- with(EPA.09.Ex.16.1.sulfate.df,
  Sulfate.ppm[Well.type == "Background"])

enorm(Sulfate.back, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
```

```
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 536.2500
#                                 sd   =  26.6927
#
#Estimation Method:               mvue
#
#Data:                            Sulfate.back
#
#Sample Size:                     8
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 513.9343
#                                 UCL = 558.5657


# Compute the estimated mean and standard deviation for the
# downgradient well.
#-----------------------------------------------------------

Sulfate.down <- with(EPA.09.Ex.16.1.sulfate.df,
  Sulfate.ppm[Well.type == "Downgradient"])

enorm(Sulfate.down, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 608.33333
#                                 sd   =  18.34848
#
#Estimation Method:               mvue
#
#Data:                            Sulfate.down
#
#Sample Size:                     6
#
#Number NA/NaN/Inf's:             2
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
```

```
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 589.0778
#                                 UCL = 627.5889


# Compute the estimated difference between the means and the confidence
# interval for the difference:
#-------------------------------------------------------------------

print(t.test(Sulfate.down, Sulfate.back, var.equal = TRUE))

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 difference in means = 0
#
#Alternative Hypothesis:          True difference in means is not equal to 0
#
#Test Name:                        Two Sample t-test
#
#Estimated Parameter(s):          mean of x = 608.3333
#                                 mean of y = 536.2500
#
#Data:                            Sulfate.down and Sulfate.back
#
#Test Statistic:                  t = 5.660985
#
#Test Statistic Parameter:        df = 12
#
#P-value:                         0.0001054306
#
#95% Confidence Interval:         LCL = 44.33974
#                                 UCL = 99.82693


# Use ciTableMean to look how the confidence interval for the difference
# between the background and downgradient means in a future study using eight
# quarterly samples at each well varies with assumed value of the pooled standard
# deviation and the observed difference between the sample means.
#-----------------------------------------------------------------------------

# Our current estimate of the pooled standard deviation is 24 ppm:

summary(lm(Sulfate.ppm ~ Well.type, data = EPA.09.Ex.16.1.sulfate.df))$sigma
#[1] 23.57759

# We can see that if this is overly optimistic and in our next study the
# pooled standard deviation is around 50 ppm, then if the observed difference
# between the means is 50 ppm, the lower end of the confidence interval for
# the difference between the two means will include 0, so we may want to
# increase our sample size.
```

```
ciTableMean(n1 = 8, n2 = 8, diff = c(100, 50, 0), SD = c(15, 25, 50), digits = 0)

#        Diff=100    Diff=50     Diff=0
#SD=15 [ 84, 116] [ 34,  66] [-16,  16]
#SD=25 [ 73, 127] [ 23,  77] [-27,  27]
#SD=50 [ 46, 154] [ -4, 104] [-54,  54]

#==========

# Clean up
#---------
rm(Sulfate.back, Sulfate.down)
```

`ciTableProp` *Table of Confidence Intervals for Proportion or Difference Between Two Proportions*

Description

Create a table of confidence intervals for probability of "success" for a binomial distribution or the difference between two proportions following Bacchetti (2010), by varying the estimated proportion or differene between the two estimated proportions given the sample size(s).

Usage

```
ciTableProp(n1 = 10, p1.hat = c(0.1, 0.2, 0.3), n2 = n1,
  p2.hat.minus.p1.hat = c(0.2, 0.1, 0), sample.type = "two.sample",
  ci.type = "two.sided", conf.level = 0.95, digits = 2, ci.method = "score",
  correct = TRUE, tol = 10^-(digits + 1))
```

Arguments

`n1`	positive integer greater than 1 specifying the sample size when `sample.type="one.sample"` or the sample size for group 1 when `sample.type="two.sample"`. The default value is `n1=10`.
`p1.hat`	numeric vector of values between 0 and 1 indicating the estimated proportion (`sample.type="one.sample"`) or the estimated proportion for group 1 (`sample.type="two.sample"`). The default value is `c(0.1, 0.2, 0.3)`. Missing (`NA`), undefined (`NaN`), an infinite (`-Inf`, `Inf`) values are not allowed.
`n2`	positive integer greater than 1 specifying the sample size for group 2 when `sample.type="two.sample"`. The default value is `n2=n1`, i.e., equal sample sizes. This argument is ignored when `sample.type="one.sample"`.
`p2.hat.minus.p1.hat`	numeric vector indicating the assumed difference between the two sample proportions when `sample.type="two.sample"`. The default value is `c(0.2, 0.1, 0)`. Missing (`NA`), undefined (`NaN`), an infinite (`-Inf`, `Inf`) values are not allowed. This argument is ignored when `sample.type="one.sample"`.

sample.type	character string specifying whether to create confidence intervals for the difference between two proportions (sample.type="two.sample"; the default) or confidence intervals for a single proportion (sample.type="one.sample").
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95.
digits	positive integer indicating how many decimal places to display in the table. The default value is digits=2.
ci.method	character string indicating the method to use to construct the confidence interval. The default value is ci.method="score" (i.e., the score method; see the help file for prop.test), which is the only method available when sample.type="two.sample". When sample.type="one.sample", you may also set ci.method="exact" (i.e., the exact method).
correct	logical scalar indicating whether to use the correction for continuity when ci.method="score" (see the help file for prop.test). The default value is correct=TRUE.
tol	numeric scalar indicating how close the values of the adjusted elements of p2.hat.minus.p1.hat have to be in order to provide a simply display of confidence intervals (see DETAILS section below). The default value is tol=10^-(digits + 1).

Details

One-Sample Case (sample.type="one.sample")
For the one-sample case, the function ciTableProp calls the R function prop.test when ci.method="score", and calls the R function binom.test, when ci.method="exact". To ensure that the user-supplied values of p1.hat are valid for the given user-supplied values of n1, values for the argument x to the function prop.test or binom.test are computed using the formula

```
x <- unique(round((p1.hat * n1), 0))
```

and the argument p.hat is then adjusted using the formula

```
p.hat <- x/n1
```

Two-Sample Case (sample.type="two.sample")
For the two-sample case, the function ciTableProp calls the R function prop.test. To ensure that the user-supplied values of p1.hat are valid for the given user-supplied values of n1, the values for the first component of the argument x to the function prop.test are computed using the formula

```
x1 <- unique(round((p1.hat * n1), 0))
```

and the argument p1.hat is then adjusted using the formula

```
p1.hat <- x1/n1
```

Next, the estimated proportions from group 2 are computed by adding together all possible combinations from the elements of p1.hat and p2.hat.minus.p1.hat. These estimated proportions from group 2 are then adjusted using the formulas:

```
x2.rep <- round((p2.hat.rep * n2), 0)
p2.hat.rep <- x2.rep/n2
```

If any of these adjusted proportions from group 2 are ≤ 0 or ≥ 1 the function terminates with a message indicating that impossible values have been supplied.

In cases where the sample sizes are small there may be instances where the user-supplied values of `p1.hat` and/or `p2.hat.minus.p1.hat` are not attainable. The argument `tol` is used to determine whether to return the table in conventional form or whether it is necessary to modify the table to include twice as many columns (see EXAMPLES section below).

Value

a data frame with elements that are character strings indicating the confidence intervals.

When `sample.type="two.sample"`, a data frame with the rows varying the estimated proportion for group 1 (i.e., the values of `p1.hat`) and the columns varying the estimated difference between the proportions from group 2 and group 1 (i.e., the values of `p2.hat.minus.p1.hat`). In cases where the sample sizes are small, it may not be possible to obtain certain differences for given values of `p1.hat`, in which case the returned data frame contains twice as many columns indicating the actual difference in one column and the compute confidence interval next to it (see EXAMPLES section below).

When `sample.type="one.sample"`, a 1-row data frame with the columns varying the estimated proportion (i.e., the values of `p1.hat`).

Note

Bacchetti (2010) presents strong arguments against the current convention in scientific research for computing sample size that is based on formulas that use a fixed Type I error (usually 5%) and a fixed minimal power (often 80%) without regard to costs. He notes that a key input to these formulas is a measure of variability (usually a standard deviation) that is difficult to measure accurately "unless there is so much preliminary data that the study isn't really needed." Also, study designers often avoid defining what a scientifically meaningful difference is by presenting sample size results in terms of the effect size (i.e., the difference of interest divided by the elusive standard deviation). Bacchetti (2010) encourages study designers to use simple tables in a sensitivity analysis to see what results of a study may look like for low, moderate, and high rates of variability and large, intermediate, and no underlying differences in the populations or processes being studied.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bacchetti, P. (2010). Current sample size conventions: Flaws, Harms, and Alternatives. *BMC Medicine* **8**, 17–23.

Also see the references in the help files for `prop.test` and `binom.test`.

See Also

`prop.test`, `binom.test`, `ciTableMean`, `ciBinomHalfWidth`, `ciBinomN`, `plotCiBinomDesign`.

Examples

```
# Reproduce Table 1 in Bacchetti (2010).  This involves planning a study with
# n1 = n2 = 935 subjects per group, where Group 1 is the control group and
# Group 2 is the treatment group.  The outcome in the study is proportion of
# subjects with serious outcomes or death.  A negative value for the difference
# in proportions between groups (Group 2 proportion - Group 1 proportion)
# indicates the treatment group has a better outcome.  In this table, the
# proportion of subjects in Group 1 with serious outcomes or death is set
# to 3%, 6.5%, and 12%, and the difference in proportions between the two
# groups is set to -2.8 percentage points, -1.4 percentage points, and 0.

ciTableProp(n1 = 935, p1.hat = c(0.03, 0.065, 0.12), n2 = 935,
  p2.hat.minus.p1.hat = c(-0.028, -0.014, 0), digits = 3)
#                  Diff=-0.028      Diff=-0.014           Diff=0
#P1.hat=0.030 [-0.040, -0.015] [-0.029,  0.001] [-0.015,  0.015]
#P1.hat=0.065 [-0.049, -0.007] [-0.036,  0.008] [-0.022,  0.022]
#P1.hat=0.120 [-0.057,  0.001] [-0.044,  0.016] [-0.029,  0.029]

#==========

# Show how the returned data frame has to be modified for cases of small
# sample sizes where not all user-supplied differenes are possible.

ciTableProp(n1 = 5, n2 = 5, p1.hat = c(0.3, 0.6, 0.12), p2.hat = c(0.2, 0.1, 0))
#           Diff            CI Diff            CI Diff            CI
#P1.hat=0.4  0.2 [-0.61, 1.00]  0.0 [-0.61, 0.61]    0 [-0.61, 0.61]
#P1.hat=0.6  0.2 [-0.55, 0.95]  0.2 [-0.55, 0.95]    0 [-0.61, 0.61]
#P1.hat=0.2  0.2 [-0.55, 0.95]  0.2 [-0.55, 0.95]    0 [-0.50, 0.50]

#==========

# Suppose we are planning a study to compare the proportion of nondetects at
# a background and downgradient well, and we can use ciTableProp to look how
# the confidence interval for the difference between the two proportions using
# say 36 quarterly samples at each well varies with the observed estimated
# proportions.  Here we'll let the argument "p1.hat" denote the proportion of
# nondetects observed at the downgradient well and set this equal to
# 20%, 40% and 60%.  The argument "p2.hat.minus.p1.hat" represents the proportion
# of nondetects at the background well minus the proportion of nondetects at the
# downgradient well.

ciTableProp(n1 = 36, p1.hat = c(0.2, 0.4, 0.6), n2 = 36,
  p2.hat.minus.p1.hat = c(0.3, 0.15, 0))
#                Diff=0.31     Diff=0.14        Diff=0
#P1.hat=0.19 [ 0.07, 0.54] [-0.09, 0.37] [-0.18, 0.18]
#P1.hat=0.39 [ 0.06, 0.55] [-0.12, 0.39] [-0.23, 0.23]
#P1.hat=0.61 [ 0.09, 0.52] [-0.10, 0.38] [-0.23, 0.23]

# We see that even if the observed difference in the proportion of nondetects
# is about 15 percentage points, all of the confidence intervals for the
# difference between the proportions of nondetects at the two wells contain 0,
# so if a difference of 15 percentage points is important to substantiate, we
```

```
# may need to increase our sample sizes.
```

---

cv *Sample Coefficient of Variation.*

Description

Compute the sample coefficient of variation.

Usage

```
cv(x, method = "moments", sd.method = "sqrt.unbiased",
  l.moment.method = "unbiased", plot.pos.cons = c(a = 0.35, b = 0),
  na.rm = FALSE)
```

Arguments

x	numeric vector of observations.
method	character string specifying what method to use to compute the sample coefficient of variation. The possible values are "moments" (product moment ratio estimator; the default), or "l.moments" (L-moment ratio estimator).
sd.method	character string specifying what method to use to compute the sample standard deviation when method="moments". The possible values are "sqrt.ubiased" (the square root of the unbiased estimate of variance; the default), or "moments" (the method of moments estimator).
l.moment.method	character string specifying what method to use to compute the L-moments when method="l.moments". The possible values are "ubiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula).
plot.pos.cons	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="l.moments" and l.moment.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b".
na.rm	logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.

Details

Let $\underline{x}$ denote a random sample of n observations from some distribution with mean μ and standard deviation σ.

Product Moment Coefficient of Variation (`method="moments"`)
The coefficient of variation (sometimes denoted CV) of a distribution is defined as the ratio of the standard deviation to the mean. That is:

$$CV = \frac{\sigma}{\mu} \qquad (1)$$

The coefficient of variation measures how spread out the distribution is relative to the size of the mean. It is usually used to characterize positive, right-skewed distributions such as the lognormal distribution.

When `sd.method="sqrt.unbiased"`, the coefficient of variation is estimated using the sample mean and the square root of the unbaised estimator of variance:

$$\widehat{CV} = \frac{s}{\bar{x}} \qquad (2)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (3)$$

$$s = [\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2]^{1/2} \qquad (4)$$

Note that the estimator of standard deviation in equation (4) is not unbiased.

When `sd.method="moments"`, the coefficient of variation is estimated using the sample mean and the square root of the method of moments estimator of variance:

$$\widehat{CV} = \frac{s_m}{\bar{x}} \qquad (5)$$

$$s = [\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2]^{1/2} \qquad (6)$$

L-Moment Coefficient of Variation (`method="l.moments"`)
Hosking (1990) defines an L-moment analog of the coefficient of variation (denoted the L-CV) as:

$$\tau = \frac{l_2}{l_1} \qquad (7)$$

that is, the second L-moment divided by the first L-moment. He shows that for a positive-valued random variable, the L-CV lies in the interval (0, 1).

When `l.moment.method="unbiased"`, the L-CV is estimated by:

$$t = \frac{l_2}{l_1} \qquad (8)$$

that is, the unbiased estimator of the second L-moment divided by the unbiased estimator of the first L-moment.

When `l.moment.method="plotting.position"`, the L-CV is estimated by:

$$\tilde{t} = \frac{\tilde{l}_2}{\tilde{l}_1} \qquad (9)$$

that is, the plotting-position estimator of the second L-moment divided by the plotting-position estimator of the first L-moment.

See the help file for `lMoment` for more information on estimating L-moments.

Value

A numeric scalar – the sample coefficient of variation.

Note

Traditionally, the coefficient of variation has been estimated using product moment estimators. Hosking (1990) introduced the idea of L-moments and the L-CV. Vogel and Fennessey (1993) argue that L-moment ratios should replace product moment ratios because of their superior performance (they are nearly unbiased and better for discriminating between distributions).

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Taylor, J.K. (1990). *Statistical Techniques for Data Analysis*. Lewis Publishers, Boca Raton, FL.

Vogel, R.M., and N.M. Fennessey. (1993). L Moment Diagrams Should Replace Product Moment Diagrams. *Water Resources Research* **29**(6), 1745–1752.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

Summary Statistics, `summaryFull`, `var`, `sd`, `skewness`, `kurtosis`.

Examples

```
# Generate 20 observations from a lognormal distribution with
# parameters mean=10 and cv=1, and estimate the coefficient of variation.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormAlt(20, mean = 10, cv = 1)
```

```
cv(dat)
#[1] 0.5077981

cv(dat, sd.method = "moments")
#[1] 0.4949403

cv(dat, method = "l.moments")
#[1] 0.2804148

#----------
# Clean up
rm(dat)
```

`detectionLimitCalibrate` *Determine Detection Limit*

Description

Determine the detection limit based on using a calibration line (or curve) and inverse regression.

Usage

```
detectionLimitCalibrate(object, coverage = 0.99, simultaneous = TRUE)
```

Arguments

`object`	an object of class `"calibrate"` that is the result of calling the function `calibrate`.
`coverage`	optional numeric scalar between 0 and 1 indicating the confidence level associated with the prediction intervals used in determining the detection limit. The default value is `coverage=0.99`.
`simultaneous`	optional logical scalar indicating whether to base the prediction intervals on simultaneous or non-simultaneous prediction limits. The default value is `simultaneous=TRUE`.

Details

The idea of a decision limit and detection limit is directly related to calibration and can be framed in terms of a hypothesis test, as shown in the table below. The null hypothesis is that the chemical is not present in the physical sample, i.e., $H_0 : C = 0$, where C denotes the concentration.

Your Decision	H_0 **True** ($C = 0$)	H_0 **False** ($C > 0$)
Reject H_0 (Declare Chemical Present)	Type I Error (Probability = α)	
Do Not Reject H_0		Type II Error

(Declare Chemical Absent) (Probability = β)

Ideally, you would like to minimize both the Type I and Type II error rates. Just as we use critical values to compare against the test statistic for a hypothesis test, we need to use a critical signal level S_D called the **decision limit** to decide whether the chemical is present or absent. If the signal is less than or equal to S_D we will declare the chemical is absent, and if the signal is greater than S_D we will declare the chemical is present.

First, suppose no chemical is present (i.e., the null hypothesis is true). If we want to guard against the mistake of declaring that the chemical is present when in fact it is absent (Type I error), then we should choose S_D so that the probability of this happening is some small value α. Thus, the value of S_D depends on what we want to use for α (the Type I error rate), and the true (but unknown) value of σ (the standard deviation of the errors assuming a constant standard deviation) (Massart et al., 1988, p. 111).

When the true concentration is 0, the decision limit is the (1-α)100th percentile of the distribution of the signal *S*. Note that the decision limit is on the scale of and in units of the signal *S*.

Now suppose that in fact the chemical is present in some concentration *C* (i.e., the null hypothesis is false). If we want to guard against the mistake of declaring that the chemical is absent when in fact it is present (Type II error), then we need to determine a minimal concentration $C_D L$ called the **detection limit (DL)** that we know will yield a signal less than the decision limit S_D only a small fraction of the time (β).

In practice we do not know the true value of the standard deviation of the errors (σ), so we cannot compute the true decision limit. Also, we do not know the true values of the intercept and slope of the calibration line, so we cannot compute the true detection limit. Instead, we usually set $\alpha = \beta$ and estimate the decision and detection limits by computing prediction limits for the calibration line and using inverse regression.

The estimated detection limit corresponds to the upper confidence bound on concentration given that the signal is equal to the estimated decision limit. Currie (1997) discusses other ways to define the detection limit, and Glaser et al. (1981) define a quantity called the method detection limit.

Value

A numeric vector of length 2 indicating the signal detection limit and the concentration detection limit. This vector has two attributes called `coverage` and `simultaneous` indicating the values of these arguments that were used in the call to `detectionLimitCalibrate`.

Note

Perhaps no other topic in environmental statistics has generated as much confusion or controversy as the topic of detection limits. After decades of disparate terminology, ISO and IUPAC provided harmonized guidance on the topic in 1995 (Currie, 1997). Intuitively, the idea of a detection limit is simple to grasp: the **detection limit** is "the smallest amount or concentration of a particular substance that can be reliably detected in a given type of sample or medium by a specific measurement process" (Currie, 1997, p. 152). Unfortunately, because of the exceedingly complex nature of measuring chemical concentrations, this simple idea is difficult to apply in practice.

Detection and quantification capabilities are fundamental performance characteristics of the **Chemical Measurement Process (CMP)** (Currie, 1996, 1997). In this help file we discuss some currently

accepted definitions of the terms decision, detection, and quantification limits. For more details, the reader should consult the references listed in this help file.

The **quantification limit** is defined as the concentration C at which the coefficient of variation (also called relative standard deviation or RSD) for the distribution of the signal S is some small value, usually taken to be 10% (Currie, 1968, 1997). In practice the quantification limit is difficult to estimate because we have to estimate both the mean and the standard deviation of the signal S for any particular concentration, and usually the standard deviation varies with concentration. Variations of the quantification limit include the quantitation limit (Keith, 1991, p. 109), minimum level (USEPA, 1993), and alternative minimum level (Gibbons et al., 1997a).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Clark, M.J.R., and P.H. Whitfield. (1994). Conflicting Perspectives About Detection Limits and About the Censoring of Environmental Data. *Water Resources Bulletin* **30**(6), 1063–1079.

Clayton, C.A., J.W. Hines, and P.D. Elkins. (1987). Detection Limits with Specified Assurance Probabilities. *Analytical Chemistry* **59**, 2506–2514.

Code of Federal Regulations. (1996). Definition and Procedure for the Determination of the Method Detection Limit–Revision 1.11. Title 40, Part 136, Appendix B, 7-1-96 Edition, pp.265–267.

Currie, L.A. (1968). Limits for Qualitative Detection and Quantitative Determination: Application to Radiochemistry. *Annals of Chemistry* **40**, 586–593.

Currie, L.A. (1988). *Detection in Analytical Chemistry: Importance, Theory, and Practice*. American Chemical Society, Washington, D.C.

Currie, L.A. (1995). Nomenclature in Evaluation of Analytical Methods Including Detection and Quantification Capabilities. *Pure & Applied Chemistry* **67**(10), 1699-1723.

Currie, L.A. (1996). Foundations and Future of Detection and Quantification Limits. *Proceedings of the Section on Statistics and the Environment*, American Statistical Association, Alexandria, VA.

Currie, L.A. (1997). Detection: International Update, and Some Emerging Di-Lemmas Involving Calibration, the Blank, and Multiple Detection Decisions. *Chemometrics and Intelligent Laboratory Systems* **37**, 151-181.

Davis, C.B. (1994). Environmental Regulatory Statistics. In Patil, G.P., and C.R. Rao, eds., *Handbook of Statistics, Vol. 12: Environmental Statistics*. North-Holland, Amsterdam, a division of Elsevier, New York, NY, Chapter 26, 817–865.

Davis, C.B. (1997). Challenges in Regulatory Environmetrics. *Chemometrics and Intelligent Laboratory Systems* **37**, 43–53.

Gibbons, R.D. (1995). Some Statistical and Conceptual Issues in the Detection of Low-Level Environmental Pollutants (with Discussion). *Environmetrics* **2**, 125-167.

Gibbons, R.D., D.E. Coleman, and R.F. Maddalone. (1997a). An Alternative Minimum Level Definition for Analytical Quantification. *Environmental Science & Technology* **31**(7), 2071–2077. Comments and Discussion in Volume **31**(12), 3727–3731, and Volume **32**(15), 2346–2353.

Gibbons, R.D., D.E. Coleman, and R.F. Maddalone. (1997b). Response to Comment on "An Alternative Minimum Level Definition for Analytical Quantification". *Environmental Science and Technology* **31**(12), 3729–3731.

Gibbons, R.D., D.E. Coleman, and R.F. Maddalone. (1998). Response to Comment on "An Alternative Minimum Level Definition for Analytical Quantification". *Environmental Science and Technology* **32**(15), 2349–2353.

Gibbons, R.D., N.E. Grams, F.H. Jarke, and K.P. Stoub. (1992). Practical Quantitation Limits. *Chemometrics Intelligent Laboratory Systems* **12**, 225–235.

Gibbons, R.D., F.H. Jarke, and K.P. Stoub. (1991). Detection Limits: For Linear Calibration Curves with Increasing Variance and Multiple Future Detection Decisions. In Tatsch, D.E., editor. *Waste Testing and Quality Assurance: Volume 3*. American Society for Testing and Materials, Philadelphi, PA.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley & Sons, Hoboken. Chapter 6, p. 111.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey. Chapter 3, p. 22.

Glasser, J.A., D.L. Foerst, G.D. McKee, S.A. Quave, and W.L. Budde. (1981). Trace Analyses for Wastewaters. *Environmental Science and Technology* **15**, 1426–1435.

Hubaux, A., and G. Vos. (1970). Decision and Detection Limits for Linear Calibration Curves. *Annals of Chemistry* **42**, 849–855.

Kahn, H.D., C.E. White, K. Stralka, and R. Kuznetsovski. (1997). Alternative Estimates of Detection. *Proceedings of the Twentieth Annual EPA Conference on Analysis of Pollutants in the Environment, May 7-8, Norfolk, VA*. U.S. Environmental Protection Agency, Washington, D.C.

Kahn, H.D., W.A. Telliard, and C.E. White. (1998). Comment on "An Alternative Minimum Level Definition for Analytical Quantification" (with Response). *Environmental Science & Technology* **32**(5), 2346–2353.

Kaiser, H. (1965). Zum Problem der Nachweisgrenze. Fresenius' Z. *Anal. Chem.* **209**, 1.

Keith, L.H. (1991). *Environmental Sampling and Analysis: A Practical Guide*. Lewis Publishers, Boca Raton, FL, Chapter 10.

Kimbrough, D.E. (1997). Comment on "An Alternative Minimum Level Definition for Analytical Quantification" (with Response). *Environmental Science & Technology* **31**(12), 3727–3731.

Lambert, D., B. Peterson, and I. Terpenning. (1991). Nondetects, Detection Limits, and the Probability of Detection. *Journal of the American Statistical Association* **86**(414), 266–277.

Massart, D.L., B.G.M. Vandeginste, S.N. Deming, Y. Michotte, and L. Kaufman. (1988). *Chemometrics: A Textbook*. Elsevier, New York, Chapter 7.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Porter, P.S., R.C. Ward, and H.F. Bell. (1988). The Detection Limit. *Environmental Science & Technology* **22**(8), 856–861.

Rocke, D.M., and S. Lorenzato. (1995). A Two-Component Model for Measurement Error in Analytical Chemistry. *Technometrics* **37**(2), 176–184.

Singh, A. (1993). Multivariate Decision and Detection Limits. *Analytica Chimica Acta* **277**, 205-214.

Spiegelman, C.H. (1997). A Discussion of Issues Raised by Lloyd Currie and a Cross Disciplinary View of Detection Limits and Estimating Parameters That Are Often At or Near Zero. *Chemometrics and Intelligent Laboratory Systems* **37**, 183–188.

USEPA. (1987c). List (Phase 1) of Hazardous Constituents for Ground-Water Monitoring; Final Rule. *Federal Register* **52**(131), 25942–25953 (July 9, 1987).

Zorn, M.E., R.D. Gibbons, and W.C. Sonzogni. (1997). Weighted Least-Squares Approach to Calculating Limits of Detection and Quantification by Modeling Variability as a Function of Concentration. *Analytical Chemistry* **69**, 3069–3075.

See Also

calibrate, inversePredictCalibrate, pointwise.

Examples

```
# The data frame EPA.97.cadmium.111.df contains calibration
# data for cadmium at mass 111 (ng/L) that appeared in
# Gibbons et al. (1997b) and were provided to them by the U.S. EPA.
#
# The Example section in the help file for calibrate shows how to
# plot these data along with the fitted calibration line and 99%
# non-simultaneous prediction limits.
#
# For the current example, we will compute the decision limit (7.68)
# and detection limit (12.36 ng/L) based on using alpha = beta = 0.01
# and a linear calibration line with constant variance. See
# Millard and Neerchal (2001, pp.566-575) for more details on this
# example.

calibrate.list <- calibrate(Cadmium ~ Spike, data = EPA.97.cadmium.111.df)

detectionLimitCalibrate(calibrate.list, simultaneous = FALSE)
#        Decision Limit (Signal) Detection Limit (Concentration)
#                       7.677842                       12.364670
#attr(,"coverage")
#[1] 0.99
#attr(,"simultaneous")
#[1] FALSE

#----------

# Clean up
#---------
rm(calibrate.list)
```

---

distChoose *Choose Best Fitting Distribution Based on Goodness-of-Fit Tests*

Description

Perform a series of goodness-of-fit tests from a (possibly user-specified) set of candidate probability distributions to determine which probability distribution provides the best fit for a data set.

Usage

```
distChoose(y, ...)

## S3 method for class 'formula'
distChoose(y, data = NULL, subset,
  na.action = na.pass, ...)

## Default S3 method:
distChoose(y, alpha = 0.05, method = "sw",
    choices = c("norm", "gamma", "lnorm"), est.arg.list = NULL,
    warn = TRUE, keep.data = TRUE, data.name = NULL,
    parent.of.data = NULL, subset.expression = NULL, ...)
```

Arguments

y	an object containing data for the goodness-of-fit tests. In the default method, the argument y must be numeric vector of observations. In the formula method, y must be a formula of the form y ~ 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
data	specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which distChoose is called.
subset	specifies an optional vector specifying a subset of observations to be used.
na.action	specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.
alpha	numeric scalar between 0 and 1 specifying the Type I error associated with each goodness-of-fit test. When method="proucl" the only allowed values for alpha are 0.01, 0.05, and 0.1. The default value is alpha=0.05.
method	character string defining which method to use. Possible values are: • "sw". Shapiro-Wilk; the default. • "sf". Shapiro-Francia. • "ppcc". Probability Plot Correlation Coefficient. • "proucl". ProUCL. See the DETAILS section below.
choices	a character vector denoting the distribution abbreviations of the candidate distributions. See the help file for Distribution.df for a list of distributions and their abbreviations. The default value is choices=c("norm", "gamma", "lnorm"), indicating the Normal, Gamma, and Lognormal distributions. This argument is ignored when method="proucl".

`est.arg.list` a list containing one or more lists of arguments to be passed to the function(s) estimating the distribution parameters. The name(s) of the components of the list must be equal to or a subset of the values of the argument `choices`. For example, if `choices=c("norm", "gamma", "lnorm")`, setting `est.arg.list=list(gamma = list(method="bcmle"))` indicates using the bias-corrected maximum-likelihood estimators of shape and scale for the gamma distribution (see the help file for `egamma`). See the help file Estimating Distribution Parameters for a list of estimating functions. The default value is `est.arg.list=NULL` so that all default values for the estimating functions are used.

When testing for some form of normality (i.e., Normal, Lognormal, Three-Parameter Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta)), the estimated parameters are provided in the output merely for information, and the choice of the method of estimation has no effect on the goodness-of-fit test statistics or p-values.

This argument is ignored when `method="proucl"`.

`warn` logical scalar indicating whether to print a warning message when observations with `NA`s, `NaN`s, or `Inf`s in `y` are removed. The default value is `TRUE`.

`keep.data` logical scalar indicating whether to return the original data. The default value is `keep.data=TRUE`.

`data.name` optional character string indicating the name of the data used for argument `y`.

`parent.of.data` character string indicating the source of the data used for the goodness-of-fit test.

`subset.expression` character string indicating the expression used to subset the data.

`...` additional arguments affecting the goodness-of-fit test.

Details

The function `distChoose` returns a list with information on the goodness-of-fit tests for various distributions and which distribution appears to best fit the data based on the p-values from the goodness-of-fit tests. This function was written in order to compare ProUCL's way of choosing the best-fitting distribution (USEPA, 2015) with other ways of choosing the best-fitting distribution.

Method Based on Shapiro-Wilk, Shapiro-Francia, or Probability Plot Correlation Test (`method="sw"`, `method="sf"`, or `method="ppcc"`)

For each value of the argument `choices`, the function `distChoose` runs the goodness-of-fit test using the data in `y` assuming that particular distribution. For example, if `choices=c("norm", "gamma", "lnorm")`, indicating the Normal, Gamma, and Lognormal distributions, and `method="sw"`, then the usual Shapiro-Wilk test is performed for the Normal and Lognormal distributions, and the extension of the Shapiro-Wilk test is performed for the Gamma distribution (see the section *Testing Goodness-of-Fit for Any Continuous Distribution* in the help file for `gofTest` for an explanation of the latter). The distribution associated with the largest p-value is the chosen distribution. In the case when all p-values are less than the value of the argument `alpha`, the distribution "Nonparametric" is chosen.

Method Based on ProUCL Algorithm (`method="proucl"`)

When method="proucl", the function distChoose uses the algorithm that ProUCL (USEPA, 2015) uses to determine the best fitting distribution. The candidate distributions are the Normal, Gamma, and Lognormal distributions. The algorithm used by ProUCL is as follows:

1. Perform the Shapiro-Wilk and Lilliefors goodness-of-fit tests for the Normal distribution, i.e., call the function gofTest with distribution = "norm", test="sw" and distribution = "norm", test="lillie". If either or both of the associated p-values are greater than or equal to the user-supplied value of alpha, then choose the Normal distribution. Otherwise, proceed to the next step.
2. Perform the "ProUCL Anderson-Darling" and "ProUCL Kolmogorov-Smirnov" goodness-of-fit tests for the Gamma distribution, i.e., call the function gofTest with distribution="gamma", test="proucl.ad.gamma" and distribution="gamma", test="proucl.ks.gamma". If either or both of the associated p-values are greater than or equal to the user-supplied value of alpha, then choose the Gamma distribution. Otherwise, proceed to the next step.
3. Perform the Shapiro-Wilk and Lilliefors goodness-of-fit tests for the Lognormal distribution, i.e., call the function gofTest with distribution="lnorm", test="sw" and distribution="lnorm", test="lillie". If either or both of the associated p-values are greater than or equal to the user-supplied value of alpha, then choose the Lognormal distribution. Otherwise, proceed to the next step.
4. If none of the goodness-of-fit tests above yields a p-value greater than or equal to the user-supplied value of alpha, then choose the "Nonparametric" distribution.

Value

a list of class "distChoose" containing the results of the goodness-of-fit tests. Objects of class "distChoose" have a special printing method. See the help files for distChoose.object for details.

Note

In practice, almost any goodness-of-fit test will *not* reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test *will* reject the null hypothesis if the number of observations is very large, since "real" data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution of "real" data is close enough to some theoretical distribution that fairly accurate results may be provided by assuming that particular theoretical distribution. One way to asses the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see qqPlot).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Birnbaum, Z.W., and F.H. Tingey. (1951). One-Sided Confidence Contours for Probability Distribution Functions. *Annals of Mathematical Statistics* **22**, 592-596.

Blom, G. (1958). *Statistical Estimates and Transformed Beta Variables*. John Wiley and Sons, New York.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Dallal, G.E., and L. Wilkinson. (1986). An Analytic Approximation to the Distribution of Lilliefor's Test for Normality. *The American Statistician* **40**, 294-296.

D'Agostino, R.B. (1970). Transformation to Normality of the Null Distribution of $g1$. *Biometrika* **57**, 679-681.

D'Agostino, R.B. (1971). An Omnibus Test of Normality for Moderate and Large Size Samples. *Biometrika* **58**, 341-348.

D'Agostino, R.B. (1986b). Tests for the Normal Distribution. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York.

D'Agostino, R.B., and E.S. Pearson (1973). Tests for Departures from Normality. Empirical Results for the Distributions of $b2$ and $\sqrt{b1}$. *Biometrika* **60**(3), 613-622.

D'Agostino, R.B., and G.L. Tietjen (1973). Approaches to the Null Distribution of $\sqrt{b1}$. *Biometrika* **60**(1), 169-173.

Fisher, R.A. (1950). *Statistical Methods for Research Workers*. 11'th Edition. Hafner Publishing Company, New York, pp.99-100.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Kendall, M.G., and A. Stuart. (1991). *The Advanced Theory of Statistics, Volume 2: Inference and Relationship*. Fifth Edition. Oxford University Press, New York.

Kim, P.J., and R.I. Jennrich. (1973). Tables of the Exact Sampling Distribution of the Two Sample Kolmogorov-Smirnov Criterion. In Harter, H.L., and D.B. Owen, eds. *Selected Tables in Mathematical Statistics, Vol. 1*. American Mathematical Society, Providence, Rhode Island, pp.79-170.

Kolmogorov, A.N. (1933). Sulla determinazione empirica di una legge di distribuzione. *Giornale dell' Istituto Italiano degle Attuari* **4**, 83-91.

Marsaglia, G., W.W. Tsang, and J. Wang. (2003). Evaluating Kolmogorov's distribution. *Journal of Statistical Software*, **8**(18). doi:10.18637/jss.v008.i18.

Moore, D.S. (1986). Tests of Chi-Squared Type. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, pp.63-95.

Pomeranz, J. (1973). Exact Cumulative Distribution of the Kolmogorov-Smirnov Statistic for Small Samples (Algorithm 487). *Collected Algorithms from ACM* ??, ???-???.

Royston, J.P. (1992a). Approximating the Shapiro-Wilk W-Test for Non-Normality. *Statistics and Computing* **2**, 117-119.

Royston, J.P. (1992b). Estimation, Reference Ranges and Goodness of Fit for the Three-Parameter Log-Normal Distribution. *Statistics in Medicine* **11**, 897-912.

Royston, J.P. (1992c). A Pocket-Calculator Algorithm for the Shapiro-Francia Test of Non-Normality: An Application to Medicine. *Statistics in Medicine* **12**, 181-184.

Royston, P. (1993). A Toolkit for Testing for Non-Normality in Complete and Censored Samples. *The Statistician* **42**, 37-43.

Ryan, T., and B. Joiner. (1973). *Normal Probability Plots and Tests for Normality*. Technical Report, Pennsylvannia State University, Department of Statistics.

Shapiro, S.S., and R.S. Francia. (1972). An Approximate Analysis of Variance Test for Normality. *Journal of the American Statistical Association* **67**(337), 215-219.

Shapiro, S.S., and M.B. Wilk. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika* **52**, 591-611.

Smirnov, N.V. (1939). Estimate of Deviation Between Empirical Distribution Functions in Two Independent Samples. *Bulletin Moscow University* **2**(2), 3-16.

Smirnov, N.V. (1948). Table for Estimating the Goodness of Fit of Empirical Distributions. *Annals of Mathematical Statistics* **19**, 279-281.

Stephens, M.A. (1970). Use of the Kolmogorov-Smirnov, Cramer-von Mises and Related Statistics Without Extensive Tables. *Journal of the Royal Statistical Society, Series B*, **32**, 115-122.

Stephens, M.A. (1986a). Tests Based on EDF Statistics. In D'Agostino, R. B., and M.A. Stevens, eds. *Goodness-of-Fit Techniques*. Marcel Dekker, New York.

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C.

Verrill, S., and R.A. Johnson. (1987). The Asymptotic Equivalence of Some Modified Shapiro-Wilk Statistics – Complete and Censored Sample Cases. *The Annals of Statistics* **15**(1), 413-419.

Verrill, S., and R.A. Johnson. (1988). Tables and Large-Sample Distribution Theory for Censored-Data Correlation Statistics for Testing Normality. *Journal of the American Statistical Association* **83**, 1192-1197.

Weisberg, S., and C. Bingham. (1975). An Approximate Analysis of Variance Test for Non-Normality Suitable for Machine Calculation. *Technometrics* **17**, 133-134.

Wilk, M.B., and S.S. Shapiro. (1968). The Joint Assessment of Normality of Several Independent Samples. *Technometrics*, **10**(4), 825-839.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`gofTest`, `distChoose.object`, `print.distChoose`.

Examples

```
# Generate 20 observations from a gamma distribution with
# parameters shape = 2 and scale = 3 and:
#
# 1) Call distChoose using the Shapiro-Wilk method.
#
# 2) Call distChoose using the Shapiro-Wilk method and specify
#    the bias-corrected method of estimating shape for the Gamma
#    distribution.
#
# 3) Compare the results in 2) above with the results using the
#    ProUCL method.
#
# Notes:  The call to set.seed lets you reproduce this example.
#
#         The ProUCL method chooses the Normal distribution, whereas the
#         Shapiro-Wilk method chooses the Gamma distribution.
```

```
set.seed(47)
dat <- rgamma(20, shape = 2, scale = 3)


# 1) Call distChoose using the Shapiro-Wilk method.
#--------------------------------------------------

distChoose(dat)

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   Shapiro-Wilk
#
#Type I Error per Test:           0.05
#
#Decision:                        Gamma
#
#Estimated Parameter(s):          shape = 1.909462
#                                 scale = 4.056819
#
#Estimation Method:               MLE
#
#Data:                            dat
#
#Sample Size:                     20
#
#Test Results:
#
#  Normal
#    Test Statistic:              W = 0.9097488
#    P-value:                     0.06303695
#
#  Gamma
#    Test Statistic:              W = 0.9834958
#    P-value:                     0.970903
#
#  Lognormal
#    Test Statistic:              W = 0.9185006
#    P-value:                     0.09271768

#--------------------

# 2) Call distChoose using the Shapiro-Wilk method and specify
#    the bias-corrected method of estimating shape for the Gamma
#    distribution.
#-----------------------------------------------------------------
```

```
distChoose(dat, method = "sw",
  est.arg.list = list(gamma = list(method = "bcmle")))

#Results of Choosing Distribution
#---------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   Shapiro-Wilk
#
#Type I Error per Test:           0.05
#
#Decision:                        Gamma
#
#Estimated Parameter(s):          shape = 1.656376
#                                 scale = 4.676680
#
#Estimation Method:               Bias-Corrected MLE
#
#Data:                            dat
#
#Sample Size:                     20
#
#Test Results:
#
#  Normal
#    Test Statistic:              W = 0.9097488
#    P-value:                     0.06303695
#
#  Gamma
#    Test Statistic:              W = 0.9834346
#    P-value:                     0.9704046
#
#  Lognormal
#    Test Statistic:              W = 0.9185006
#    P-value:                     0.09271768

#--------------------

# 3) Compare the results in 2) above with the results using the
#    ProUCL method.
#-------------------------------------------------------------------

distChoose(dat, method = "proucl")

#Results of Choosing Distribution
#---------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
```

```
#
#Choice Method:                  ProUCL
#
#Type I Error per Test:          0.05
#
#Decision:                       Normal
#
#Estimated Parameter(s):         mean = 7.746340
#                                sd   = 5.432175
#
#Estimation Method:              mvue
#
#Data:                           dat
#
#Sample Size:                    20
#
#Test Results:
#
#  Normal
#    Shapiro-Wilk GOF
#      Test Statistic:            W = 0.9097488
#      P-value:                   0.06303695
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:            D = 0.1547851
#      P-value:                   0.238092
#
#  Gamma
#    ProUCL Anderson-Darling Gamma GOF
#      Test Statistic:            A = 0.1853826
#      P-value:                   >= 0.10
#    ProUCL Kolmogorov-Smirnov Gamma GOF
#      Test Statistic:            D = 0.0988692
#      P-value:                   >= 0.10
#
#  Lognormal
#    Shapiro-Wilk GOF
#      Test Statistic:            W = 0.9185006
#      P-value:                   0.09271768
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:            D = 0.149317
#      P-value:                   0.2869177

#--------------------

# Clean up
#---------

rm(dat)

#======================================================================

# Example 10-2 of USEPA (2009, page 10-14) gives an example of
# using the Shapiro-Wilk test to test the assumption of normality
```

```
# for nickel concentrations (ppb) in groundwater collected over
# 4 years.  The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

EPA.09.Ex.10.1.nickel.df
#   Month   Well Nickel.ppb
#1      1 Well.1       58.8
#2      3 Well.1        1.0
#3      6 Well.1      262.0
#4      8 Well.1       56.0
#5     10 Well.1        8.7
#6      1 Well.2       19.0
#7      3 Well.2       81.5
#8      6 Well.2      331.0
#9      8 Well.2       14.0
#10    10 Well.2       64.4
#11     1 Well.3       39.0
#12     3 Well.3      151.0
#13     6 Well.3       27.0
#14     8 Well.3       21.4
#15    10 Well.3      578.0
#16     1 Well.4        3.1
#17     3 Well.4      942.0
#18     6 Well.4       85.6
#19     8 Well.4       10.0
#20    10 Well.4      637.0

# Use distChoose with the probability plot correlation method,
# and for the lognormal distribution specify the
# mean and CV parameterization:
#-------------------------------------------------------------

distChoose(Nickel.ppb ~ 1, data = EPA.09.Ex.10.1.nickel.df,
  choices = c("norm", "gamma", "lnormAlt"), method = "ppcc")

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   PPCC
#
#Type I Error per Test:           0.05
#
#Decision:                        Lognormal
#
#Estimated Parameter(s):          mean = 213.415628
#                                 cv   =   2.809377
#
#Estimation Method:               mvue
#
```

```
#Data:                             Nickel.ppb
#
#Data Source:                      EPA.09.Ex.10.1.nickel.df
#
#Sample Size:                      20
#
#Test Results:
#
#  Normal
#    Test Statistic:               r = 0.8199825
#    P-value:                      5.753418e-05
#
#  Gamma
#    Test Statistic:               r = 0.9749044
#    P-value:                      0.317334
#
#  Lognormal
#    Test Statistic:               r = 0.9912528
#    P-value:                      0.9187852

#--------------------

# Repeat the above example using the ProUCL method.
#-------------------------------------------------

distChoose(Nickel.ppb ~ 1, data = EPA.09.Ex.10.1.nickel.df,
  method = "proucl")

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:          Normal
#                                  Gamma
#                                  Lognormal
#
#Choice Method:                    ProUCL
#
#Type I Error per Test:            0.05
#
#Decision:                         Gamma
#
#Estimated Parameter(s):           shape =   0.5198727
#                                  scale = 326.0894272
#
#Estimation Method:                MLE
#
#Data:                             Nickel.ppb
#
#Data Source:                      EPA.09.Ex.10.1.nickel.df
#
#Sample Size:                      20
#
#Test Results:
```

```
#
#  Normal
#    Shapiro-Wilk GOF
#      Test Statistic:           W = 0.6788888
#      P-value:                  2.17927e-05
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:           D = 0.3267052
#      P-value:                  5.032807e-06
#
#  Gamma
#    ProUCL Anderson-Darling Gamma GOF
#      Test Statistic:           A = 0.5076725
#      P-value:                  >= 0.10
#    ProUCL Kolmogorov-Smirnov Gamma GOF
#      Test Statistic:           D = 0.1842904
#      P-value:                  >= 0.10
#
#  Lognormal
#    Shapiro-Wilk GOF
#      Test Statistic:           W = 0.978946
#      P-value:                  0.9197735
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:           D = 0.08405167
#      P-value:                  0.9699648

#=====================================================================

## Not run:
# 1) Simulate 1000 trials where for each trial you:
#    a) Generate 20 observations from a Gamma distribution with
#       parameters mean = 10 and CV = 1.
#    b) Use distChoose with the Shapiro-Wilk method.
#    c) Use distChoose with the ProUCL method.
#
#  2) Compare the proportion of times the
#     Normal vs. Gamma vs. Lognormal vs. Nonparametric distribution
#     is chosen for b) and c) above.
#---------------------------------------------------------------------

set.seed(58)
N <- 1000

Choose.fac <- factor(rep("", N), levels = c("Normal", "Gamma", "Lognormal", "Nonparametric"))
Choose.df <- data.frame(SW = Choose.fac, ProUCL = Choose.fac)

for(i in 1:N) {
  dat <- rgammaAlt(20, mean = 10, cv = 1)
  Choose.df[i, "SW"]     <- distChoose(dat, method = "sw")$decision
  Choose.df[i, "ProUCL"] <- distChoose(dat, method = "proucl")$decision
}

summaryStats(Choose.df, digits = 0)
```

```
#               ProUCL(N) ProUCL(Pct) SW(N) SW(Pct)
#Normal              443          44    41       4
#Gamma               546          55   733      73
#Lognormal            9           1   215      22
#Nonparametric        2           0    11       1
#Combined           1000         100  1000     100


#--------------------


# Repeat above example for the Lognormal Distribution with mean=10 and CV = 1.
#-----------------------------------------------------------------------------

set.seed(297)
N <- 1000

Choose.fac <- factor(rep("", N), levels = c("Normal", "Gamma", "Lognormal", "Nonparametric"))
Choose.df <- data.frame(SW = Choose.fac, ProUCL = Choose.fac)

for(i in 1:N) {
  dat <- rlnormAlt(20, mean = 10, cv = 1)
  Choose.df[i, "SW"]     <- distChoose(dat, method = "sw")$decision
  Choose.df[i, "ProUCL"] <- distChoose(dat, method = "proucl")$decision
}

summaryStats(Choose.df, digits = 0)

#               ProUCL(N) ProUCL(Pct) SW(N) SW(Pct)
#Normal              313          31    15       2
#Gamma               556          56   254      25
#Lognormal           121          12   706      71
#Nonparametric        10           1    25       2
#Combined           1000         100  1000     100


#--------------------


# Clean up
#---------

rm(N, Choose.fac, Choose.df, i, dat)

## End(Not run)
```

---

distChoose.object *S3 Class "distChoose"*

Description

Objects of S3 class "distChoose" are returned by the **EnvStats** function distChoose.

Details

Objects of S3 class "distChoose" are lists that contain information about the candidate distributions, the estimated distribution parameters for each candidate distribution, and the test statistics and p-values associated with each candidate distribution.

Value

Required Components

The following components must be included in a legitimate list of class "distChoose".

choices — a character vector containing the full names of the candidate distributions. (see Distribution.df).

method — a character string denoting which method was used.

decision — a character vector containing the full name of the chosen distribution.

alpha — a numeric scalar between 0 and 1 specifying the Type I error associated with each goodness-of-fit test.

distribution.parameters — a numeric vector containing the estimated parameters associated with the chosen distribution.

estimation.method — a character string indicating the method used to compute the estimated parameters associated with the chosen distribution. The value of this component will depend on the available estimation methods (see Distribution.df).

sample.size — a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit tests.

test.results — a list with the same number of components as the number of elements in the component choices. The names of the list are the distribution abbreviations of the candidate distributions. (See the help file for Distribution.df for a list of distributions and their abbreviations.) Each component is an object of class gof containing the results of the goodness-of-fit test for that particular hypothesized distribution.

data.name — character string indicating the name of the data object used for the goodness-of-fit tests.

Optional Components

The following component is included in the result of calling distChoose when the argument keep.data=TRUE:

data — numeric vector containing the data actually used for the goodness-of-fit tests (i.e., the original data without any missing or infinite values).

The following component is included in the result of calling distChoose when missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values are present:

bad.obs — numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to choosing a distribution.

Methods

Generic functions that have methods for objects of class "distChoose" include:
print.

Note

Since objects of class "distChoose" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

distChoose, print.distChoose, Goodness-of-Fit Tests, Distribution.df.

Examples

```
# Create an object of class "distChoose", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(47)
dat <- rgamma(20, shape = 2, scale = 3)

distChoose.obj <- distChoose(dat)

mode(distChoose.obj)
#[1] "list"

class(distChoose.obj)
#[1] "distChoose"

names(distChoose.obj)
#[1] "choices"                 "method"
#[3] "decision"                "alpha"
#[5] "distribution.parameters" "estimation.method"
#[7] "sample.size"             "test.results"
#[9] "data"                    "data.name"

distChoose.obj

#Results of Choosing Distribution
```

```
#--------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   Shapiro-Wilk
#
#Type I Error per Test:           0.05
#
#Decision:                        Gamma
#
#Estimated Parameter(s):          shape = 1.909462
#                                 scale = 4.056819
#
#Estimation Method:               MLE
#
#Data:                            dat
#
#Sample Size:                     20
#
#Test Results:
#
#  Normal
#    Test Statistic:              W = 0.9097488
#    P-value:                     0.06303695
#
#  Gamma
#    Test Statistic:              W = 0.9834958
#    P-value:                     0.970903
#
#  Lognormal
#    Test Statistic:              W = 0.9185006
#    P-value:                     0.09271768

#==========

# Extract the choices
#--------------------

distChoose.obj$choices
#[1] "Normal"    "Gamma"     "Lognormal"

#==========

# Clean up
#---------
rm(dat, distChoose.obj)
```

---

distChooseCensored *Choose Best Fitting Distribution Based on Goodness-of-Fit Tests for Censored Data*

Description

Perform a series of goodness-of-fit tests for censored data from a (possibly user-specified) set of candidate probability distributions to determine which probability distribution provides the best fit for a data set.

Usage

```
distChooseCensored(x, censored, censoring.side = "left", alpha = 0.05,
    method = "sf", choices = c("norm", "gamma", "lnorm"),
    est.arg.list = NULL, prob.method = "hirsch-stedinger",
    plot.pos.con = 0.375, warn = TRUE, keep.data = TRUE,
    data.name = NULL, censoring.name = NULL)
```

Arguments

x	a numeric vector containing data for the goodness-of-fit tests. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
alpha	numeric scalar between 0 and 1 specifying the Type I error associated with each goodness-of-fit test. When method="proucl" the only allowed values for alpha are 0.01, 0.05, and 0.1. The default value is alpha=0.05.
method	character string defining which method to use. Possible values are: • "sw". Shapiro-Wilk. • "sf". Shapiro-Francia; the default. • "ppcc". Probability Plot Correlation Coefficient. • "proucl". ProUCL. The Shapiro-Wilk method is only available for singly censored data. See the DETAILS section for more information.
choices	a character vector denoting the distribution abbreviations of the candidate distributions. See the help file for Distribution.df for a list of distributions and their abbreviations. The default value is choices=c("norm", "gamma", "lnorm"), indicating the Normal, Gamma, and Lognormal distributions. This argument is ignored when method="proucl".
est.arg.list	a list containing one or more lists of arguments to be passed to the function(s) estimating the distribution parameters. The name(s) of the components of the list must be equal to or a subset of the values of the argument choices. For example, if choices=c("norm", "gammaAlt", "lnormAlt"), setting est.arg.list=list(lnormAlt=list(method="bcmle")) indicates using the

bias-corrected maximum-likelihood estimators (see the help file for elnormAltCensored). See the section **Estimating Distribution Parameters** in the help file EnvStats Functions for Censored Data for a list of available estimating functions for censored data. The default value is est.arg.list=NULL so that all default values for the estimating functions are used.

In the course of testing for some form of normality (i.e., Normal, Lognormal), the estimated parameters are saved in the test.results component of the returned object, but the choice of the method of estimation has no effect on the goodness-of-fit test statistic or p-value.

This argument is ignored when method="proucl".

prob.method — character string indicating what method to use to compute the plotting positions (empirical probabilities) when test="sf" or test="ppcc". Possible values are:

- "modified kaplan-meier" (modification of product-limit method of Kaplan and Meier (1958))
- "nelson" (hazard plotting method of Nelson (1972))
- "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986))
- "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987))

The default value is prob.method="hirsch-stedinger".

The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" method is only available for censoring.side="left". See the help files for gofTestCensored and ppointsCensored for more information.

plot.pos.con — numeric scalar between 0 and 1 containing the value of the plotting position constant to use when test="sf" or test="ppcc". The default value is plot.pos.con=0.375. See the help files for gofTestCensored and ppointsCensored for more information.

warn — logical scalar indicating whether to print a warning message when observations with NAs, NaNs, or Infs in y are removed. The default value is warn=TRUE.

keep.data — logical scalar indicating whether to return the original data. The default value is keep.data=TRUE.

data.name — optional character string indicating the name of the data used for argument x.

censoring.name — optional character string indicating the name for the data used for argument censored.

Details

The function distChooseCensored returns a list with information on the goodness-of-fit tests for various distributions and which distribution appears to best fit the data based on the p-values from the goodness-of-fit tests. This function was written in order to compare ProUCL's way of choosing the best-fitting distribution (USEPA, 2015) with other ways of choosing the best-fitting distribution.

Method Based on Shapiro-Wilk, Shapiro-Francia, or Probability Plot Correlation Test (method="sw", method="sf", or method="ppcc")

For each value of the argument choices, the function distChooseCensored runs the goodness-of-fit test using the data in x assuming that particular distribution. For example, if choices=c("norm", "gamma", "lnorm"), indicating the Normal, Gamma, and Lognormal distributions, and method="sf", then the usual Shapiro-Francia test is performed for the Normal and Lognormal distributions, and the extension of the Shapiro-Francia test is performed for the Gamma distribution (see the section *Testing Goodness-of-Fit for Any Continuous Distribution* in the help file for gofTestCensored for an explanation of the latter). The distribution associated with the largest p-value is the chosen distribution. In the case when all p-values are less than the value of the argument alpha, the distribution "Nonparametric" is chosen.

Method Based on ProUCL Algorithm (method="proucl")

When method="proucl", the function distChooseCensored uses the algorithm that ProUCL (USEPA, 2015) uses to determine the best fitting distribution. The candidate distributions are the Normal, Gamma, and Lognormal distributions. The algorithm used by ProUCL is as follows:

1. Remove all censored observations and use only the uncensored observations.
2. Perform the Shapiro-Wilk and Lilliefors goodness-of-fit tests for the Normal distribution, i.e., call the function gofTest with distribution="norm", test="sw" and distribution = "norm", test="lillie". If either or both of the associated p-values are greater than or equal to the user-supplied value of alpha, then choose the Normal distribution. Otherwise, proceed to the next step.
3. Perform the "ProUCL Anderson-Darling" and "ProUCL Kolmogorov-Smirnov" goodness-of-fit tests for the Gamma distribution, i.e., call the function gofTest with distribution="gamma", test="proucl.ad.gamma" and distribution="gamma", test="proucl.ks.gamma". If either or both of the associated p-values are greater than or equal to the user-supplied value of alpha, then choose the Gamma distribution. Otherwise, proceed to the next step.
4. Perform the Shapiro-Wilk and Lilliefors goodness-of-fit tests for the Lognormal distribution, i.e., call the function gofTest with distribution = "lnorm", test="sw" and distribution = "lnorm", test="lillie". If either or both of the associated p-values are greater than or equal to the user-supplied value of alpha, then choose the Lognormal distribution. Otherwise, proceed to the next step.
5. If none of the goodness-of-fit tests above yields a p-value greater than or equal to the user-supplied value of alpha, then choose the "Nonparametric" distribution.

Value

a list of class "distChooseCensored" containing the results of the goodness-of-fit tests. Objects of class "distChooseCensored" have a special printing method. See the help file for distChooseCensored.object for details.

Note

In practice, almost any goodness-of-fit test will *not* reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test *will* reject the null hypothesis if the number of observations is very large, since "real" data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution

of "real" data is close enough to some theoretical distribution that fairly accurate results may be provided by assuming that particular theoretical distribution. One way to asses the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see `qqPlotCensored`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Birnbaum, Z.W., and F.H. Tingey. (1951). One-Sided Confidence Contours for Probability Distribution Functions. *Annals of Mathematical Statistics* **22**, 592-596.

Blom, G. (1958). *Statistical Estimates and Transformed Beta Variables*. John Wiley and Sons, New York.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Dallal, G.E., and L. Wilkinson. (1986). An Analytic Approximation to the Distribution of Lilliefor's Test for Normality. *The American Statistician* **40**, 294-296.

D'Agostino, R.B. (1970). Transformation to Normality of the Null Distribution of $g1$. *Biometrika* **57**, 679-681.

D'Agostino, R.B. (1971). An Omnibus Test of Normality for Moderate and Large Size Samples. *Biometrika* **58**, 341-348.

D'Agostino, R.B. (1986b). Tests for the Normal Distribution. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York.

D'Agostino, R.B., and E.S. Pearson (1973). Tests for Departures from Normality. Empirical Results for the Distributions of $b2$ and $\sqrt{b1}$. *Biometrika* **60**(3), 613-622.

D'Agostino, R.B., and G.L. Tietjen (1973). Approaches to the Null Distribution of $\sqrt{b1}$. *Biometrika* **60**(1), 169-173.

Fisher, R.A. (1950). *Statistical Methods for Research Workers*. 11'th Edition. Hafner Publishing Company, New York, pp.99-100.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Kendall, M.G., and A. Stuart. (1991). *The Advanced Theory of Statistics, Volume 2: Inference and Relationship*. Fifth Edition. Oxford University Press, New York.

Kim, P.J., and R.I. Jennrich. (1973). Tables of the Exact Sampling Distribution of the Two Sample Kolmogorov-Smirnov Criterion. In Harter, H.L., and D.B. Owen, eds. *Selected Tables in Mathematical Statistics, Vol. 1*. American Mathematical Society, Providence, Rhode Island, pp.79-170.

Kolmogorov, A.N. (1933). Sulla determinazione empirica di una legge di distribuzione. *Giornale dell' Istituto Italiano degle Attuari* **4**, 83-91.

Marsaglia, G., W.W. Tsang, and J. Wang. (2003). Evaluating Kolmogorov's distribution. *Journal of Statistical Software*, **8**(18). doi:10.18637/jss.v008.i18.

Moore, D.S. (1986). Tests of Chi-Squared Type. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, pp.63-95.

Pomeranz, J. (1973). Exact Cumulative Distribution of the Kolmogorov-Smirnov Statistic for Small Samples (Algorithm 487). *Collected Algorithms from ACM* ??, ???-???.

Royston, J.P. (1992a). Approximating the Shapiro-Wilk W-Test for Non-Normality. *Statistics and Computing* **2**, 117-119.

Royston, J.P. (1992b). Estimation, Reference Ranges and Goodness of Fit for the Three-Parameter Log-Normal Distribution. *Statistics in Medicine* **11**, 897-912.

Royston, J.P. (1992c). A Pocket-Calculator Algorithm for the Shapiro-Francia Test of Non-Normality: An Application to Medicine. *Statistics in Medicine* **12**, 181-184.

Royston, P. (1993). A Toolkit for Testing for Non-Normality in Complete and Censored Samples. *The Statistician* **42**, 37-43.

Ryan, T., and B. Joiner. (1973). *Normal Probability Plots and Tests for Normality*. Technical Report, Pennsylvania State University, Department of Statistics.

Shapiro, S.S., and R.S. Francia. (1972). An Approximate Analysis of Variance Test for Normality. *Journal of the American Statistical Association* **67**(337), 215-219.

Shapiro, S.S., and M.B. Wilk. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika* **52**, 591-611.

Smirnov, N.V. (1939). Estimate of Deviation Between Empirical Distribution Functions in Two Independent Samples. *Bulletin Moscow University* **2**(2), 3-16.

Smirnov, N.V. (1948). Table for Estimating the Goodness of Fit of Empirical Distributions. *Annals of Mathematical Statistics* **19**, 279-281.

Stephens, M.A. (1970). Use of the Kolmogorov-Smirnov, Cramer-von Mises and Related Statistics Without Extensive Tables. *Journal of the Royal Statistical Society, Series B*, **32**, 115-122.

Stephens, M.A. (1986a). Tests Based on EDF Statistics. In D'Agostino, R. B., and M.A. Stevens, eds. *Goodness-of-Fit Techniques*. Marcel Dekker, New York.

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C.

Verrill, S., and R.A. Johnson. (1987). The Asymptotic Equivalence of Some Modified Shapiro-Wilk Statistics – Complete and Censored Sample Cases. *The Annals of Statistics* **15**(1), 413-419.

Verrill, S., and R.A. Johnson. (1988). Tables and Large-Sample Distribution Theory for Censored-Data Correlation Statistics for Testing Normality. *Journal of the American Statistical Association* **83**, 1192-1197.

Weisberg, S., and C. Bingham. (1975). An Approximate Analysis of Variance Test for Non-Normality Suitable for Machine Calculation. *Technometrics* **17**, 133-134.

Wilk, M.B., and S.S. Shapiro. (1968). The Joint Assessment of Normality of Several Independent Samples. *Technometrics*, **10**(4), 825-839.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`gofTestCensored`, `distChooseCensored.object`, `print.distChooseCensored`, `distChoose`.

Examples

```
# Generate 30 observations from a gamma distribution with
# parameters mean=10 and cv=1 and then censor observations less than 5.
# Then:
#
# 1) Call distChooseCensored using the Shapiro-Wilk method and specify
#    choices of the
#      normal,
#      gamma (alternative parameterzation), and
#      lognormal (alternative parameterization)
#    distributions.
#
# 2) Compare the results in 1) above with the results using the
#    ProUCL method.
#
# Notes:  The call to set.seed lets you reproduce this example.
#
#         The ProUCL method chooses the Normal distribution, whereas the
#         Shapiro-Wilk method chooses the Gamma distribution.

set.seed(598)

dat <- sort(rgammaAlt(30, mean = 10, cv = 1))
dat
# [1]  0.5313509  1.4741833  1.9936208  2.7980636  3.4509840
# [6]  3.7987348  4.5542952  5.5207531  5.5253596  5.7177872
#[11]  5.7513827  9.1086375  9.8444090 10.6247123 10.9304922
#[16] 11.7925398 13.3432689 13.9562777 14.6029065 15.0563342
#[21] 15.8730642 16.0039936 16.6910715 17.0288922 17.8507891
#[26] 19.1105522 20.2657141 26.3815970 30.2912797 42.8726101

dat.censored <- dat
censored <- dat.censored < 5
dat.censored[censored] <- 5


# 1) Call distChooseCensored using the Shapiro-Wilk method.
#----------------------------------------------------------

distChooseCensored(dat.censored, censored, method = "sw",
  choices = c("norm", "gammaAlt", "lnormAlt"))

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   Shapiro-Wilk
#
#Type I Error per Test:           0.05
```

```
#
#Decision:                          Gamma
#
#Estimated Parameter(s):           mean = 12.4911448
#                                   cv   =  0.7617343
#
#Estimation Method:                 MLE
#
#Data:                              dat.censored
#
#Sample Size:                       30
#
#Censoring Side:                    left
#
#Censoring Variable:                censored
#
#Censoring Level(s):                5
#
#Percent Censored:                  23.33333%
#
#Test Results:
#
#  Normal
#    Test Statistic:                W = 0.9372741
#    P-value:                       0.1704876
#
#  Gamma
#    Test Statistic:                W = 0.9613711
#    P-value:                       0.522329
#
#  Lognormal
#    Test Statistic:                W = 0.9292406
#    P-value:                       0.114511

#--------------------

# 2) Compare the results in 1) above with the results using the
#    ProUCL method.
#-----------------------------------------------------------------

distChooseCensored(dat.censored, censored, method = "proucl")

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:           Normal
#                                   Gamma
#                                   Lognormal
#
#Choice Method:                     ProUCL
#
#Type I Error per Test:             0.05
#
```

```
#Decision:                          Normal
#
#Estimated Parameter(s):            mean = 15.397584
#                                   sd   =  8.688302
#
#Estimation Method:                 mvue
#
#Data:                              dat.censored
#
#Sample Size:                       30
#
#Censoring Side:                    left
#
#Censoring Variable:                censored
#
#Censoring Level(s):                5
#
#Percent Censored:                  23.33333%
#
#ProUCL Sample Size:                23
#
#Test Results:
#
#  Normal
#    Shapiro-Wilk GOF
#      Test Statistic:              W = 0.861652
#      P-value:                     0.004457924
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:              D = 0.1714435
#      P-value:                     0.07794315
#
#  Gamma
#    ProUCL Anderson-Darling Gamma GOF
#      Test Statistic:              A = 0.3805556
#      P-value:                     >= 0.10
#    ProUCL Kolmogorov-Smirnov Gamma GOF
#      Test Statistic:              D = 0.1035271
#      P-value:                     >= 0.10
#
#  Lognormal
#    Shapiro-Wilk GOF
#      Test Statistic:              W = 0.9532604
#      P-value:                     0.3414187
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:              D = 0.115588
#      P-value:                     0.5899259

#--------------------

# Clean up
#---------

rm(dat, censored, dat.censored)
```

```
#====================================================================

# Check the assumption that the silver data stored in Helsel.Cohn.88.silver.df
# follows a lognormal distribution.
# Note that the small p-value and the shape of the Q-Q plot
# (an inverted S-shape) suggests that the log transformation is not quite strong
# enough to "bring in" the tails (i.e., the log-transformed silver data has tails
# that are slightly too long relative to a normal distribution).
# Helsel and Cohn (1988, p.2002) note that the gross outlier of 560 mg/L tends to
# make the shape of the data resemble a gamma distribution, but
# the distChooseCensored function decision is neither Gamma nor Lognormal,
# but instead Nonparametric.

# First create a lognormal Q-Q plot
#----------------------------------

dev.new()
with(Helsel.Cohn.88.silver.df,
  qqPlotCensored(Ag, Censored, distribution = "lnorm",
    points.col = "blue", add.line = TRUE))

#----------

# Now call the distChooseCensored function using the default settings.
#-------------------------------------------------------------------------

with(Helsel.Cohn.88.silver.df,
  distChooseCensored(Ag, Censored))

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   Shapiro-Francia
#
#Type I Error per Test:           0.05
#
#Decision:                        Nonparametric
#
#Data:                            Ag
#
#Sample Size:                     56
#
#Censoring Side:                  left
#
#Censoring Variable:              Censored
#
#Censoring Level(s):            0.1  0.2  0.3  0.5  1.0  2.0  2.5  5.0  6.0 10.0 20.0 25.0
#
```

```
#Percent Censored:                60.71429%
#
#Test Results:
#
#  Normal
#    Test Statistic:              W = 0.3065529
#    P-value:                     8.346126e-08
#
#  Gamma
#    Test Statistic:              W = 0.6254148
#    P-value:                     1.884155e-05
#
#  Lognormal
#    Test Statistic:              W = 0.8957198
#    P-value:                     0.03490314

#----------

# Clean up
#---------

graphics.off()

#====================================================================

# Chapter 15 of USEPA (2009) gives several examples of looking
# at normal Q-Q plots and estimating the mean and standard deviation
# for manganese concentrations (ppb) in groundwater at five background
# wells (USEPA, 2009, p. 15-10).  The Q-Q plot shown in Figure 15-4
# on page 15-13 clearly indicates that the Lognormal distribution
# is a good fit for these data.
# In EnvStats these data are stored in the data frame EPA.09.Ex.15.1.manganese.df.

# Here we will call the distChooseCensored function to determine
# whether the data appear to come from a normal, gamma, or lognormal
# distribution.
#
# Note that using the Probability Plot Correlation Coefficient method
# (equivalent to using the Shapiro-Francia method) yields a decision of
# Lognormal, but using the ProUCL method yields a decision of Gamma.
#--------------------------------------------------------------------


# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#...
```

```
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Use distChooseCensored with the probability plot correlation method,
# and for the gamma and lognormal distribution specify the
# mean and CV parameterization:
#------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  distChooseCensored(Manganese.ppb, Censored,
    choices = c("norm", "gamma", "lnormAlt"), method = "ppcc"))

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   PPCC
#
#Type I Error per Test:           0.05
#
#Decision:                        Lognormal
#
#Estimated Parameter(s):          mean = 23.003987
#                                 cv   =  2.300772
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Sample Size:                     25
#
#Censoring Side:                  left
#
#Censoring Variable:              Censored
#
#Censoring Level(s):              2 5
#
```

```
#Percent Censored:                24%
#
#Test Results:
#
#  Normal
#    Test Statistic:              r = 0.9147686
#    P-value:                     0.004662658
#
#  Gamma
#    Test Statistic:              r = 0.9844875
#    P-value:                     0.6836625
#
#  Lognormal
#    Test Statistic:              r = 0.9931982
#    P-value:                     0.9767731

#--------------------

# Repeat the above example using the ProUCL method.
#--------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  distChooseCensored(Manganese.ppb, Censored, method = "proucl"))

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:          Normal
#                                  Gamma
#                                  Lognormal
#
#Choice Method:                    ProUCL
#
#Type I Error per Test:            0.05
#
#Decision:                         Gamma
#
#Estimated Parameter(s):           shape =  1.284882
#                                  scale = 19.813413
#
#Estimation Method:                MLE
#
#Data:                             Manganese.ppb
#
#Sample Size:                      25
#
#Censoring Side:                   left
#
#Censoring Variable:               Censored
#
#Censoring Level(s):               2 5
#
#Percent Censored:                 24%
```

```
#
#ProUCL Sample Size:             19
#
#Test Results:
#
#  Normal
#    Shapiro-Wilk GOF
#      Test Statistic:            W = 0.7423947
#      P-value:                   0.0001862975
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:            D = 0.2768771
#      P-value:                   0.0004771155
#
#  Gamma
#    ProUCL Anderson-Darling Gamma GOF
#      Test Statistic:            A = 0.6857121
#      P-value:                   0.05 <= p < 0.10
#    ProUCL Kolmogorov-Smirnov Gamma GOF
#      Test Statistic:            D = 0.1830034
#      P-value:                   >= 0.10
#
#  Lognormal
#    Shapiro-Wilk GOF
#      Test Statistic:            W = 0.969805
#      P-value:                   0.7725528
#    Lilliefors (Kolmogorov-Smirnov) GOF
#      Test Statistic:            D = 0.138547
#      P-value:                   0.4385195

#====================================================================

## Not run:
# 1) Simulate 1000 trials where for each trial you:
#    a) Generate 30 observations from a Gamma distribution with
#       parameters mean = 10 and CV = 1.
#    b) Censor observations less than 5 (the 39th percentile).
#    c) Use distChooseCensored with the Shapiro-Francia method.
#    d) Use distChooseCensored with the ProUCL method.
#
#  2) Compare the proportion of times the
#     Normal vs. Gamma vs. Lognormal vs. Nonparametric distribution
#     is chosen for c) and d) above.
#--------------------------------------------------------------------

set.seed(58)
N <- 1000

Choose.fac <- factor(rep("", N), levels = c("Normal", "Gamma", "Lognormal", "Nonparametric"))
Choose.df <- data.frame(SW = Choose.fac, ProUCL = Choose.fac)

for(i in 1:N) {
  dat <- rgammaAlt(30, mean = 10, cv = 1)
  censored <- dat < 5
```

```
  dat[censored] <- 5
  Choose.df[i, "SW"]     <- distChooseCensored(dat, censored, method = "sw")$decision
 Choose.df[i, "ProUCL"] <- distChooseCensored(dat, censored, method = "proucl")$decision
}

summaryStats(Choose.df, digits = 0)

#                ProUCL(N) ProUCL(Pct) SW(N) SW(Pct)
#Normal              520          52   398      40
#Gamma               336          34   375      38
#Lognormal           105          10   221      22
#Nonparametric        39           4     6       1
#Combined           1000         100  1000     100

#--------------------


# Repeat above example for the Lognormal Distribution with mean=10 and CV = 1.
# In this case, 5 is the 34th percentile.
#-------------------------------------------------------------------------------

set.seed(297)
N <- 1000

Choose.fac <- factor(rep("", N), levels = c("Normal", "Gamma", "Lognormal", "Nonparametric"))
Choose.df <- data.frame(SW = Choose.fac, ProUCL = Choose.fac)

for(i in 1:N) {
  dat <- rlnormAlt(30, mean = 10, cv = 1)
  censored <- dat < 5
  dat[censored] <- 5
  Choose.df[i, "SW"]     <- distChooseCensored(dat, censored, method = "sf")$decision
 Choose.df[i, "ProUCL"] <- distChooseCensored(dat, censored, method = "proucl")$decision
}

summaryStats(Choose.df, digits = 0)

#               ProUCL(N) ProUCL(Pct) SW(N) SW(Pct)
#Normal              277          28    92       9
#Gamma               393          39   231      23
#Lognormal           190          19   624      62
#Nonparametric       140          14    53       5
#Combined           1000         100  1000     100

#--------------------


# Clean up
#---------

rm(N, Choose.fac, Choose.df, i, dat, censored)

## End(Not run)
```

distChooseCensored.object
S3 Class "distChooseCensored"

Description

Objects of S3 class "distChooseCensored" are returned by the **EnvStats** function distChooseCensored.

Details

Objects of S3 class "distChooseCensored" are lists that contain information about the candidate distributions, the estimated distribution parameters for each candidate distribution, and the test statistics and p-values associated with each candidate distribution.

Value

Required Components

The following components must be included in a legitimate list of class "distChooseCensored".

choices — a character vector containing the full names of the candidate distributions. (see Distribution.df).

method — a character string denoting which method was used.

decision — a character vector containing the full name of the chosen distribution.

alpha — a numeric scalar between 0 and 1 specifying the Type I error associated with each goodness-of-fit test.

distribution.parameters — a numeric vector containing the estimated parameters associated with the chosen distribution.

estimation.method — a character string indicating the method used to compute the estimated parameters associated with the chosen distribution. The value of this component will depend on the available estimation methods (see Distribution.df).

sample.size — a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit tests.

censoring.side — character string indicating whether the data are left- or right-censored.

censoring.levels — numeric scalar or vector indicating the censoring level(s).

percent.censored — numeric scalar indicating the percent of non-missing observations that are censored.

`test.results` a list with the same number of components as the number of elements in the component `choices`. The names of the list are the distribution abbreviations of the candidate distributions. (See the help file for `Distribution.df` for a list of distributions and their abbreviations.) Each component is an object of class `gofCensored` containing the results of the goodness-of-fit test for that particular hypothesized distribution.

`data.name` character string indicating the name of the data object used for the goodness-of-fit tests.

`censoring.name` character string indicating the name of the data object used to identify which values are censored.

Optional Components

The following components are included in the result of calling `distChooseCensored` when the argument `keep.data=TRUE`:

`data` numeric vector containing the data actually used for the goodness-of-fit tests (i.e., the original data without any missing or infinite values).

`censored` logical vector containing the censoring status for the data actually used for the goodness-of-fit tests (i.e., the original data without any missing or infinite values).

The following component is included in the result of calling `distChooseCensored` when missing (`NA`), undefined (`NaN`) and/or infinite (`Inf`, `-Inf`) values are present:

`bad.obs` numeric scalar indicating the number of missing (`NA`), undefined (`NaN`) and/or infinite (`Inf`, `-Inf`) values that were removed from the data object prior to choosing a distribution.

Methods

Generic functions that have methods for objects of class `"distChooseCensored"` include: `print`.

Note

Since objects of class `"distChooseCensored"` are lists, you may extract their components with the `$` and `[[` operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

`distChooseCensored`, `print.distChooseCensored`, Censored Data, Goodness-of-Fit Tests, `Distribution.df`.

Examples

```
# Create an object of class "distChooseCensored", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(598)

dat <- rgammaAlt(30, mean = 10, cv = 1)
censored <- dat < 5
dat[censored] <- 5

distChooseCensored.obj <- distChooseCensored(dat, censored,
  method = "sw", choices = c("norm", "gammaAlt", "lnormAlt"))

mode(distChooseCensored.obj)
#[1] "list"

class(distChooseCensored.obj)
#[1] "distChooseCensored"

names(distChooseCensored.obj)
# [1] "choices"                 "method"
# [3] "decision"                "alpha"
# [5] "distribution.parameters" "estimation.method"
# [7] "sample.size"             "censoring.side"
# [9] "censoring.levels"        "percent.censored"
#[11] "test.results"            "data"
#[13] "censored"                "data.name"
#[15] "censoring.name"

distChooseCensored.obj

#Results of Choosing Distribution
#--------------------------------
#
#Candidate Distributions:         Normal
#                                 Gamma
#                                 Lognormal
#
#Choice Method:                   Shapiro-Wilk
#
#Type I Error per Test:           0.05
#
#Decision:                        Gamma
#
#Estimated Parameter(s):          mean = 12.4911448
#                                 cv   =  0.7617343
#
#Estimation Method:               MLE
#
#Data:                            dat.censored
#
```

```
#Sample Size:                    30
#
#Censoring Side:                 left
#
#Censoring Variable:             censored
#
#Censoring Level(s):             5
#
#Percent Censored:               23.33333%
#
#Test Results:
#
#  Normal
#    Test Statistic:             W = 0.9372741
#    P-value:                    0.1704876
#
#  Gamma
#    Test Statistic:             W = 0.9613711
#    P-value:                    0.522329
#
#  Lognormal
#    Test Statistic:             W = 0.9292406
#    P-value:                    0.114511

#==========

# Extract the choices
#--------------------

distChooseCensored.obj$choices
#[1] "Normal"    "Gamma"     "Lognormal"

#==========

# Clean up
#---------
rm(dat, censored, distChooseCensored.obj)
```

`Distribution.df` *Data Frame Summarizing Available Probability Distributions and Estimation Methods*

Description

Data frame summarizing information about available probability distributions in R and the **EnvStats** package, and which distributions have associated functions for estimating distribution parameters.

Usage

```
Distribution.df
```

Format

A data frame with 35 rows corresponding to 35 different available probability distributions, and 25 columns containing information associated with these probability distributions.

`Name` a character vector containing the name of the probability distribution (see the column labeled **Name** in the table below).

`Type` a character vector indicating the type of distribution (see the column labeled **Type** in the table below). Possible values are `"Finite Discrete"`, `"Discrete"`, `"Continuous"`, and `"Mixed"`.

`Support.Min` a character vector indicating the minimum value the random variable can assume (see the column labeled **Range** in the table below). The reason this is a character vector instead of a numeric vector is because some distributions have a lower bound that depends on the value of a distribution parameter. For example, the minimum value for a Uniform distribution is given by the value of the parameter `min`.

`Support.Max` a character vector indicating the maximum value the random variable can assume (see the column labeled **Range** in the table below). The reason this is a character vector instead of a numeric vector is because some distributions have an upper bound that depends on the value of a distribution parameter. For example, the maximum value for a Uniform distribution is given by the value of the parameter `max`.

`Estimation.Method(s)` a character vector indicating the names of the methods available to estimate the distribution parameter(s) (see the column labeled **Estimation Method(s)** in the table below). Possible values include `"mle"` (maximum likelihood), `"mme"` (method of moments), `"mmue"` (method of moments based on the unbiased estimate of variance), `"mvue"` (minimum variance unbiased), `"qmle"` (quasi-mle), etc., or some combination of these. In cases where an estimator is more than one kind, a slash (`/`) is used to denote all methods covered by the single estimator. For example, for the Binomial distribution, the sample proportion is the maximum likelihood, method of moments, and minimum variance unbiased estimator, so this method is denoted as `"mle/mme/mvue"`. See the help files for the specific function listed under Estimating Distribution Parameters for an explanation of each of these estimation methods.

`Quantile.Estimation.Method(s)` a character vector indicating the names of the methods available to estimate the distribution quantiles. For many distributions, these are the same as `Estimation.Method(s)`. See the help files for the specific function listed under Estimating Distribution Quantiles for an explanation of each of these estimation methods.

`Prediction.Interval.Method(s)` a character vector indicating the names of the methods available to create prediction intervals. See the help files for the specific function listed under Prediction Intervals for an explanation of each of these estimation methods.

`Singly.Censored.Estimation.Method(s)` a character vector indicating the names of the methods available to estimate the distribution parameter(s) for Type I singly-censored data. See the help files for the specific function listed under *Estimating Distribution Parameters* in the help file for Censored Data for an explanation of each of these estimation methods.

`Multiply.Censored.Estimation.Method(s)` a character vector indicating the names of the methods available to estimate the distribution parameter(s) for Type I multiply-censored data. See the help files for the specific function listed under *Estimating Distribution Parameters* in the help file for Censored Data for an explanation of each of these estimation methods.

`Number.parameters` a numeric vector indicating the number of parameters associated with the distribution (see the column labeled **Parameters** in the table below).

`Parameter.1` the columns labeled `Parameter.1`, `Parameter.2`, ..., `Parameter.5` are character vectors containing the names of the distribution parameters (see the column labeled **Parameters** in the table below). If a distribution has n parameters and $n < 5$, then the columns labeled `Parameter.n+1`, ..., `Parameter.5` are empty. For example, the Normal distribution has only two parameters associated with it (`mean` and `sd`), so the fields in `Parameter.3`, `Parameter.4`, and `Parameter.5` are empty.

`Parameter.2` see `Parameter.1`

`Parameter.3` see `Parameter.1`

`Parameter.4` see `Parameter.1`

`Parameter.5` see `Parameter.1`

`Parameter.1.Min` the columns labeled `Parameter.1.Min`, `Parameter.2.Min`, ..., `Parameter.5.Min` are character vectors containing the minimum values that can be assumed by the distribution parameters (see the column labeled **Parameter Range(s)** in the table below).

The reason these are character vectors instead of numeric vectors is because some parameters have a lower bound of `0` but must be strictly bigger than `0` (e.g., the parameter `sd` for the Normal distribution), in which case the lower bound is `.Machine$double.eps`, which may vary from machine to machine. Also, some parameters have a lower bound that depends on the value of another parameter. For example, the parameter `max` for a Uniform distribution is bounded below by the value of the parameter `min`.

If a distribution has n parameters and $n < 5$, then the columns labeled `Parameter.n+1.Min`, ..., `Parameter.5.Min` have the missing value code (`NA`). For example, the Normal distribution has only two parameters associated with it (`mean` and `sd`) so the fields in `Parameter.3.Min`, `Parameter.4.Min`, and `Parameter.5.Min` have `NA`s in them.

`Parameter.2.Min` see `Parameter.1.Min`

`Parameter.3.Min` see `Parameter.1.Min`

`Parameter.4.Min` see `Parameter.1.Min`

`Parameter.5.Min` see `Parameter.1.Min`

`Parameter.1.Max` the columns labeled `Parameter.1.Max`, `Parameter.2.Max`, ..., `Parameter.5.Max` are character vectors containing the maximum values that can be assumed by the distribution parameters (see the column labeled **Parameter Range(s)** in the table below).

The reason these are character vectors instead of numeric vectors is because some parameters have an upper bound that depends on the value of another parameter. For example, the parameter `min` for a Uniform distribution is bounded above by the value of the parameter `max`.

If a distribution has n parameters and $n < 5$, then the columns labeled `Parameter.n+1.Max`, ..., `Parameter.5.Max` have the missing value code (`NA`). For example, the Normal distribution has only two parameters associated with it (`mean` and `sd`) so the fields in `Parameter.3.Max`, `Parameter.4.Max`, and `Parameter.5.Max` have `NA`s in them.

`Parameter.2.Max` see `Parameter.1.Max`

`Parameter.3.Max` see `Parameter.1.Max`

`Parameter.4.Max` see `Parameter.1.Max`

`Parameter.5.Max` see `Parameter.1.Max`

Details

The table below summarizes the probability distributions available in R and **EnvStats**. For each distribution, there are four associated functions for computing density values, percentiles, quantiles, and random numbers. The form of the names of these functions are d*abb*, p*abb*, q*abb*, and r*abb*, where *abb* is the abbreviated name of the distribution (see table below). These functions are described in the help file with the name of the distribution (see the first column of the table below). For example, the help file for Beta describes the behavior of `dbeta`, `pbeta`, `qbeta`, and `rbeta`.

For most distributions, there is also an associated function for estimating the distribution parameters, and the form of the names of these functions is e*abb*, where *abb* is the abbreviated name of the distribution (see table below). All of these functions are listed in the help file Estimating Distribution Parameters. For example, the function `ebeta` estimates the shape parameters of a Beta distribution based on a random sample of observations from this distribution.

For some distributions, there are functions to estimate distribution parameters based on Type I censored data. The form of the names of these functions is e*abb*`SinglyCensored` for singly censored data and e*abb*`MultiplyCensored` for multiply censored data. All of these functions are listed under the heading *Estimating Distribution Parameters* in the help file Censored Data.

Table 1a. Available Distributions: Name, Abbreviation, Type, and Range

Name	Abbreviation	Type	Range
Beta	`beta`	Continuous	$[0, 1]$
Binomial	`binom`	Finite Discrete	$[0, size]$ (integer)
Cauchy	`cauchy`	Continuous	$(-\infty, \infty)$
Chi	`chi`	Continuous	$[0, \infty)$
Chi-square	`chisq`	Continuous	$[0, \infty)$
Exponential	`exp`	Continuous	$[0, \infty)$
Extreme Value	`evd`	Continuous	$(-\infty, \infty)$
F	`f`	Continuous	$[0, \infty)$
Gamma	`gamma`	Continuous	$[0, \infty)$
Gamma (Alternative)	`gammaAlt`	Continuous	$[0, \infty)$
Generalized Extreme Value	`gevd`	Continuous	$(-\infty, \infty)$ for $shape = 0$ $(-\infty, location + \frac{scale}{shape}]$ for $shape > 0$

			$[location + \frac{scale}{shape}, \infty)$ for $shape < 0$
Geometric	`geom`	Discrete	$[0, \infty)$ (integer)
Hypergeometric	`hyper`	Finite Discrete	$[0, min(k, m)]$ (integer)
Logistic	`logis`	Continuous	$(-\infty, \infty)$
Lognormal	`lnorm`	Continuous	$(0, \infty)$
Lognormal (Alternative)	`lnormAlt`	Continuous	$(0, \infty)$
Lognormal Mixture	`lnormMix`	Continuous	$(0, \infty)$
Lognormal Mixture (Alternative)	`lnormMixAlt`	Continuous	$(0, \infty)$
Three-Parameter Lognormal	`lnorm3`	Continuous	$[threshold, \infty)$
Truncated Lognormal	`lnormTrunc`	Continuous	$[min, max]$
Truncated Lognormal (Alternative)	`lnormTruncAlt`	Continuous	$[min, max]$
Negative Binomial	`nbinom`	Discrete	$[0, \infty)$ (integer)
Normal	`norm`	Continuous	$(-\infty, \infty)$
Normal Mixture	`normMix`	Continuous	$(-\infty, \infty)$
Truncated Normal	`normTrunc`	Continuous	$[min, max]$
Pareto	`pareto`	Continuous	$[location, \infty)$

Poisson	`pois`	Discrete	$[0, \infty)$ (integer)
Student's t	`t`	Continuous	$(-\infty, \infty)$
Triangular	`tri`	Continuous	$[min, max]$
Uniform	`unif`	Continuous	$[min, max]$
Weibull	`weibull`	Continuous	$[0, \infty)$
Wilcoxon Rank Sum	`wilcox`	Finite Discrete	$[0, mn]$ (integer)
Zero-Modified Lognormal (Delta)	`zmlnorm`	Mixed	$[0, \infty)$
Zero-Modified Lognormal (Delta) (Alternative)	`zmlnormAlt`	Mixed	$[0, \infty)$
Zero-Modified Normal	`zmnorm`	Mixed	$(-\infty, \infty)$

Table 1b. Available Distributions: Name, Parameters, Parameter Default Values, Parameter Ranges, Estimation Method(s)

Name	Parameter(s)	Default Value(s)	Parameter Range(s)	Estimation Method(s)
Beta	`shape1`		$(0, \infty)$	mle, mme, mmue
	`shape2`		$(0, \infty)$	
	`ncp`	`0`	$(0, \infty)$	
Binomial	`size`		$[0, \infty)$	mle/mme/mvue
	`prob`		$[0, 1]$	
Cauchy	`location`	`0`	$(-\infty, \infty)$	
	`scale`	`1`	$(0, \infty)$	
Chi	`df`		$(0, \infty)$	
Chi-square	`df`		$(0, \infty)$	
	`ncp`	`0`	$(-\infty, \infty)$	
Exponential	`rate`	`1`	$(0, \infty)$	mle/mme

Extreme Value	`location`	`0`	$(-\infty, \infty)$	mle, mme, mmue, pwme
	`scale`	`1`	$(0, \infty)$	
F	`df1`		$(0, \infty)$	
	`df2`		$(0, \infty)$	
	`ncp`	`0`	$(0, \infty)$	
Gamma	`shape`		$(0, \infty)$	mle, bcmle, mme, mmue
	`scale`	`1`	$(0, \infty)$	
Gamma (Alternative)	`mean`		$(0, \infty)$	mle, bcmle, mme, mmue
	`cv`	`1`	$(0, \infty)$	
Generalized Extreme Value	`location`	`0`	$(-\infty, \infty)$	mle, pwme, tsoe
	`scale`	`1`	$(0, \infty)$	
	`shape`	`0`	$(-\infty, \infty)$	
Geometric	`prob`		$(0, 1)$	mle/mme, mvue
Hypergeometric	`m`		$[0, \infty)$	mle, mvue
	`n`		$[0, \infty)$	
	`k`		$[1, m + n]$	
Logistic	`location`	`0`	$(-\infty, \infty)$	mle, mme, mmue
	`scale`	`1`	$(0, \infty)$	
Lognormal	`meanlog`	`0`	$(-\infty, \infty)$	mle/mme, mvue
	`sdlog`	`1`	$(0, \infty)$	
Lognormal (Alternative)	`mean`	`exp(1/2)`	$(0, \infty)$	mle, mme, mmue,
	`cv`	`sqrt(exp(1)-1)`	$(0, \infty)$	mvue, qmle
Lognormal Mixture	`meanlog1`	`0`	$(-\infty, \infty)$	
	`sdlog1`	`1`	$(0, \infty)$	
	`meanlog2`	`0`	$(-\infty, \infty)$	
	`sdlog2`	`1`	$(0, \infty)$	
	`p.mix`	`0.5`	$[0, 1]$	
Lognormal Mixture (Alternative)	`mean1`	`exp(1/2)`	$(0, \infty)$	
	`cv1`	`sqrt(exp(1)-1)`	$(0, \infty)$	
	`mean2`	`exp(1/2)`	$(0, \infty)$	
	`cv2`	`sqrt(exp(1)-1)`	$(0, \infty)$	
	`p.mix`	`0.5`	$[0, 1]$	
Three-Parameter Lognormal	`meanlog`	`0`	$(-\infty, \infty)$	lmle, mme,
	`sdlog`	`1`	$(0, \infty)$	mmue, mmme,
	`threshold`	`0`	$(-\infty, \infty)$	royston.skew,

				zero.skew
Truncated Lognormal	meanlog	0	$(-\infty, \infty)$	
	sdlog	1	$(0, \infty)$	
	min	0	$[0, max)$	
	max	Inf	(min, ∞)	
Truncated Lognormal (Alternative)	mean	exp(1/2)	$(0, \infty)$	
	cv	sqrt(exp(1)-1)	$(0, \infty)$	
	min	0	$[0, max)$	
	max	Inf	(min, ∞)	
Negative Binomial	size		$[1, \infty)$	mle/mme, mvue
	prob		$(0, 1]$	
	mu		$(0, \infty)$	
Normal	mean	0	$(-\infty, \infty)$	mle/mme, mvue
	sd	1	$(0, \infty)$	
Normal Mixture	mean1	0	$(-\infty, \infty)$	
	sd1	1	$(0, \infty)$	
	mean2	0	$(-\infty, \infty)$	
	sd2	1	$(0, \infty)$	
	p.mix	0.5	$[0, 1]$	
Truncated Normal	mean	0	$(-\infty, \infty)$	
	sd	1	$(0, \infty)$	
	min	-Inf	$(-\infty, max)$	
	max	Inf	(min, ∞)	
Pareto	location		$(0, \infty)$	lse, mle
	shape	1	$(0, \infty)$	
Poisson	lambda		$(0, \infty)$	mle/mme/mvue
Student's t	df		$(0, \infty)$	
	ncp	0	$(-\infty, \infty)$	
Triangular	min	0	$(-\infty, max)$	
	max	1	(min, ∞)	
	mode	0.5	(min, max)	
Uniform	min	0	$(-\infty, max)$	mle, mme, mmue
	max	1	(min, ∞)	
Weibull	shape		$(0, \infty)$	mle, mme, mmue
	scale	1	$(0, \infty)$	

Wilcoxon Rank Sum	m		$[1, \infty)$	
	n		$[1, \infty)$	
Zero-Modified Lognormal (Delta)	meanlog	0	$(-\infty, \infty)$	mvue
	sdlog	1	$(0, \infty)$	
	p.zero	0.5	$[0, 1]$	
Zero-Modified Lognormal (Delta) (Alternative)	mean	exp(1/2)	$(0, \infty)$	mvue
	cv	sqrt(exp(1)-1)	$(0, \infty)$	
	p.zero	0.5	$[0, 1]$	
Zero-Modified Normal	mean	0	$(-\infty, \infty)$	mvue
	sd	1	$(0, \infty)$	
	p.zero	0.5	$[0, 1]$	

Source

The **EnvStats** package.

References

Millard, S.P. (2013). *EnvStats: An R Package for Environmental Statistics*. Springer, New York. https://link.springer.com/book/10.1007/978-1-4614-8456-1.

`ebeta` *Estimate Parameters of a Beta Distribution*

Description

Estimate the shape parameters of a beta distribution.

Usage

```
ebeta(x, method = "mle")
```

Arguments

x	numeric vector of observations. All observations must be between greater than 0 and less than 1.
method	character string specifying the method of estimation. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a beta distribution with parameters shape1=ν and shape2=ω.

Maximum Likelihood Estimation (method="mle")
The maximum likelihood estimators (mle's) of the shape parameters ν and ω are the solutions of the simultaneous equations:

$$\Psi(\hat{\nu}) - \Psi(\hat{\nu} + \hat{\omega}) = (1/n) \sum_{i=1}^{n} log(x_i)$$

$$\Psi(\hat{\nu}) - \Psi(\hat{\nu} + \hat{\omega}) = (1/n) \sum_{i=1}^{n} log(1 - x_i)$$

where $\Psi()$ is the digamma function (Forbes et al., 2011).

Method of Moments Estimators (method="mme")
The method of moments estimators (mme's) of the shape parameters ν and ω are given by (Forbes et al., 2011):

$$\hat{\nu} = \bar{x}\{[\bar{x}(1 - \bar{x})/s_m^2] - 1\}$$

$$\hat{\omega} = (1 - \bar{x})\{[\bar{x}(1 - \bar{x})/s_m^2] - 1\}$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i; \; s_m^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Method of Moments Estimators Based on the Unbiased Estimator of Variance (method="mmue")
These estimators are the same as the method of moments estimators except that the method of moments estimator of variance is replaced with the unbiased estimator of variance:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The beta distribution takes real values between 0 and 1. Special cases of the beta are the Uniform[0,1] when shape1=1 and shape2=1, and the arcsin distribution when shape1=0.5 and shape2=0.5. The arcsin distribution appears in the theory of random walks. The beta distribution is used in Bayesian analyses as a conjugate to the binomial distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

Beta.

Examples

```
# Generate 20 observations from a beta distribution with parameters
# shape1=2 and shape2=4, then estimate the parameters via
# maximum likelihood.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rbeta(20, shape1 = 2, shape2 = 4)
ebeta(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Beta
#
#Estimated Parameter(s):          shape1 =  5.392221
#                                 shape2 = 11.823233
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20

#==========

# Repeat the above, but use the method of moments estimators:

ebeta(dat, method = "mme")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Beta
#
#Estimated Parameter(s):          shape1 =  5.216311
#                                 shape2 = 11.461341
#
#Estimation Method:               mme
#
```

```
#Data:                          dat
#
#Sample Size:                   20

#==========

# Clean up
#---------
rm(dat)
```

---

ebinom *Estimate Parameter of a Binomial Distribution*

Description

Estimate p (the probability of "success") for a binomial distribution, and optionally construct a confidence interval for p.

Usage

```
ebinom(x, size = NULL, method = "mle/mme/mvue", ci = FALSE,
ci.type = "two-sided", ci.method = "score", correct = TRUE,
var.denom = "n", conf.level = 0.95, warn = TRUE)
```

Arguments

x	numeric or logical vector of observations. When size is not supplied, x must be a numeric vector of 0s ("failures") and 1s ("successes"), or else a logical vector of FALSE values ("failures") and TRUE values ("successes"). When size is supplied, x must be a non-negative integer containing the number of "successes" out of the number of trials indicated by size. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
size	positive integer indicating the of number of trials; size must be at least as large as the value of x.
method	character string specifying the method of estimation. The only possible value is "mle/mme/mvue" (maximum likelihood, method of moments, and minimum variance unbiased). See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating which method to use to construct the confidence interval. Possible values are "score" (the default), "exact", "adjusted Wald", and "Wald". This argument is ignored if ci=FALSE.

`correct` logical scalar indicating whether to use the continuity correction when `ci.method="score"` or `ci.method="Wald"`. The default value is `correct=TRUE`.

`var.denom` character string indicating what value to use in the denominator of the variance estimator when `ci.method="Wald"`. Possible values are `"n"` (the default) and `"n-1"`. This argument is ignored if `ci=FALSE`.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

`warn` a logical scalar indicating whether to issue a waning in the case when `ci=TRUE`, `ci.method="Wald"`, and any of the following conditions is true: the estimated proportion is less than 0.2, the estimated proportion is greater than 0.8, the number of successes or failures is less than 5. The default value is `warn=TRUE`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

If $\underline{x}$ is a vector of n observations from a binomial distribution with parameters `size=`1 and `prob=`p, then the sum of all the values in $\underline{x}$ is an observation from a binomial distribution with parameters `size=`n and `prob=`p.

If x is an observation from a binomial distribution with parameters `size=`n and `prob=`p, the maximum likelihood estimator (mle), method of moments estimator (mme), and minimum variance unbiased estimator (mvue) of p is simply x/n.

Confidence Intervals.

`ci.method="score"` The confidence interval for p based on the score method was developed by Wilson (1927) and is discussed by Newcombe (1998a), Agresti and Coull (1998), and Agresti and Caffo (2000). When `ci=TRUE` and `ci.method="score"`, the function `ebinom` calls the R function `prop.test` to compute the confidence interval. This method has been shown to provide the best performance (in terms of actual coverage matching assumed coverage) of all the methods provided here, although unlike the exact method, the actual coverage can fall below the assumed coverage.

`ci.method="exact"` The confidence interval for p based on the exact (Clopper-Pearson) method is discussed by Newcombe (1998a), Agresti and Coull (1998), and Zar (2010, pp.543-547). This is the method used in the R function `binom.test`. This method ensures the actual coverage is greater than or equal to the assumed coverage.

`ci.method="Wald"` The confidence interval for p based on the Wald method (with or without a correction for continuity) is the usual "normal approximation" method and is discussed by Newcombe (1998a), Agresti and Coull (1998), Agresti and Caffo (2000), and Zar (2010, pp.543-547). This method is **never** recommended but is included for historical purposes.

`ci.method="adjusted Wald"` The confidence interval for p based on the adjusted Wald method is discussed by Agresti and Coull (1998), Agresti and Caffo (2000), and Zar (2010, pp.543-547). This is a simple modification of the Wald method and performs surpringly well.

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", p, is the same on each trial. A binomial discrete random variable X is the number of "successes" in n independent trials. A special case of the binomial distribution occurs when $n = 1$, in which case X is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143). The binomial distribution is also used to compute an upper bound on the overall Type I error rate for deciding whether a facility or location is in compliance with some set standard. Assume the null hypothesis is that the facility is in compliance. If a test of hypothesis is conducted periodically over time to test compliance and/or several tests are performed during each time period, and the facility or location is always in compliance, and each single test has a Type I error rate of α, and the result of each test is independent of the result of any other test (usually not a reasonable assumption), then the number of times the facility is declared out of compliance when in fact it is in compliance is a binomial random variable with probability of "success" $p = \alpha$ being the probability of being declared out of compliance (see USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Agresti, A., and B.A. Coull. (1998). Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions. *The American Statistician*, **52**(2), 119–126.

Agresti, A., and B. Caffo. (2000). Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures. *The American Statistician*, **54**(4), 280–288.

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapters 2 and 15.

Cochran, W.G. (1977). *Sampling Techniques*. John Wiley and Sons, New York, Chapter 3.

Fisher, R.A., and F. Yates. (1963). *Statistical Tables for Biological, Agricultural, and Medical Research*. 6th edition. Hafner, New York, 146pp.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. Second Edition. John Wiley and Sons, New York, Chapters 1-2.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, Chapter 11.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 3.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Newcombe, R.G. (1998a). Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods. *Statistics in Medicine*, **17**, 857–872.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL, Chapter 4.

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.6-38.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 24.

See Also

Binomial, `prop.test`, `binom.test`, `ciBinomHalfWidth`, `ciBinomN`, `plotCiBinomDesign`.

Examples

```
# Generate 20 observations from a binomial distribution with
# parameters size=1 and prob=0.2, then estimate the 'prob' parameter.
# (Note: the call to set.seed simply allows you to reproduce this
# example. Also, the only parameter estimated is 'prob'; 'size' is
# specified in the call to ebinom.  The parameter 'size' is printed
# inorder to show all of the parameters associated with the
# distribution.)

set.seed(251)
dat <- rbinom(20, size = 1, prob = 0.2)
ebinom(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Binomial
#
#Estimated Parameter(s):          size = 20.0
#                                 prob =  0.1
#
#Estimation Method:               mle/mme/mvue for 'prob'
#
#Data:                            dat
#
#Sample Size:                     20
```

```
#----------------------------------------------------------------

# Generate one observation from a binomial distribution with
# parameters size=20 and prob=0.2, then estimate the "prob"
# parameter and compute a confidence interval:

set.seed(763)
dat <- rbinom(1, size=20, prob=0.2)
ebinom(dat, size = 20, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Binomial
#
#Estimated Parameter(s):          size = 20.00
#                                 prob =  0.35
#
#Estimation Method:               mle/mme/mvue for 'prob'
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         prob
#
#Confidence Interval Method:      Score normal approximation
#                                 (With continuity correction)
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 0.1630867
#                                 UCL = 0.5905104

#----------------------------------------------------------------

# Using the data from the last example, compare confidence
# intervals based on the various methods

ebinom(dat, size = 20, ci = TRUE,
  ci.method = "score", correct = TRUE)$interval$limits
#      LCL       UCL
#0.1630867 0.5905104


ebinom(dat, size = 20, ci = TRUE,
  ci.method = "score", correct = FALSE)$interval$limits
#      LCL       UCL
#0.1811918 0.5671457
```

```
ebinom(dat, size = 20, ci = TRUE,
  ci.method = "exact")$interval$limits
#      LCL       UCL
#0.1539092 0.5921885

ebinom(dat, size = 20, ci = TRUE,
  ci.method = "adjusted Wald")$interval$limits
#      LCL       UCL
#0.1799264 0.5684112


ebinom(dat, size = 20, ci = TRUE,
  ci.method = "Wald", correct = TRUE)$interval$limits
#      LCL       UCL
#0.1159627 0.5840373


ebinom(dat, size = 20, ci = TRUE,
  ci.method = "Wald", correct = FALSE)$interval$limits
#      LCL       UCL
#0.1409627 0.5590373


#----------------------------------------------------------------

# Use the cadmium data on page 8-6 of USEPA (1989b) to compute
# two-sided 95% confidence intervals for the probability of
# detection at background and compliance wells.  The data are
# stored in EPA.89b.cadmium.df.

EPA.89b.cadmium.df
#   Cadmium.orig Cadmium Censored  Well.type
#1           0.1   0.100    FALSE Background
#2          0.12   0.120    FALSE Background
#3           BDL   0.000     TRUE Background
#...
#86          BDL   0.000     TRUE Compliance
#87          BDL   0.000     TRUE Compliance
#88          BDL   0.000     TRUE Compliance

attach(EPA.89b.cadmium.df)

# Probability of detection at Background well:
#-----------------------------------------

ebinom(!Censored[Well.type=="Background"], ci=TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Binomial
#
```

```
#Estimated Parameter(s):          size = 24.0000000
#                                 prob =  0.3333333
#
#Estimation Method:               mle/mme/mvue for 'prob'
#
#Data:                            !Censored[Well.type == "Background"]
#
#Sample Size:                     24
#
#Confidence Interval for:         prob
#
#Confidence Interval Method:      Score normal approximation
#                                 (With continuity correction)
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 0.1642654
#                                 UCL = 0.5530745


# Probability of detection at Compliance well:
#---------------------------------------------

ebinom(!Censored[Well.type=="Compliance"], ci=TRUE)

#Results of Distribution Parameter Estimation
#---------------------------------------------
#
#Assumed Distribution:            Binomial
#
#Estimated Parameter(s):          size = 64.000
#                                 prob =  0.375
#
#Estimation Method:               mle/mme/mvue for 'prob'
#
#Data:                            !Censored[Well.type == "Compliance"]
#
#Sample Size:                     64
#
#Confidence Interval for:         prob
#
#Confidence Interval Method:      Score normal approximation
#                                 (With continuity correction)
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 0.2597567
#                                 UCL = 0.5053034
```

```
#----------------------------------------------------------

# Clean up
rm(dat)
detach("EPA.89b.cadmium.df")
```

---

ecdfPlot *Empirical Cumulative Distribution Function Plot*

Description

Produce an empirical cumulative distribution function plot.

Usage

```
ecdfPlot(x, discrete = FALSE,
  prob.method = ifelse(discrete, "emp.probs", "plot.pos"),
  plot.pos.con = 0.375, plot.it = TRUE, add = FALSE, ecdf.col = "black",
  ecdf.lwd = 3 * par("cex"), ecdf.lty = 1, curve.fill = FALSE,
  curve.fill.col = "cyan", ..., type = ifelse(discrete, "s", "l"),
  main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
discrete	logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).
prob.method	character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are plot.pos (plotting positions, the default if discrete=FALSE) and emp.probs (empirical probabilities, the default if discrete=TRUE). See the DETAILS section for more explanation.
plot.pos.con	numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is ignored if prob.method="emp.probs".
plot.it	logical scalar indicating whether to produce a plot or add to the current plot (see add) on the current graphics device. The default value is plot.it=TRUE.
add	logical scalar indicating whether to add the empirical cdf to the current plot (add=TRUE) or generate a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.
ecdf.col	a numeric scalar or character string determining the color of the empirical cdf line or points. The default value is ecdf.col=1. See the entry for col in the help file for par for more information.
ecdf.lwd	a numeric scalar determining the width of the empirical cdf line. The default value is ecdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.

`ecdf.lty` a numeric scalar determining the line type of the empirical cdf line. The default value is `ecdf.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`curve.fill` a logical scalar indicating whether to fill in the area below the empirical cdf curve with the color specified by `curve.fill.col`. The default value is `curve.fill=FALSE`.

`curve.fill.col` a numeric scalar or character string indicating what color to use to fill in the area below the empirical cdf curve. The default value is `curve.fill.col=5`. This argument is ignored if `curve.fill=FALSE`.

`type`, `main`, `xlab`, `ylab`, `xlim`, `ylim`, `...`
additional graphical parameters (see `lines` and `par`). In particular, the argument `type` specifies the kind of line type. By default, the function `ecdfPlot` plots a step function (`type="s"`) when `discrete=TRUE`, and plots a straight line between points (`type="l"`) when `discrete=FALSE`. The user may override these defaults by supplying the graphics parameter `type` (`type="s"` for a step function, `type="l"` for linear interpolation, `type="p"` for points only, etc.).

Details

The ***cumulative distribution function (cdf)*** of a random variable X is the function F such that

$$F(x) = Pr(X \leq x) \qquad (1)$$

for all values of x. That is, if $p = F(x)$, then p is the proportion of the population that is less than or equal to x, and x is called the p'th ***quantile***, or the $100p$'th percentile. A plot of quantiles on the x-axis (i.e., the possible value for the random variable X) vs. the fraction of the population less than or equal to that number on the y-axis is called the ***cumulative distribution function plot***, and the y-axis is usually labeled as the "cumulative probability" or "cumulative frequency".

When we have a sample of data from some population, we usually do not know what percentiles our observations correspond to because we do not know the form of the cumulative distribution function F, so we have to use the sample data to estimate the cdf F. An ***emprical cumulative distribution function (ecdf) plot***, also called a ***quantile plot***, is a plot of the observed quantiles (i.e., the ordered observations) on the x-axis vs. the estimated cumulative probabilities on the y-axis (Chambers et al., 1983, pp. 11-19; Cleveland, 1993, pp. 17-20; Cleveland, 1994, pp. 136-139; Helsel and Hirsch, 1992, pp. 21-24).

(Note: Some authors (e.g., Chambers et al., 1983, pp.11-16; Cleveland, 1993, pp.17-20) reverse the axes on a quantile plot, i.e., the observed order statistics from the random sample are on the y-axis and the estimated cumulative probabilities are on the x-axis.)

The ***empirical cumulative distribution function (ecdf)*** is an estimate of the cdf based on a random sample of n observations from the distribution. Let $x_1, x_2, \ldots, x_n$ denote the n observations, and let $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ denote the ordered observations (i.e., the order statistics). The cdf is usually estimated by either the ***empirical probabilities estimator*** or the ***plotting-position estimator***. The empirical probabilities estimator is given by:

$$\hat{F}[x_{(i)}] = \hat{p}_i = \frac{\#[x_j \leq x_{(i)}]}{n} \qquad (2)$$

where $\#[x_j \leq x_{(i)}]$ denotes the number of observations less than or equal to $x_{(i)}$. The plotting-position estimator is given by:

$$\hat{F}[x_{(i)}] = \hat{p}_i = \frac{i - a}{n - 2a + 1} \qquad (3)$$

where $0 \leq a \leq 1$ (Cleveland, 1993, p. 18; D'Agostino, 1986a, pp. 8,25).

For any value x such that $x_{(1)} < x < x_{(n)}$, the ecdf is usually defined as either a step function:

$$\hat{F}(x) = \hat{F}[x_{(i)}], \qquad x_{(i)} \leq x < x_{(i+1)} \qquad (4)$$

(e.g., D'Agostino, 1986a), or linear interpolation between order statistics is used:

$$\hat{F}(x) = (1 - r)\hat{F}[x_{(i)}] + r\hat{F}[x_{(i+1)}], \qquad x_{(i)} \leq x < x_{(i+1)} \qquad (5)$$

where

$$r = \frac{x - x_{(i)}}{x_{(i+1)} - x_{(i)}} \qquad (6)$$

(e.g., Chambers et al., 1983). For the step function version, the ecdf stays flat until it hits a value on the x-axis corresponding to one of the order statistics, then it makes a jump. For the linear interpolation version, the ecdf plot looks like lines connecting the points. By default, the function `ecdfPlot` uses the step function version when `discrete=TRUE`, and the linear interpolation version when `discrete=FALSE`. The user may override these defaults by supplying the graphics parameter `type` (`type="s"` for a step function, `type="l"` for linear interpolation, `type="p"` for points only, etc.).

The empirical probabilities estimator is intuitively appealing. This is the estimator used when `prob.method="emp.probs"`. The disadvantage of this estimator is that it implies the largest observed value is the maximum possible value of the distribution (i.e., the 100'th percentile). This may be satisfactory if the underlying distribution is known to be discrete, but it is usually not satisfactory if the underlying distribution is known to be continuous.

The plotting-position estimator with various values of a is often used when the goal is to produce a probability plot (see `qqPlot`) rather than an empirical cdf plot. It is used to compute the estimated expected values or medians of the order statistics for a probability plot. This is the estimator used when `prob.method="plot.pos"`. The argument `plot.pos.con` refers to the variable a. Based on certain principles from statistical theory, certain values of the constant a make sense for specific underlying distributions (see the help file for `qqPlot` for more information).

Because x is a random sample, the emprical cdf changes from sample to sample and the variability in these estimates can be dramatic for small sample sizes.

Value

`ecdfPlot` invisibly returns a list with the following components:

`Order.Statistics`
: numeric vector of the ordered observations.

`Cumulative.Probabilities`
: numeric vector of the associated plotting positions.

Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the characteristics of a set of data. It is easy to determine quartiles and the minimum and maximum values from such a plot. Also, ecdf plots allow you to assess local density: a higher density of observations occurs where the slope is steep.

Chambers et al. (1983, pp.11-16) plot the observed order statistics on the y-axis vs. the ecdf on the x-axis and call this a quantile plot.

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots (see `cdfPlot` and `cdfCompare`) to graphically assess whether a sample of observations comes from a particular distribution. The Kolmogorov-Smirnov goodness-of-fit test (see `gofTest`) is the statistical companion of this kind of comparison; it is based on the maximum vertical distance between the empirical cdf plot and the theoretical cdf plot. More often, however, quantile-quantile (Q-Q) plots are used instead of ecdf plots to graphically assess departures from an assumed distribution (see `qqPlot`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11-16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7-62.

See Also

`ppoints`, `cdfPlot`, `cdfCompare`, `qqPlot`, `ecdfPlotCensored`.

Examples

```
# Generate 20 observations from a normal distribution with
# mean=0 and sd=1 and create an ecdf plot.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rnorm(20)
dev.new()
ecdfPlot(x)

#----------

# Repeat the above example, but fill in the area under the
# empirical cdf curve.

dev.new()
```

```
ecdfPlot(x, curve.fill = TRUE)

#----------

# Repeat the above example, but plot only the points.

dev.new()
ecdfPlot(x, type = "p")

#----------

# Repeat the above example, but force a step function.

dev.new()
ecdfPlot(x, type = "s")

#----------

# Clean up
rm(x)

#--------------------------------------------------------------------------------

# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area.  These data are strored
# in the data frame EPA.94b.tccb.df.
#
# Create an empirical CDF plot for the reference area data.

dev.new()
with(EPA.94b.tccb.df,
  ecdfPlot(TcCB[Area == "Reference"], xlab = "TcCB (ppb)"))

#==========

# Clean up
#---------
graphics.off()
```

---

ecdfPlotCensored *Empirical Cumulative Distribution Function Plot Based on Type I Censored Data*

Description

Produce an empirical cumulative distribution function plot for Type I left-censored or right-censored data.

Usage

```
ecdfPlotCensored(x, censored, censoring.side = "left", discrete = FALSE,
  prob.method = "michael-schucany", plot.pos.con = 0.375, plot.it = TRUE,
  add = FALSE, ecdf.col = 1, ecdf.lwd = 3 * par("cex"), ecdf.lty = 1,
  include.cen = FALSE, cen.pch = ifelse(censoring.side == "left", 6, 2),
  cen.cex = par("cex"), cen.col = 4, ...,
  type = ifelse(discrete, "s", "l"), main = NULL, xlab = NULL, ylab = NULL,
  xlim = NULL, ylim = NULL)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
discrete	logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).
prob.method	character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany". The "nelson" method is only available for censoring.side="right". See the DETAILS section for more explanation.
plot.pos.con	numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".
plot.it	logical scalar indicating whether to produce a plot or add to the current plot (see add) on the current graphics device. The default value is plot.it=TRUE.
add	logical scalar indicating whether to add the empirical cdf to the current plot (add=TRUE) or generate a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.
ecdf.col	a numeric scalar or character string determining the color of the empirical cdf line or points. The default value is ecdf.col=1. See the entry for col in the help file for par for more information.

ecdf.lwd — a numeric scalar determining the width of the empirical cdf line. The default value is ecdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.

ecdf.lty — a numeric scalar determining the line type of the empirical cdf line. The default value is ecdf.lty=1. See the entry for lty in the help file for par for more information.

include.cen — logical scalar indicating whether to include censored values in the plot. The default value is include.cen=FALSE. If include.cen=TRUE, censored values are plotted using the plotting character indicated by the argument cen.pch (see below).

cen.pch — numeric scalar or character string indicating the plotting character to use to plot censored values. The default value is cen.pch=2 (hollow triangle pointing up) when censoring.side="right", and cen.pch=6 (hollow triangle pointing down) when censoring.side="left". See the help file for points for a list of other possible plotting characters. This argument is ignored if include.cen=FALSE.

cen.cex — numeric scalar that determines the size of the plotting character used to plot censored values. The default value is the current value of the cex graphics parameter. See the entry for cex in the help file for par for more information. This argument is ignored if include.cen=FALSE.

cen.col — numeric scalar or character string that determines the color of the plotting character used to plot censored values. The default value is cen.col=4. See the entry for col in the help file for par for more information. This argument is ignored if include.cen=FALSE.

type, main, xlab, ylab, xlim, ylim, ... — additional graphical parameters (see lines and par). In particular, the argument type specifies the kind of line type. By default, the function ecdfPlotCensored plots a step function (type="s") when discrete=TRUE, and plots a straight line between points (type="l") when discrete=FALSE. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).

Details

The function ecdfPlotCensored does exactly the same thing as ecdfPlot, except it calls the function ppointsCensored to compute the plotting positions (estimated cumulative probabilities) for the uncensored observations.

If plot.it=TRUE, the estimated cumulative probabilities for the uncensored observations are plotted against the uncensored observations. By default, the function ecdfPlotCensored plots a step function when discrete=TRUE, and plots a straight line between points when discrete=FALSE. The user may override these defaults by supplying the graphics parameter type (type="s" for a step function, type="l" for linear interpolation, type="p" for points only, etc.).

If include.cen=TRUE, censored observations are included on the plot as points. The arguments cen.pch, cen.cex, and cen.col control the appearance of these points.

In cases where x is a random sample, the emprical cdf will change from sample to sample and the variability in these estimates can be dramatic for small sample sizes. Caution must be used in interpreting the empirical cdf when a large percentage of the observations are censored.

Value

ecdfPlotCensored returns a list with the following components:

Order.Statistics
: numeric vector of the "ordered" observations.

Cumulative.Probabilities
: numeric vector of the associated plotting positions.

Censored
: logical vector indicating which of the ordered observations are censored.

Censoring.Side
: character string indicating whether the data are left- or right-censored. This is same value as the argument censoring.side.

Prob.Method
: character string indicating what method was used to compute the plotting positions. This is the same value as the argument prob.method.

Optional Component (only present when prob.method="michael-schucany" or prob.method="hirsch-stedinger"):

Plot.Pos.Con
: numeric scalar containing the value of the plotting position constant that was used. This is the same as the argument plot.pos.con.

Note

An empirical cumulative distribution function (ecdf) plot is a graphical tool that can be used in conjunction with other graphical tools such as histograms, strip charts, and boxplots to assess the characteristics of a set of data.

Censored observations complicate the procedures used to graphically explore data. Techniques from survival analysis and life testing have been developed to generalize the procedures for constructing plotting positions, empirical cdf plots, and q-q plots to data sets with censored observations (see ppointsCensored).

Empirical cumulative distribution function (ecdf) plots are often plotted with theoretical cdf plots (see cdfPlot and cdfCompareCensored) to graphically assess whether a sample of observations comes from a particular distribution. More often, however, quantile-quantile (Q-Q) plots are used instead (see qqPlot and qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11-16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7-62.

Gillespie, B.W., Q. Chen, H. Reichert, A. Franzblau, E. Hedgeman, J. Lepkowski, P. Adriaens, A. Demond, W. Luksemburg, and D.H. Garabrant. (2010). Estimating Population Distributions

When Some Data Are Below a Limit of Detection by Using a Reverse Kaplan-Meier Estimator. *Epidemiology* **21**(4), S64–S70.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997-2004.

Hirsch, R.M., and J.R. Stedinger. (1987). Plotting Positions for Historical Floods and Their Precision. *Water Resources Research* **23**(4), 715-727.

Kaplan, E.L., and P. Meier. (1958). Nonparametric Estimation From Incomplete Observations. *Journal of the American Statistical Association* **53**, 457-481.

Lee, E.T., and J.W. Wang. (2003). *Statistical Methods for Survival Data Analysis, Third Edition*. John Wiley & Sons, Hoboken, New Jersey, 513pp.

Michael, J.R., and W.R. Schucany. (1986). Analysis of Data from Censored Samples. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, 560pp, Chapter 11, 461-496.

Nelson, W. (1972). Theory and Applications of Hazard Plotting for Censored Failure Data. *Technometrics* **14**, 945-966.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. Chapter 15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

ppoints, ppointsCensored, ecdfPlot, qqPlot, qqPlotCensored, cdfPlot, cdfCompareCensored.

Examples

```
# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then generate an empirical cdf plot
# for the complete data set and the censored data set.  Note that the empirical
# cdf plot for the censored data set starts at the first ordered uncensored
# observation, and that for values of x > 18 the two emprical cdf plots are
# exactly the same.  This is because there is only one censoring level and
# no uncensored observations fall below the censored observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(333)
x <- rnorm(20, mean=20, sd=5)
censored <- x < 18

sum(censored)
#[1] 7
```

```
new.x <- x
new.x[censored] <- 18

dev.new()
ecdfPlot(x, xlim = range(pretty(x)),
  main = "Empirical CDF Plot for\nComplete Data Set")

dev.new()
ecdfPlotCensored(new.x, censored, xlim = range(pretty(x)),
  main="Empirical CDF Plot for\nCensored Data Set")

# Clean up
#---------
rm(x, censored, new.x)

#--------------------------------------------------------------------------------

# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# computing plotting positions based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring
# wells.  The data for this example are stored in
# EPA.09.Ex.15.1.manganese.df.  Here we will create an empirical
# CDF plot based on the Kaplan-Meier method.

EPA.09.Ex.15.1.manganese.df
#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#4       4 Well.1               21.6          21.6    FALSE
#5       5 Well.1                 <2           2.0     TRUE
#...
#21      1 Well.5               17.9          17.9    FALSE
#22      2 Well.5               22.7          22.7    FALSE
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  ecdfPlotCensored(Manganese.ppb, Censored,
    prob.method = "kaplan-meier", ecdf.col = "blue",
    main = "Empirical CDF of Manganese Data\nBased on Kaplan-Meier"))

#==========

# Clean up
#---------
graphics.off()
```

---

eevd *Estimate Parameters of an Extreme Value (Gumbel) Distribution*

Description

Estimate the location and scale parameters of an extreme value distribution, and optionally construct a confidence interval for one of the parameters.

Usage

```
eevd(x, method = "mle", pwme.method = "unbiased",
  plot.pos.cons = c(a = 0.35, b = 0), ci = FALSE,
  ci.parameter = "location", ci.type = "two-sided",
  ci.method = "normal.approx", conf.level = 0.95)
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default), "mme" (methods of moments), "mmue" (method of moments based on the unbiased estimator of variance), and "pwme" (probability-weighted moments). See the DETAILS section for more information on these estimation methods.
pwme.method	character string specifying what method to use to compute the probability-weighted moments when method="pwme". The possible values are "ubiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section in this help file and the help file for pwMoment for more information. This argument is ignored if method is not equal to "pwme".
plot.pos.cons	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="pwme" and pwme.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section in this help file and the help file for pwMoment for more information. This argument is ignored if method is not equal to "pwme" or if pwme.method="ubiased".
ci	logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is FALSE.
ci.parameter	character string indicating the parameter for which the confidence interval is desired. The possible values are "location" (the default) and "scale". This argument is ignored if ci=FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is "normal.approx" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from an extreme value distribution with parameters `location`=η and `scale`=θ.

Estimation

Maximum Likelihood Estimation (`method="mle"`)
The maximum likelihood estimators (mle's) of η and θ are the solutions of the simultaneous equations (Forbes et al., 2011):

$$\hat{\eta}_mle = \hat{\theta}_mle\, log[\frac{1}{n}\sum_{i=1}^{n} exp(\frac{-x_i}{\hat{\theta}_mle})]$$

$$\hat{\theta}_mle = \bar{x} - \frac{\sum_{i=1}^{n} x_i exp(\frac{-x_i}{\hat{\theta}_mle})}{\sum_{i=1}^{n} exp(\frac{-x_i}{\hat{\theta}_mle})}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

.

Method of Moments Estimation (`method="mme"`)
The method of moments estimators (mme's) of η and θ are given by (Johnson et al., 1995, p.27):

$$\hat{\eta}_{mme} = \bar{x} - \epsilon\hat{\theta}_{mme}$$

$$\hat{\theta}_{mme} = \frac{\sqrt{6}}{\pi} s_m$$

where ϵ denotes Euler's constant and s_m denotes the square root of the method of moments estimator of variance:

$$s_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Method of Moments Estimators Based on the Unbiased Estimator of Variance (`method="mmue"`)
These estimators are the same as the method of moments estimators except that the method of moments estimator of variance is replaced with the unbiased estimator of variance:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Probability-Weighted Moments Estimation (`method="pwme"`)
Greenwood et al. (1979) show that the relationship between the distribution parameters η and θ and the probability-weighted moments is given by:

$$\eta = M(1,0,0) - \epsilon\theta$$

$$\theta = \frac{M(1,0,0) - 2M(1,0,1)}{log(2)}$$

where $M(i,j,k)$ denotes the ijk'th probability-weighted moment and ϵ denotes Euler's constant. The probability-weighted moment estimators (pwme's) of η and θ are computed by simply replacing the $M(i,j,k)$'s in the above two equations with estimates of the $M(i,j,k)$'s (and for the estimate of η, replacing θ with its estimated value). See the help file for `pwMoment` for more information on how to estimate the $M(i,j,k)$'s. Also, see Landwehr et al. (1979) for an example of this method of estimation using the unbiased (U-statistic type) probability-weighted moment estimators. Hosking et al. (1985) note that this method of estimation using the U-statistic type probability-weighted moments is equivalent to Downton's (1966) linear estimates with linear coefficients.

Confidence Intervals

When `ci=TRUE`, an approximate $(1-\alpha)100\%$ confidence intervals for η can be constructed assuming the distribution of the estimator of η is approximately normally distributed. A two-sided confidence interval is constructed as:

$$[\hat{\eta} - t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\eta}},\ \hat{\eta} + t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\eta}}]$$

where $t(\nu, p)$ is the p'th quantile of Student's t-distribution with ν degrees of freedom, and the quantity

$$\hat{\sigma}_{\hat{\eta}}$$

denotes the estimated asymptotic standard deviation of the estimator of η.

Similarly, a two-sided confidence interval for θ is constructed as:

$$[\hat{\theta} - t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\theta}},\ \hat{\theta} + t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\theta}}]$$

One-sided confidence intervals for η and θ are computed in a similar fashion.

Maximum Likelihood (`method="mle"`)

Downton (1966) shows that the estimated asymptotic variances of the mle's of η and θ are given by:

$$\hat{\sigma}^2_{\hat{\eta}_m le} = \frac{\hat{\theta}_m le^2}{n}[1 + \frac{6(1-\epsilon)^2}{\pi^2}] = \frac{1.10867\hat{\theta}_m le^2}{n}$$

$$\hat{\sigma}^2_{\hat{\theta}_m le} = \frac{6}{\pi^2}\frac{\hat{\theta}_m le^2}{n} = \frac{0.60793\hat{\theta}_m le^2}{n}$$

where ϵ denotes Euler's constant.

Method of Moments (`method="mme"` or `method="mmue"`)

Tiago de Oliveira (1963) and Johnson et al. (1995, p.27) show that the estimated asymptotic variance of the mme's of η and θ are given by:

$$\hat{\sigma}^2_{\hat{\eta}_m me} = \frac{\hat{\theta}_m me^2}{n}[\frac{\pi^2}{6} + \frac{\epsilon^2}{4}(\beta_2 - 1) - \frac{\pi\epsilon\sqrt{\beta_1}}{\sqrt{6}}] = \frac{1.1678\hat{\theta}_m me^2}{n}$$

$$\hat{\sigma}^2_{\hat{\theta}_m me} = \frac{\hat{\theta}_m le^2}{n}\frac{(\beta_2 - 1)}{4} = \frac{1.1\hat{\theta}_m me^2}{n}$$

where the quantities

$$\sqrt{\beta_1},\ \beta_2$$

denote the skew and kurtosis of the distribution, and ϵ denotes Euler's constant.

The estimated asymptotic variances of the mmue's of η and θ are the same, except replace the mme of θ in the above equations with the mmue of θ.

Probability-Weighted Moments (`method="pwme"`)
As stated above, Hosking et al. (1985) note that this method of estimation using the U-statistic type probability-weighted moments is equivalent to Downton's (1966) linear estimates with linear coefficients. Downton (1966) provides exact values of the variances of the estimates of location and scale parameters for the smallest extreme value distribution. For the largest extreme value distribution, the formula for the estimate of scale is the same, but the formula for the estimate of location must be modified. Thus, Downton's (1966) equation (3.4) is modified to:

$$\hat{\eta}_p wme = \frac{(n-1)log(2) + (n+1)\epsilon}{n(n-1)log(2)} v - \frac{2\epsilon}{n(n-1)log(2)} w$$

where ϵ denotes Euler's constant, and v and w are defined in Downton (1966, p.8). Using Downton's (1966) equations (3.9)-(3.12), the exact variance of the pwme of η can be derived. Note that when `method="pwme"` and `pwme.method="plotting.position"`, these are only the asymptotically correct variances.

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Note

There are three families of extreme value distributions. The one described here is the Type I, also called the Gumbel extreme value distribution or simply Gumbel distribution. The name "extreme value" comes from the fact that this distribution is the limiting distribution (as n approaches infinity) of the greatest value among n independent random variables each having the same continuous distribution.

The Gumbel extreme value distribution is related to the exponential distribution as follows. Let Y be an exponential random variable with parameter `rate`=λ. Then $X = \eta - log(Y)$ has an extreme value distribution with parameters `location`=η and `scale`=$1/\lambda$.

The distribution described above and assumed by `eevd` is the *largest* extreme value distribution. The smallest extreme value distribution is the limiting distribution (as n approaches infinity) of the smallest value among n independent random variables each having the same continuous distribution. If X has a largest extreme value distribution with parameters `location`=η and `scale`=θ, then $Y = -X$ has a smallest extreme value distribution with parameters `location`=$-\eta$ and `scale`=θ. The smallest extreme value distribution is related to the Weibull distribution as follows. Let Y be a Weibull random variable with parameters `shape`=β and `scale`=α. Then $X = log(Y)$ has a smallest extreme value distribution with parameters `location`=$log(\alpha)$ and `scale`=$1/\beta$.

The extreme value distribution has been used extensively to model the distribution of streamflow, flooding, rainfall, temperature, wind speed, and other meteorological variables, as well as material strength and life data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Castillo, E. (1988). *Extreme Value Theory in Engineering*. Academic Press, New York, pp.184–198.

Downton, F. (1966). Linear Estimates of Parameters in the Extreme Value Distribution. *Technometrics* **8**(1), 3–17.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Greenwood, J.A., J.M. Landwehr, N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments: Definition and Relation to Parameters of Several Distributions Expressible in Inverse Form. *Water Resources Research* **15**(5), 1049–1054.

Hosking, J.R.M., J.R. Wallis, and E.F. Wood. (1985). Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments. *Technometrics* **27**(3), 251–261.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

Landwehr, J.M., N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments Compared With Some Traditional Techniques in Estimating Gumbel Parameters and Quantiles. *Water Resources Research* **15**(5), 1055–1064.

Tiago de Oliveira, J. (1963). Decision Results for the Parameters of the Extreme Value (Gumbel) Distribution Based on the Mean and Standard Deviation. *Trabajos de Estadistica* **14**, 61–81.

See Also

Extreme Value Distribution, Euler's Constant.

Examples

```
# Generate 20 observations from an extreme value distribution with
# parameters location=2 and scale=1, then estimate the parameters
# and construct a 90% confidence interval for the location parameter.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- revd(20, location = 2)
eevd(dat, ci = TRUE, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Extreme Value
#
#Estimated Parameter(s):          location = 1.9684093
#                                 scale    = 0.7481955
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20
#
```

```
#Confidence Interval for:         location
#
#Confidence Interval Method:      Normal Approximation
#                                 (t Distribution)
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                90%
#
#Confidence Interval:             LCL = 1.663809
#                                 UCL = 2.273009

#----------

#Compare the values of the different types of estimators:

eevd(dat, method = "mle")$parameters
# location     scale
#1.9684093 0.7481955

eevd(dat, method = "mme")$parameters
# location     scale
#1.9575980 0.8339256

eevd(dat, method = "mmue")$parameters
# location     scale
#1.9450932 0.8555896

eevd(dat, method = "pwme")$parameters
# location     scale
#1.9434922 0.8583633

#----------

# Clean up
#---------
rm(dat)
```

---

eexp *Estimate Rate Parameter of an Exponential Distribution*

Description

Estimate the rate parameter of an exponential distribution, and optionally construct a confidence interval for the rate parameter.

Usage

```
eexp(x, method = "mle/mme", ci = FALSE, ci.type = "two-sided",
  ci.method = "exact", conf.level = 0.95)
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Currently the only possible value is "mle/mme" (maximum likelihood/method of moments; the default). See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is "exact" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from an exponential distribution with parameter rate=λ.

Estimation

The maximum likelihood estimator (mle) of λ is given by:

$$\hat{\lambda}_{mle} = \frac{1}{\bar{x}}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

(Forbes et al., 2011). That is, the mle is the reciprocal of the sample mean.

Sometimes the exponential distribution is parameterized with a scale parameter instead of a rate parameter. The scale parameter is the reciprocal of the rate parameter, and the sample mean is both the mle and the minimum variance unbiased estimator (mvue) of the scale parameter.

Confidence Interval

When ci=TRUE, an exact $(1-\alpha)100\%$ confidence intervals for λ can be constructed based on the relationship between the exponential distribution, the gamma distribution, and the chi-square distribution. An exponential distribution with parameter rate=λ is equivalent to a gamma distribution with parameters shape=1 and scale=$1/\lambda$. The sum of n iid gamma random variables with parameters shape=1 and scale=$1/\lambda$ is a gamma random variable with parameters shape=n and scale=$1/\lambda$. Finally, a gamma distribution with parameters shape=n and scale=$1/\lambda$ is equivalent to 0.5 times a chi-square distribution with degrees of freedom df=$2n$. Thus, the quantity $2n\bar{x}$ has a chi-square distribution with degrees of freedom df=$2n$.

A two-sided $(1-\alpha)100\%$ confidence interval for λ is therefore constructed as:

$$[\frac{\chi^2(2n, \alpha/2)}{2n\bar{x}}, \frac{chi^2(2n, 1-\alpha/2)}{2n\bar{x}}]$$

where $\chi^2(\nu, p)$ is the p'th quantile of a chi-square distribution with ν degrees of freedom.

One-sided confidence intervals are computed in a similar fashion.

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Note

The exponential distribution is a special case of the gamma distribution, and takes on positive real values. A major use of the exponential distribution is in life testing where it is used to model the lifetime of a product, part, person, etc.

The exponential distribution is the only continuous distribution with a "lack of memory" property. That is, if the lifetime of a part follows the exponential distribution, then the distribution of the time until failure is the same as the distribution of the time until failure given that the part has survived to time t.

The exponential distribution is related to the double exponential (also called Laplace) distribution, and to the extreme value distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

Exponential.

Examples

```
# Generate 20 observations from an exponential distribution with parameter
# rate=2, then estimate the parameter and construct a 90% confidence interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rexp(20, rate = 2)
eexp(dat, ci=TRUE, conf = 0.9)
```

```
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Exponential
#
#Estimated Parameter(s):          rate = 2.260587
#
#Estimation Method:               mle/mme
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         rate
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                90%
#
#Confidence Interval:             LCL = 1.498165
#                                 UCL = 3.151173

#----------

# Clean up
#---------
rm(dat)
```

---

egamma *Estimate Parameters of Gamma Distribution*

Description

Estimate the shape and scale parameters (or the mean and coefficient of variation) of a Gamma distribution.

Usage

```
egamma(x, method = "mle", ci = FALSE,
  ci.type = "two-sided", ci.method = "normal.approx",
  normal.approx.transform = "kulkarni.powar", conf.level = 0.95)

egammaAlt(x, method = "mle", ci = FALSE,
  ci.type = "two-sided", ci.method = "normal.approx",
  normal.approx.transform = "kulkarni.powar", conf.level = 0.95)
```

Arguments

x	numeric vector of non-negative observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
method	character string specifying the method of estimation. The possible values are: "mle" (maximum likelihood; the default), "bcmle" (bias-corrected mle), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating which method to use to construct the confidence interval. Possible values are "normal.approx" (the default), "profile.likelihood", "chisq.approx", and "chisq.adj". This argument is ignored if ci=FALSE.
normal.approx.transform	character string indicating which power transformation to use when ci.method="normal.approx". Possible values are "kulkarni.powar" (the default), "cube.root", and "fourth.root". See the DETAILS section for more informaiton. This argument is ignored if ci=FALSE or ci.method="chisq.approx".
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a random sample of n observations from a gamma distribution with parameters shape=κ and scale=θ. The relationship between these parameters and the mean (mean=μ) and coefficient of variation (cv=τ) of this distribution is given by:

$$\kappa = \tau^{-2} \quad (1)$$

$$\theta = \mu/\kappa \quad (2)$$

$$\mu = \kappa\,\theta \quad (3)$$

$$\tau = \kappa^{-1/2} \quad (4)$$

The function egamma returns estimates of the shape and scale parameters. The function egammaAlt returns estimates of the mean (μ) and coefficient of variation (cv) based on the estimates of the shape and scale parameters.

Estimation

Maximum Likelihood Estimation (`method="mle"`)
The maximum likelihood estimators (mle's) of the shape and scale parameters κ and θ are solutions of the simultaneous equations:

$$\hat{\kappa}_{mle} = \frac{1}{n}\sum_{i=1}^{n} log(x_i) - log(\bar{x}) = \psi(\hat{\kappa}_{mle}) - log(\hat{\kappa}_{mle}) \qquad (5)$$

$$\hat{\theta}_{mle} = \bar{x}/\hat{\kappa}_{mle} \qquad (6)$$

where ψ denotes the digamma function, and $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (7)$$

(Forbes et al., 2011, chapter 22; Johnson et al., 1994, chapter 17).

Bias-Corrected Maximum Likelihood Estimation (`method="bcmle"`)
The "bias-corrected" maximum likelihood estimator of the shape parameter is based on the suggestion of Anderson and Ray (1975; see also Johnon et al., 1994, p.366 and Singh et al., 2010b, p.48), who noted that the bias of the maximum likelihood estimator of the shape parameter can be considerable when the sample size is small. This estimator is given by:

$$\hat{\kappa}_{bcmle} = \frac{n-3}{n}\hat{\kappa}_{mle} + \frac{2}{3n} \qquad (8)$$

When `method="bcmle"`, Equation (6) above is modified so that the estimate of the scale paramter is based on the "bias-corrected" maximum likelihood estimator of the shape parameter:

$$\hat{\theta}_{bcmle} = \bar{x}/\hat{\kappa}_{bcmle} \qquad (9)$$

Method of Moments Estimation (`method="mme"`)
The method of moments estimators (mme's) of the shape and scale parameters κ and θ are:

$$\hat{\kappa}_{mme} = (\bar{x}/s_m)^2 \qquad (10)$$

$$\hat{\theta}_{mme} = s_m^2/\bar{x} \qquad (11)$$

where s_m^2 denotes the method of moments estimator of variance:

$$s_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (12)$$

Method of Moments Estimation Based on the Unbiased Estimator of Variance (`method="mmue"`)
The method of moments estimators based on the unbiased estimator of variance are exactly the same as the method of moments estimators, except that the method of moments estimator of variance is replaced with the unbiased estimator of variance:

$$\hat{\kappa}_{mmue} = (\bar{x}/s)^2 \qquad (13)$$

$$\hat{\theta}_{mmue} = s^2/\bar{x} \qquad (14)$$

where s^2 denotes the unbiased estimator of variance:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (15)$$

Confidence Intervals

This section discusses how confidence intervals for the mean μ are computed.

Normal Approximation (`ci.method="normal.approx"`)

The normal approximation method is based on the method of Kulkarni and Powar (2010), who use a power transformation of the the original data to approximate a sample from a normal distribuiton, compute the confidence interval for the mean on the transformed scale using the usual formula for a confidence interval for the mean of a normal distribuiton, and then tranform the limits back to the original space using equations based on the expected value of a gamma random variable raised to a power.

The particular power used for the normal approximation is defined by the argument `normal.approx.transform`. The value `normal.approx.transform="cube.root"` uses the cube root transformation suggested by Wilson and Hilferty (1931), and the value `"fourth.root"` uses the fourth root transformation suggested by Hawkins and Wixley (1986). The default value `"kulkarni.powar"` uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010), who show this method performs the best in terms of maintining coverage and minimizing confidence interval width compared to eight other methods. The "optimum" power p is determined by:

$$\begin{array}{ll} p = -0.0705 - 0.178\hat{\kappa} + 0.475\sqrt{\hat{\kappa}} & \text{if } \hat{\kappa} \leq 1.5 \\ p = 0.246 & \text{if } \hat{\kappa} > 1.5 \end{array} \qquad (16)$$

where $\hat{\kappa}$ denotes the estimate of the shape parameter. Kulkarni and Powar (2010) derived this equation by determining what power transformation yields a skew closest to 0 and a kurtosis closest to 3 for a gamma random variable with a given shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power to use to induce approximate normality, for the functions `egamma` and `egammaAlt` the power is based on whatever estimate of shape is used (e.g., `method="mle"`, `method="bcmle"`, etc.).

Likelihood Profile (`ci.method="profile.likelihood"`)

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean μ while treating the coefficient of variation τ as a nuisance parameter.

The likelihood function is given by:

$$L(\mu, \tau|\underline{x}) = \prod_{i=1}^{n} \frac{x_i^{\kappa-1} e^{-x_i/\theta}}{\theta^{\kappa}\Gamma(\kappa)} \qquad (17)$$

where κ, θ, μ, and τ are defined in Equations (1)-(4) above, and $\Gamma(t)$ denotes the Gamma function evaluated at t.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and coefficient of variation by (μ^*, τ^*). The likelihood ratio test statistic (G^2) of the hypothesis $H_0 : \mu = \mu_0$ (where μ_0 is a fixed value) equals the drop in $2log(L)$ between the "full" model and the reduced model with μ fixed at μ_0, i.e.,

$$G^2 = 2\{log[L(\mu^*, \tau^*)] - log[L(\mu_0, \tau_0^*)]\} \qquad (18)$$

where τ_0^* is the maximum likelihood estimate of τ for the reduced model (i.e., when $\mu = \mu_0$). Under the null hypothesis, the test statistic G^2 follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function L_1 for the mean μ, which is obtained from the usual likelihood function by maximizing over the parameter τ, i.e.,

$$L_1(\mu) = max_\tau L(\mu, \tau) \qquad (19)$$

Then we have

$$G^2 = 2\{log[L_1(\mu^*)] - log[L_1(\mu_0)]\} \qquad (20)$$

A two-sided $(1 - \alpha)100\%$ confidence interval for the mean μ consists of all values of μ_0 for which the test is not significant at level $alpha$:

$$\mu_0 : G^2 \leq \chi^2_{1,1-\alpha} \qquad (21)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-squared distribution with ν degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1 - \alpha$ in Equation (21) is replaced with $1 - 2\alpha$.

Chi-Square Approximation (`ci.method="chisq.approx"`)
This method is based on the relationship between the sample mean of the gamma distribution and the chi-squared distribution (Grice and Bain, 1980):

$$\frac{2n\bar{x}}{\theta} \sim \chi^2_{2n\kappa} \qquad (22)$$

Therefore, an exact one-sided upper $(1 - \alpha)100\%$ confidence interval for the mean μ is given by:

$$[0, \frac{2n\bar{x}\kappa}{\chi^2_{2n\kappa,\alpha}}] \qquad (23)$$

an exact one-sided lower $(1 - \alpha)100\%$ confidence interval is given by:

$$[\frac{2n\bar{x}\kappa}{\chi^2_{2n\kappa,1-\alpha}}, \infty] \qquad (24)$$

and a two-sided $(1 - \alpha)100\%$ confidence interval is given by:

$$[\frac{2n\bar{x}\kappa}{\chi^2_{2n\kappa,1-\alpha/2}}, \frac{2n\bar{x}\kappa}{\chi^2_{2n\kappa,\alpha/2}}] \qquad (25)$$

Because this method is exact only when the shape parameter κ is known, the method used here is called the "chi-square approximation" method because the estimate of the shape parameter, $\hat{\kappa}$, is used in place of κ in Equations (23)-(25) above. The Chi-Square Approximation method is **not** the method proposed by Grice and Bain (1980) in which the confidence interval is adjusted based on adjusting for the fact that the shape parameter κ is estimated (see the explanation of the Chi-Square Adjusted method below). The Chi-Square Approximation method used by `egamma` and `egammaAlt` is equivalent to the "approximate gamma" method of *ProUCL* (USEPA, 2015, equation (2-34), p.62).

Chi-Square Adjusted (`ci.method="chisq.adj"`)
This is the same method as the Chi-Square Approximation method discussed above, execpt that the value of α is adjusted to account for the fact that the shape parameter κ is estimated rather than known. Grice and Bain (1980) performed Monte Carlo simulations to determine how to adjust α and the values in their Table 2 are given in the matrix `Grice.Bain.80.mat`. This method requires that the sample size n is at least 5 and the confidence level is between 75% and 99.5% (except when $n = 5$, in which case the confidence level must be less than 99%). For values of the sample size n and/or α that are not listed in the table, linear interpolation is used (when the sample size n is greater than 40, linear interpolation on $1/n$ is used, as recommended by Grice and Bain (1980)). The Chi-Square Adjusted method used by `egamma` and `egammaAlt` is equivalent to the "adjusted gamma" method of *ProUCL* (USEPA, 2015, equation (2-35), p.63).

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Warning

When `ci=TRUE` and `ci.method="normal.approx"`, it is possible for the lower confidence limit based on the transformed data to be less than 0. In this case, the lower confidence limit on the original scale is set to 0 and a warning is issued stating that the normal approximation is not accurate in this case.

Note

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter a tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribuiton instead.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Anderson, C.W., and W.D. Ray. (1975). Improved Maximum Likelihood Estimators for the Gamma Distribution. *Communications in Statistics*, **4**, 437–448.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). *Statistical Distributions, Fourth Edition*. John Wiley and Sons, Hoboken, NJ.

Grice, J.V., and L.J. Bain. (1980). Inferences Concerning the Mean of the Gamma Distribution. *Journal of the American Statistician*, **75**, 929-933.

Hawkins, D. M., and R.A.J. Wixley. (1986). A Note on the Transformation of Chi-Squared Variables to Normality. *The American Statistician*, **40**, 296–298.

Johnson, N.L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, Chapter 17.

Kulkarni, H.V., and S.K. Powar. (2010). A New Method for Interval Estimation of the Mean of the Gamma Distribution. *Lifetime Data Analysis*, **16**, 431–447.

Singh, A., A.K. Singh, and R.J. Iaci. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C.

Wilson, E.B., and M.M. Hilferty. (1931). The Distribution of Chi-Squares. *Proceedings of the National Academy of Sciences*, **17**, 684–688.

See Also

GammaDist, estimate.object, eqgamma, predIntGamma, tolIntGamma.

Examples

```
# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then estimate the parameters.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
egamma(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 2.203862
#                                 scale = 2.174928
#
#Estimation Method:               mle
#
```

```
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Optimum Power Normal Approximation
#                                 of Kulkarni & Powar (2010)
#                                 using mle of 'shape'
#
#Normal Transform Power:          0.246
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 3.361652
#                                 UCL = 6.746794

# Clean up
rm(dat)

#==========================================================================

# Using the reference area TcCB data in EPA.94b.tccb.df, assume a
# gamma distribution, estimate the parameters based on the
# bias-corrected mle of shape, and compute a one-sided upper 90%
# confidence interval for the mean.

#----------
# First test to see whether the data appear to follow a gamma
# distribution.

with(EPA.94b.tccb.df,
  gofTest(TcCB[Area == "Reference"], dist = "gamma",
    est.arg.list = list(method = "bcmle"))
)

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Shapiro-Wilk GOF Based on
#                                 Chen & Balakrisnan (1995)
#
#Hypothesized Distribution:       Gamma
#
#Estimated Parameter(s):          shape = 4.5695247
#                                 scale = 0.1309788
#
#Estimation Method:               bcmle
#
#Data:                            TcCB[Area == "Reference"]
#
```

```
#Sample Size:                     47
#
#Test Statistic:                  W = 0.9703827
#
#Test Statistic Parameter:        n = 47
#
#P-value:                         0.2739512
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Gamma Distribution.

#----------
# Now estimate the paramters and compute the upper confidence limit.

with(EPA.94b.tccb.df,
  egamma(TcCB[Area == "Reference"], method = "bcmle", ci = TRUE,
    ci.type = "upper", conf.level = 0.9)
)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 4.5695247
#                                 scale = 0.1309788
#
#Estimation Method:               bcmle
#
#Data:                            TcCB[Area == "Reference"]
#
#Sample Size:                     47
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Optimum Power Normal Approximation
#                                 of Kulkarni & Powar (2010)
#                                 using bcmle of 'shape'
#
#Normal Transform Power:          0.246
#
#Confidence Interval Type:        upper
#
#Confidence Level:                90%
#
#Confidence Interval:             LCL = 0.0000000
#                                 UCL = 0.6561838

#----------------------------------------------------------------

# Repeat the above example but use the alternative parameterization.

with(EPA.94b.tccb.df,
```

```
  egammaAlt(TcCB[Area == "Reference"], method = "bcmle", ci = TRUE,
    ci.type = "upper", conf.level = 0.9)
)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          mean = 0.5985106
#                                 cv   = 0.4678046
#
#Estimation Method:               bcmle of 'shape'
#
#Data:                            TcCB[Area == "Reference"]
#
#Sample Size:                     47
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Optimum Power Normal Approximation
#                                 of Kulkarni & Powar (2010)
#                                 using bcmle of 'shape'
#
#Normal Transform Power:          0.246
#
#Confidence Interval Type:        upper
#
#Confidence Level:                90%
#
#Confidence Interval:             LCL = 0.0000000
#                                 UCL = 0.6561838

#-------------------------------------------------------------------

# Compare the upper confidence limit based on
# 1) the default method:
#    normal approximation method based on Kulkarni and Powar (2010)
# 2) Profile Likelihood
# 3) Chi-Square Approximation
# 4) Chi-Square Adjusted

# Default Method
#---------------
with(EPA.94b.tccb.df,
  egamma(TcCB[Area == "Reference"], method = "bcmle", ci = TRUE,
    ci.type = "upper", conf.level = 0.9)$interval$limits["UCL"]
)

#      UCL
#0.6561838

# Profile Likelihood
```

```
#-------------------
with(EPA.94b.tccb.df,
  egamma(TcCB[Area == "Reference"], method = "mle", ci = TRUE,
    ci.type = "upper", conf.level = 0.9,
    ci.method = "profile.likelihood")$interval$limits["UCL"]
)

#      UCL
#0.6527009


# Chi-Square Approximation
#-------------------------
with(EPA.94b.tccb.df,
  egamma(TcCB[Area == "Reference"], method = "mle", ci = TRUE,
    ci.type = "upper", conf.level = 0.9,
    ci.method = "chisq.approx")$interval$limits["UCL"]
)

#      UCL
#0.6532188


# Chi-Square Adjusted
#--------------------
with(EPA.94b.tccb.df,
  egamma(TcCB[Area == "Reference"], method = "mle", ci = TRUE,
    ci.type = "upper", conf.level = 0.9,
    ci.method = "chisq.adj")$interval$limits["UCL"]
)

#    UCL
#0.65467
```

`egammaAltCensored` *Estimate Mean and Coefficient of Variation for a Gamma Distribution Based on Type I Censored Data*

Description

Estimate the mean and coefficient of variation of a gamma distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

```
egammaAltCensored(x, censored, method = "mle", censoring.side = "left",
  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
  ci.sample.size = sum(!censored))
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
method	character string specifying the method of estimation. Currently, the only available method is maximum likelihood (method="mle").
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
ci	logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profile.likelihood" (profile likelihood; the default), "normal.approx" (normal approximation), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
n.bootstraps	numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".
pivot.statistic	character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample size (see below). The argument pivot.statistic is ignored if ci=FALSE.
ci.sample.size	numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method="normal.approx". The default value is the number of uncensored observations.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x}$ denote a vector of N observations from a gamma distribution with parameters shape=κ and scale=θ. The relationship between these parameters and the mean μ and coefficient of variation τ of this distribution is given by:

$$\kappa = \tau^{-2} \quad (1)$$

$$\theta = \mu/\kappa \quad (2)$$

$$\mu = \kappa\,\theta \quad (3)$$

$$\tau = \kappa^{-1/2} \quad (4)$$

Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_k;\ k \geq 1 \quad (5)$$

For the case when $k \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $k = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j = 1, 2, \ldots, k$, so that

$$\sum_{i=1}^{k} c_j = c \quad (6)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations.

Estimation

Maximum Likelihood Estimation (method="mle")
For Type I left censored data, the likelihood function is given by:

$$L(\mu, \tau|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \quad (7)$$

where f and F denote the probability density function (pdf) and cumulative distribution function (cdf) of the population (Cohen, 1963; Cohen, 1991, pp.6, 50). That is,

$$f(t) = \frac{t^{\kappa-1} e^{-t/\theta}}{\theta^{\kappa}\Gamma(\kappa)} \quad (8)$$

(Johnson et al., 1994, p.343), where κ and θ are defined in terms of μ and τ by Equations (1) and (2) above.

For left singly censored data, Equation (7) simplifies to:

$$L(\mu, \tau|\underline{x}) = \binom{N}{c}[F(T)]^c \prod_{i=c+1}^{n} f[x_{(i)}] \qquad (9)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\mu, \tau|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k}[1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (10)$$

and for right singly censored data this simplifies to:

$$L(\kappa, \theta|\underline{x}) = \binom{N}{c}[1 - F(T)]^c \prod_{i=1}^{n} f[x_{(i)}] \qquad (11)$$

The maximum likelihood estimators are computed by minimizing the negative log-likelihood function.

Confidence Intervals

This section explains how confidence intervals for the mean μ are computed.

Likelihood Profile (`ci.method="profile.likelihood"`)

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean μ while treating the coefficient of variation τ as a nuisance parameter. Equation (7) above shows the form of the likelihood function $L(\mu, \tau|\underline{x})$ for multiply left-censored data, where μ and τ are defined by Equations (3) and (4), and Equation (10) shows the function for multiply right-censored data.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and coefficient of variation by (μ^*, τ^*). The likelihood ratio test statistic (G^2) of the hypothesis $H_0 : \mu = \mu_0$ (where μ_0 is a fixed value) equals the drop in $2log(L)$ between the "full" model and the reduced model with μ fixed at μ_0, i.e.,

$$G^2 = 2\{log[L(\mu^*, \tau^*)] - log[L(\mu_0, \tau_0^*)]\} \qquad (12)$$

where τ_0^* is the maximum likelihood estimate of τ for the reduced model (i.e., when $\mu = \mu_0$). Under the null hypothesis, the test statistic G^2 follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function L_1 for the mean μ, which is obtained from the usual likelihood function by maximizing over the parameter τ, i.e.,

$$L_1(\mu) = max_\tau L(\mu, \tau) \qquad (13)$$

Then we have

$$G^2 = 2\{log[L_1(\mu^*)] - log[L_1(\mu_0)]\} \qquad (14)$$

A two-sided $(1-\alpha)100\%$ confidence interval for the mean μ consists of all values of μ_0 for which the test is not significant at level $alpha$:

$$\mu_0 : G^2 \leq \chi^2_{1,1-\alpha} \qquad (15)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-squared distribution with ν degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1-\alpha$ in Equation (15) is replaced with $1-2\alpha$.

Normal Approximation (`ci.method="normal.approx"`)
This method constructs approximate $(1-\alpha)100\%$ confidence intervals for μ based on the assumption that the estimator of μ is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$ confidence interval for μ is constructed as:

$$[\hat{\mu} - t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}},\ \hat{\mu} + t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}}] \qquad (16)$$

where $\hat{\mu}$ denotes the estimate of μ, $\hat{\sigma}_{\hat{\mu}}$ denotes the estimated asymptotic standard deviation of the estimator of μ, m denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument `ci.sample.size` determines the value of m and by default is equal to the number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When `pivot.statistic="z"`, the p'th quantile from the standard normal distribution is used in place of the p'th quantile from Student's t-distribution.

The standard deviation of the mle of μ is estimated based on the inverse of the Fisher Information matrix.

Bootstrap and Bias-Corrected Bootstrap Approximation (`ci.method="bootstrap"`)
The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1-\alpha)100\%$ confidence interval for the population mean μ, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size N from the observations in $\underline{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\underline{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).
2. Estimate μ based on the bootstrap sample created in Step 1, using the same method that was used to estimate μ using the original observations in $\underline{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of μ based on the bootstrap sample will usually differ from the original estimate based on $\underline{x}$.
3. Repeat Steps 1 and 2 B times, where B is some large number. For the function `egammaAltCensored`, the number of bootstraps B is determined by the argument `n.bootstraps` (see the section ARGUMENTS above). The default value of `n.bootstraps` is `1000`.

4. Use the B estimated values of μ to compute the empirical cumulative distribution function of this estimator of μ (see ecdfPlot), and then create a confidence interval for μ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}),\ \hat{G}^{-1}(1-\frac{\alpha}{2})] \qquad (17)$$

where $\hat{G}(t)$ denotes the empirical cdf evaluated at t and thus $\hat{G}^{-1}(p)$ denotes the p'th empirical quantile, that is, the p'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha),\ \infty] \qquad (18)$$

and a one-sided upper confidence interval is computed as:

$$[0,\ \hat{G}^{-1}(1-\alpha)] \qquad (19)$$

The function egammaAltCensored calls the R function quantile to compute the empirical quantiles used in Equations (17)-(19).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of μ can be off by $k/\sqrt{N}$, where $k$is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of μ may be off by k/N instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1),\ \hat{G}^{-1}(\alpha_2)] \qquad (20)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1-\hat{a}(z_0 + z_{\alpha/2})}] \qquad (21)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1-\hat{a}(z_0 + z_{1-\alpha/2})}] \qquad (22)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\mu})] \qquad (23)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^3}{6[\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^2]^{3/2}} \qquad (24)$$

where the quantity $\hat{\mu}_{(i)}$ denotes the estimate of μ using all the values in $\underline{x}$ except the i'th one, and

$$\hat{\mu}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\hat{\mu_{(i)}} \qquad (25)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1),\ \infty] \qquad (26)$$

and a one-sided upper confidence interval is given by:

$$[0,\ \hat{G}^{-1}(\alpha_2)] \qquad (27)$$

where α_1 and α_2 are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with α in Equations (21) and (22).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of μ with respect to the true value of μ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of μ does not depend on the value of μ, hence the acceleration constant is not really necessary.

When `ci.method="bootstrap"`, the function `egammaAltCensored` computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

Value

a list of class `"estimateCensored"` containing the estimated parameters and other information. See `estimateCensored.object` for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation. Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators for parameters of a normal or lognormal distribution based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation (or coefficient of variation), rather than rely on a single point-estimate of the mean. Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and coefficient of variation of a gamma distribution when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

Cohen, A.C. (1963). Progressively Censored Samples in Life Testing. *Technometrics* **5**, 327–339

Cohen, A.C. (1991). *Truncated and Censored Samples*. Marcel Dekker, New York, New York, 312pp.

Cox, D.R. (1970). *Analysis of Binary Data*. Chapman & Hall, London. 142pp.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). *Statistical Distributions, Fourth Edition*. John Wiley and Sons, Hoboken, NJ.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Johnson, N.L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, Chapter 17.

Millard, S.P., P. Dixon, and N.K. Neerchal. (2014; in preparation). *Environmental Statistics with R*. CRC Press, Boca Raton, Florida.

Nelson, W. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, 634pp.

Royston, P. (2007). Profile Likelihood for Estimation and Confdence Intervals. *The Stata Journal* **7**(3), pp. 376–387.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Stryhn, H., and J. Christensen. (2003). *Confidence Intervals by the Profile Likelihood Method, with Applications in Veterinary Epidemiology*. Contributed paper at ISVEE X (November 2003, Chile). https://gilvanguedes.com/wp-content/uploads/2019/05/Profile-Likelihood-CI.pdf.

Venzon, D.J., and S.H. Moolgavkar. (1988). A Method for Computing Profile-Likelihood-Based Confidence Intervals. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **37**(1), pp. 87–94.

See Also

egammaCensored, GammaDist, egamma, estimateCensored.object.

Examples

```
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and coefficient of variation
# ON THE ORIGINAL SCALE using the MLE and
# assuming a gamma distribution.

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
```

```
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now estimate the mean and coefficient of variation
# using the MLE, and compute a confidence interval
# for the mean using the profile-likelihood method.
#--------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  egammaAltCensored(Manganese.ppb, Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#----------------------------------------
#
#Assumed Distribution:            Gamma
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          mean = 19.664797
#                                 cv   =  1.252936
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Profile Likelihood
#
```

```
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 12.25151
#                                 UCL = 34.35332

#----------

# Compare the confidence interval for the mean
# based on assuming a lognormal distribution versus
# assuming a gamma distribution.

with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored,
    ci = TRUE))$interval$limits
#     LCL      UCL
#12.37629 69.87694

with(EPA.09.Ex.15.1.manganese.df,
  egammaAltCensored(Manganese.ppb, Censored,
    ci = TRUE))$interval$limits
#     LCL      UCL
#12.25151 34.35332
```

---

egammaCensored *Estimate Shape and Scale Parameters for a Gamma Distribution Based on Type I Censored Data*

Description

Estimate the shape and scale parameters of a gamma distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

```
egammaCensored(x, censored, method = "mle", censoring.side = "left",
  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
  ci.sample.size = sum(!censored))
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric",

it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.

method — character string specifying the method of estimation. Currently, the only available method is maximum likelihood (method="mle").

censoring.side — character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

ci — logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.

ci.method — character string indicating what method to use to construct the confidence interval for the mean. The possible values are "profile.likelihood" (profile likelihood; the default), "normal.approx" (normal approximation), and "bootstrap" (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if ci=FALSE.

ci.type — character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level — a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

n.bootstraps — numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

pivot.statistic — character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample size (see below). The argument pivot.statistic is ignored if ci=FALSE.

ci.sample.size — numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method="normal.approx". The default value is the number of uncensored observations.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x}$ denote a vector of N observations from a gamma distribution with parameters shape=κ and scale=θ. The relationship between these parameters and the mean μ and coefficient of variation τ of this distribution is given by:

$$\kappa = \tau^{-2} \qquad (1)$$

$$\theta = \mu/\kappa \qquad (2)$$

$$\mu = \kappa\,\theta \qquad (3)$$

$$\tau = \kappa^{-1/2} \qquad (4)$$

Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_k;\ k \geq 1 \qquad (5)$$

For the case when $k \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $k = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j = 1, 2, \ldots, k$, so that

$$\sum_{i=1}^{k} c_j = c \qquad (6)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations.

Estimation

Maximum Likelihood Estimation (`method="mle"`)
For Type I left censored data, the likelihood function is given by:

$$L(\kappa, \theta|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (7)$$

where f and F denote the probability density function (pdf) and cumulative distribution function (cdf) of the population (Cohen, 1963; Cohen, 1991, pp.6, 50). That is,

$$f(t) = \frac{t^{\kappa-1} e^{-t/\theta}}{\theta^{\kappa} \Gamma(\kappa)} \qquad (8)$$

(Johnson et al., 1994, p.343). For left singly censored data, Equation (7) simplifies to:

$$L(\kappa, \theta|\underline{x}) = \binom{N}{c} [F(T)]^{c} \prod_{i=c+1}^{n} f[x_{(i)}] \qquad (9)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\kappa, \theta|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (10)$$

and for right singly censored data this simplifies to:

$$L(\kappa, \theta|\underline{x}) = \binom{N}{c}[1 - F(T)]^c \prod_{i=1}^{n} f[x_{(i)}] \qquad (11)$$

The maximum likelihood estimators are computed by minimizing the negative log-likelihood function.

Confidence Intervals

This section explains how confidence intervals for the mean μ are computed.

Likelihood Profile (`ci.method="profile.likelihood"`)

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean μ while treating the coefficient of variation τ as a nuisance parameter. Equation (7) above shows the form of the likelihood function $L(\mu, \tau|\underline{x})$ for multiply left-censored data and Equation (10) shows the function for multiply right-censored data, where μ and τ are defined by Equations (3) and (4).

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and coefficient of variation by (μ^*, τ^*). The likelihood ratio test statistic (G^2) of the hypothesis $H_0 : \mu = \mu_0$ (where μ_0 is a fixed value) equals the drop in $2log(L)$ between the "full" model and the reduced model with μ fixed at μ_0, i.e.,

$$G^2 = 2\{log[L(\mu^*, \tau^*)] - log[L(\mu_0, \tau_0^*)]\} \qquad (12)$$

where τ_0^* is the maximum likelihood estimate of τ for the reduced model (i.e., when $\mu = \mu_0$). Under the null hypothesis, the test statistic G^2 follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function L_1 for the mean μ, which is obtained from the usual likelihood function by maximizing over the parameter τ, i.e.,

$$L_1(\mu) = max_\tau L(\mu, \tau) \qquad (13)$$

Then we have

$$G^2 = 2\{log[L_1(\mu^*)] - log[L_1(\mu_0)]\} \qquad (14)$$

A two-sided $(1 - \alpha)100\%$ confidence interval for the mean μ consists of all values of μ_0 for which the test is not significant at level $alpha$:

$$\mu_0 : G^2 \leq \chi^2_{1,1-\alpha} \qquad (15)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-squared distribution with ν degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1 - \alpha$ in Equation (15) is replaced with $1 - 2\alpha$.

Normal Approximation (`ci.method="normal.approx"`)

This method constructs approximate $(1 - \alpha)100\%$ confidence intervals for μ based on the assumption that the estimator of μ is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$

confidence interval for μ is constructed as:

$$[\hat{\mu} - t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}},\ \hat{\mu} + t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}}] \quad (16)$$

where $\hat{\mu}$ denotes the estimate of μ, $\hat{\sigma}_{\hat{\mu}}$ denotes the estimated asymptotic standard deviation of the estimator of μ, m denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument `ci.sample.size` determines the value of m and by default is equal to the number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When `pivot.statistic="z"`, the p'th quantile from the standard normal distribution is used in place of the p'th quantile from Student's t-distribution.

The standard deviation of the mle of μ is estimated based on the inverse of the Fisher Information matrix.

Bootstrap and Bias-Corrected Bootstrap Approximation (`ci.method="bootstrap"`)
The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1-\alpha)100\%$ confidence interval for the population mean μ, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size N from the observations in $\underline{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\underline{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).
2. Estimate μ based on the bootstrap sample created in Step 1, using the same method that was used to estimate μ using the original observations in $\underline{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of μ based on the bootstrap sample will usually differ from the original estimate based on $\underline{x}$.
3. Repeat Steps 1 and 2 B times, where B is some large number. For the function `egammaCensored`, the number of bootstraps B is determined by the argument `n.bootstraps` (see the section ARGUMENTS above). The default value of `n.bootstraps` is `1000`.
4. Use the B estimated values of μ to compute the empirical cumulative distribution function of this estimator of μ (see `ecdfPlot`), and then create a confidence interval for μ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}),\ \hat{G}^{-1}(1-\frac{\alpha}{2})] \quad (17)$$

where $\hat{G}(t)$ denotes the empirical cdf evaluated at t and thus $\hat{G}^{-1}(p)$ denotes the p'th empirical quantile, that is, the p'th quantile associated with the empirical cdf. Similarly, a one-sided lower

confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha),\ \infty] \qquad (18)$$

and a one-sided upper confidence interval is computed as:

$$[0,\ \hat{G}^{-1}(1-\alpha)] \qquad (19)$$

The function `egammaCensored` calls the R function `quantile` to compute the empirical quantiles used in Equations (17)-(19).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of μ can be off by $k/\sqrt{N}$, where $k$is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of μ may be off by k/N instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1),\ \hat{G}^{-1}(\alpha_2)] \qquad (20)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(z_0 + z_{\alpha/2})}] \qquad (21)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}(z_0 + z_{1-\alpha/2})}] \qquad (22)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\mu})] \qquad (23)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^3}{6[\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^2]^{3/2}} \qquad (24)$$

where the quantity $\hat{\mu}_{(i)}$ denotes the estimate of μ using all the values in $\underline{x}$ except the i'th one, and

$$\hat{\mu}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\hat{\mu_{(i)}} \qquad (25)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1),\ \infty] \qquad (26)$$

and a one-sided upper confidence interval is given by:

$$[0,\ \hat{G}^{-1}(\alpha_2)] \qquad (27)$$

where α_1 and α_2 are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with α in Equations (21) and (22).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of μ with respect to the true value of μ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of μ does not depend on the value of μ, hence the acceleration constant is not really necessary.

When `ci.method="bootstrap"`, the function `egammaCensored` computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

Value

a list of class `"estimateCensored"` containing the estimated parameters and other information. See `estimateCensored.object` for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation. Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators for parameters of a normal or lognormal distribution based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation (or coefficient of variation), rather than rely on a single point-estimate of the mean. Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and coefficient of variation of a gamma distribution when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

Cohen, A.C. (1963). Progressively Censored Samples in Life Testing. *Technometrics* **5**, 327–339

Cohen, A.C. (1991). *Truncated and Censored Samples*. Marcel Dekker, New York, New York, 312pp.

Cox, D.R. (1970). *Analysis of Binary Data*. Chapman & Hall, London. 142pp.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). *Statistical Distributions, Fourth Edition*. John Wiley and Sons, Hoboken, NJ.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Johnson, N.L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, Chapter 17.

Millard, S.P., P. Dixon, and N.K. Neerchal. (2014; in preparation). *Environmental Statistics with R*. CRC Press, Boca Raton, Florida.

Nelson, W. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, 634pp.

Royston, P. (2007). Profile Likelihood for Estimation and Confdence Intervals. *The Stata Journal* **7**(3), pp. 376–387.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Stryhn, H., and J. Christensen. (2003). *Confidence Intervals by the Profile Likelihood Method, with Applications in Veterinary Epidemiology*. Contributed paper at ISVEE X (November 2003, Chile). https://gilvanguedes.com/wp-content/uploads/2019/05/Profile-Likelihood-CI.pdf.

Venzon, D.J., and S.H. Moolgavkar. (1988). A Method for Computing Profile-Likelihood-Based Confidence Intervals. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **37**(1), pp. 87–94.

See Also

egammaAltCensored, GammaDist, egamma, estimateCensored.object.

Examples

```
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the shape and scale parameters using
# the data ON THE ORIGINAL SCALE, using the MLE and
# assuming a gamma distribution.

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
```

```
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now estimate the shape and scale parameters
# using the MLE, and compute a confidence interval
# for the mean using the profile-likelihood method.
#--------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  egammaCensored(Manganese.ppb, Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#-------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          shape =  0.6370043
#                                 scale = 30.8707533
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Profile Likelihood
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 12.25151
#                                 UCL = 34.35332

#----------

# Compare the confidence interval for the mean
# based on assuming a lognormal distribution versus
# assuming a gamma distribution.
```

```
with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored,
    ci = TRUE))$interval$limits
#     LCL      UCL
#12.37629 69.87694

with(EPA.09.Ex.15.1.manganese.df,
  egammaCensored(Manganese.ppb, Censored,
    ci = TRUE))$interval$limits
#     LCL      UCL
#12.25151 34.35332
```

---

egeom *Estimate Probability Parameter of a Geometric Distribution*

Description

Estimate the probability parameter of a geometric distribution.

Usage

```
egeom(x, method = "mle/mme")
```

Arguments

x	vector of non-negative integers indicating the number of trials that took place *before* the first "success" occurred. (The total number of trials that took place is x+1). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. If length(x)=n and n is greater than 1, it is assumed that x represents observations from n separate geometric experiments that all had the same probability of success (prob).
method	character string specifying the method of estimation. Possible values are "mle/mme" (maximum likelihood and method of moments; the default) and "mvue" (minimum variance unbiased). You cannot use method="mvue" if length(x)=1. See the DETAILS section for more information on these estimation methods.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n independent observations from a geometric distribution with parameter prob=p.

It can be shown (e.g., Forbes et al., 2011) that if X is defined as:

$$X = \sum_{i=1}^{n} x_i$$

then X is an observation from a negative binomial distribution with parameters prob=p and size=n.

Estimation
The maximum likelihood and method of moments estimator (mle/mme) of p is given by:

$$\hat{p}_{mle} = \frac{n}{X + n}$$

and the minimum variance unbiased estimator (mvue) of p is given by:

$$\hat{p}_{mvue} = \frac{n - 1}{X + n - 1}$$

(Forbes et al., 2011). Note that the mvue of p is not defined for $n = 1$.

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The geometric distribution with parameter prob=p is a special case of the negative binomial distribution with parameters size=1 and prob=p.

The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 5.

See Also

Geometric, enbinom, NegBinomial.

Examples

```
# Generate an observation from a geometric distribution with parameter
# prob=0.2, then estimate the parameter prob.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rgeom(1, prob = 0.2)
```

```
dat
#[1] 4

egeom(dat)
#Results of Distribution Parameter Estimation
#-----------------------------------------
#
#Assumed Distribution:            Geometric
#
#Estimated Parameter(s):          prob = 0.2
#
#Estimation Method:               mle/mme
#
#Data:                            dat
#
#Sample Size:                     1

#----------

# Generate 3 observations from a geometric distribution with parameter
# prob=0.2, then estimate the parameter prob with the mvue.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(200)
dat <- rgeom(3, prob = 0.2)
dat
#[1] 0 1 2

egeom(dat, method = "mvue")
#Results of Distribution Parameter Estimation
#-----------------------------------------
#
#Assumed Distribution:            Geometric
#
#Estimated Parameter(s):          prob = 0.4
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     3

#----------

# Clean up
#---------
rm(dat)
```

---

egevd *Estimate Parameters of a Generalized Extreme Value Distribution*

Description

Estimate the location, scale and shape parameters of a generalized extreme value distribution, and optionally construct a confidence interval for one of the parameters.

Usage

```
egevd(x, method = "mle", pwme.method = "unbiased", tsoe.method = "med",
  plot.pos.cons = c(a = 0.35, b = 0), ci = FALSE, ci.parameter = "location",
 ci.type = "two-sided", ci.method = "normal.approx", information = "observed",
  conf.level = 0.95)
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default), "pwme" (probability-weighted moments), and "tsoe" (two-stage order-statistics estimator of Castillo and Hadi (1994)). See the DETAILS section for more information on these estimation methods.
pwme.method	character string specifying what method to use to compute the probability-weighted moments when method="pwme". The possible values are "ubiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section in this help file and the help file for pwMoment for more information. This argument is ignored if method is not equal to "pwme".
tsoe.method	character string specifying the robust function to apply in the second stage of the two-stage order-statistics estimator when method="tsoe". Possible values are "med" (median; the default), and "lms" (least median of squares). See the DETAILS section for more information on these estimation methods. This argument is ignored if method is not equal to "tsoe".
plot.pos.cons	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="pwme" and pwme.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section in this help file and the help file for pwMoment for more information. This argument is used only if method="tsoe", or if both method="pwme" and pwme.method="plotting.position".
ci	logical scalar indicating whether to compute a confidence interval for the location, scale, or shape parameter. The default value is FALSE.
ci.parameter	character string indicating the parameter for which the confidence interval is desired. The possible values are "location" (the default), "scale", or "shape". This argument is ignored if ci=FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

`ci.method` character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is `"normal.approx"` (the default). See the DETAILS section for more information. This argument is ignored if `ci=FALSE`.

`information` character string indicating which kind of Fisher information to use when computing the variance-covariance matrix of the maximum likelihood estimators. The possible values are `"observed"` (the default) and `"expected"`. See the DETAILS section for more information. This argument is used only when `method="mle"` and `ci=TRUE`.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a generalized extreme value distribution with parameters `location`=η, `scale`=θ, and `shape`=κ.

Estimation

Maximum Likelihood Estimation (`method="mle"`)
The log likelihood function is given by:

$$L(\eta, \theta, \kappa) = -n\,log(\theta) - (1-\kappa)\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} e^{y_i}$$

where

$$y_i = -\frac{1}{\kappa} log[\frac{1-\kappa(x_i-\eta)}{\theta}]$$

(see, for example, Jenkinson, 1969; Prescott and Walden, 1980; Prescott and Walden, 1983; Hosking, 1985; MacLeod, 1989). The maximum likelihood estimators (MLE's) of η, θ, and κ are those values that maximize the likelihood function, subject to the following constraints:

$$\theta > 0$$

$$\kappa \leq 1$$

$$x_i < \eta + \frac{\theta}{\kappa}\ if \kappa > 0$$

$$x_i > \eta + \frac{\theta}{\kappa}\ if \kappa < 0$$

Although in theory the value of κ may lie anywhere in the interval $(-\infty, \infty)$ (see GEVD), the constraint $\kappa \leq 1$ is imposed because when $\kappa > 1$ the likelihood can be made infinite and thus the MLE does not exist (Castillo and Hadi, 1994). Hence, **this method of estimation is not valid when the true value of κ is larger than 1**. Hosking (1985) and Hosking et al. (1985) note that in practice the value of κ tends to lie in the interval $-1/2 < \kappa < 1/2$.

The value of $-L$ is minimized using the R function `nlminb`. Prescott and Walden (1983) give formulas for the gradient and Hessian. Only the gradient is supplied in the call to `nlminb`. The

values of the PWME (see below) are used as the starting values. If the starting value of κ is less than 0.001 in absolute value, it is reset to `sign(k) * 0.001`, as suggested by Hosking (1985).

Probability-Weighted Moments Estimation (`method="pwme"`)
The idea of probability-weighted moments was introduced by Greenwood et al. (1979). Landwehr et al. (1979) derived probability-weighted moment estimators (PWME's) for the parameters of the Type I (Gumbel) extreme value distribution. Hosking et al. (1985) extended these results to the generalized extreme value distribution. See the abstract for Hosking et al. (1985) for details on how these estimators are computed.

Two-Stage Order Statistics Estimation (`method="tsoe"`)
The two-stage order statistics estimator (TSOE) was introduced by Castillo and Hadi (1994) as an alternative to the MLE and PWME. Unlike the MLE and PWME, the TSOE of κ exists for all combinations of sample values and possible values of κ. See the abstract for Castillo and Hadi (1994) for details on how these estimators are computed. In the second stage, Castillo and Hadi (1984) suggest using either the median or the least median of squares as the robust function. The function `egevd` allows three options for the robust function: median (`tsoe.method="med"`; see the R help file for `median`), least median of squares (`tsoe.method="lms"`; see the help file for `lmsreg` in the package **MASS**), and least trimmed squares (`tsoe.method="lts"`; see the help file for `ltsreg` in the package **MASS**).

Confidence Intervals
When `ci=TRUE`, an approximate $(1-\alpha)100\%$ confidence intervals for η can be constructed assuming the distribution of the estimator of η is approximately normally distributed. A two-sided confidence interval is constructed as:

$$[\hat{\eta} - t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\eta}},\ \hat{\eta} + t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\eta}}]$$

where $t(\nu, p)$ is the p'th quantile of Student's t-distribution with ν degrees of freedom, and the quantity

$$\hat{\sigma}_{\hat{\eta}}$$

denotes the estimated asymptotic standard deviation of the estimator of η.

Similarly, a two-sided confidence interval for θ is constructed as:

$$[\hat{\theta} - t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\theta}},\ \hat{\theta} + t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\theta}}]$$

and a two-sided confidence interval for κ is constructed as:

$$[\hat{\kappa} - t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\kappa}},\ \hat{\kappa} + t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\kappa}}]$$

One-sided confidence intervals for η, θ, and κ are computed in a similar fashion.

Maximum Likelihood Estimator (`method="mle"`)
Prescott and Walden (1980) derive the elements of the Fisher information matrix (the expected information). The inverse of this matrix, evaluated at the values of the MLE, is the estimated asymptotic variance-covariance matrix of the MLE. This method is used to estimate the standard deviations of the estimated distribution parameters when `information="expected"`. The necessary regularity conditions hold for $\kappa < 1/2$. Thus, **this method of constructing confidence intervals is not valid when the true value of κ is greater than or equal to 1/2**.

Prescott and Walden (1983) derive expressions for the observed information matrix (i.e., the Hessian). This matrix is used to compute the estimated asymptotic variance-covariance matrix of the MLE when `information="observed"`.

In computer simulations, Prescott and Walden (1983) found that the variance-covariance matrix based on the observed information gave slightly more accurate estimates of the variance of MLE of κ compared to the estimated variance based on the expected information.

Probability-Weighted Moments Estimator (`method="pwme"`)
Hosking et al. (1985) show that these estimators are asymptotically multivariate normal and derive the asymptotic variance-covariance matrix. See the abstract for Hosking et al. (1985) for details on how this matrix is computed.

Two-Stage Order Statistics Estimator (`method="tsoe"`)
Currently there is no built-in method in **EnvStats** for computing confidence intervals when `method="tsoe"`. Castillo and Hadi (1994) suggest using the bootstrap or jackknife method.

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Note

Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930's to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter $\kappa = 0$, the GEVD reduces to the Type I extreme value (Gumbel) distribution. (The function `zTestGevdShape` allows you to test the null hypothesis $H_0 : \kappa = 0$.) When $\kappa > 0$, the GEVD is the same as the Type II extreme value distribution, and when $\kappa < 0$ it is the same as the Type III extreme value distribution.

Hosking et al. (1985) compare the asymptotic and small-sample statistical properties of the PWME with the MLE and Jenkinson's (1969) method of sextiles. Castillo and Hadi (1994) compare the small-sample statistical properties of the MLE, PWME, and TSOE. Hosking and Wallis (1995) compare the small-sample properties of unbaised L-moment estimators vs. plotting-position L-moment estimators. (PWMEs can be written as linear combinations of L-moments and thus have equivalent statistical properties.) Hosking and Wallis (1995) conclude that unbiased estimators should be used for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Castillo, E., and A. Hadi. (1994). Parameter and Quantile Estimation for the Generalized Extreme-Value Distribution. *Environmetrics* **5**, 417–432.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Greenwood, J.A., J.M. Landwehr, N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments: Definition and Relation to Parameters of Several Distributions Expressible in Inverse Form. *Water Resources Research* **15**(5), 1049–1054.

Hosking, J.R.M. (1984). Testing Whether the Shape Parameter is Zero in the Generalized Extreme-Value Distribution. *Biometrika* **71**(2), 367–374.

Hosking, J.R.M. (1985). Algorithm AS 215: Maximum-Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Applied Statistics* **34**(3), 301–310.

Hosking, J.R.M., J.R. Wallis, and E.F. Wood. (1985). Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments. *Technometrics* **27**(3), 251–261.

Jenkinson, A.F. (1969). Statistics of Extremes. *Technical Note 98*, World Meteorological Office, Geneva.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

Landwehr, J.M., N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments Compared With Some Traditional Techniques in Estimating Gumbel Parameters and Quantiles. *Water Resources Research* **15**(5), 1055–1064.

Macleod, A.J. (1989). Remark AS R76: A Remark on Algorithm AS 215: Maximum Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Applied Statistics* **38**(1), 198–199.

Prescott, P., and A.T. Walden. (1980). Maximum Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Biometrika* **67**(3), 723–724.

Prescott, P., and A.T. Walden. (1983). Maximum Likelihood Estimation of the Three-Parameter Generalized Extreme-Value Distribution from Censored Samples. *Journal of Statistical Computing and Simulation* **16**, 241–250.

See Also

Generalized Extreme Value Distribution, `zTestGevdShape`, Extreme Value Distribution, `eevd`.

Examples

```
# Generate 20 observations from a generalized extreme value distribution
# with parameters location=2, scale=1, and shape=0.2, then compute the
# MLE and construct a 90% confidence interval for the location parameter.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(498)
dat <- rgevd(20, location = 2, scale = 1, shape = 0.2)
egevd(dat, ci = TRUE, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Generalized Extreme Value
#
#Estimated Parameter(s):          location = 1.6144631
#                                 scale    = 0.9867007
```

```
#                                shape    = 0.2632493
#
#Estimation Method:              mle
#
#Data:                           dat
#
#Sample Size:                    20
#
#Confidence Interval for:        location
#
#Confidence Interval Method:     Normal Approximation
#                                (t Distribution) based on
#                                observed information
#
#Confidence Interval Type:       two-sided
#
#Confidence Level:               90%
#
#Confidence Interval:            LCL = 1.225249
#                                UCL = 2.003677

#----------

# Compare the values of the different types of estimators:

egevd(dat, method = "mle")$parameters
# location     scale     shape
#1.6144631 0.9867007 0.2632493

egevd(dat, method = "pwme")$parameters
# location     scale     shape
#1.5785779 1.0187880 0.2257948

egevd(dat, method = "pwme", pwme.method = "plotting.position")$parameters
# location     scale     shape
#1.5509183 0.9804992 0.1657040

egevd(dat, method = "tsoe")$parameters
# location     scale     shape
#1.5372694 1.0876041 0.2927272

egevd(dat, method = "tsoe", tsoe.method = "lms")$parameters
#location    scale    shape
#1.519469 1.081149 0.284863

egevd(dat, method = "tsoe", tsoe.method = "lts")$parameters
# location     scale     shape
#1.4840198 1.0679549 0.2691914

#----------

# Clean up
#---------
```

```
rm(dat)
```

---

ehyper *Estimate Parameter of a Hypergeometric Distribution*

Description

Estimate m, the number of white balls in the urn, or $m + n$, the total number of balls in the urn, for a hypergeometric distribution.

Usage

```
ehyper(x, m = NULL, total = NULL, k, method = "mle")
```

Arguments

x	non-negative integer indicating the number of white balls out of a sample of size k drawn without replacement from the urn. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
m	non-negative integer indicating the number of white balls in the urn. You must supply m or total, but not both. Missing values (NAs) are not allowed.
total	positive integer indicating the total number of balls in the urn (i.e., m+n). You must supply m or total, but not both. Missing values (NAs) are not allowed.
k	positive integer indicating the number of balls drawn without replacement from the urn. Missing values (NAs) are not allowed.
method	character string specifying the method of estimation. Possible values are "mle" (maximum likelihood; the default) and "mvue" (minimum variance unbiased). The mvue method is only available when you are estimating m (i.e., when you supply the argument total). See the DETAILS section for more information on these estimation methods.

Details

Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

Let x be an observation from a hypergeometric distribution with parameters m=M, n=N, and k=K. In R nomenclature, x represents the number of white balls drawn out of a sample of K balls drawn *without* replacement from an urn containing M white balls and N black balls. The total number of balls in the urn is thus $M + N$. Denote the total number of balls by $T = M + N$.

Estimation

Estimating M, Given T and K are known

When T and K are known, the maximum likelihood estimator (mle) of M is given by (Forbes et al., 2011):

$$\hat{M}_{mle} = floor[(T + 1)x/K] \quad (1)$$

where $floor()$ represents the floor function. That is, $floor(y)$ is the largest integer less than or equal to y.

If the quantity $floor[(T+1)x/K]$ is an integer, then the mle of M is also given by (Johnson et al., 1992, p.263):

$$\hat{M}_{mle} = [(T+1)x/K] - 1 \quad (2)$$

which is what the function ehyper uses for this case.

The minimum variance unbiased estimator (mvue) of M is given by (Forbes et al., 2011):

$$\hat{M}_{mvue} = (Tx/K) \quad (3)$$

Estimating T, given M and K are known
When M and K are known, the maximum likelihood estimator (mle) of T is given by (Forbes et al., 2011):

$$\hat{T}_{mle} = floor(KM/x) \quad (4)$$

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The hypergeometric distribution can be described by an urn model with M white balls and N black balls. If K balls are drawn *with* replacement, then the number of white balls in the sample of size K follows a binomial distribution with parameters size=K and prob=$M/(M+N)$. If K balls are drawn *without* replacement, then the number of white balls in the sample of size K follows a hypergeometric distribution with parameters m=M, n=N, and k=K.

The name "hypergeometric" comes from the fact that the probabilities associated with this distribution can be written as successive terms in the expansion of a function of a Gaussian hypergeometric series.

The hypergeometric distribution is applied in a variety of fields, including quality control and estimation of animal population size. It is also the distribution used to compute probabilities for Fishers's exact test for a 2x2 contingency table.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 6.

See Also

Hypergeometric.

Examples

```
# Generate an observation from a hypergeometric distribution with
# parameters m=10, n=30, and k=5, then estimate the parameter m.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter actually estimated is m; once m is estimated,
# n is computed by subtracting the estimated value of m (8 in this example)
# from the given of value of m+n (40 in this example).  The parameters
# n and k are shown in the output in order to provide information on
# all of the parameters associated with the hypergeometric distribution.

set.seed(250)
dat <- rhyper(nn = 1, m = 10, n = 30, k = 5)
dat
#[1] 1

ehyper(dat, total = 40, k = 5)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Hypergeometric
#
#Estimated Parameter(s):          m =  8
#                                 n = 32
#                                 k =  5
#
#Estimation Method:               mle for 'm'
#
#Data:                            dat
#
#Sample Size:                     1

#----------

# Use the same data as in the previous example, but estimate m+n instead.
# Note: The only parameter estimated is m+n. Once this is estimated,
# n is computed by subtracting the given value of m (10 in this case)
# from the estimated value of m+n (50 in this example).

ehyper(dat, m = 10, k = 5)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Hypergeometric
#
#Estimated Parameter(s):          m = 10
#                                 n = 40
```

```
#                                k =  5
#
#Estimation Method:              mle for 'm+n'
#
#Data:                           dat
#
#Sample Size:                    1


#----------

# Clean up
#---------
rm(dat)
```

---

elnorm *Estimate Parameters of a Lognormal Distribution (Log-Scale)*

Description

Estimate the mean and standard deviation parameters of the logarithm of a lognormal distribution, and optionally construct a confidence interval for the mean.

Usage

```
elnorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "exact", conf.level = 0.95)
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Possible values are "mvue" (minimum variance unbiased; the default), and "mle/mme" (maximum likelihood/method of moments). See the DETAILS section for more information on these estimation methods.
ci	logical scalar indicating whether to compute a confidence interval for the mean. The default value is FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the mean or variance. The only possible value is "exact" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let X denote a random variable with a lognormal distribution with parameters meanlog=μ and sdlog=σ. Then $Y = log(X)$ has a normal (Gaussian) distribution with parameters mean=μ and sd=σ. Thus, the function elnorm simply calls the function enorm using the log-transformed values of x.

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean or variance. This is done with confidence intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special references to its uses in economics)*. Cambridge University Press, London, Chapter 5.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, Chapter 2.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Limpert, E., W.A. Stahel, and M. Abbt. (2001). Log-Normal Distributions Across the Sciences: Keys and Clues. *BioScience* **51**, 341–352.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

Lognormal, LognormalAlt, Normal.

Examples

```
# Using the Reference area TcCB data in the data frame EPA.94b.tccb.df,
# estimate the mean and standard deviation of the log-transformed distribution,
# and construct a 95% confidence interval for the mean.

with(EPA.94b.tccb.df, elnorm(TcCB[Area == "Reference"], ci = TRUE))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = -0.6195712
#                                 sdlog   =  0.4679530
#
#Estimation Method:               mvue
#
#Data:                            TcCB[Area == "Reference"]
#
#Sample Size:                     47
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = -0.7569673
#                                 UCL = -0.4821751
```

---

elnorm3 *Estimate Parameters of a Three-Parameter Lognormal Distribution (Log-Scale)*

Description

Estimate the mean, standard deviation, and threshold parameters for a three-parameter lognormal distribution, and optionally construct a confidence interval for the threshold or the median of the distribution.

Usage

```
elnorm3(x, method = "lmle", ci = FALSE, ci.parameter = "threshold",
  ci.method = "avar", ci.type = "two-sided", conf.level = 0.95,
  threshold.lb.sd = 100, evNormOrdStats.method = "royston")
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Possible values are: • "lmle" (local maximum likelihood; the default) • "mme" (method of moments) • "mmue" (method of moments using an unbaised estimate of variance) • "mmme" (modified method of moments due to Cohen and Whitten (1980)) • "zero.skew" (zero-skewness estimator due to Griffiths (1980)) • "royston.skew" (estimator based on Royston's (1992b) index of skewness). See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for either the threshold or median of the distribution. The default value is FALSE.
ci.parameter	character string indicating the parameter for which the confidence interval is desired. The possible values are "threshold" (the default) and "median". This argument is ignored if ci=FALSE.
ci.method	character string indicating the method to use to construct the confidence interval. The possible values are "avar" (asymptotic variance; the default), "likelihood.profile", and "skewness" (method suggested by Royston (1992b) for method="zero.skew"). This argument is ignored if ci=FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
threshold.lb.sd	a positive numeric scalar specifying the range over which to look for the local maximum likelihood (method="lmle") or zero-skewness (method="zero.skewness") estimator of threshold. The range is set to [mean(x) - threshold.lb.sd * sd(x), min(x)]. If you receive a warning message that elnorm3 is unable to find an acceptable estimate of threshold in this range, it may be because of convergence problems specific to the data in x. When this occurs, try changing the value of threshold.lb.sd. This same range is used in constructing confidence intervals for the threshold parameter. The default value is threshold.lb.sd=100. This argument is relevant only if method="lmle", method="zero.skew", ci.method="likelihood.profile", and/or ci.method="skewness".
evNormOrdStats.method	character string indicating which method to use in the call to link{evNormOrdStatsScalar} when method="mmme". See the DETAILS section for more information.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let X denote a random variable from a three-parameter lognormal distribution with parameters meanlog=μ, sdlog=σ, and threshold=γ. Let $\underline{x}$ denote a vector of n observations from this distribution. Furthermore, let $x_{(i)}$ denote the i'th order statistic in the sample, so that $x_{(1)}$ denotes the smallest value and $x_{(n)}$ denote the largest value in $\underline{x}$. Finally, denote the sample mean and variance by:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (1)$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (2)$$

Note that the sample variance is the unbiased version. Denote the method of moments estimator of variance by:

$$s_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (3)$$

Estimation

Local Maximum Likelihood Estimation (method="lmle")
Hill (1963) showed that the likelihood function approaches infinity as γ approaches $x_{(1)}$, so that the global maximum likelihood estimators of (μ, σ, γ) are $(-\infty, \infty, x_{(1)})$, which are inadmissible, since γ must be smaller than $x_{(1)}$. Cohen (1951) suggested using local maximum likelihood estimators (lmle's), derived by equating partial derivatives of the log-likelihood function to zero. These estimators were studied by Harter and Moore (1966), Calitz (1973), Cohen and Whitten (1980), and Griffiths (1980), and appear to possess most of the desirable properties ordinarily associated with maximum likelihood estimators.

Cohen (1951) showed that the lmle of γ is given by the solution to the following equation:

$$[\sum_{i=1}^{n}\frac{1}{w_i}]\{\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} y_i^2 + \frac{1}{n}[\sum_{i=1}^{n} y_i]^2\} - n\sum_{i=1}^{n}\frac{y_i}{w_i} = 0 \quad (4)$$

where

$$w_i = x_i - \hat{\gamma} \quad (5)$$

$$y_i = log(x_i - \hat{\gamma}) = log(w_i) \quad (6)$$

and that the lmle's of μ and σ then follow as:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i = \bar{y} \quad (7)$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \quad (8)$$

Unfortunately, while equation (4) simplifies the task of computing the lmle's, for certain data sets there still may be convergence problems (Calitz, 1973), and occasionally multiple roots of equation

(4) may exist. When multiple roots to equation (4) exisit, Cohen and Whitten (1980) recommend using the one that results in closest agreement between the mle of μ (equation (7)) and the sample mean (equation (1)).

On the other hand, Griffiths (1980) showed that for a given value of the threshold parameter γ, the maximized value of the log-likelihood (the "profile likelihood" for γ) is given by:

$$log[L(\gamma)] = \frac{-n}{2}[1 + log(2\pi) + 2\hat{\mu} + log(\hat{\sigma}^2)] \quad (9)$$

where the estimates of μ and σ are defined in equations (7) and (8), so the lmle of γ reduces to an iterative search over the values of γ. Griffiths (1980) noted that the distribution of the lmle of γ is far from normal and that $log[L(\gamma)]$ is not quadratic near the lmle of γ. He suggested a better parameterization based on

$$\eta = -log(x_{(1)} - \gamma) \quad (10)$$

Thus, once the lmle of η is found using equations (9) and (10), the lmle of γ is given by:

$$\hat{\gamma} = x_{(1)} - exp(-\hat{\eta}) \quad (11)$$

When `method="lmle"`, the function `elnorm3` uses the function `nlminb` to search for the minimum of $-2log[L(\eta)]$, using the modified method of moments estimator (`method="mmme"`; see below) as the starting value for γ. Equation (11) is then used to solve for the lmle of γ, and equation (4) is used to "fine tune" the estimated value of γ. The lmle's of μ and σ are then computed using equations (6)-(8).

Method of Moments Estimation (`method="mme"`)
Denote the r'th sample central moment by:

$$m_r = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^r \quad (12)$$

and note that

$$s_m^2 = m_2 \quad (13)$$

Equating the sample first moment (the sample mean) with its population value (the population mean), and equating the second and third sample central moments with their population values yields (Johnson et al., 1994, p.228):

$$\bar{x} = \gamma + \beta\sqrt{\omega} \quad (14)$$

$$m_2 = s_m^2 = \beta^2\omega(\omega - 1) \quad (15)$$

$$m_3 = \beta^3\omega^{3/2}(\omega - 1)^2(\omega + 2) \quad (16)$$

where

$$\beta = exp(\mu) \quad (17)$$

$$\omega = exp(\sigma^2) \quad (18)$$

Combining equations (15) and (16) yields:

$$b_1 = \frac{m_3}{m_2^{3/2}} = (\omega + 2)\sqrt{\omega - 1} \quad (19)$$

The quantity on the left-hand side of equation (19) is the usual estimator of skewness. Solving equation (19) for ω yields:

$$\hat{\omega} = (d + h)^{1/3} + (d - h)^{1/3} - 1 \quad (20)$$

where

$$d = 1 + \frac{b_1}{2} \quad (21)$$

$$h = sqrtd^2 - 1 \quad (22)$$

Using equation (18), the method of moments estimator of σ is then computed as:

$$\hat{\sigma}^2 = log(\hat{\omega}) \quad (23)$$

Combining equations (15) and (17), the method of moments estimator of μ is computed as:

$$\hat{\mu} = \frac{1}{2} log[\frac{s_m^2}{om\hat{e}ga(\hat{\omega} - 1)}] \quad (24)$$

Finally, using equations (14), (17), and (18), the method of moments estimator of γ is computed as:

$$\bar{x} - exp(\hat{mu} + \frac{\hat{\sigma}^2}{2}) \quad (25)$$

There are two major problems with using method of moments estimators for the three-parameter lognormal distribution. First, they are subject to very large sampling error due to the use of second and third sample moments (Cohen, 1988, p.121; Johnson et al., 1994, p.228). Second, Heyde (1963) showed that the lognormal distribution is not uniquely determined by its moments.

Method of Moments Estimators Using an Unbiased Estimate of Variance (`method="mmue"`)
This method of estimation is exactly the same as the method of moments (`method="mme"`), except that the unbiased estimator of variance (equation (3)) is used in place of the method of moments one (equation (4)). This modification is given in Cohen (1988, pp.119-120).

Modified Method of Moments Estimation (`method="mmme"`)
This method of estimation is described by Cohen (1988, pp.125-132). It was introduced by Cohen and Whitten (1980; their MME-II with r=1) and was further investigated by Cohen et al. (1985). It is motivated by the fact that the first order statistic in the sample, $x_{(1)}$, contains more information about the threshold parameter γ than any other observation and often more information than all of the other observations combined (Cohen, 1988, p.125).

The first two sets of equations are the same as for the modified method of moments estimators (`method="mmme"`), i.e., equations (14) and (15) with the unbiased estimator of variance (equation (3)) used in place of the method of moments one (equation (4)). The third equation replaces equation (16) by equating a function of the first order statistic with its expected value:

$$log(x_{(1)} - \gamma) = \mu + \sigma E[Z_{(1,n)}] \quad (26)$$

where $E[Z_{(i,n)}]$ denotes the expected value of the i'th order statistic in a random sample of n observations from a standard normal distribution. (See the help file for `evNormOrdStats` for information on how $E[Z_{(i,n)}]$ is computed.) Using equations (17) and (18), equation (26) can be rewritten as:

$$x_{(1)} = \gamma + \beta exp\{\sqrt{log(\omega)}\, E[Z_{(i,n)}]\} \quad (27)$$

Combining equations (14), (15), (17), (18), and (27) yields the following equation for the estimate of ω:

$$\frac{s^2}{[\bar{x} - x_{(1)}]^2} = \frac{\hat{\omega}(\hat{\omega} - 1)}{[\sqrt{\hat{\omega}} - exp\{\sqrt{log(\omega)}\, E[Z_{(i,n)}]\}]^2} \quad (28)$$

After equation (28) is solved for $\hat{\omega}$, the estimate of σ is again computed using equation (23), and the estimate of μ is computed using equation (24), where the unbiased estimate of variaince is used in place of the biased one (just as for `method="mmue"`).

Zero-Skewness Estimation (`method="zero.skew"`)
This method of estimation was introduced by Griffiths (1980), and elaborated upon by Royston (1992b). The idea is that if the threshold parameter γ were known, then the distribution of:

$$Y = log(X - \gamma) \quad (29)$$

is normal, so the skew of Y is 0. Thus, the threshold parameter γ is estimated as that value that forces the sample skew (defined in equation (19)) of the observations defined in equation (6) to be 0. That is, the zero-skewness estimator of γ is the value that satisfies the following equation:

$$0 = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^3}{[\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2]^{3/2}} \quad (30)$$

where

$$y_i = log(x_i - \hat{\gamma}) \quad (31)$$

Note that since the denominator in equation (30) is always positive (assuming there are at least two unique values in $\underline{x}$), only the numerator needs to be used to determine the value of $\hat{\gamma}$.

Once the value of $\hat{\gamma}$ has been determined, μ and σ are estimated using equations (7) and (8), except the unbiased estimator of variance is used in equation (8).

Royston (1992b) developed a modification of the Shaprio-Wilk goodness-of-fit test for normality based on tranforming the data using equation (6) and the zero-skewness estimator of γ (see `gofTest`).

Estimators Based on Royston's Index of Skewness (`method="royston.skew"`)
This method of estimation is discussed by Royston (1992b), and is similar to the zero-skewness method discussed above, except a different measure of skewness is used. Royston's (1992b) index of skewness is given by:

$$q = \frac{y_{(n)} - \tilde{y}}{\tilde{y} - y_{(1)}} \quad (32)$$

where $y_{(i)}$ denotes the i'th order statistic of y and y is defined in equation (31) above, and $\tilde{y}$ denotes the median of y. Royston (1992b) shows that the value of γ that yields a value of $q = 0$ is given by:

$$\hat{\gamma} = \frac{y_{(1)}y_{(n)} - \tilde{y}^2}{y_{(1)} + y_{(n)} - 2\tilde{y}} \quad (33)$$

Again, as for the zero-skewness method, once the value of $\hat{\gamma}$ has been determined, μ and σ are estimated using equations (7) and (8), except the unbiased estimator of variance is used in equation (8).

Royston (1992b) developed this estimator as a quick way to estimate γ.

Confidence Intervals

This section explains three different methods for constructing confidence intervals for the threshold parameter γ, or the median of the three-parameter lognormal distribution, which is given by:

$$Med[X] = \gamma + exp(\mu) = \gamma + \beta \quad (34)$$

Normal Approximation Based on Asymptotic Variances and Covariances (`ci.method="avar"`)

Formulas for asymptotic variances and covariances for the three-parameter lognormal distribution, based on the information matrix, are given in Cohen (1951), Cohen and Whitten (1980), Cohen et al., (1985), and Cohen (1988). The relevant quantities for γ and the median are:

$$Var(\hat{\gamma}) = \sigma^2_{\hat{\gamma}} = \frac{\sigma^2}{n}(\frac{\beta^2}{\omega})H \quad (35)$$

$$Var(\hat{\beta}) = \sigma^2_{\hat{\beta}} = \frac{\sigma^2}{n}\beta^2(1+H) \quad (36)$$

$$Cov(\hat{\gamma},\hat{\beta}) = \sigma_{\hat{\gamma},\hat{\beta}} = \frac{-\sigma^3}{n}(\frac{\beta^2}{\sqrt{\omega}})H \quad (37)$$

where

$$H = [\omega(1+\sigma^2) - 2\sigma^2 - 1]^{-1} \quad (38)$$

A two-sided $(1-\alpha)100\%$ confidence interval for γ is computed as:

$$\hat{\gamma} - t_{n-2,1-\alpha/2}\hat{\sigma}_{\hat{\gamma}},\ \hat{\gamma} + t_{n-2,1-\alpha/2}\hat{\sigma}_{\hat{\gamma}} \quad (39)$$

where $t_{\nu,p}$ denotes the p'th quantile of Student's t-distribution with n degrees of freedom, and the quantity $\hat{\sigma}_{\hat{\gamma}}$ is computed using equations (35) and (38) and substituting estimated values of β, ω, and σ. One-sided confidence intervals are computed in a similar manner.

A two-sided $(1-\alpha)100\%$ confidence interval for the median (see equation (34) above) is computed as:

$$\hat{\gamma} + \hat{\beta} - t_{n-2,1-\alpha/2}\hat{\sigma}_{\hat{\gamma}+\hat{\beta}},\ \hat{\gamma} + \hat{\beta} + t_{n-2,1-\alpha/2}\hat{\sigma}_{\hat{\gamma}+\hat{\beta}} \quad (40)$$

where

$$\hat{\sigma}^2_{\hat{\gamma}+\hat{\beta}} = \hat{\sigma}^2_{\hat{\gamma}} + \hat{\sigma}^2_{\hat{\beta}} + \hat{\sigma}_{\hat{\gamma},\hat{\beta}} \quad (41)$$

is computed using equations (35)-(38) and substituting estimated values of β, ω, and σ. One-sided confidence intervals are computed in a similar manner.

This method of constructing confidence intervals is analogous to using the Wald test (e.g., Silvey, 1975, pp.115-118) to test hypotheses on the parameters.

Because of the regularity problems associated with the global maximum likelihood estimators, it is questionble whether the asymptotic variances and covariances shown above apply to local maximum likelihood estimators. Simulation studies, however, have shown that these estimates of variance and covariance perform reasonably well (Harter and Moore, 1966; Cohen and Whitten, 1980).

Note that this method of constructing confidence intervals can be used with estimators other than the lmle's. Cohen and Whitten (1980) and Cohen et al. (1985) found that the asymptotic variances

and covariances are reasonably close to corresponding simulated variances and covariances for the modified method of moments estimators (`method="mmme"`).

Likelihood Profile (`ci.method="likelihood.profile"`)
Griffiths (1980) suggested constructing confidence intervals for the threshold parameter γ based on the profile likelihood function given in equations (9) and (10). Royston (1992b) further elaborated upon this procedure. A two-sided $(1-\alpha)100\%$ confidence interval for η is constructed as:

$$[\eta_{LCL}, \eta_{UCL}] \quad (42)$$

by finding the two values of η (one larger than the lmle of η and one smaller than the lmle of η) that satisfy:

$$log[L(\eta)] = log[L(\hat{\eta}_{lmle})] - \frac{1}{2}\chi^2_{1,\alpha/2} \quad (43)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-square distribution with ν degrees of freedom. Once these values are found, the two-sided confidence for γ is computed as:

$$[\gamma_{LCL}, \gamma_{UCL}] \quad (44)$$

where

$$\gamma_{LCL} = x_{(1)} - exp(-\eta_{LCL}) \quad (45)$$

$$\gamma_{UCL} = x_{(1)} - exp(-\eta_{UCL}) \quad (46)$$

One-sided intervals are construced in a similar manner.

This method of constructing confidence intervals is analogous to using the likelihood-ratio test (e.g., Silvey, 1975, pp.108-115) to test hypotheses on the parameters.

To construct a two-sided $(1-\alpha)100\%$ confidence interval for the median (see equation (34)), Royston (1992b) suggested the following procedure:

1. Construct a confidence interval for γ using the likelihood profile procedure.
2. Construct a confidence interval for β as:

$$[\beta_{LCL}, \beta_{UCL}] = [exp(\hat{\mu} - t_{n-2,1-\alpha/2}\frac{\hat{\sigma}}{n}),\ exp(\hat{\mu} + t_{n-2,1-\alpha/2}\frac{\hat{\sigma}}{n})] \quad (47)$$

3. Construct the confidence interval for the median as:

$$[\gamma_{LCL} + \beta_{LCL}, \gamma_{UCL} + \beta_{UCL}] \quad (48)$$

Royston (1992b) actually suggested using the quantile from the standard normal distribution instead of Student's t-distribution in step 2 above. The function `elnorm3`, however, uses the Student's t quantile.

Note that this method of constructing confidence intervals can be used with estimators other than the lmle's.

Royston's Confidence Interval Based on Significant Skewness (`ci.method="skewness"`)
Royston (1992b) suggested constructing confidence intervals for the threshold parameter γ based on the idea behind the zero-skewness estimator (`method="zero.skew"`). A two-sided $(1-\alpha)100\%$ confidence interval for γ is constructed by finding the two values of γ that yield a p-value of $\alpha/2$ for

the test of zero-skewness on the observations $\underline{y}$ defined in equation (6) (see gofTest). One-sided confidence intervals are constructed in a similar manner.

To construct $(1-\alpha)100\%$ confidence intervals for the median (see equation (34)), the exact same procedure is used as for `ci.method="likelihood.profile"`, except that the confidence interval for γ is based on the zero-skewness method just described instead of the likelihood profile method.

Value

a list of class `"estimate"` containing the estimated parameters and other information. See estimate.object for details.

Note

The problem of estimating the parameters of a three-parameter lognormal distribution has been extensively discussed by Aitchison and Brown (1957, Chapter 6), Calitz (1973), Cohen (1951), Cohen (1988), Cohen and Whitten (1980), Cohen et al. (1985), Griffiths (1980), Harter and Moore (1966), Hill (1963), and Royston (1992b). Stedinger (1980) and Hoshi et al. (1984) discuss fitting the three-parameter lognormal distribution to hydrologic data.

The global maximum likelihood estimates are inadmissible. In the past, several researchers have found that the local maximum likelihood estimates (lmle's) occasionally fail because of convergence problems, but they were not using the likelihood profile and reparameterization of Griffiths (1980). Cohen (1988) recommends the modified methods of moments estimators over lmle's because they are easy to compute, they are unbiased with respect to μ and σ^2 (the mean and standard deviation on the log-scale), their variances are minimal or near minimal, and they do not suffer from regularity problems.

Because the distribution of the lmle of the threshold parameter γ is far from normal for moderate sample sizes (Griffiths, 1980), it is questionable whether confidence intervals for γ or the median based on asymptotic variances and covariances will perform well. Cohen and Whitten (1980) and Cohen et al. (1985), however, found that the asymptotic variances and covariances are reasonably close to corresponding simulated variances and covariances for the modified method of moments estimators (`method="mmme"`). In a simulation study (5000 monte carlo trials), Royston (1992b) found that the coverage of confidence intervals for γ based on the likelihood profile (`ci.method="likelihood.profile"`) was very close the nominal level (94.1% for a nominal level of 95%), although not symmetric. Royston (1992b) also found that the coverage of confidence intervals for γ based on the skewness method (`ci.method="skewness"`) was also very close (95.4%) and symmetric.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special references to its uses in economics)*. Cambridge University Press, London, Chapter 5.

Calitz, F. (1973). Maximum Likelihood Estimation of the Parameters of the Three-Parameter Lognormal Distribution–a Reconsideration. *Australian Journal of Statistics* **15**(3), 185–190.

Cohen, A.C. (1951). Estimating Parameters of Logarithmic-Normal Distributions by Maximum Likelihood. *Journal of the American Statistical Association* **46**, 206–212.

Cohen, A.C. (1988). Three-Parameter Estimation. In Crow, E.L., and K. Shimizu, eds. *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, Chapter 4.

Cohen, A.C., and B.J. Whitten. (1980). Estimation in the Three-Parameter Lognormal Distribution. *Journal of the American Statistical Association* **75**, 399–404.

Cohen, A.C., B.J. Whitten, and Y. Ding. (1985). Modified Moment Estimation for the Three-Parameter Lognormal Distribution. *Journal of Quality Technology* **17**, 92–99.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, Chapter 2.

Griffiths, D.A. (1980). Interval Estimation for the Three-Parameter Lognormal Distribution via the Likelihood Function. *Applied Statistics* **29**, 58–68.

Harter, H.L., and A.H. Moore. (1966). Local-Maximum-Likelihood Estimation of the Parameters of Three-Parameter Lognormal Populations from Complete and Censored Samples. *Journal of the American Statistical Association* **61**, 842–851.

Heyde, C.C. (1963). On a Property of the Lognormal Distribution. *Journal of the Royal Statistical Society, Series B* **25**, 392–393.

Hill, .B.M. (1963). The Three-Parameter Lognormal Distribution and Bayesian Analysis of a Point-Source Epidemic. *Journal of the American Statistical Association* **58**, 72–84.

Hoshi, K., J.R. Stedinger, and J. Burges. (1984). Estimation of Log-Normal Quantiles: Monte Carlo Results and First-Order Approximations. *Journal of Hydrology* **71**, 1–30.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Royston, J.P. (1992b). Estimation, Reference Ranges and Goodness of Fit for the Three-Parameter Log-Normal Distribution. *Statistics in Medicine* **11**, 897–912.

Stedinger, J.R. (1980). Fitting Lognormal Distributions to Hydrologic Data. *Water Resources Research* **16**(3), 481–490.

See Also

Lognormal3, Lognormal, LognormalAlt, Normal.

Examples

```
# Generate 20 observations from a 3-parameter lognormal distribution
# with parameters meanlog=1.5, sdlog=1, and threshold=10, then use
# Cohen and Whitten's (1980) modified moments estimators to estimate
# the parameters, and construct a confidence interval for the
# threshold based on the estimated asymptotic variance.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnorm3(20, meanlog = 1.5, sdlog = 1, threshold = 10)
elnorm3(dat, method = "mmme", ci = TRUE)

#Results of Distribution Parameter Estimation
```

```
#------------------------------------------
#
#Assumed Distribution:            3-Parameter Lognormal
#
#Estimated Parameter(s):          meanlog   = 1.5206664
#                                 sdlog     = 0.5330974
#                                 threshold = 9.6620403
#
#Estimation Method:               mmme
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         threshold
#
#Confidence Interval Method:      Normal Approximation
#                                 Based on Asymptotic Variance
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =  6.985258
#                                 UCL = 12.338823

#----------

# Repeat the above example using the other methods of estimation
# and compare.

round(elnorm3(dat, "lmle")$parameters, 1)
#meanlog     sdlog threshold
#    1.3       0.7      10.5

round(elnorm3(dat, "mme")$parameters, 1)
#meanlog     sdlog threshold
#    2.1       0.3       6.0

round(elnorm3(dat, "mmue")$parameters, 1)
#meanlog     sdlog threshold
#    2.2       0.3       5.8

round(elnorm3(dat, "mmme")$parameters, 1)
#meanlog     sdlog threshold
#    1.5       0.5       9.7

round(elnorm3(dat, "zero.skew")$parameters, 1)
#meanlog     sdlog threshold
#    1.3       0.6      10.3

round(elnorm3(dat, "royston")$parameters, 1)
#meanlog     sdlog threshold
```

```
#    1.4       0.6      10.1

#----------

# Compare methods for computing a two-sided 95% confidence interval
# for the threshold:
# modified method of moments estimator using asymptotic variance,
# lmle using asymptotic variance,
# lmle using likelihood profile, and
# zero-skewness estimator using the skewness method.

elnorm3(dat, method = "mmme", ci = TRUE,
  ci.method = "avar")$interval$limits
#      LCL       UCL
# 6.985258 12.338823

elnorm3(dat, method = "lmle", ci = TRUE,
  ci.method = "avar")$interval$limits
#       LCL       UCL
#  9.017223 11.980107

elnorm3(dat, method = "lmle", ci = TRUE,
  ci.method="likelihood.profile")$interval$limits
#      LCL       UCL
# 3.699989 11.266029


elnorm3(dat, method = "zero.skew", ci = TRUE,
  ci.method = "skewness")$interval$limits
#      LCL       UCL
#-25.18851  11.18652

#----------

# Now construct a confidence interval for the median of the distribution
# based on using the modified method of moments estimator for threshold
# and the asymptotic variances and covariances.  Note that the true median
# is given by threshold + exp(meanlog) = 10 + exp(1.5) = 14.48169.

elnorm3(dat, method = "mmme", ci = TRUE, ci.parameter = "median")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            3-Parameter Lognormal
#
#Estimated Parameter(s):          meanlog   = 1.5206664
#                                 sdlog     = 0.5330974
#                                 threshold = 9.6620403
#
#Estimation Method:               mmme
#
#Data:                            dat
```

```
#
#Sample Size:                     20
#
#Confidence Interval for:         median
#
#Confidence Interval Method:      Normal Approximation
#                                 Based on Asymptotic Variance
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 11.20541
#                                 UCL = 17.26922

#----------

# Compare methods for computing a two-sided 95% confidence interval
# for the median:
# modified method of moments estimator using asymptotic variance,
# lmle using asymptotic variance,
# lmle using likelihood profile, and
# zero-skewness estimator using the skewness method.

elnorm3(dat, method = "mmme", ci = TRUE, ci.parameter = "median",
  ci.method = "avar")$interval$limits
#     LCL      UCL
#11.20541 17.26922

elnorm3(dat, method = "lmle", ci = TRUE, ci.parameter = "median",
  ci.method = "avar")$interval$limits
#     LCL      UCL
#12.28326 15.87233

elnorm3(dat, method = "lmle", ci = TRUE, ci.parameter = "median",
  ci.method = "likelihood.profile")$interval$limits
#      LCL       UCL
# 6.314583 16.165525

elnorm3(dat, method = "zero.skew", ci = TRUE, ci.parameter = "median",
  ci.method = "skewness")$interval$limits
#      LCL       UCL
#-22.38322  16.33569

#----------

# Clean up
#---------

rm(dat)
```

elnormAlt *Estimate Parameters of a Lognormal Distribution (Original Scale)*

Description

Estimate the mean and coefficient of variation of a lognormal distribution, and optionally construct a confidence interval for the mean.

Usage

```
elnormAlt(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "land", conf.level = 0.95, parkin.list = NULL)
```

Arguments

x	numeric vector of positive observations.
method	character string specifying the method of estimation. Possible values are "mvue" (minimum variance unbiased; the default), "qmle" (quasi maximum likelihood), "mle" (maximum likelihood), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimate of variance). See the DETAILS section for more information on these estimation methods.
ci	logical scalar indicating whether to compute a confidence interval for the mean. The default value is FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the mean. The possible values are "land" (Land's method; the default), zou (Zou et al.'s method), "parkin" (Parkin et al.'s method), "cox" (Cox's approximation), and "normal.approx" (normal approximation). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
parkin.list	a list containing arguments for the function eqnpar. The components of this list are lcl.rank (set to NULL by default), ucl.rank (set to NULL by default), ci.method (set to "exact" if the sample size is ≤ 20, otherwise set to "normal.approx"), and approx.conf.level (set to the value of conf.level). This argument is ignored unless ci=TRUE and ci.method="parkin".

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x}$ be a vector of n observations from a lognormal distribution with parameters mean=θ and cv=τ. Let η denote the standard deviation of this distribution, so that $\eta = \theta\tau$. Set $\underline{y} = log(\underline{x})$. Then $\underline{y}$ is a vector of observations from a normal distribution with parameters mean=μ and sd=σ. See the help file for LognormalAlt for the relationship between θ, τ, η, μ, and σ.

Estimation

This section explains how each of the estimators of mean=θ and cv=τ are computed. The approach is to first compute estimates of θ and η^2 (the mean and variance of the lognormal distribution), say $\hat{\theta}$ and $\hat{\eta}^2$, then compute the estimate of the cv τ by $\hat{\tau} = \hat{\eta}/\hat{\theta}$.

Minimum Variance Unbiased Estimation (method="mvue")

The minimum variance unbiased estimators (mvue's) of θ and η^2 were derived by Finney (1941) and are discussed in Gilbert (1987, pp. 164-167) and Cohn et al. (1989). These estimators are computed as:

$$\hat{\theta}_{mvue} = e^{\bar{y}} g_{n-1}(\frac{s^2}{2}) \quad (1)$$

$$\hat{\eta}^2_{mvue} = e^{2\bar{y}}\{g_{n-1}(2s^2) - g_{n-1}[\frac{(n-2)s^2}{n-1}]\} \quad (2)$$

where

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad (3)$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \quad (4)$$

$$g_m(z) = \sum_{i=0}^{\infty} \frac{m^i(m+2i)}{m(m+2)\cdots(m+2i)}(\frac{m}{m+1})^i(\frac{z^i}{i!}) \quad (5)$$

The expected value and variance of the mvue of θ are (Bradu and Mundlak, 1970; Cohn et al., 1989):

$$E[\hat{\theta}_{mvue}] = \theta \quad (6)$$

$$Var[\hat{\theta}_{mvue}] = e^{2\mu}\{e^{[(2+n-1)\sigma^2]/n} g_{n-1}(\frac{\sigma^4}{4n}) - e^{\sigma^2}\} \quad (7)$$

Maximum Likelihood Estimation (method="mle")

The maximum likelihood estimators (mle's) of θ and η^2 are given by:

$$\hat{\theta}_{mle} = exp(\bar{y} + \frac{\hat{\sigma}^2_{mle}}{2}) \quad (8)$$

$$\hat{\eta}^2_{mle} = \hat{\theta}^2_{mle}\hat{\tau}^2_{mle} \quad (9)$$

where

$$\hat{\tau}^2_{mle} = exp(\hat{\sigma}^2_{mle}) - 1 \quad (10)$$

$$\hat{\sigma}^2_{mle} = \frac{n-1}{n}s^2 \quad (11)$$

The expected value and variance of the mle of θ are (after Cohn et al., 1989):

$$E[\hat{\theta}_{mle}] = \theta exp[\frac{-(n-1)\sigma^2}{2n}](1 - \frac{\sigma^2}{n})^{-(n-1)/2} \quad (12)$$

$$Var[\hat{\theta}_{mle}] = exp(2\mu + \frac{\sigma^2}{n})\{exp(\frac{\sigma^2}{n})[1 - \frac{2\sigma^2}{n}]^{-(n-1)/2} - [1 - \frac{\sigma^2}{n}]^{-(n-1)}\} \quad (13)$$

As can be seen from equation (12), the expected value of the mle of θ does not exist when $\sigma^2 > n$. In general, the p'th moment of the mle of θ does not exist when $\sigma^2 > n/p$.

Quasi Maximum Likelihood Estimation (`method="qmle"`)
The quasi maximum likelihood estimators (qmle's; Cohn et al., 1989; Gilbert, 1987, p.167) of θ and η^2 are the same as the mle's, except the mle of σ^2 in equations (8) and (10) is replaced with the more commonly used mvue of σ^2 shown in equation (4):

$$\hat{\theta}_{qmle} = exp(\bar{y} + \frac{s^2}{2}) \quad (14)$$

$$\hat{\eta}^2_{qmle} = \hat{\theta}^2_{qmle}\hat{\tau}^2_{qmle} \quad (15)$$

$$\hat{\tau}^2_{qmle} = exp(s^2) - 1 \quad (16)$$

The expected value and variance of the qmle of θ are (Cohn et al., 1989):

$$E[\hat{\theta}_{mle}] = \theta exp[\frac{-(n-1)\sigma^2}{2n}](1 - \frac{\sigma^2}{n-1})^{-(n-1)/2} \quad (17)$$

$$Var[\hat{\theta}_{mle}] = exp(2\mu + \frac{\sigma^2}{n})\{exp(\frac{\sigma^2}{n})[1 - \frac{2\sigma^2}{n-1}]^{-(n-1)/2} - [1 - \frac{\sigma^2}{n-1}]^{-(n-1)}\} \quad (18)$$

As can be seen from equation (17), the expected value of the qmle of θ does not exist when $\sigma^2 > (n-1)$. In general, the p'th moment of the mle of θ does not exist when $\sigma^2 > (n-1)/p$.

Note that Gilbert (1987, p. 167) incorrectly presents equation (12) rather than equation (17) as the expected value of the qmle of θ. For large values of n relative to σ^2, however, equations (12) and (17) are virtually identical.

Method of Moments Estimation (`method="mme"`)
The method of moments estimators (mme's) of θ and η^2 are found by equating the sample mean and variance with their population values:

$$\hat{\theta}_{mme} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (19)$$

$$\hat{\eta}_{mme} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (20)$$

Note that the estimator of variance in equation (20) is biased.

The expected value and variance of the mme of θ are:

$$E[\hat{\theta}_{mme}] = \theta \quad (21)$$

$$Var[\hat{\theta}_{mme}] = \frac{\eta^2}{n} = \frac{1}{n}exp(2\mu + \sigma^2)[exp(\sigma^2) - 1] \quad (22)$$

Method of Moments Estimation Based on the Unbiased Estimate of Variance (`method="mmue"`)
These estimators are exactly the same as the method of moments estimators described above, except that the usual unbiased estimate of variance is used:

$$\hat{\theta}_{mmue} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (23)$$

$$\hat{\eta}_{mmue} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (24)$$

Since the mmue of θ is equivalent to the mme of θ, so are its mean and varaince.

Confidence Intervals

This section explains the different methods for constructing confidence intervals for θ, the mean of the lognormal distribution.

Land's Method (`ci.method="land"`)

Land (1971, 1975) derived a method for computing one-sided (lower or upper) uniformly most accurate unbiased confidence intervals for θ. A two-sided confidence interval can be constructed by combining an optimal lower confidence limit with an optimal upper confidence limit. This procedure for two-sided confidence intervals is only asymptotically optimal, but for most purposes should be acceptable (Land, 1975, p.387).

As shown in equation (3) in the help file for LognormalAlt, the mean θ of a lognormal random variable is related to the mean μ and standard deviation σ of the log-transformed random variable by the following relationship:

$$\theta = e^{\beta} \quad (25)$$

where

$$\beta = \mu + \frac{\sigma^2}{2} \quad (26)$$

Land (1971) developed confidence bounds for the quantity β. The mvue of β is given by:

$$\hat{\beta}_{mvue} = \bar{y} + \frac{s^2}{2} \quad (27)$$

Note that $\hat{\theta}_{qmle} = exp(\hat{\beta}_{mvue})$. The $(1-\alpha)100\%$ two-sided confidence interval for β is given by:

$$[\hat{\beta}_{mvue} + s\frac{C_{\alpha/2}}{\sqrt{n-1}},\ \hat{\beta}_{mvue} + s\frac{C_{1-\alpha/2}}{\sqrt{n-1}}] \quad (28)$$

the $(1-\alpha)100\%$ one-sided upper confidence interval for β is given by:

$$[-\infty,\ \hat{\beta}_{mvue} + s\frac{C_{1-\alpha}}{\sqrt{n-1}}] \quad (29)$$

and the $(1-\alpha)100\%$ one-sided lower confidence interval for β is given by:

$$[\hat{\beta}_{mvue} + s\frac{C_{\alpha}}{\sqrt{n-1}},\ \infty] \quad (30)$$

where s is the estimate of σ (see equation (4) above), and the factor C is given in tables in Land (1975).

Thus, by equations (25)-(30), the two-sided $(1-\alpha)100\%$ confidence interval for θ is given by:

$$\{\hat{\theta}_{qmle}exp[s\frac{C_{\alpha/2}}{\sqrt{n-1}}],\ \hat{\theta}_{qmle}exp[s\frac{C_{1-\alpha/2}}{\sqrt{n-1}}]\} \quad (31)$$

the $(1-\alpha)100\%$ one-sided upper confidence interval for θ is given by:

$$\{0,\ \hat{\theta}_{qmle}exp[s\frac{C_{1-\alpha}}{\sqrt{n-1}}]\} \quad (32)$$

and the $(1-\alpha)100\%$ one-sided lower confidence interval for θ is given by:

$$\{\hat{\theta}_{qmle} exp[s\frac{C_\alpha}{\sqrt{n-1}}],\ \infty\} \quad (33)$$

Note that Gilbert (1987, pp. 169-171, 264-265) denotes the quantity C above as H and reproduces a subset of Land's (1975) tables. Some guidance documents (e.g., USEPA, 1992d) refer to this quantity as the H-statistic.

Zou et al.'s Method (`ci.method="zou"`)
Zou et al. (2009) proposed the following approximation for the two-sided $(1-\alpha)100\%$ confidence intervals for θ. The lower limit LL is given by:

$$LL = \hat{\theta}_{qmle} exp\{-[\frac{z^2_{1-\alpha/2}s^2}{n} + (\frac{s^2}{2} - \frac{(n-1)s^2}{2\chi^2_{1-\alpha/2,n-1}})^2]^{1/2}\} \quad (34)$$

and the upper limit UL is given by:

$$UL = \hat{\theta}_{qmle} exp\{[\frac{z^2_{1-\alpha/2}s^2}{n} + (\frac{(n-1)s^2}{2\chi^2_{\alpha/2,n-1}} - \frac{s^2}{2})^2]^{1/2}\} \quad (35)$$

where z_p denotes the p'th quantile of the standard normal distribuiton, and $\chi_{p,\nu}$ denotes the p'th quantile of the chi-square distribution with ν degrees of freedom. The $(1-\alpha)100\%$ one-sided lower confidence limit and one-sided upper confidence limit are given by equations (34) and (35), respectively, with $\alpha/2$ replaced by α.

Parkin et al.'s Method (`ci.method="parkin"`)
This method was developed by Parkin et al. (1990). It can be shown that the mean of a lognormal distribution corresponds to the p'th quantile, where

$$p = \Phi(\frac{\sigma}{2}) \quad (36)$$

and Φ denotes the cumulative distribution function of the standard normal distribution. Parkin et al. (1990) suggested estimating p by replacing σ in equation (36) with the estimate s as computed in equation (4). Once an estimate of p is obtained, a nonparametric confidence interval can be constructed for p, assuming p is equal to its estimated value (see `eqnpar`).

Cox's Method (`ci.method="cox"`)
This method was suggested by Professor D.R. Cox and is illustrated in Land (1972). El-Shaarawi (1989) adapts this method to the case of censored water quality data. Cox's idea is to construct an approximate $(1-\alpha)100\%$ confidence interval for the quantity β defined in equation (26) above assuming the estimate of β is approximately normally distributed, and then exponentiate the confidence limits. That is, a two-sided $(1-\alpha)100\%$ confidence interval for θ is constructed as:

$$[exp(\hat{\beta} - t_{1-\alpha/2,n-1}\hat{\sigma}_{\hat{\beta}}),\ exp(\hat{\beta} + t_{1-\alpha/2,n-1}\hat{\sigma}_{\hat{\beta}})] \quad (37)$$

where $t(p,\nu)$ denotes the p'th quantile of Student's t-distribution with ν degrees of freedom. Note that this method, unlike the normal approximation method discussed below, guarantees a positive value for the lower confidence limit. One-sided confidence intervals are computed in a similar fashion.

Define an estimator of β by:

$$\hat{\beta} = \hat{\mu} + \frac{\hat{\sigma}^2}{2} \quad (38)$$

Then the variance of this estimator is given by:

$$Var(\hat{\beta}) = Var(\hat{\mu}) + Cov(\hat{\mu}, \hat{\sigma}^2) + \frac{1}{4}Var(\hat{\sigma}^2) \quad (39)$$

The function `elnormAlt` follows Land (1972) and uses the minimum variance unbiased estimator for β shown in equation (27) above, so the variance and estimated variance of this estimator are:

$$Var(\hat{\beta}_{mvue}) = \frac{\sigma^2}{n} + \frac{\sigma^4}{2(n-1)} \quad (40)$$

$$\hat{\sigma}^2_{\hat{\beta}} = \frac{s^2}{n} + \frac{s^4}{2(n+1)} \quad (41)$$

Note that El-Shaarawi (1989, equation 5) simply replaces the value of s^2 in equation (41) with some estimator of σ^2 (the mle or mvue of σ^2), rather than using the mvue of the variance of β as shown in equation (41).

Normal Approximation (`ci.method="normal.approx"`) This method constructs approximate $(1-\alpha)100\%$ confidence intervals for θ based on the assumption that the estimator of θ is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$ confidence interval for θ is constructed as:

$$[\hat{\theta} - t_{1-\alpha/2,n-1}\hat{\sigma}_{\hat{\theta}},\ \hat{\theta} + t_{1-\alpha/2,n-1}\hat{\sigma}_{\hat{\theta}}] \quad (42)$$

One-sided confidence intervals are computed in a similar fashion.

When `method="mvue"` is used to estimate θ, an unbiased estimate of the variance of the estimator of θ is used in equation (42) (Bradu and Mundlak, 1970, equation 4.3; Gilbert, 1987, equation 13.5):

$$\hat{\sigma^2}_{\hat{\theta}} = e^{2\bar{y}}\{[g_{n-1}(\frac{s^2}{2})]^2 - g_{n-1}[\frac{s^2(n-2)}{n-1}]\} \quad (43)$$

When `method="mle"` is used to estimate θ, the estimate of the variance of the estimator of θ is computed by replacing μ and σ^2 in equation (13) with their mle's:

$$\hat{\sigma}^2_{\hat{\theta}} = exp(2\bar{y} + \frac{\hat{\sigma}^2_{mle}}{n})\{exp(\frac{\hat{\sigma}^2_{mle}}{n})[1 - \frac{2\hat{\sigma}^2_{mle}}{n}]^{-(n-1)/2} - [1 - \frac{\hat{\sigma}^2_{mle}}{n}]^{-(n-1)}\} \quad (44)$$

When `method="qmle"` is used to estimate θ, the estimate of the variance of the estimator of θ is computed by replacing μ and σ^2 in equation (18) with their mvue's:

$$\hat{\sigma}^2_{\hat{\theta}} = exp(2\bar{y} + \frac{s^2}{n})\{exp(\frac{s^2}{n})[1 - \frac{2s^2}{n-1}]^{-(n-1)/2} - [1 - \frac{s^2}{n-1}]^{-(n-1)}\} \quad (45)$$

Note that equation (45) is exactly the same as Gilbert's (1987, p. 167) equation 13.8a, except that Gilbert (1987) erroneously uses n where he should use $n-1$ instead. For large values of n relative to s^2, however, this makes little difference.

When `method="mme"`, the estimate of the variance of the estimator of θ is computed by replacing eta^2 in equation (22) with the mme of η^2 defined in equation (20):

$$\hat{\sigma}^2_{\hat{\theta}} = \frac{\hat{\eta}^2_{mme}}{n} = \frac{1}{n^2}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (46)$$

When `method="mmue"`, the estimate of the variance of the estimator of θ is computed by replacing eta^2 in equation (22) with the mmue of η^2 defined in equation (24):

$$\hat{\sigma}^2_{\hat{\theta}} = \frac{\hat{\eta}^2_{mmue}}{n} = \frac{1}{n(n-1)}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (47)$$

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean or variance. This is done with confidence intervals.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribuiton instead.

USEPA (1992d) directs persons involved in risk assessment for Superfund sites to use Land's (1971, 1975) method (ci.method="land") for computing the upper 95% confidence interval for the mean, assuming the data follow a lognormal distribution (the guidance document cites Gilbert (1987) as a source of descriptions and tables for this method). The last example in the EXAMPLES section below reproduces an example from this guidance document.

In the past, some authors suggested using the geometric mean, also called the "rating curve" estimator (Cohn et al., 1989), as the estimator of the mean, θ. This estimator is computed as:

$$\hat{\theta}_{rc} = e^{\bar{y}} \quad (48)$$

Cohn et al. (1989) cite several authors who have pointed out this estimator is biased and is not even a consistent estimator of the mean. In fact, it is the maximum likelihood estimator of the median of the distribution (see eqlnorm.)

Finney (1941) computed the efficiency of the method of moments estimators of the mean (θ) and variance (η^2) of the lognormal distribution (equations (19)-(20)) relative to the mvue's (equations (1)-(2)) as a function of σ^2 (the variance of the log-transformed observations), and found that while the mme of θ is reasonably efficient compared to the mvue of θ, the mme of η^2 performs quite poorly relative to the mvue of η^2.

Cohn et al. (1989) and Parkin et al. (1988) have shown that the qmle and the mle of the mean can be severely biased for typical environmental data, and suggest always using the mvue.

Parkin et al. (1990) studied the performance of various methods for constructing a confidence interval for the mean via Monte Carlo simulation. They compared approximate methods to Land's optimal method (ci.method="land"). They used four parent lognormal distributions to generate observations; all had mean 10, but differed in coefficient of variation: 50, 100, 200, and 500%. They also generated sample sizes from 6 to 100 in increments of 2. For each combination of parent distribution and sample size, they generated 25,000 Monte Carlo trials. Parkin et al. found that for small sample sizes ($n < 20$), none of the approximate methods ("parkin", "cox", "normal.approx") worked very well. For $n > 20$, their method ("parkin") provided reasonably accurate coverage. Cox's method ("cox") worked well for $n > 60$, and performed slightly better than Parkin et al.'s method ("parkin") for highly skewed populations.

Zou et al. (2009) used Monte Carlo simulation to compare the performance of their method with the CGI method of Krishnamoorthy and Mathew (2003) and the modified Cox method of Armstrong

(1992) and El-Shaarawi and Lin (2007). Performance was assessed based on 1) percentage of times the interval contained the parameter value (coverage%), 2) balance between left and right tail errors, and 3) confidence interval width. All three methods showed acceptable coverage percentages. The modified Cox method showed unbalanced tail errors, and Zou et al.'s method showed consistently narrower average width.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special references to its uses in economics)*. Cambridge University Press, London, Chapter 5.

Armstrong, B.G. (1992). Confidence Intervals for Arithmetic Means of Lognormally Distributed Exposures. *American Industrial Hygiene Association Journal* **53**, 481–485.

Bradu, D., and Y. Mundlak. (1970). Estimation in Lognormal Linear Models. *Journal of the American Statistical Association* **65**, 198–211.

Cohn, T.A., L.L. DeLong, E.J. Gilroy, R.M. Hirsch, and D.K. Wells. (1989). Estimating Constituent Loads. *Water Resources Research* **25**(5), 937–942.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, Chapter 2.

El-Shaarawi, A.H., and J. Lin. (2007). Interval Estimation for Log-Normal Mean with Applications to Water Quality. *Environmetrics* **18**, 1–10.

El-Shaarawi, A.H., and R. Viveros. (1997). Inference About the Mean in Log-Regression with Environmental Applications. *Environmetrics* **8**, 569–582.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Finney, D.J. (1941). On the Distribution of a Variate Whose Logarithm is Normally Distributed. *Supplement to the Journal of the Royal Statistical Society* **7**, 155–161.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Krishnamoorthy, K., and T.P. Mathew. (2003). Inferences on the Means of Lognormal Distributions Using Generalized p-Values and Generalized Confidence Intervals. *Journal of Statistical Planning and Inference* **115**, 103–121.

Land, C.E. (1971). Confidence Intervals for Linear Functions of the Normal Mean and Variance. *The Annals of Mathematical Statistics* **42**(4), 1187–1205.

Land, C.E. (1972). An Evaluation of Approximate Confidence Interval Estimation Methods for Lognormal Means. *Technometrics* **14**(1), 145–158.

Land, C.E. (1973). Standard Confidence Limits for Linear Functions of the Normal Mean and Variance. *Journal of the American Statistical Association* **68**(344), 960–963.

Land, C.E. (1975). Tables of Confidence Limits for Linear Functions of the Normal Mean and Variance, in *Selected Tables in Mathematical Statistics, Vol. III*. American Mathematical Society, Providence, RI, pp. 385–419.

Likes, J. (1980). Variance of the MVUE for Lognormal Variance. *Technometrics* **22**(2), 253–258.

Limpert, E., W.A. Stahel, and M. Abbt. (2001). Log-Normal Distributions Across the Sciences: Keys and Clues. *BioScience* **51**, 341–352.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Parkin, T.B., J.J. Meisinger, S.T. Chester, J.L. Starr, and J.A. Robinson. (1988). Evaluation of Statistical Estimation Methods for Lognormally Distributed Variables. *Journal of the Soil Science Society of America* **52**, 323–329.

Parkin, T.B., S.T. Chester, and J.A. Robinson. (1990). Calculating Confidence Intervals for the Mean of a Lognormally Distributed Variable. *Journal of the Soil Science Society of America* **54**, 321–326.

Singh, A., A.K. Singh, and R.J. Iaci. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (1992d). *Supplemental Guidance to RAGS: Calculating the Concentration Term*. Publication 9285.7-081, May 1992. Intermittenet Bulletin, Volume 1, Number 1. Office of Emergency and Remedial Response, Hazardous Site Evaluation Division, OS-230. Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

Zou, G.Y., C.Y. Huo, and J. Taleban. (2009). Simple Confidence Intervals for Lognormal Means and their Differences with Environmental Applications. *Environmetrics* **20**, 172–180.

See Also

LognormalAlt, Lognormal, Normal.

Examples

```
# Using the Reference area TcCB data in the data frame EPA.94b.tccb.df,
# estimate the mean and coefficient of variation,
# and construct a 95% confidence interval for the mean.
```

```
with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], ci = TRUE))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          mean = 0.5989072
#                                 cv   = 0.4899539
#
#Estimation Method:               mvue
#
#Data:                            TcCB[Area == "Reference"]
#
#Sample Size:                     47
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Land
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 0.5243787
#                                 UCL = 0.7016992

#----------

# Compare the different methods of estimating the distribution parameters using the
# Reference area TcCB data.

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mvue"))$parameters
#     mean        cv
#0.5989072 0.4899539

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "qmle"))$parameters
#     mean        cv
#0.6004468 0.4947791

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mle"))$parameters
#     mean        cv
#0.5990497 0.4888968

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mme"))$parameters
#     mean        cv
#0.5985106 0.4688423

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"], method = "mmue"))$parameters
#     mean        cv
#0.5985106 0.4739110
```

```
#----------

# Compare the different methods of constructing the confidence interval for
# the mean using the Reference area TcCB data.

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
  method = "mvue", ci = TRUE, ci.method = "land"))$interval$limits
#      LCL       UCL
#0.5243787 0.7016992

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
  method = "mvue", ci = TRUE, ci.method = "zou"))$interval$limits
#      LCL       UCL
#0.5230444 0.6962071

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
  method = "mvue", ci = TRUE, ci.method = "parkin"))$interval$limits
# LCL  UCL
#0.50 0.74

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
   method = "mvue", ci = TRUE, ci.method = "cox"))$interval$limits
#      LCL       UCL
#0.5196213 0.6938444

with(EPA.94b.tccb.df, elnormAlt(TcCB[Area == "Reference"],
   method = "mvue", ci = TRUE, ci.method = "normal.approx"))$interval$limits
#      LCL       UCL
#0.5130160 0.6847984

#----------

# Reproduce the example in Highlights 7 and 8 of USEPA (1992d).  This example shows
# how to compute the upper 95% confidence limit of the mean of a lognormal distribution
# and compares it to the result of computing the upper 95% confidence limit assuming a
# normal distribution. The data for this example are chromium concentrations (mg/kg) in
# soil samples collected randomly over a Superfund site, and are stored in the data frame
# EPA.92d.chromium.vec.

# First look at the data

EPA.92d.chromium.vec
# [1]   10   13   20   36   41   59   67  110  110  136  140  160  200  230 1300

stripChart(EPA.92d.chromium.vec, ylab = "Chromium (mg/kg)")

# Note there is one very large "outlier" (1300).
# Perform a goodness-of-fit test to determine whether a lognormal distribution
# is appropriate:

gof.list <- gofTest(EPA.92d.chromium.vec, dist = 'lnormAlt')
gof.list
```

```
#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#
#Hypothesized Distribution:       Lognormal
#
#Estimated Parameter(s):          mean = 159.855185
#                                 cv   =   1.493994
#
#Estimation Method:               mvue
#
#Data:                            EPA.92d.chromium.vec
#
#Sample Size:                     15
#
#Test Statistic:                  W = 0.9607179
#
#Test Statistic Parameter:        n = 15
#
#P-value:                         0.7048747
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.

plot(gof.list, digits = 2)

# The lognormal distribution seems to provide an adequate fit, although the largest
# observation (1300) is somewhat suspect, and given the small sample size there is
# not much power to detect any kind of mild deviation from a lognormal distribution.

# Now compute the one-sided 95% upper confidence limit for the mean.
# Note that the value of 502 mg/kg shown in Hightlight 7 of USEPA (1992d) is a bit
# larger than the exact value of 496.6 mg/kg shown below.
# This is simply due to rounding error.

elnormAlt(EPA.92d.chromium.vec, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:          Lognormal
#
#Estimated Parameter(s):        mean = 159.855185
#                               cv   =   1.493994
#
#Estimation Method:             mvue
#
#Data:                          EPA.92d.chromium.vec
#
#Sample Size:                   15
#
#Confidence Interval for:       mean
```

```
#
#Confidence Interval Method:    Land
#
#Confidence Interval Type:      upper
#
#Confidence Level:              95%
#
#Confidence Interval:           LCL =   0
#                               UCL = 496.6282

# Now compare this result with the upper 95% confidence limit based on assuming
# a normal distribution.  Again note that the value of 325 mg/kg shown in
# Hightlight 8 is slightly larger than the exact value of 320.3 mg/kg shown below.
# This is simply due to rounding error.

enorm(EPA.92d.chromium.vec, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 175.4667
#                                 sd   = 318.5440
#
#Estimation Method:               mvue
#
#Data:                            EPA.92d.chromium.vec
#
#Sample Size:                     15
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =     -Inf
#                                 UCL = 320.3304

#----------

# Clean up
#---------

rm(gof.list)
```

`elnormAltCensored` *Estimate Parameters for a Lognormal Distribution (Original Scale) Based on Type I Censored Data*

Description

Estimate the mean and coefficient of variation of a lognormal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

```
elnormAltCensored(x, censored, method = "mle", censoring.side = "left",
  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z", ...)
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`censored`	numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is `"logical"`, `TRUE` values correspond to elements of `x` that are censored, and `FALSE` values correspond to elements of `x` that are not censored. If the mode of `censored` is `"numeric"`, it must contain only `1`'s and `0`'s; `1` corresponds to `TRUE` and `0` corresponds to `FALSE`. Missing (`NA`) values are allowed but will be removed.
`method`	character string specifying the method of estimation. For singly censored data, the possible values are: `"mle"` (maximum likelihood; the default), `"qmvue"` (quasi minimum variance unbiased estimation), `"bcmle"` (bias-corrected maximum likelihood), `"rROS"` or `"impute.w.qq.reg"` (moment estimation based on imputation using quantile-quantile regression; also called *robust regression on order statistics* and abbreviated rROS), `"impute.w.qq.reg.w.cen.level"` (moment estimation based on imputation using the `qq.reg.w.cen.level` method), `"impute.w.mle"` (moment estimation based on imputation using the mle), and `"half.cen.level"` (moment estimation based on setting the censored observations to half the censoring level). For multiply censored data, the possible values are: `"mle"` (maximum likelihood; the default), `"qmvue"` (quasi minimum variance unbiased estimation), `"bcmle"` (bias-corrected maximum likelihood), `"rROS"` or `"impute.w.qq.reg"` (moment estimation based on imputation using quantile-quantile regression; also called *robust regression on order statistics* and abbreviated rROS), and `"half.cen.level"` (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

`censoring.side` character string indicating on which side the censoring occurs. The possible values are `"left"` (the default) and `"right"`.

`ci` logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is `ci=FALSE`.

`ci.method` character string indicating what method to use to construct the confidence interval for the mean. The possible values are `"profile.likelihood"` (profile likelihood; the default),
`"cox"` (Cox's approximation),
`"delta"` (normal approximation based on the delta method),
`"normal.approx"` (normal approximation), and
`"bootstrap"` (based on bootstrapping).

The confidence interval method `"profile.likelihood"` is valid only when `method="mle"`.

The confidence interval methods `"delta"` and `"cox"` are valid only when `method` is one of `"mle"`, `"bcmle"`, or `"qmvue"`.

The confidence interval method `"normal.approx"` is valid only when `method` is one of `"rROS"`, `"impute.w.qq.reg"`, `"impute.w.qq.reg.w.cen.level"`, `"impute.w.mle"`, or `"half.cen.level"`.

See the DETAILS section for more information. This argument is ignored if `ci=FALSE`.

`ci.type` character string indicating what kind of confidence interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`. This argument is ignored if `ci=FALSE`.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

`n.bootstraps` numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when `ci.type="bootstrap"`. This argument is ignored if `ci=FALSE` and/or `ci.method` does not equal `"bootstrap"`.

`pivot.statistic`

character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when `ci.method` is equal to `"delta"`, `"cox"`, or `"normal.approx"` (see the DETAILS section). The possible values are `pivot.statistic="z"` (the default) and `pivot.statistic="t"`. When `pivot.statistic="t"` you may supply the argument `ci.sample size` (see below). The argument `pivot.statistic` is ignored if `ci=FALSE`.

`...` additional arguments to pass to other functions.

- `prob.method`. Character string indicating what method to use to compute the plotting positions (empirical probabilities) when `method` is one of `"rROS"`, `"impute.w.qq.reg"`, `"impute.w.qq.reg.w.cen.level"`, or `"impute.w.mle"`. Possible values are `"kaplan-meier"` (product-limit method of Kaplan and Meier (1958)), `"nelson"` (hazard plotting method of Nelson (1972)), `"michael-schucany"` (generalization of the product-limit method due to Michael and Schucany (1986)), and `"hirsch-stedinger"` (generalization of the product-limit method due to Hirsch and Stedinger (1987)).

The default value is prob.method="hirsch-stedinger". The "nelson" method is only available for censoring.side="right". See the DETAILS section and the help file for ppointsCensored for more information.

- plot.pos.con. Numeric scalar between 0 and 1 containing the value of the plotting position constant to use when method is one of "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", or "impute.w.mle". The default value is plot.pos.con=0.375. See the DETAILS section and the help file for ppointsCensored for more information.
- ci.sample.size. Numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method is equal to "delta", "cox", or "normal.approx". When method equals "mle", "bcmle", or "qmvue", the default value is the expected number of uncensored observations, otherwise it is the observed number of uncensored observations.
- lb.impute. Numeric scalar indicating the lower bound for imputed observations when method is one of "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", or "impute.w.mle". Imputed values smaller than this value will be set to this value. The default is lb.impute=-Inf.
- ub.impute. Numeric scalar indicating the upper bound for imputed observations when method is one of "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", or "impute.w.mle". Imputed values larger than this value will be set to this value. The default is ub.impute=Inf.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x}$ be a vector of n observations from a lognormal distribution with parameters mean=θ and cv=τ. Let η denote the standard deviation of this distribution, so that $\eta = \theta\tau$. Set $\underline{y} = log(\underline{x})$. Then $\underline{y}$ is a vector of observations from a normal distribution with parameters mean=μ and sd=σ. See the help file for LognormalAlt for the relationship between θ, τ, η, μ, and σ.

Let $\underline{x}$ denote a vector of N observations from a lognormal distribution with parameters mean=θ and cv=τ. Let η denote the standard deviation of this distribution, so that $\eta = \theta\tau$. Set $\underline{y} = log(\underline{x})$. Then $\underline{y}$ is a vector of observations from a normal distribution with parameters mean=μ and sd=σ. See the help file for LognormalAlt for the relationship between θ, τ, η, μ, and σ.

Assume n ($0 < n < N$) of the N observations are known and c ($c = N - n$) of the observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_k;\ k \geq 1 \qquad (1)$$

For the case when $k \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $k = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j =$

$1, 2, \ldots, k$, so that

$$\sum_{i=1}^{k} c_j = c \qquad (2)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations.

ESTIMATION

This section explains how each of the estimators of `mean`=θ and `cv`=τ are computed. The approach is to first compute estimates of θ and η^2 (the mean and variance of the lognormal distribution), say $\hat{\theta}$ and $\hat{\eta}^2$, then compute the estimate of the cv τ by $\hat{\tau} = \hat{\eta}/\hat{\theta}$.

Maximum Likelihood Estimation (`method="mle"`)
The maximum likelihood estimators of θ, τ, and η are computed as:

$$\hat{\theta}_{mle} = exp(\hat{\mu}_{mle} + \frac{\hat{\sigma}^2_{mle}}{2}) \qquad (3)$$

$$\hat{\tau}_{mle} = [exp(\hat{\sigma}^2_{mle}) - 1]^{1/2} \qquad (4)$$

$$\hat{\eta}_{mle} = \hat{\theta}_{mle}\ \hat{\tau}_{mle} \qquad (5)$$

where $\hat{\mu}_{mle}$ and $\hat{\sigma}_{mle}$ denote the maximum likelihood estimators of μ and σ. See the help for for `enormCensored` for information on how $\hat{\mu}_{mle}$ and $\hat{\sigma}_{mle}$ are computed.

Quasi Minimum Variance Unbiased Estimation Based on the MLE's (`method="qmvue"`)
The maximum likelihood estimators of θ and η^2 are biased. Even for complete (uncensored) samples these estimators are biased (see equation (12) in the help file for `elnormAlt`). The bias tends to 0 as the sample size increases, but it can be considerable for small sample sizes. (Cohn et al., 1989, demonstrate the bias for complete data sets.) For the case of complete samples, the minimum variance unbiased estimators (mvue's) of θ and η^2 were derived by Finney (1941) and are discussed in Gilbert (1987, pp.164-167) and Cohn et al. (1989). These estimators are computed as:

$$\hat{\theta}_{mvue} = e^{\bar{y}} g_{n-1}(\frac{s^2}{2}) \qquad (6)$$

$$\hat{\eta}^2_{mvue} = e^{2\bar{y}}\{g_{n-1}(2s^2) - g_{n-1}[\frac{(n-2)s^2}{n-1}]\} \qquad (7)$$

where

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad (8)$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \qquad (9)$$

$$g_m(z) = \sum_{i=0}^{\infty}\frac{m^i(m+2i)}{m(m+2)\cdots(m+2i)}(\frac{m}{m+1})^i(\frac{z^i}{i!}) \qquad (10)$$

(see the help file for `elnormAlt`).

For Type I censored samples, the quasi minimum variance unbiased estimators (qmvue's) of θ and η^2 are computed using equations (6) and (7) and estimating μ and σ with their mle's (see `elnormCensored`).

For singly censored data, this is apparently the LM method of Gilliom and Helsel (1986, p.137) (it is not clear from their description on page 137 whether their LM method is the straight `method="mle"` described above or `method="qmvue"` described here). This method was also used by Newman et al. (1989, p.915, equations 10-11).

For multiply censored data, this is apparently the MM method of Helsel and Cohn (1988, p.1998). (It is not clear from their description on page 1998 and the description in Gilliom and Helsel, 1986, page 137 whether Helsel and Cohn's (1988) MM method is the straight `method="mle"` described above or `method="qmvue"` described here.)

Bias-Corrected Maximum Likelihood Estimation (`method="bcmle"`)
This method was derived by El-Shaarawi (1989) and can be applied to complete or censored data sets. For complete data, the exact relative bias of the mle of the mean θ is given as:

$$B_{mle} = \frac{E[\hat{\theta}_{mle}]}{\theta} = exp[\frac{-(n-1)\sigma^2}{2n}](1 - \frac{\sigma^2}{n})^{-(n-1)/2} \qquad (11)$$

(see equation (12) in the help file for `elnormAlt`).

For the case of complete or censored data, El-Shaarawi (1989) proposed the following "bias-corrected" maximum likelihood estimator:

$$\hat{\theta}_{bcmle} = \frac{\hat{\theta}_{mle}}{\hat{B}_{mle}} \qquad (12)$$

where

$$\hat{B}_{mle} = exp[\frac{1}{2}(\hat{V}_{11} + 2\hat{\sigma}_{mle}\hat{V}_{12} + \hat{\sigma}^2_{mle}\hat{V}_{22})] \qquad (13)$$

and V denotes the asymptotic variance-covariance of the mle's of μ and σ, which is based on the observed information matrix, formulas for which are given in Cohen (1991). El-Shaarawi (1989) does not propose a bias-corrected estimator of the variance η^2, so the mle of η is computed when `method="bcmle"`.

Robust Regression on Order Statistics (`method="rROS"`) or
Imputation Using Quantile-Quantile Regression (`method ="impute.w.qq.reg"`)
This is the robust Regression on Order Statistics (rROS) method discussed in USEPA (2009) and Helsel (2012). This method involves using quantile-quantile regression on the log-transformed observations to fit a regression line (and thus initially estimate the mean μ and standard deviation σ in log-space), imputing the log-transformed values of the c censored observations by predicting them from the regression equation, transforming the log-scale imputed values back to the original

scale, and then computing the method of moments estimates of the mean and standard deviation based on the observed and imputed values.

The steps are:

1. Estimate μ and σ by computing the least-squares estimates in the following model:

$$y_{(i)} = \mu + \sigma\Phi^{-1}(p_i) + \epsilon_i, \; i \in \Omega \qquad (14)$$

   where p_i denotes the plotting position associated with the i'th largest value, a is a constant such that $0 \leq a \leq 1$ (the default value is 0.375), Φ denotes the cumulative distribution function (cdf) of the standard normal distribution and Ω denotes the set of n subscripts associated with the uncensored observations in the ordered sample. The plotting positions are computed by calling the function `ppointsCensored`.

2. Compute the log-scale imputed values as:

$$\hat{y}_{(i)} = \hat{\mu}_{qqreg} + \hat{\sigma}_{qqreg}\Phi^{-1}(p_i), \; i \notin \Omega \qquad (15)$$

3. Retransform the log-scale imputed values:

$$\hat{x}_{(i)} = exp[\hat{y}_{(i)}], \; i \notin \Omega \qquad (16)$$

4. Compute the usual method of moments estimates of the mean and variance.

$$\hat{\theta} = \frac{1}{N}[\sum_{i \notin \Omega} \hat{x}_{(i)} + \sum_{i \in \Omega} x_{(i)}] \qquad (17)$$

$$\hat{\eta}^2 = \frac{1}{N-1}[\sum_{i \notin \Omega} (\hat{x}_{(i)} - \hat{\theta}^2) + \sum_{i \in \Omega} (x_{(i)} - \hat{\theta}^2)] \qquad (18)$$

   Note that the estimate of variance is actually the usual unbiased one (not the method of moments one) in the case of complete data.

For sinlgy censored data, this method is discussed by Hashimoto and Trussell (1983), Gilliom and Helsel (1986), and El-Shaarawi (1989), and is referred to as the LR (Log-Regression) or Log-Probability Method.

For multiply censored data, this is the MR method of Helsel and Cohn (1988, p.1998). They used it with the probability method of Hirsch and Stedinger (1987) and Weibull plotting positions (i.e., `prob.method="hirsch-stedinger"` and `plot.pos.con=0`).

The argument `plot.pos.con` (see the entry for `...` in the ARGUMENTS section above) determines the value of the plotting positions computed in equations (14) and (15) when `method` equals `"hirsch-stedinger"` or `"michael-schucany"`. The default value is `plot.pos.con=0.375`. See the help file for `ppointsCensored` for more information.

The arguments `lb.impute` and `ub.impute` (see the entry for `...` in the ARGUMENTS section above) determine the lower and upper bounds for the imputed values. Imputed values smaller than

`lb.impute` are set to this value. Imputed values larger than `ub.impute` are set to this value. The default values are `lb.impute=0` and `ub.impute=Inf`.

Imputation Using Quantile-Quantile Regression Including the Censoring Level (`method ="impute.w.qq.reg.w.cen.level"`)
This method is only available for sinlgy censored data. This method was proposed by El-Shaarawi (1989), which he denoted as the Modified LR Method. It is exactly the same method as imputation using quantile-quantile regression (`method="impute.w.qq.reg"`), except that the quantile-quantile regression includes the censoring level. For left singly censored data, the modification involves adding the point $[\Phi^{-1}(p_c), T]$ to the plot before fitting the least-squares line. For right singly censored data, the point $[\Phi^{-1}(p_{n+1}), T]$ is added to the plot before fitting the least-squares line.

Imputation Using Maximum Likelihood (`method ="impute.w.mle"`)
This method is only available for sinlgy censored data. This is exactly the same method as robust Regression on Order Statistics (i.e., the same as using `method="rROS"` or `method="impute.w.qq.reg"`), except that the maximum likelihood method (`method="mle"`) is used to compute the initial estimates of the mean and standard deviation. In the context of log-normal data, this method is discussed by El-Shaarawi (1989), which he denotes as the Modified Maximum Likelihood Method.

Setting Censored Observations to Half the Censoring Level (`method="half.cen.level"`)
This method is applicable only to left censored data that is bounded below by 0. This method involves simply replacing all the censored observations with half their detection limit, and then computing the usual moment estimators of the mean and variance. That is, all censored observations are imputed to be half the detection limit, and then Equations (17) and (18) are used to estimate the mean and varaince.

This method is included only to allow comparison of this method to other methods. ***Setting left-censored observations to half the censoring level is not recommended***. In particular, El-Shaarawi and Esterby (1992) show that these estimators are biased and inconsistent (i.e., the bias remains even as the sample size increases).

CONFIDENCE INTERVALS

This section explains how confidence intervals for the mean θ are computed. f

Likelihood Profile (`ci.method="profile.likelihood"`)
This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean θ while treating the coefficient of variation τ as a nuisance parameter.

For Type I left censored data, the likelihood function is given by:

$$L(\theta, \tau|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (19)$$

where f and F denote the probability density function (pdf) and cumulative distribution function (cdf) of the population. That is,

$$f(t) = \phi(\frac{t-\mu}{\sigma}) \qquad (20)$$

$$F(t) = \Phi(\frac{t-\mu}{\sigma}) \qquad (21)$$

where

$$\mu = log(\frac{\theta}{\sqrt{\tau^2+1}}) \qquad (22)$$

$$\sigma = [log(\tau^2+1)]^{1/2} \qquad (23)$$

and ϕ and Φ denote the pdf and cdf of the standard normal distribution, respectively (Cohen, 1963; 1991, pp.6, 50). For left singly censored data, equation (3) simplifies to:

$$L(\mu,\sigma|\underline{x}) = \binom{N}{c}[F(T)]^c \prod_{i=c+1}^{n} f[x_{(i)}] \qquad (24)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\mu,\sigma|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k}[1-F(T_j)]^{c_j} \prod_{i\in\Omega} f[x_{(i)}] \qquad (25)$$

and for right singly censored data this simplifies to:

$$L(\mu,\sigma|\underline{x}) = \binom{N}{c}[1-F(T)]^c \prod_{i=1}^{n} f[x_{(i)}] \qquad (26)$$

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and coefficient of variation by (θ^*, τ^*). The likelihood ratio test statistic (G^2) of the hypothesis $H_0 : \theta = \theta_0$ (where θ_0 is a fixed value) equals the drop in $2log(L)$ between the "full" model and the reduced model with θ fixed at θ_0, i.e.,

$$G^2 = 2\{log[L(\theta^*,\tau^*)] - log[L(\theta_0,\tau_0^*)]\} \qquad (27)$$

where τ_0^* is the maximum likelihood estimate of τ for the reduced model (i.e., when $\theta = \theta_0$). Under the null hypothesis, the test statistic G^2 follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function L_1 for the mean θ, which is obtained from the usual likelihood function by maximizing over the parameter τ, i.e.,

$$L_1(\theta) = max_\tau L(\theta,\tau) \qquad (28)$$

Then we have

$$G^2 = 2\{log[L_1(\theta^*)] - log[L_1(\theta_0)]\} \qquad (29)$$

A two-sided $(1-\alpha)100\%$ confidence interval for the mean θ consists of all values of θ_0 for which the test is not significant at level $alpha$:

$$\theta_0 : G^2 \leq \chi^2_{1,1-\alpha} \qquad (30)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-squared distribution with ν degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1-\alpha$ in Equation (30) is replaced with $1-2\alpha$.

Direct Normal Approximations (`ci.method="delta"` or `ci.method="normal.approx"`)
An approximate $(1-\alpha)100\%$ confidence interval for θ can be constructed assuming the distribution of the estimator of θ is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$ confidence interval for θ is constructed as:

$$[\hat{\theta} - t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\theta}},\ \hat{\theta} + t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\theta}}] \qquad (31)$$

where $\hat{\theta}$ denotes the estimate of θ, $\hat{\sigma}_{\hat{\theta}}$ denotes the estimated asymptotic standard deviation of the estimator of θ, m denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument `ci.sample.size` determines the value of m (see see the entry for ... in the ARGUMENTS section above). When `method` equals `"mle"`, `"qmvue"`, or `"bcmle"` and the data are singly censored, the default value is the expected number of uncensored observations, otherwise it is n, the observed number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When `pivot.statistic="z"`, the p'th quantile from the standard normal distribution is used in place of the p'th quantile from Student's t-distribution.

Direct Normal Approximation Based on the Delta Method (`ci.method="delta"`)
This method is usually applied with the maximum likelihood estimators (`method="mle"`). It should also work approximately for the quasi minimum variance unbiased estimators (`method="qmvue"`) and the bias-corrected maximum likelihood estimators (`method="bcmle"`).

When `method="mle"`, the variance of the mle of θ can be estimated based on the variance-covariance matrix of the mle's of μ and σ (denoted V), and the delta method:

$$\hat{\sigma}^2_{\hat{\theta}} = (\frac{\partial\theta}{\partial\underline{\lambda}})'_{\hat{\underline{\lambda}}}\hat{V}(\frac{\partial\theta}{\partial\underline{\lambda}})_{\hat{\underline{\lambda}}} \qquad (32)$$

where

$$\underline{\lambda}' = (\mu, \sigma) \qquad (33)$$

$$\frac{\partial\theta}{\partial\mu} = exp(\mu + \frac{\sigma^2}{2}) \qquad (34)$$

$$\frac{\partial\theta}{\partial\sigma} = \sigma exp(\mu + \frac{\sigma^2}{2}) \qquad (35)$$

(Shumway et al., 1989). The variance-covariance matrix V of the mle's of μ and σ is estimated based on the inverse of the observed Fisher Information matrix, formulas for which are given in Cohen (1991).

Direct Normal Approximation Based on the Moment Estimators (`ci.method="normal.approx"`)
This method is valid only for the moment estimators based on imputed values
(i.e., `method="impute.w.qq.reg"` or `method="half.cen.level"`). For these cases, the standard deviation of the estimated mean is assumed to be approximated by

$$\hat{\sigma}_{\hat{\theta}} = \frac{\hat{\eta}}{\sqrt{m}} \qquad (36)$$

where, as already noted, m denotes the assumed sample size. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

Cox's Method (`ci.method="cox"`)
This method may be applied with the maximum likelihood estimators (`method="mle"`), the quasi minimum variance unbiased estimators (`method="qmvue"`), and the bias-corrected maximum likelihood estimators (`method="bcmle"`).

This method was proposed by El-Shaarawi (1989) and is an extension of the method derived by Cox and presented in Land (1972) for the case of complete data (see the explanation of `ci.method="cox"` in the help file for `elnormAlt`). The idea is to construct an approximate $(1-\alpha)100\%$ confidence interval for the quantity

$$\beta = exp(\mu + \frac{\sigma^2}{2}) \qquad (37)$$

assuming the estimate of β

$$\hat{\beta} = exp(\hat{\mu} + \frac{\hat{\sigma}^2}{2}) \qquad (38)$$

is approximately normally distributed, and then exponentiate the confidence limits. That is, a two-sided $(1-\alpha)100\%$ confidence interval for θ is constructed as:

$$[exp(\hat{\beta} - h),\ exp(\hat{\beta} + h)] \qquad (39)$$

where

$$h = t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\beta}} \qquad (40)$$

and $\hat{\sigma}_{\hat{\beta}}$ denotes the estimated asymptotic standard deviation of the estimator of β, m denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom.

El-Shaarawi (1989) shows that the standard deviation of the mle of β can be estimated by:

$$\hat{\sigma}_{\hat{\beta}} = \sqrt{\hat{V}_{11} + 2\hat{\sigma}\hat{V}_{12} + \hat{\sigma}^2\hat{V}_{22}} \qquad (41)$$

where V denotes the variance-covariance matrix of the mle's of μ and σ and is estimated based on the inverse of the Fisher Information matrix.

One-sided confidence intervals are computed in a similar fashion.

Bootstrap and Bias-Corrected Bootstrap Approximation (`ci.method="bootstrap"`)
The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1-\alpha)100\%$ confidence interval for the population mean θ, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size N from the observations in $\underline{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\underline{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate θ based on the bootstrap sample created in Step 1, using the same method that was used to estimate θ using the original observations in $\underline{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of θ based on the bootstrap sample will usually differ from the original estimate based on $\underline{x}$.
3. Repeat Steps 1 and 2 B times, where B is some large number. The number of bootstraps B is determined by the argument `n.bootstraps` (see the section ARGUMENTS above). The default value of `n.bootstraps` is `1000`.
4. Use the B estimated values of θ to compute the empirical cumulative distribution function of this estimator of θ (see `ecdfPlot`), and then create a confidence interval for θ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}),\ \hat{G}^{-1}(1-\frac{\alpha}{2})] \qquad (42)$$

where $\hat{G}(t)$ denotes the empirical cdf evaluated at t and thus $\hat{G}^{-1}(p)$ denotes the p'th empirical quantile, that is, the p'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha),\ \infty] \qquad (43)$$

and a one-sided upper confidence interval is computed as:

$$[0,\ \hat{G}^{-1}(1-\alpha)] \qquad (44)$$

The function `elnormAltCensored` calls the R function `quantile` to compute the empirical quantiles used in Equations (42)-(44).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of θ can be off by $k/\sqrt{N}$, where $k$is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of θ may be off by k/N instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1),\ \hat{G}^{-1}(\alpha_2)] \qquad (45)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1-\hat{a}(z_0+z_{\alpha/2})}] \qquad (46)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1-\hat{a}(z_0+z_{1-\alpha/2})}] \qquad (47)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\theta})] \qquad (48)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\theta}_{(\cdot)} - \hat{\theta}_{(i)})^3}{6[\sum_{i=1}^{N}(\hat{\theta}_{(\cdot)} - \hat{\theta}_{(i)})^2]^{3/2}} \qquad (49)$$

where the quantity $\hat{\theta}_{(i)}$ denotes the estimate of θ using all the values in $\underline{x}$ except the i'th one, and

$$\hat{\theta}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\hat{\theta_{(i)}} \qquad (50)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1),\ \infty] \qquad (51)$$

and a one-sided upper confidence interval is given by:

$$[0,\ \hat{G}^{-1}(\alpha_2)] \qquad (52)$$

where α_1 and α_2 are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with α in Equations (51) and (52).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of θ with respect to the true value of θ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of θ does not depend on the value of θ, hence the acceleration constant is not really necessary.

When `ci.method="bootstrap"`, the function `elnormAltCensored` computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

This method of constructing confidence intervals for censored data was studied by Shumway et al. (1989).

Value

a list of class `"estimateCensored"` containing the estimated parameters and other information. See `estimateCensored.object` for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation **on the original scale, not the log-scale,** when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bain, L.J., and M. Engelhardt. (1991). *Statistical Analysis of Reliability and Life-Testing Models*. Marcel Dekker, New York, 496pp.

Cohen, A.C. (1959). Simplified Estimators for the Normal Distribution When Samples are Singly Censored or Truncated. *Technometrics* **1**(3), 217–237.

Cohen, A.C. (1963). Progressively Censored Samples in Life Testing. *Technometrics* **5**, 327–339

Cohen, A.C. (1991). *Truncated and Censored Samples*. Marcel Dekker, New York, New York, 312pp.

Cox, D.R. (1970). *Analysis of Binary Data*. Chapman & Hall, London. 142pp.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

El-Shaarawi, A.H. (1989). Inferences About the Mean from Censored Water Quality Data. *Water Resources Research* **25**(4) 685–690.

El-Shaarawi, A.H., and D.M. Dolan. (1989). Maximum Likelihood Estimation of Water Quality Concentrations from Censored Data. *Canadian Journal of Fisheries and Aquatic Sciences* **46**, 1033–1039.

El-Shaarawi, A.H., and S.R. Esterby. (1992). Replacement of Censored Observations by a Constant: An Evaluation. *Water Research* **26**(6), 835–844.

El-Shaarawi, A.H., and A. Naderi. (1991). Statistical Inference from Multiply Censored Environmental Data. *Environmental Monitoring and Assessment* **17**, 339–347.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135–146.

Gleit, A. (1985). Estimation for Small Normal Data Sets with Detection Limits. *Environmental Science and Technology* **19**, 1201–1206.

Haas, C.N., and P.A. Scheff. (1990). Estimation of Averages in Truncated Samples. *Environmental Science and Technology* **24**(6), 912–919.

Hashimoto, L.K., and R.R. Trussell. (1983). Evaluating Water Quality Data Near the Detection Limit. Paper presented at the Advanced Technology Conference, American Water Works Association, Las Vegas, Nevada, June 5-9, 1983.

Helsel, D.R. (1990). Less than Obvious: Statistical Treatment of Data Below the Detection Limit. *Environmental Science and Technology* **24**(12), 1766–1774.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997–2004.

Hirsch, R.M., and J.R. Stedinger. (1987). Plotting Positions for Historical Floods and Their Precision. *Water Resources Research* **23**(4), 715–727.

Korn, L.R., and D.E. Tyler. (2001). Robust Estimation for Chemical Concentration Data Subject to Detection Limits. In Fernholz, L., S. Morgenthaler, and W. Stahel, eds. *Statistics in Genetics and in the Environmental Sciences*. Birkhauser Verlag, Basel, pp.41–63.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Michael, J.R., and W.R. Schucany. (1986). Analysis of Data from Censored Samples. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, 560pp, Chapter 11, 461–496.

Millard, S.P., P. Dixon, and N.K. Neerchal. (2014; in preparation). *Environmental Statistics with R*. CRC Press, Boca Raton, Florida.

Nelson, W. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, 634pp.

Newman, M.C., P.M. Dixon, B.B. Looney, and J.E. Pinder. (1989). Estimating Mean and Variance for Environmental Samples with Below Detection Limit Observations. *Water Resources Bulletin* **25**(4), 905–916.

Pettitt, A. N. (1983). Re-Weighted Least Squares Estimation with Censored and Grouped Data: An Application of the EM Algorithm. *Journal of the Royal Statistical Society, Series B* **47**, 253–260.

Regal, R. (1982). Applying Order Statistic Censored Normal Confidence Intervals to Time Censored Data. Unpublished manuscript, University of Minnesota, Duluth, Department of Mathematical Sciences.

Royston, P. (2007). Profile Likelihood for Estimation and Confdence Intervals. *The Stata Journal* **7**(3), pp. 376–387.

Saw, J.G. (1961b). The Bias of the Maximum Likelihood Estimators of Location and Scale Parameters Given a Type II Censored Normal Sample. *Biometrika* **48**, 448–451.

Schmee, J., D.Gladstein, and W. Nelson. (1985). Confidence Limits for Parameters of a Normal Distribution from Singly Censored Samples, Using Maximum Likelihood. *Technometrics* **27**(2) 119–128.

Schneider, H. (1986). *Truncated and Censored Samples from Normal Populations*. Marcel Dekker, New York, New York, 273pp.

Shumway, R.H., A.S. Azari, and P. Johnson. (1989). Estimating Mean Concentrations Under Transformations for Environmental Data With Detection Limits. *Technometrics* **31**(3), 347–356.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Stryhn, H., and J. Christensen. (2003). *Confidence Intervals by the Profile Likelihood Method, with Applications in Veterinary Epidemiology*. Contributed paper at ISVEE X (November 2003, Chile). https://gilvanguedes.com/wp-content/uploads/2019/05/Profile-Likelihood-CI.pdf.

Travis, C.C., and M.L. Land. (1990). Estimating the Mean of Data Sets with Nondetectable Values. *Environmental Science and Technology* **24**, 961–962.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. Chapter 15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Venzon, D.J., and S.H. Moolgavkar. (1988). A Method for Computing Profile-Likelihood-Based Confidence Intervals. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **37**(1), pp. 87–94.

See Also

LognormalAlt, elnormAlt, elnormCensored, enormCensored, estimateCensored.object.

Examples

```
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and coefficient of variation
# ON THE ORIGINAL SCALE using the MLE, QMVUE,
# and robust ROS (imputation with Q-Q regression).

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)
```

```
#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now estimate the mean and coefficient of variation
# using the MLE:
#-----------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#-------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          mean = 23.003987
#                                 cv   =  2.300772
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%

# Now compare the MLE with the QMVUE and the
# estimator based on robust ROS
#-------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored))$parameters
#     mean        cv
#23.003987  2.300772

with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored,
  method = "qmvue"))$parameters
#     mean        cv
#21.566945  1.841366
```

```
with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored,
  method = "rROS"))$parameters
#     mean        cv
#19.886180  1.298868

#----------

# The method used to estimate quantiles for a Q-Q plot is
# determined by the argument prob.method.  For the function
# elnormCensoredAlt, for any estimation method that involves
# Q-Q regression, the default value of prob.method is
# "hirsch-stedinger" and the default value for the
# plotting position constant is plot.pos.con=0.375.

# Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger
# probability method but set the plotting position constant to 0.

with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored,
  method = "rROS", plot.pos.con = 0))$parameters
#     mean        cv
#19.827673  1.304725

#----------

# Using the same data as above, compute a confidence interval
# for the mean using the profile-likelihood method.

with(EPA.09.Ex.15.1.manganese.df,
  elnormAltCensored(Manganese.ppb, Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#----------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          mean = 23.003987
#                                 cv   =  2.300772
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
```

```
#Percent Censored:                24%
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Profile Likelihood
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 12.37629
#                                 UCL = 69.87694
```

`elnormCensored` *Estimate Parameters for a Lognormal Distribution (Log-Scale) Based on Type I Censored Data*

Description

Estimate the mean and standard deviation parameters of the logarithm of a lognormal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

```
elnormCensored(x, censored, method = "mle", censoring.side = "left",
  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
  nmc = 1000, seed = NULL, ...)
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`censored`	numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is `"logical"`, `TRUE` values correspond to elements of `x` that are censored, and `FALSE` values correspond to elements of `x` that are not censored. If the mode of `censored` is `"numeric"`, it must contain only `1`'s and `0`'s; `1` corresponds to `TRUE` and `0` corresponds to `FALSE`. Missing (`NA`) values are allowed but will be removed.
`method`	character string specifying the method of estimation. For singly censored data, the possible values are: `"mle"` (maximum likelihood; the default), `"bcmle"` (bias-corrected maximum likelihood), `"ROS"` or `"qq.reg"` (quantile-quantile regression; also called *regression on order statistics* and abbreviated ROS), `"qq.reg.w.cen.level"` (quantile-quantile regression including the censoring

level),
"rROS" or "impute.w.qq.reg" (moment estimation based on imputation using quantile-quantile regression; also called *robust regression on order statistics* and abbreviated rROS),
"impute.w.qq.reg.w.cen.level" (moment estimation based on imputation using the qq.reg.w.cen.level method),
"impute.w.mle" (moment estimation based on imputation using the mle),
"iterative.impute.w.qq.reg" (moment estimation based on iterative imputation using the qq.reg method),
"m.est" (robust M-estimation), and
"half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

For multiply censored data, the possible values are:
"mle" (maximum likelihood; the default),
"ROS" or "qq.reg" (quantile-quantile regression; also called *regression on order statistics* and abbreviated ROS),
"rROS" or "impute.w.qq.reg" (moment estimation based on imputation using quantile-quantile regression; also called *robust regression on order statistics* and abbreviated rROS), and
"half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

`censoring.side` character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

`ci` logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.

`ci.method` character string indicating what method to use to construct the confidence interval for the mean. The possible values are:
"profile.likelihood" (profile likelihood; the default),
"normal.approx" (normal approximation),
"normal.approx.w.cov" (normal approximation taking into account the covariance between the estimated mean and standard deviation; only available for singly censored data),
"gpq" (generalized pivotal quantity), and
"bootstrap" (based on bootstrapping).

See the DETAILS section for more information. This argument is ignored if ci=FALSE.

`ci.type` character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

`n.bootstraps` numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

pivot.statistic
: character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample size (see below). The argument pivot.statistic is ignored if ci=FALSE.

nmc
: numeric scalar indicating the number of Monte Carlo simulations to run when ci.method="gpq". The default is nmc=1000. This argument is ignored if ci=FALSE.

seed
: integer supplied to the function set.seed and used when ci.method="bootstrap" or ci.method="gpq". The default value is seed=NULL, in which case the current value of .Random.seed is used. This argument is ignored when ci=FALSE.

...
: additional arguments to pass to other functions.

- prob.method. Character string indicating what method to use to compute the plotting positions (empirical probabilities) when method is one of "ROS", "qq.reg", "qq.reg.w.cen.level", "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Possible values are:
  "kaplan-meier" (product-limit method of Kaplan and Meier (1958)),
  "nelson" (hazard plotting method of Nelson (1972)),
  "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and
  "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)).
  The default value is prob.method="hirsch-stedinger". The "nelson" method is only available for censoring.side="right". See the DETAILS section and the help file for ppointsCensored for more information.
- plot.pos.con. Numeric scalar between 0 and 1 containing the value of the plotting position constant to use when method is one of "qq.reg", "qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". The default value is plot.pos.con=0.375. See the DETAILS section and the help file for ppointsCensored for more information.
- ci.sample.size. Numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method="normal.approx" or ci.method="normal.approx.w.cov". When method equals "mle" or "bcmle", the default value is the expected number of uncensored observations, otherwise it is the observed number of uncensored observations.
- lb.impute. Numeric scalar indicating the lower bound for imputed observations when method is one of "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values smaller than this value will be set to this value. The default is lb.impute=-Inf.

- ub.impute. Numeric scalar indicating the upper bound for imputed observations when method is one of "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values larger than this value will be set to this value. The default is ub.impute=Inf.
- convergence. Character string indicating the kind of convergence criterion when method="iterative.impute.w.qq.reg". The possible values are "relative" (the default) and "absolute". See the DETAILS section for more information.
- tol. Numeric scalar indicating the convergence tolerance when method="iterative.impute.w.qq.reg". The default value is tol=1e-6. If convergence="relative", then the relative difference in the old and new estimates of the mean and the relative difference in the old and new estimates of the standard deviation must be less than tol for convergence to be achieved. If convergence="absolute", then the absolute difference in the old and new estimates of the mean and the absolute difference in the old and new estimates of the standard deviation must be less than tol for convergence to be achieved.
- max.iter. Numeric scalar indicating the maximum number of iterations when method="iterative.impute.w.qq.reg".
- t.df. Numeric scalar greater than or equal to 1 that determines the robustness and efficiency properties of the estimator when method="m.est". The default value is t.df=3.

Details

If x or censored contain any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let X denote a random variable with a lognormal distribution with parameters meanlog=μ and sdlog=σ. Then $Y = log(X)$ has a normal (Gaussian) distribution with parameters mean=μ and sd=σ. Thus, the function elnormCensored simply calls the function enormCensored using the log-transformed values of x.

Value

a list of class "estimateCensored" containing the estimated parameters and other information. See estimateCensored.object for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed al-

ternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Schmee et al. (1985) studied Type II censoring for a normal distribution and noted that the bias and variances of the maximum likelihood estimators are of the order $1/N$, and that the bias is negligible for $N = 100$ and as much as 90% censoring. (If the proportion of censored observations is less than 90%, the bias becomes negligible for smaller sample sizes.) For small samples with moderate to high censoring, however, the bias of the mle's causes confidence intervals based on them using a normal approximation (e.g., `method="mle"` and `ci.method="normal.approx"`) to be too short. Schmee et al. (1985) provide tables for exact confidence intervals for sample sizes up to $N = 100$ that were created based on Monte Carlo simulation. Schmee et al. (1985) state that these tables should work well for Type I censored data as well.

Shumway et al. (1989) evaluated the coverage of 90% confidence intervals for the mean based on using a Box-Cox transformation to induce normality, computing the mle's based on the normal distribution, then computing the mean in the original scale. They considered three methods of constructing confidence intervals: the delta method, the bootstrap, and the bias-corrected bootstrap. Shumway et al. (1989) used three parent distributions in their study: Normal(3,1), the square of this distribuiton, and the exponentiation of this distribution (i.e., a lognormal distribution). Based on sample sizes of 10 and 50 with a censoring level at the 10'th or 20'th percentile, Shumway et al. (1989) found that the delta method performed quite well and was superior to the bootstrap method.

Millard et al. (2014; in preparation) show that the coverage of profile likelihood method is excellent.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bain, L.J., and M. Engelhardt. (1991). *Statistical Analysis of Reliability and Life-Testing Models*. Marcel Dekker, New York, 496pp.

Cohen, A.C. (1959). Simplified Estimators for the Normal Distribution When Samples are Singly Censored or Truncated. *Technometrics* **1**(3), 217–237.

Cohen, A.C. (1963). Progressively Censored Samples in Life Testing. *Technometrics* **5**, 327–339

Cohen, A.C. (1991). *Truncated and Censored Samples*. Marcel Dekker, New York, New York, 312pp.

Cox, D.R. (1970). *Analysis of Binary Data*. Chapman & Hall, London. 142pp.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

El-Shaarawi, A.H. (1989). Inferences About the Mean from Censored Water Quality Data. *Water Resources Research* **25**(4) 685–690.

El-Shaarawi, A.H., and D.M. Dolan. (1989). Maximum Likelihood Estimation of Water Quality Concentrations from Censored Data. *Canadian Journal of Fisheries and Aquatic Sciences* **46**, 1033–1039.

El-Shaarawi, A.H., and S.R. Esterby. (1992). Replacement of Censored Observations by a Constant: An Evaluation. *Water Research* **26**(6), 835–844.

El-Shaarawi, A.H., and A. Naderi. (1991). Statistical Inference from Multiply Censored Environmental Data. *Environmental Monitoring and Assessment* **17**, 339–347.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135–146.

Gleit, A. (1985). Estimation for Small Normal Data Sets with Detection Limits. *Environmental Science and Technology* **19**, 1201–1206.

Haas, C.N., and P.A. Scheff. (1990). Estimation of Averages in Truncated Samples. *Environmental Science and Technology* **24**(6), 912–919.

Hashimoto, L.K., and R.R. Trussell. (1983). Evaluating Water Quality Data Near the Detection Limit. Paper presented at the Advanced Technology Conference, American Water Works Association, Las Vegas, Nevada, June 5-9, 1983.

Helsel, D.R. (1990). Less than Obvious: Statistical Treatment of Data Below the Detection Limit. *Environmental Science and Technology* **24**(12), 1766–1774.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997–2004.

Hirsch, R.M., and J.R. Stedinger. (1987). Plotting Positions for Historical Floods and Their Precision. *Water Resources Research* **23**(4), 715–727.

Korn, L.R., and D.E. Tyler. (2001). Robust Estimation for Chemical Concentration Data Subject to Detection Limits. In Fernholz, L., S. Morgenthaler, and W. Stahel, eds. *Statistics in Genetics and in the Environmental Sciences*. Birkhauser Verlag, Basel, pp.41–63.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Michael, J.R., and W.R. Schucany. (1986). Analysis of Data from Censored Samples. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, 560pp, Chapter 11, 461–496.

Millard, S.P., P. Dixon, and N.K. Neerchal. (2014; in preparation). *Environmental Statistics with R*. CRC Press, Boca Raton, Florida.

Nelson, W. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, 634pp.

Newman, M.C., P.M. Dixon, B.B. Looney, and J.E. Pinder. (1989). Estimating Mean and Variance for Environmental Samples with Below Detection Limit Observations. *Water Resources Bulletin* **25**(4), 905–916.

Pettitt, A. N. (1983). Re-Weighted Least Squares Estimation with Censored and Grouped Data: An Application of the EM Algorithm. *Journal of the Royal Statistical Society, Series B* **47**, 253–260.

Regal, R. (1982). Applying Order Statistic Censored Normal Confidence Intervals to Time Censored Data. Unpublished manuscript, University of Minnesota, Duluth, Department of Mathematical Sciences.

Royston, P. (2007). Profile Likelihood for Estimation and Confdence Intervals. *The Stata Journal* **7**(3), pp. 376–387.

Saw, J.G. (1961b). The Bias of the Maximum Likelihood Estimators of Location and Scale Parameters Given a Type II Censored Normal Sample. *Biometrika* **48**, 448–451.

Schmee, J., D.Gladstein, and W. Nelson. (1985). Confidence Limits for Parameters of a Normal Distribution from Singly Censored Samples, Using Maximum Likelihood. *Technometrics* **27**(2) 119–128.

Schneider, H. (1986). *Truncated and Censored Samples from Normal Populations*. Marcel Dekker, New York, New York, 273pp.

Shumway, R.H., A.S. Azari, and P. Johnson. (1989). Estimating Mean Concentrations Under Transformations for Environmental Data With Detection Limits. *Technometrics* **31**(3), 347–356.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Stryhn, H., and J. Christensen. (2003). *Confidence Intervals by the Profile Likelihood Method, with Applications in Veterinary Epidemiology*. Contributed paper at ISVEE X (November 2003, Chile). https://gilvanguedes.com/wp-content/uploads/2019/05/Profile-Likelihood-CI.pdf.

Travis, C.C., and M.L. Land. (1990). Estimating the Mean of Data Sets with Nondetectable Values. *Environmental Science and Technology* **24**, 961–962.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. Chapter 15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Venzon, D.J., and S.H. Moolgavkar. (1988). A Method for Computing Profile-Likelihood-Based Confidence Intervals. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **37**(1), pp. 87–94.

See Also

enormCensored, Lognormal, elnorm, estimateCensored.object.

Examples

```
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and standard deviation using the MLE,
# Q-Q regression (also called parametric regression on order statistics
# or ROS; e.g., USEPA, 2009 and Helsel, 2012), and imputation with Q-Q
# regression (also called robust ROS or rROS).

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now estimate the mean and standard deviation on the log-scale
# using the MLE:
#---------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  elnormCensored(Manganese.ppb, Censored))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
```

```
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          meanlog = 2.215905
#                                 sdlog   = 1.356291
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%

# Now compare the MLE with the estimators based on
# Q-Q regression (ROS) and imputation with Q-Q regression (rROS)
#---------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  elnormCensored(Manganese.ppb, Censored))$parameters
# meanlog    sdlog
#2.215905 1.356291

with(EPA.09.Ex.15.1.manganese.df,
  elnormCensored(Manganese.ppb, Censored,
  method = "ROS"))$parameters
# meanlog    sdlog
#2.293742 1.283635

with(EPA.09.Ex.15.1.manganese.df,
  elnormCensored(Manganese.ppb, Censored,
  method = "rROS"))$parameters
# meanlog    sdlog
#2.298656 1.238104

#----------

# The method used to estimate quantiles for a Q-Q plot is
# determined by the argument prob.method.  For the functions
# enormCensored and elnormCensored, for any estimation
# method that involves Q-Q regression, the default value of
# prob.method is "hirsch-stedinger" and the default value for the
# plotting position constant is plot.pos.con=0.375.

# Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger
# probability method but set the plotting position constant to 0.

with(EPA.09.Ex.15.1.manganese.df,
  elnormCensored(Manganese.ppb, Censored,
  method = "rROS", plot.pos.con = 0))$parameters
# meanlog    sdlog
```

```
#2.277175 1.261431

#----------

# Using the same data as above, compute a confidence interval
# for the mean on the log-scale using the profile-likelihood
# method.

with(EPA.09.Ex.15.1.manganese.df,
  elnormCensored(Manganese.ppb, Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          meanlog = 2.215905
#                                 sdlog   = 1.356291
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Confidence Interval for:         meanlog
#
#Confidence Interval Method:      Profile Likelihood
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 1.595062
#                                 UCL = 2.771197
```

---

elogis *Estimate Parameters of a Logistic Distribution*

Description

Estimate the location and scale parameters of a logistic distribution, and optionally construct a confidence interval for the location parameter.

Usage

```
elogis(x, method = "mle", ci = FALSE, ci.type = "two-sided",
  ci.method = "normal.approx", conf.level = 0.95)
```

Arguments

`x`	numeric vector of observations.
`method`	character string specifying the method of estimation. Possible values are `"mle"` (maximum likelihood; the default), `"mme"` (methods of moments), and `"mmue"` (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.
`ci`	logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is `FALSE`.
`ci.type`	character string indicating what kind of confidence interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`. This argument is ignored if `ci=FALSE`.
`ci.method`	character string indicating what method to use to construct the confidence interval for the location or scale parameter. Currently, the only possible value is `"normal.approx"` (the default). See the DETAILS section for more information. This argument is ignored if `ci=FALSE`.
`conf.level`	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from an logistic distribution with parameters `location`=η and `scale`=θ.

Estimation

Maximum Likelihood Estimation (`method="mle"`)
The maximum likelihood estimators (mle's) of η and θ are the solutions of the simultaneous equations (Forbes et al., 2011):

$$\sum_{i=1}^{n} \frac{1}{1+e^{z_i}} = \frac{n}{2} \quad (1)$$

$$\sum_{i=1}^{n} z_i \left[\frac{1-e^{z_i}}{1+e^{z_i}} = n \quad (2)\right.$$

where

$$z_i = \frac{x_i - \hat{eta}_{mle}}{\hat{\theta}_{mle}} \quad (3)$$

Method of Moments Estimation (`method="mme"`)
The method of moments estimators (mme's) of η and θ are given by:

$$\hat{\eta}_{mme} = \bar{x} \quad (4)$$

$$\hat{\theta}_{mme} = \frac{\sqrt{3}}{\pi} s_m \quad (5)$$

where

$$\bar{x} = \sum_{i=1}^{n} x_i \quad (6)$$

$$s_m^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (7)$$

that is, s_m denotes the square root of the method of moments estimator of variance.

Method of Moments Estimators Based on the Unbiased Estimator of Variance (`method="mmue"`)
These estimators are exactly the same as the method of moments estimators given in equations (4-7) above, except that the method of moments estimator of variance in equation (7) is replaced with the unbiased estimator of variance:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (8)$$

Confidence Intervals
When `ci=TRUE`, an approximate $(1 - \alpha)100\%$ confidence intervals for η can be constructed assuming the distribution of the estimator of η is approximately normally distributed. A two-sided confidence interval is constructed as:

$$[\hat{\eta} - t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\eta}},\ \hat{\eta} + t(n-1, 1-\alpha/2)\hat{\sigma}_{\hat{\eta}}]$$

where $t(\nu, p)$ is the p'th quantile of Student's t-distribution with ν degrees of freedom, and the quantity

$$\hat{\sigma}_{\hat{\eta}} = \frac{\pi\hat{\theta}}{\sqrt{3n}} \quad (9)$$

denotes the estimated asymptotic standard deviation of the estimator of η.

One-sided confidence intervals for η and θ are computed in a similar fashion.

Value

a list of class `"estimate"` containing the estimated parameters and other information.
See `estimate.object` for details.

Note

The logistic distribution is defined on the real line and is unimodal and symmetric about its location parameter (the mean). It has longer tails than a normal (Gaussian) distribution. It is used to model growth curves and bioassay data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

Logistic.

Examples

```
# Generate 20 observations from a logistic distribution with
# parameters location=0 and scale=1, then estimate the parameters
# and construct a 90% confidence interval for the location parameter.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlogis(20)
elogis(dat, ci = TRUE, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Logistic
#
#Estimated Parameter(s):          location = -0.2181845
#                                 scale    =  0.8152793
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         location
#
#Confidence Interval Method:      Normal Approximation
#                                 (t Distribution)
#
#Confidence Interval Type:        two-sided
```

```
#
#Confidence Level:                90%
#
#Confidence Interval:             LCL = -0.7899382
#                                 UCL =  0.3535693

#----------

# Clean up
#---------
rm(dat)
```

`Empirical` *The Empirical Distribution Based on a Set of Observations*

Description

Density, distribution function, quantile function, and random generation for the empirical distribution based on a set of observations

Usage

```
demp(x, obs, discrete = FALSE, density.arg.list = NULL)
pemp(q, obs, discrete = FALSE,
  prob.method = ifelse(discrete, "emp.probs", "plot.pos"),
  plot.pos.con = 0.375)
qemp(p, obs, discrete = FALSE,
  prob.method = ifelse(discrete, "emp.probs", "plot.pos"),
  plot.pos.con = 0.375)
remp(n, obs)
```

Arguments

`x`	vector of quantiles.
`q`	vector of quantiles.
`p`	vector of probabilities between 0 and 1.
`n`	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
`obs`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`discrete`	logical scalar indicating whether the assumed parent distribution of `x` is discrete (`discrete=TRUE`) or continuous (`discrete=FALSE`). The default value is `FALSE`.
`density.arg.list`	list with arguments to the R `density` function. The default value is `NULL`. (See the help file for `density` for more information on the arguments to density.) The argument `density.arg.list` is ignored if `discrete=TRUE`.

`prob.method` character string indicating what method to use to compute the empirical probabilities. Possible values are `"emp.probs"` (empirical probabilities, default if `discrete=TRUE`) and `"plot.pos"` (plotting positions, default if `discrete=FALSE`). See the DETAILS section for more explanation.

`plot.pos.con` numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is `plot.pos.con=0.375`. See the DETAILS section for more information. This argument is ignored if `prob.method="emp.probs"`.

Details

Let $x_1, x_2, \ldots, x_n$ denote a random sample of n observations from some unknown probability distribution (i.e., the elements of the argument `obs`), and let $x_{(i)}$ denote the i^{th} order statistic, that is, the i^{th} largest observation, for $i = 1, 2, \ldots, n$.

Estimating Density

The function `demp` computes the empirical probability density function. If the observations are assumed to come from a discrete distribution, the probability density (mass) function is estimated by:

$$\hat{f}(x) = \widehat{Pr}(X = x) = \frac{\sum_{i=1}^{n} I_{[x]}(x_i)}{n}$$

where I is the indicator function:

$$I_{[x]}(y) = \begin{array}{ll} 1 & \text{if } y = x, \\ 0 & \text{if } y \neq x \end{array}$$

That is, the estimated probability of observing the value x is simply the observed proportion of observations equal to x.

If the observations are assumed to come from a continuous distribution, the function `demp` calls the R function `density` to compute the estimated density based on the values specified in the argument `obs`, and then uses linear interpolation to estimate the density at the values specified in the argument `x`. See the R help file for `density` for more information on how the empirical density is computed in the continuous case.

Estimating Probabilities

The function `pemp` computes the estimated cumulative distribution function (cdf), also called the empirical cdf (ecdf). If the observations are assumed to come from a discrete distribution, the value of the cdf evaluated at the i^{th} order statistic is usually estimated by:

$$\hat{F}[x_{(i)}] = \widehat{Pr}(X \leq x_{(i)}) = \hat{p}_i = \frac{\sum_{j=1}^{n} I_{(-\infty, x_{(i)}]}(x_j)}{n}$$

where:

$$I_{(-\infty, x]}(y) = \begin{array}{ll} 1 & \text{if } y \leq x, \\ 0 & \text{if } y > x \end{array}$$

(D'Agostino, 1986a). That is, the estimated value of the cdf at the i^{th} order statistic is simply the observed proportion of observations less than or equal to the i^{th} order statistic. This estimator is sometimes called the "empirical probabilities" estimator and is intuitively appealing. The function `pemp` uses the above equations to compute the empirical cdf when `prob.method="emp.probs"`.

For any general value of x, when the observations are assumed to come from a discrete distribution, the value of the cdf is estimated by:

$$\hat{F}(x) = \begin{array}{ll} 0 & \text{if } x < x_{(1)}, \\ \hat{p}_i & \text{if } x_{(i)} \le x < x_{(i+1)}, \\ 1 & \text{if } x \ge x_{(n)} \end{array}$$

The function `pemp` uses the above equation when `discrete=TRUE`.

If the observations are assumed to come from a continuous distribution, the value of the cdf evaluated at the i^{th} order statistic is usually estimated by:

$$\hat{F}[x_{(i)}] = \hat{p}_i = \frac{i - a}{n - 2a + 1}$$

where a denotes the plotting position constant and $0 \le a \le 1$ (Cleveland, 1993, p.18; D'Agostino, 1986a, pp.8,25). The estimators defined by the above equation are called *plotting positions* and are used to construct probability plots. The function `pemp` uses the above equation when `prob.method="plot.pos"`.

For any general value of x, the value of the cdf is estimated by linear interpolation:

$$\hat{F}(x) = \begin{array}{ll} \hat{p}_1 & \text{if } x < x_{(1)}, \\ (1-r)\hat{p}_i + r\hat{p}_{i+1} & \text{if } x_{(i)} \le x < x_{(i+1)}, \\ \hat{p}_n & \text{if } x \ge x_{(n)} \end{array}$$

where

$$r = \frac{x - x_{(i)}}{x_{(i+1)} - x_{(i)}}$$

(Chambers et al., 1983). The function `pemp` uses the above two equations when `discrete=FALSE`.

Estimating Quantiles

The function `qemp` computes the estimated quantiles based on the observed data. If the observations are assumed to come from a discrete distribution, the p^{th} quantile is usually estimated by:

$$\hat{x}_p = \begin{array}{ll} x_{(1)} & \text{if } p \le \hat{p}_1, \\ x_{(i)} & \text{if } \hat{p}_{i-1} < p \le \hat{p}_i, \\ x_n & \text{if } p > \hat{p}_n \end{array}$$

The function `qemp` uses the above equation when `discrete=TRUE`.

If the observations are assumed to come from a continuous distribution, the p^{th} quantile is usually estimated by linear interpolation:

$$\hat{x}_p = \begin{array}{ll} x_{(1)} & \text{if } p \le \hat{p}_1, \\ (1-r)x_{(i-1)} + rx_{(i)} & \text{if } \hat{p}_{i-1} < p \le \hat{p}_i, \\ x_n & \text{if } p > \hat{p}_n \end{array}$$

where

$$r = \frac{p - \hat{p}_{i-1}}{\hat{p}_i - \hat{p}_{i-1}}$$

The function qemp uses the above two equations when discrete=FALSE.

Generating Random Numbers From the Empirical Distribution
The function remp simply calls the R function sample to sample the elements of obs with replacement.

Value

density (demp), probability (pemp), quantile (qemp), or random sample (remp) for the empirical distribution based on the data contained in the vector obs.

Note

The function demp let's you perform nonparametric density estimation. The function pemp computes the value of the empirical cumulative distribution function (ecdf) for user-specified quantiles. The ecdf is a nonparametric estimate of the true cdf (see ecdfPlot). The function qemp computes nonparametric estimates of quantiles (see the help files for eqnpar and quantile). The function remp let's you sample a set of observations with replacement, which is often done while bootstrapping or performing some other kind of Monte Carlo simulation.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11–16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7–62.

Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice and Visualization*. John Wiley and Sons, New York.

Sheather, S. J. and Jones M. C. (1991). A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation. *Journal of the Royal Statististical Society B*, 683–690.

Silverman, B.W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London.

Wegman, E.J. (1972). Nonparametric Probability Density Estimation. *Technometrics* **14**, 533-546.

See Also

density, approx, epdfPlot, ecdfPlot, cdfCompare, qqplot, eqnpar, quantile, sample, simulateVector, simulateMvMatrix.

Examples

```
# Create a set of 100 observations from a gamma distribution with
# parameters shape=4 and scale=5.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(3)
obs <- rgamma(100, shape=4, scale=5)

# Now plot the empirical distribution (with a histogram) and the true distribution:

dev.new()
hist(obs, col = "cyan", xlim = c(0, 65), freq = FALSE,
  ylab = "Relative Frequency")

pdfPlot('gamma', list(shape = 4, scale = 5), add = TRUE)

box()

# Now plot the empirical distribution (based on demp) with the
# true distribution:

x <- qemp(p = seq(0, 1, len = 100), obs = obs)
y <- demp(x, obs)

dev.new()
plot(x, y, xlim = c(0, 65), type = "n",
  xlab = "Value of Random Variable",
  ylab = "Relative Frequency")
lines(x, y, lwd = 2, col = "cyan")

pdfPlot('gamma', list(shape = 4, scale = 5), add = TRUE)

# Alternatively, you can create the above plot with the function
# epdfPlot:

dev.new()
epdfPlot(obs, xlim = c(0, 65), epdf.col = "cyan",
  xlab = "Value of Random Variable",
  main = "Empirical and Theoretical PDFs")

pdfPlot('gamma', list(shape = 4, scale = 5), add = TRUE)




# Clean Up
#---------
rm(obs, x, y)
```

enbinom *Estimate Probability Parameter of a Negative Binomial Distribution*

Description

Estimate the probability parameter of a negative binomial distribution.

Usage

```
enbinom(x, size, method = "mle/mme")
```

Arguments

x	vector of non-negative integers indicating the number of trials that took place *before* size "successes" occurred. (The total number of trials that took place is x+1). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. If length(x)=n and n is greater than 1, it is assumed that x represents observations from n separate negative binomial experiments that all had the same probability of success (prob), but possibly different values of size.
size	vector of positive integers indicating the number of "successes" that must be observed before the trials are stopped. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The length of size must be 1 or else the same length as x.
method	character string specifying the method of estimation. Possible values are: "mle/mme" (maximum likelihood and method of moments; the default) and "mvue" (minimum variance unbiased). You cannot use method="mvue" if the sum of the elements in size is 1. See the DETAILS section for more information on these estimation methods.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n independent observations from negative binomial distributions with parameters prob=p and size=$\underline{k}$, where where $\underline{k} = c(k_1, k_2, \ldots, k_n)$ is a vector of n (possibly different) values.

It can be shown (e.g., Forbes et al., 2011) that if X is defined as:

$$X = \sum_{i=1}^{n} x_i$$

then X is an observation from a negative binomial distribution with parameters prob=p and size=K, where

$$K = \sum_{i=1}^{n} k_i$$

Estimation
The maximum likelihood and method of moments estimator (mle/mme) of p is given by:

$$\hat{p}_{mle} = \frac{K}{X + K}$$

and the minimum variance unbiased estimator (mvue) of p is given by:

$$\hat{p}_{mvue} = \frac{K - 1}{X + K - 1}$$

(Forbes et al., 2011). Note that the mvue of p is not defined for $K = 1$.

Value

a list of class `"estimate"` containing the estimated parameters and other information.
See `estimate.object` for details.

Note

The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.

The geometric distribution with parameter `prob`=p is a special case of the negative binomial distribution with parameters `size=1` and `prob`=p.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 5.

See Also

NegBinomial, `egeom`, Geometric.

Examples

```
# Generate an observation from a negative binomial distribution with
# parameters size=2 and prob=0.2, then estimate the parameter prob.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter that is estimated is prob; the parameter
# size is supplied in the call to enbinom.  The parameter size is printed in
# order to show all of the parameters associated with the distribution.
```

```
set.seed(250)
dat <- rnbinom(1, size = 2, prob = 0.2)
dat
#[1] 5

enbinom(dat, size = 2)
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Negative Binomial
#
#Estimated Parameter(s):          size = 2.0000000
#                                 prob = 0.2857143
#
#Estimation Method:               mle/mme for 'prob'
#
#Data:                            dat, 2
#
#Sample Size:                     1

#----------

# Generate 3 observations from negative binomial distributions with
# parameters size=c(2,3,4) and prob=0.2, then estimate the parameter
# prob using the mvue.
# (Note: the call to set.seed simply allows you to reproduce this example.)

size.vec <- 2:4
set.seed(250)
dat <- rnbinom(3, size = size.vec, prob = 0.2)
dat
#[1]  5 19 12

enbinom(dat, size = size.vec, method = "mvue")
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Negative Binomial
#
#Estimated Parameter(s):          size = 9.0000000
#                                 prob = 0.1818182
#
#Estimation Method:               mvue for 'prob'
#
#Data:                            dat, size.vec
#
#Sample Size:                     3

#----------

# Clean up
#---------
rm(dat)
```

enorm *Estimate Parameters of a Normal (Gaussian) Distribution*

Description

Estimate the mean and standard deviation parameters of a normal (Gaussian) distribution, and optionally construct a confidence interval for the mean or the variance.

Usage

```
enorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "exact", conf.level = 0.95, ci.param = "mean")
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Possible values are "mvue" (minimum variance unbiased; the default), and "mle/mme" (maximum likelihood/method of moments). See the DETAILS section for more information on these estimation methods.
ci	logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the mean or variance. The only possible value is "exact" (the default). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
ci.param	character string indicating which parameter to create a confidence interval for. The possible values are ci.param="mean" (the default) and ci.param="variance". This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from an normal (Gaussian) distribution with parameters mean=μ and sd=σ.

Estimation

Minimum Variance Unbiased Estimation (`method="mvue"`)
The minimum variance unbiased estimators (mvue's) of the mean and variance are:

$$\hat{\mu}_{mvue} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (1)$$

$$\hat{\sigma}^2_{mvue} = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (2)$$

(Johnson et al., 1994; Forbes et al., 2011). Note that when `method="mvue"`, the estimated standard deviation is the square root of the mvue of the variance, but is not itself an mvue.

Maximum Likelihood/Method of Moments Estimation (`method="mle/mme"`)
The maximum likelihood estimator (mle) and method of moments estimator (mme) of the mean are both the same as the mvue of the mean given in equation (1) above. The mle and mme of the variance is given by:

$$\hat{\sigma}^2_{mle} = s^2_m = \frac{n-1}{n}s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (3)$$

When `method="mle/mme"`, the estimated standard deviation is the square root of the mle of the variance, and is itself an mle.

Confidence Intervals

Confidence Interval for the Mean (`ci.param="mean"`)
When `ci=TRUE` and `ci.param = "mean"`, the usual confidence interval for μ is constructed as follows. If `ci.type="two-sided"`, a the $(1-\alpha)100\%$ confidence interval for μ is given by:

$$[\hat{\mu} - t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}},\ \hat{\mu} + t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}}] \quad (4)$$

where $t(\nu, p)$ is the p'th quantile of Student's t-distribution with ν degrees of freedom (Zar, 2010; Gilbert, 1987; Ott, 1995; Helsel and Hirsch, 1992).

If `ci.type="lower"`, the $(1-\alpha)100\%$ confidence interval for μ is given by:

$$[\hat{\mu} - t(n-1, 1-\alpha)\frac{\hat{\sigma}}{\sqrt{n}},\ \infty] \quad (5)$$

and if `ci.type="upper"`, the confidence interval is given by:

$$[-\infty,\ \hat{\mu} + t(n-1, 1-\alpha/2)\frac{\hat{\sigma}}{\sqrt{n}}] \quad (6)$$

Confidence Interval for the Variance (`ci.param="variance"`)
When `ci=TRUE` and `ci.param = "variance"`, the usual confidence interval for σ^2 is constructed as follows. A two-sided $(1-\alpha)100\%$ confidence interval for σ^2 is given by:

$$[\frac{(n-1)s^2}{\chi^2_{n-1,1-\alpha/2}},\ \frac{(n-1)s^2}{\chi^2_{n-1,\alpha/2}}] \quad (7)$$

Similarly, a one-sided upper $(1-\alpha)100\%$ confidence interval for the population variance is given by:

$$[0, \frac{(n-1)s^2}{\chi^2_{n-1,\alpha}}] \quad (8)$$

and a one-sided lower $(1-\alpha)100\%$ confidence interval for the population variance is given by:

$$[\frac{(n-1)s^2}{\chi^2_{n-1,1-\alpha}}, \infty] \quad (9)$$

(van Belle et al., 2004; Zar, 2010).

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean or variance. This is done with confidence intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery

Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

van Belle, G., L.D. Fisher, Heagerty, P.J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences, 2nd Edition*. John Wiley & Sons, New York.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

Normal.

Examples

```
# Generate 20 observations from a N(3, 2) distribution, then estimate
# the parameters and create a 95% confidence interval for the mean.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rnorm(20, mean = 3, sd = 2)
enorm(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 2.861160
#                                 sd   = 1.180226
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 2.308798
#                                 UCL = 3.413523

#----------

# Using the same data, construct an upper 90% confidence interval for
# the variance.

enorm(dat, ci = TRUE, ci.type = "upper", ci.param = "variance")$interval
```

```
#Confidence Interval for:         variance
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 0.000000
#                                 UCL = 2.615963

#----------

# Clean up
#---------
rm(dat)

#----------

# Using the Reference area TcCB data in the data frame EPA.94b.tccb.df,
# estimate the mean and standard deviation of the log-transformed data,
# and construct a 95% confidence interval for the mean.

with(EPA.94b.tccb.df, enorm(log(TcCB[Area == "Reference"]), ci = TRUE))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = -0.6195712
#                                 sd   =  0.4679530
#
#Estimation Method:               mvue
#
#Data:                            log(TcCB[Area == "Reference"])
#
#Sample Size:                     47
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = -0.7569673
#                                 UCL = -0.4821751
```

enormCensored *Estimate Parameters for a Normal Distribution Based on Type I Censored Data*

Description

Estimate the mean and standard deviation of a normal (Gaussian) distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

```
enormCensored(x, censored, method = "mle", censoring.side = "left",
  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
  nmc = 1000, seed = NULL, ...)
```

Arguments

`x` numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.

`censored` numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is `"logical"`, `TRUE` values correspond to elements of `x` that are censored, and `FALSE` values correspond to elements of `x` that are not censored. If the mode of `censored` is `"numeric"`, it must contain only `1`'s and `0`'s; `1` corresponds to `TRUE` and `0` corresponds to `FALSE`. Missing (`NA`) values are allowed but will be removed.

`method` character string specifying the method of estimation.

For singly censored data, the possible values are:
`"mle"` (maximum likelihood; the default),
`"bcmle"` (bias-corrected maximum likelihood),
`"ROS"` or `"qq.reg"` (quantile-quantile regression; also called *regression on order statistics* and abbreviated ROS),
`"qq.reg.w.cen.level"` (quantile-quantile regression including the censoring level),
`"rROS"` or `"impute.w.qq.reg"` (moment estimation based on imputation using quantile-quantile regression; also called *robust regression on order statistics* and abbreviated rROS),
`"impute.w.qq.reg.w.cen.level"` (moment estimation based on imputation using the `qq.reg.w.cen.level` method),
`"impute.w.mle"` (moment estimation based on imputation using the mle),
`"iterative.impute.w.qq.reg"` (moment estimation based on iterative imputation using the `qq.reg` method),
`"m.est"` (robust M-estimation), and
`"half.cen.level"` (moment estimation based on setting the censored observations to half the censoring level).

For multiply censored data, the possible values are:
`"mle"` (maximum likelihood; the default),

"ROS" or "qq.reg" (quantile-quantile regression; also called *regression on order statistics* and abbreviated ROS),
"rROS" or "impute.w.qq.reg" (moment estimation based on imputation using quantile-quantile regression; also called *robust regression on order statistics* and abbreviated rROS), and
"half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

`censoring.side` character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

`ci` logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.

`ci.method` character string indicating what method to use to construct the confidence interval for the mean. The possible values are:
"profile.likelihood" (profile likelihood; the default),
"normal.approx" (normal approximation),
"normal.approx.w.cov" (normal approximation taking into account the covariance between the estimated mean and standard deviation; only available for singly censored data),
"gpq" (generalized pivotal quantity), and
"bootstrap" (based on bootstrapping).

See the DETAILS section for more information. This argument is ignored if ci=FALSE.

`ci.type` character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

`n.bootstraps` numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when ci.type="bootstrap". This argument is ignored if ci=FALSE and/or ci.method does not equal "bootstrap".

`pivot.statistic`
character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when ci.method="normal.approx" or ci.method="normal.approx.w.cov" (see the DETAILS section). The possible values are pivot.statistic="z" (the default) and pivot.statistic="t". When pivot.statistic="t" you may supply the argument ci.sample size (see below). The argument pivot.statistic is ignored if ci=FALSE.

`nmc` numeric scalar indicating the number of Monte Carlo simulations to run when ci.method="gpq". The default is nmc=1000. This argument is ignored if ci=FALSE.

`seed` integer supplied to the function set.seed and used when ci.method="bootstrap" or ci.method="gpq". The default value is seed=NULL, in which case the current value of .Random.seed is used. This argument is ignored when ci=FALSE.

... additional arguments to pass to other functions.

- prob.method. Character string indicating what method to use to compute the plotting positions (empirical probabilities) when method is one of "ROS", "qq.reg", "qq.reg.w.cen.level", "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Possible values are:
"kaplan-meier" (product-limit method of Kaplan and Meier (1958)),
"nelson" (hazard plotting method of Nelson (1972)),
"michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and
"hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)).
The default value is prob.method="hirsch-stedinger". The "nelson" method is only available for censoring.side="right". See the DETAILS section and the help file for ppointsCensored for more information.
- plot.pos.con. Numeric scalar between 0 and 1 containing the value of the plotting position constant to use when method is one of "ROS", "qq.reg", "qq.reg.w.cen.level", "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". The default value is plot.pos.con=0.375. See the DETAILS section and the help file for ppointsCensored for more information.
- ci.sample.size. Numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if pivot.statistic="t" and ci.method="normal.approx" or ci.method="normal.approx.w.cov". When method equals "mle" or "bcmle", the default value is the expected number of uncensored observations, otherwise it is the observed number of uncensored observations.
- lb.impute. Numeric scalar indicating the lower bound for imputed observations when method is one of "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values smaller than this value will be set to this value. The default is lb.impute=-Inf.
- ub.impute. Numeric scalar indicating the upper bound for imputed observations when method is one of "rROS", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", or "iterative.impute.w.qq.reg". Imputed values larger than this value will be set to this value. The default is ub.impute=Inf.
- convergence. Character string indicating the kind of convergence criterion when method="iterative.impute.w.qq.reg". The possible values are "relative" (the default) and "absolute". See the DETAILS section for more information.
- tol. Numeric scalar indicating the convergence tolerance when method="iterative.impute.w.qq.reg". The default value is tol=1e-6. If convergence="relative", then the relative difference in the old and new estimates of the mean and the relative difference in the old and new estimates of the standard deviation must be less than tol for convergence to be achieved. If convergence="absolute", then the absolute difference

in the old and new estimates of the mean and the absolute difference in the old and new estimates of the standard deviation must be less than `tol` for convergence to be achieved.

- `max.iter`. Numeric scalar indicating the maximum number of iterations when `method="iterative.impute.w.qq.reg"`.
- `t.df`. Numeric scalar greater than or equal to 1 that determines the robustness and efficiency properties of the estimator when `method="m.est"`. The default value is `t.df=3`.

Details

If `x` or `censored` contain any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x}$ denote a vector of N observations from a normal distribution with mean μ and standard deviation σ. Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_k;\ k \geq 1 \qquad (1)$$

For the case when $k \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $k = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j = 1, 2, \ldots, k$, so that

$$\sum_{i=1}^{k} c_j = c \qquad (2)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations.

ESTIMATION

Estimation Methods for Multiply and Singly Censored Data

The following methods are available for multiply and singly censored data.

Maximum Likelihood Estimation (`method="mle"`)

For Type I left censored data, the likelihood function is given by:

$$L(\mu, \sigma|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (3)$$

where f and F denote the probability density function (pdf) and cumulative distribution function (cdf) of the population. That is,

$$f(t) = \phi(\frac{t-\mu}{\sigma}) \qquad (4)$$

$$F(t) = \Phi(\frac{t-\mu}{\sigma}) \qquad (5)$$

where ϕ and Φ denote the pdf and cdf of the standard normal distribution, respectively (Cohen, 1963; 1991, pp.6, 50). For left singly censored data, Equation (3) simplifies to:

$$L(\mu, \sigma|\underline{x}) = \binom{N}{c}[F(T)]^c \prod_{i=c+1}^{n} f[x_{(i)}] \qquad (6)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\mu, \sigma|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k}[1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (7)$$

and for right singly censored data this simplifies to:

$$L(\mu, \sigma|\underline{x}) = \binom{N}{c}[1 - F(T)]^c \prod_{i=1}^{n} f[x_{(i)}] \qquad (8)$$

The maximum likelihood estimators are computed by maximizing the likelihood function. For right-censored data, Cohen (1963; 1991, pp.50-51) shows that taking partial derivatives of the log-likelihood function with respect to μ and σ and setting these to 0 produces the following two simultaneous equations:

$$\bar{x} - \mu = -\sigma \sum_{i=1}^{k} (\frac{c_j}{n}) Q_j \qquad (9)$$

$$s^2 + (\bar{x} - \mu)^2 = \sigma^2[1 - \sum_{j=1}^{k} \zeta_j (\frac{c_j}{n}) Q_j] \qquad (10)$$

where

$$\bar{x} = \frac{1}{n} \sum_{i \in \Omega} x_{(i)} \qquad (11)$$

$$s^2 = \frac{1}{n} \sum_{i \in \Omega} (x_{(i)} - \bar{x})^2 \qquad (12)$$

$$Q_j = Q(\zeta_j) \qquad (13)$$

$$\zeta_j = \frac{T_j - \mu}{\sigma} \qquad (14)$$

$$Q(t) = \frac{\phi(t)}{1 - \Phi(t)} \qquad (15)$$

Note that the quantity defined in Equation (11) is simply the mean of the uncensored observations, the quantity defined in Equation (12) is simply the method of moments estimator of variance based

on the uncensored observations, and the function $Q()$ defined in Equation (15) is the hazard function for the standard normal distribution.

For left-censored data, Equations (9) and (10) stay the same, except ζ is replaced with $-\zeta$.

The function `enormCensored` computes the maximum likelihood estimators by solving Equations (9) and (10) and uses the quantile-quantile regression estimators (see below) as initial values.

Regression on Order Statistics (`method="ROS"`) or
Quantile-Quantile Regression (`method="qq.reg"`)
This method is sometimes called the *probability plot method* (Nelson, 1982, Chapter 3; Gilbert, 1987, pp.134-136; Helsel and Hirsch, 1992, p. 361), and more recently also called *parametric regression on order statistics* or *ROS* (USEPA, 2009; Helsel, 2012). In the case of no censoring, it is well known (e.g., Nelson, 1982, p.113; Cleveland, 1993, p.31) that for the standard normal (Gaussian) quantile-quantile plot (i.e., the plot of the sorted observations (empirical quantiles) versus standard normal quantiles; see `qqPlot`), the intercept and slope of the fitted least-squares line estimate the mean and standard deviation, respectively. Specifically, the estimates of μ and σ are found by computing the least-squares estimates in the following model:

$$x_{(i)} = \mu + \sigma\Phi^{-1}(p_i) + \epsilon_i, \; i = 1, 2, \ldots, N \qquad (16)$$

where

$$p_i = \frac{i - a}{N - 2a + 1} \qquad (17)$$

denotes the plotting position associated with the i'th largest value, a is a constant such that $0 \leq a \leq 1$ (the plotting position constant), and Φ denotes the cumulative distribution function (cdf) of the standard normal distribution. The default value of a is 0.375 (see below).

This method can be adapted to the case of left (right) singly censored data as follows. Plot the n uncensored observations against the n largest (smallest) normal quantiles, where the normal quantiles are computed based on a sample size of N, fit the least-squares line to this plot, and estimate the mean and standard deviation from the intercept and slope, respectively. That is, use Equations (16) and (17), but for right singly censored data use $i = 1, 2, \ldots, n$, and for left singly censored data use $i = (c + 1), (c + 2), \ldots, N$.

The argument `plot.pos.con` (see the entry for ... in the ARGUMENTS section above) determines the value of the plotting positions computed in Equation (18). The default value is `plot.pos.con=0.375`. See the help file for `qqPlot` for more information.

This method is discussed by Haas and Scheff (1990). In the context of lognormal data, Travis and Land (1990) suggest exponentiating the predicted 50'th percentile from this fit to estimate the geometric mean (i.e., the median of the lognormal distribution).

This method is easily extended to multiply censored data. Equation (16) becomes

$$x_{(i)} = \mu + \sigma\Phi^{-1}(p_i) + \epsilon_i, \; i \in \Omega \qquad (18)$$

where Ω denotes the set of n subscripts associated with the uncensored observations in the ordered sample. The plotting positions are computed by calling the **EnvStats** function `ppointsCensored`. The argument `prob.method` determines the method of computing the plotting positions (default is `prob.method="hirsch-stedinger"`), and the argument `plot.pos.con` determines the plotting position constant (default is `plot.pos.con=0.375`). (See the entry for ... in the ARGUMENTS section above.) Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger probability

method but set the plotting position constant to 0.

Robust Regression on Order Statistics (`method="rROS"`) or
Imputation Using Quantile-Quantile Regression (`method="impute.w.qq.reg"`)
This is the robust Regression on Order Statistics (rROS) method discussed in USEPA (2009) and Helsel (2012). It involves using the quantile-quantile regression method (`method="qq.reg"` or `method="ROS"`) to fit a regression line (and thus initially estimate the mean and standard deviation), and then imputing the values of the censored observations by predicting them from the regression equation. The final estimates of the mean and standard deviation are then computed using the usual formulas (see `enorm`) based on the observed and imputed values.

The imputed values are computed as:

$$\hat{x}_{(i)} = \hat{\mu}_{qqreg} + \hat{\sigma}_{qqreg}\Phi^{-1}(p_i),\ i \notin \Omega \qquad (19)$$

See the help file for `ppointsCensored` for information on how the plotting positions for the censored observations are computed.

The argument `prob.method` determines the method of computing the plotting positions (default is `prob.method="hirsch-stedinger"`), and the argument `plot.pos.con` determines the plotting position constant (default is `plot.pos.con=0.375`). (See the entry for ... in the ARGUMENTS section above.) Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger probability method but set the plotting position constant to 0.

The arguments `lb.impute` and `ub.impute` determine the lower and upper bounds for the imputed values. Imputed values smaller than `lb.impute` are set to this value. Imputed values larger than `ub.impute` are set to this value. The default values are `lb.impute=-Inf` and `ub.impute=Inf`. See the entry for ... in the ARGUMENTS section above.

For singly censored data, this is the NR method of Gilliom and Helsel (1986, p. 137). In the context of lognormal data, this method is discussed by Hashimoto and Trussell (1983), Gilliom and Helsel (1986), and El-Shaarawi (1989), and is referred to as the LR or Log-Probability Method.

For multiply censored data, this method was developed in the context of lognormal data by Helsel and Cohn (1988) using the formulas for plotting positions given in Hirsch and Stedinger (1987) and Weibull plotting positions (i.e., `prob.method="hirsch-stedinger"` and `plot.pos.con=0`).

Setting Censored Observations to Half the Censoring Level (`method="half.cen.level"`)
This method is applicable only to left censored data that is bounded below by 0. This method involves simply replacing all the censored observations with half their detection limit, and then computing the mean and standard deviation with the usual formulas (see `enorm`).

This method is included only to allow comparison of this method to other methods. ***Setting left-censored observations to half the censoring level is not recommended***.

For singly censored data, this method is discussed by Gleit (1985), Haas and Scheff (1990), and El-Shaarawi and Esterby (1992). El-Shaarawi and Esterby (1992) show that these estimators are biased and inconsistent (i.e., the bias remains even as the sample size increases).

For multiply censored data, this method was studied by Helsel and Cohn (1988).

Estimation Methods for Singly Censored Data
The following methods are available only for singly censored data.

Bias-Corrected Maximum Likelihood Estimation (`method="bcmle"`)
The maximum likelihood estimates of μ and σ are biased. The bias tends to 0 as the sample size increases, but it can be considerable for small sample sizes, especially in the case of a large percentage of censored observations (Saw, 1961b). Schmee et al. (1985) note that bias and variances of the mle's are of the order $1/N$ (see for example, Bain and Engelhardt, 1991), and that for 90% censoring the bias is negligible if N is at least 100. (For less intense censoring, even fewer observations are needed.)

The exact bias of each estimator is extremely difficult to compute. Saw (1961b), however, derived the first-order term (i.e., the term of order $1/N$) in the bias of the mle's of μ and σ and proposed bias-corrected mle's. His bias-corrected estimators were derived for the case of Type II singly censored data. Schneider (1986, p.110) and Haas and Scheff (1990), however, state that this bias correction should reduce the bias of the estimators in the case of Type I censoring as well.

Based on the tables of bias-correction terms given in Saw (1961b), Schneider (1986, pp.107-110) performed a least-squares fit to produce the following computational formulas for right-censored data:

$$B_\mu = -exp[2.692 - 5.493\frac{n}{N+1}] \qquad (20)$$

$$B_\sigma = -[0.312 + 0.859\frac{n}{N+1}]^{-2} \qquad (21)$$

$$\hat{\mu}_{bcmle} = \hat{\mu}_{mle} - \frac{\hat{\sigma}_{mle}}{N+1}B_\mu \qquad (22)$$

$$\hat{\sigma}_{bcmle} = \hat{\sigma}_{mle} - \frac{\hat{\sigma}_{mle}}{N+1}B_\sigma \qquad (23)$$

For left-censored data, Equation (22) becomes:

$$\hat{\mu}_{bcmle} = \hat{\mu}_{mle} + \frac{\hat{\sigma}_{mle}}{N+1}B_\mu \qquad (22)$$

Quantile-Quantile Regression Including the Censoring Level (`method="qq.reg.w.cen.level"`)
This is a modification of the quantile-quantile regression method and was proposed by El-Shaarawi (1989) in the context of lognormal data. El-Shaarawi's idea is to include the censoring level and an associated plotting position, along with the uncensored observations and their associated plotting positions, in order to include information about the value of the censoring level T.

For left singly censored data, the modification involves adding the point $[\Phi^{-1}(p_c), T]$ to the plot before fitting the least-squares line. For right singly censored data, the point $[\Phi^{-1}(p_{n+1}), T]$ is added to the plot before fitting the least-squares line.

El-Shaarawi (1989) also proposed replacing the estimated normal quantiles with the exact expected values of normal order statistics, and using the values in their variance-covariance matrix to perform a weighted least least-squared regression. These last two modifications are not incorporated here.

Imputation Using Quantile-Quantile Regression Including the Censoring Level
(`method ="impute.w.qq.reg.w.cen.level"`)
This is exactly the same method as imputation using quantile-quantile regression (`method="impute.w.qq.reg"`), except that the quantile-quantile regression including the censoring level method (`method="qq.reg.w.cen.level"`) is used to fit the regression line. In the context

of lognormal data, this method is discussed by El-Shaarawi (1989), which he denotes as the Modified LR Method.

Imputation Using Maximum Likelihood (`method ="impute.w.mle"`)
This is exactly the same method as imputation with quantile-quantile regression (`method="impute.w.qq.reg"`), except that the maximum likelihood method (`method="mle"`) is used to compute the initial estimates of the mean and standard deviation. In the context of lognormal data, this method is discussed by El-Shaarawi (1989), which he denotes as the Modified Maximum Likelihood Method.

Iterative Imputation Using Quantile-Quantile Regression (`method="iterative.impute.w.qq.reg"`)
This method is similar to the imputation with quantile-quantile regression method (`method="impute.w.qq.reg"`), but iterates until the estimates of the mean and standard deviation converge. The algorithm is:

1. Compute the initial estimates of μ and σ using the `"impute.w.qq.reg"` method. (Actually, any suitable estimates will do.)
2. Using the current values of μ and σ and Equation (19), compute new imputed values of the censored observations.
3. Use the new imputed values along with the uncensored observations to compute new estimates of μ and σ based on the usual formulas (see `enorm`).
4. Repeat Steps 2 and 3 until the estimates converge (the convergence criterion is determined by the arguments `tol` and `convergence`; see the entry for ... in the ARGUMENTS section above).

This method is discussed by Gleit (1985), which he denotes as "Fill-In with Expected Values".

M-Estimators (`method="m.est"`)
This method was contributed by Leo R. Korn (Korn and Tyler, 2001). This method finds location and scale estimates that are consistent at the normal model and robust to deviations from the normal model, including both outliers on the right and outliers on the left above and below the limit of detection. The estimates are found by solving the simultaneous equations:

$$\sum_{i=1}^{c} h_\nu(\frac{T-\mu}{\sigma}) + \sum_{i=c+1}^{N} \psi_\nu(\frac{x_i-\mu}{\sigma}) = 0 \qquad (23)$$

$$\sum_{i=1}^{c} \lambda_\nu(\frac{T-\mu}{\sigma}) + \sum_{i=c+1}^{N} \chi_\nu(\frac{x_i-\mu}{\sigma}) = 0 \qquad (24)$$

where

$$H_\nu(r) = -log[F_\nu(r)] \qquad (25)$$

$$h_\nu(r) = \frac{d}{dr} H_\nu(r) = H'_\nu(r) \qquad (26)$$

$$\rho_\nu(r) = -log[f_\nu(r)] \qquad (27)$$

$$\psi_\nu(r) = \frac{d}{dr} \rho_\nu(r) = \rho'_\nu(r) \qquad (28)$$

$$\lambda_\nu(r) = r h_\nu(r) \qquad (29)$$

$$\chi_\nu(r) = r\psi_\nu(r) - 1 \qquad (30)$$

and f_ν and F_ν denote the probability density function (pdf) and cumulative distribution function (cdf) of Student's t-distribution with ν degrees of freedom.

This results in an M-estimating equation based on the t-density function (Korn and Tyler., 2001). Since the t-density has heavier tails than the normal density, this M-estimator will tend to down-weight values that are far away from the center of the data. When censoring is present, neither the location nor the scale estimates are consistent at the normal model. A computational correction is performed that converts the above M-estimator to another M-estimator that is consistent at the normal model, even under censoring.

The degrees of freedom parameter ν is set by the argument `t.df` and may be viewed as a tuning parameter that will determine the robustness and efficiency properties. When `t.df` is large, the estimator is similar to the usual mle and the output will then be very close to that when `method="mle"`. As `t.df` decreases, the efficiency will decline and the outlier rejection property will increase in strength. Choosing `t.df=3` (the default) provides a good combination of efficiency and robustness. A reasonable strategy is to transform the data so that they are approximately symmetric (often the log transformation for environmental data is appropriate) and then apply the M-estimator using `t.df=3`.

CONFIDENCE INTERVALS

This section explains how confidence intervals for the mean μ are computed.

Likelihood Profile (`ci.method="profile.likelihood"`)

This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean μ while treating the standard deviation σ as a nuisance parameter. Equation (3) above shows the form of the likelihood function $L(\mu, \sigma|\underline{x})$ for multiply left-censored data, and Equation (7) shows the function for multiply right-censored data.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimates of the mean and standard deviation by (μ^*, σ^*). The likelihood ratio test statistic (G^2) of the hypothesis $H_0 : \mu = \mu_0$ (where μ_0 is a fixed value) equals the drop in $2log(L)$ between the "full" model and the reduced model with μ fixed at μ_0, i.e.,

$$G^2 = 2\{log[L(\mu^*, \sigma^*)] - log[L(\mu_0, \sigma_0^*)]\} \qquad (30)$$

where σ_0^* is the maximum likelihood estimate of σ for the reduced model (i.e., when $\mu = \mu_0$). Under the null hypothesis, the test statistic G^2 follows a chi-squared distribution with 1 degree of freedom.

Alternatively, we may express the test statistic in terms of the profile likelihood function L_1 for the mean μ, which is obtained from the usual likelihood function by maximizing over the parameter σ, i.e.,

$$L_1(\mu) = max_\sigma L(\mu, \sigma) \qquad (31)$$

Then we have

$$G^2 = 2\{log[L_1(\mu^*)] - log[L_1(\mu_0)]\} \qquad (32)$$

A two-sided $(1-\alpha)100\%$ confidence interval for the mean μ consists of all values of μ_0 for which the test is not significant at level $alpha$:

$$\mu_0 : G^2 \leq \chi^2_{1,1-\alpha} \qquad (33)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-squared distribution with ν degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1-\alpha$ in Equation (33) is replaced with $1-2\alpha$.

Normal Approximation (`ci.method="normal.approx"`)
This method constructs approximate $(1-\alpha)100\%$ confidence intervals for μ based on the assumption that the estimator of μ is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$ confidence interval for μ is constructed as:

$$[\hat{\mu} - t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}},\ \hat{\mu} + t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}}] \qquad (34)$$

where $\hat{\mu}$ denotes the estimate of μ, $\hat{\sigma}_{\hat{\mu}}$ denotes the estimated asymptotic standard deviation of the estimator of μ, m denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument `ci.sample.size` determines the value of m (see see the entry for ... in the ARGUMENTS section above). When `method` equals `"mle"` or `"bcmle"`, the default value is the expected number of uncensored observations, otherwise it is the observed number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When `pivot.statistic="z"`, the p'th quantile from the standard normal distribution is used in place of the p'th quantile from Student's t-distribution.

Approximate Confidence Interval Based on Maximum Likelihood Estimators
When `method="mle"`, the standard deviation of the mle of μ is estimated based on the inverse of the Fisher Information matrix. The estimated variance-covariance matrix for the estimates of μ and σ are based on the observed information matrix, formulas for which are given in Cohen (1991).

Approximate Confidence Interval Based on Bias-Corrected Maximum Likelihood Estimators
When `method="bcmle"` (available only for singly censored data), the same procedures are used to construct the confidence interval as for `method="mle"`. The true variance of the bias-corrected mle of μ is necessarily larger than the variance of the mle of μ (although the differences in the variances goes to 0 as the sample size gets large). Hence this method of constructing a confidence interval leads to intervals that are too short for small sample sizes, but these intervals should be better centered about the true value of μ.

Approximate Confidence Interval Based on Other Estimators
When method is some value other than `"mle"`, the standard deviation of the estimated mean is approximated by

$$\hat{\sigma}_{\hat{\mu}} = \frac{\hat{\sigma}}{\sqrt{m}} \qquad (35)$$

where, as already noted, m denotes the assumed sample size. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

Normal Approximation Using Covariance (`ci.method="normal.approx.w.cov"`) This method is only available for singly censored data and only applicable when `method="mle"` or `method="bcmle"`.

It was proposed by Schneider (1986, pp. 191-193) for the case of Type II censoring, but is applicable to any situation where the estimated mean and standard deviation are consistent estimators and are correlated. In particular, the mle's of μ and σ are correlated under Type I censoring as well.

Schneider's idea is to determine two positive quantities z_1, z_2 such that

$$Pr(\hat{\mu} + z_1\hat{\sigma} < \mu) = \frac{\alpha}{2} \qquad (36)$$

$$Pr(\hat{\mu} - z_2\hat{\sigma} > \mu) = \frac{\alpha}{2} \qquad (37)$$

so that

$$[\hat{\mu} - z_2\hat{\sigma},\ \hat{\mu} + z_1\hat{\sigma}] \qquad (38)$$

is a $(1-\alpha)100\%$ confidence interval for μ.

For cases where the estimators of μ and σ are independent (e.g., complete samples), it is well known that setting

$$z_1 = z_2 = \frac{t_{1-\alpha/2,N}}{\sqrt{N}} \qquad (39)$$

yields an exact confidence interval and setting

$$z_1 = z_2 = \frac{z_{1-\alpha/2}}{\sqrt{N}} \qquad (40)$$

where z_p denotes the p'th quantile of the standard normal distribution yields an approximate confidence interval that is asymptotically correct.

For the general case, Schneider (1986) considers the random variable

$$W(z) = \hat{\mu} + z\hat{\sigma} \qquad (41)$$

and provides formulas for z_1 and z_2.

Note that the resulting confidence interval for the mean is not symmetric about the estimated mean. Also note that the quantity m is a random variable for Type I censoring, while Schneider (1986) assumed it to be fixed since he derived the result for Type II censoring (in which case $m = n$).

Bootstrap and Bias-Corrected Bootstrap Approximation (`ci.method="bootstrap"`)
The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1-\alpha)100\%$ confidence interval for the population mean μ, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size N from the observations in $\underline{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\underline{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).

2. Estimate μ based on the bootstrap sample created in Step 1, using the same method that was used to estimate μ using the original observations in $\underline{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of μ based on the bootstrap sample will usually differ from the original estimate based on $\underline{x}$.
3. Repeat Steps 1 and 2 B times, where B is some large number. For the function `enormCensored`, the number of bootstraps B is determined by the argument `n.bootstraps` (see the section ARGUMENTS above). The default value of `n.bootstraps` is `1000`.
4. Use the B estimated values of μ to compute the empirical cumulative distribution function of this estimator of μ (see `ecdfPlot`), and then create a confidence interval for μ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}),\ \hat{G}^{-1}(1-\frac{\alpha}{2})] \qquad (42)$$

where $\hat{G}(t)$ denotes the empirical cdf evaluated at t and thus $\hat{G}^{-1}(p)$ denotes the p'th empirical quantile, that is, the p'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha),\ \infty] \qquad (43)$$

and a one-sided upper confidence interval is computed as:

$$[-\infty,\ \hat{G}^{-1}(1-\alpha)] \qquad (44)$$

The function `enormCensored` calls the R function `quantile` to compute the empirical quantiles used in Equations (42)-(44).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of μ can be off by $k/\sqrt{N}$, where k is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of μ may be off by k/N instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1),\ \hat{G}^{-1}(\alpha_2)] \qquad (45)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1-\hat{a}(z_0+z_{\alpha/2})}] \qquad (46)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1-\hat{a}(z_0+z_{1-\alpha/2})}] \qquad (47)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\mu})] \qquad (48)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^3}{6[\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^2]^{3/2}} \qquad (49)$$

where the quantity $\hat{\mu}_{(i)}$ denotes the estimate of μ using all the values in $\underline{x}$ except the i'th one, and

$$\hat{\mu}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\hat{\mu_{(i)}} \qquad (50)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1),\ \infty] \qquad (51)$$

and a one-sided upper confidence interval is given by:

$$[-\infty,\ \hat{G}^{-1}(\alpha_2)] \qquad (52)$$

where α_1 and α_2 are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with α in Equations (46) and (47).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of μ with respect to the true value of μ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of μ does not depend on the value of μ, hence the acceleration constant is not really necessary.

When `ci.method="bootstrap"`, the function `enormCensored` computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

This method of constructing confidence intervals for censored data was studied by Shumway et al. (1989).

Generalized Pivotal Quantity (`ci.method="gpq"`)
This method was introduced by Schmee et al. (1985) and is discussed by Krishnamoorthy and Mathew (2009). The idea is essentially to use a parametric bootstrap to estimate the correct pivotal quantities z_1 and z_2 in Equation (38) above. For singly censored data, these quantities are computed as follows:

1. Generate a random sample of N observations from a standard normal (i.e., N(0,1)) distribution and let $z_{(1)}, z_{(2)}, \ldots, z_{(N)}$ denote the ordered (sorted) observations.
2. Set the smallest `c` observations to be censored.
3. Compute the estimates of μ and σ using the method specified by the `method` argument, and denote these estimates as $\hat{\mu}^*,\ \hat{\sigma}^*$.
4. Compute the t-like pivotal quantity $\hat{t} = \hat{\mu}^*/\hat{\sigma}^*$.
5. Repeat steps 1-4 `nmc` times to produce an empirical distribution of the t-like pivotal quantity.

The function `enormCensored` calls the function `gpqCiNormSinglyCensored` to generate the distribution of pivotal quantities in the case of singly censored data. A two-sided $(1-\alpha)100\%$ confidence interval for μ is then computed as:

$$[\hat{\mu} - \hat{t}_{1-(\alpha/2)}\hat{\sigma},\ \hat{\mu} - \hat{t}_{\alpha/2}\hat{\sigma}] \qquad (49)$$

where $\hat{t}_p$ denotes the p'th empirical quantile of the `nmc` generated $\hat{t}$ values.

Schmee at al. (1985) derived this method in the context of Type II singly censored data (for which these limits are exact within Monte Carlo error), but state that according to Regal (1982) this method produces confidence intervals that are close apporximations to the correct limits for Type I censored data.

For multiply censored data, this method has been extended as follows. The algorithm stays the same, except that Step 2 becomes:

2. Set the i'th ordered generated observation to be censored or not censored according to whether the i'th observed observation in the original data is censored or not censored.

The function `enormCensored` calls the function `gpqCiNormMultiplyCensored` to generate the distribution of pivotal quantities in the case of multiply censored data.

Value

a list of class `"estimateCensored"` containing the estimated parameters and other information. See `estimateCensored.object` for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Schmee et al. (1985) studied Type II censoring for a normal distribution and noted that the bias and variances of the maximum likelihood estimators are of the order $1/N$, and that the bias is negligible for $N = 100$ and as much as 90% censoring. (If the proportion of censored observations is less than 90%, the bias becomes negligible for smaller sample sizes.) For small samples with moderate to high censoring, however, the bias of the mle's causes confidence intervals based on them using a normal approximation (e.g., `method="mle"` and `ci.method="normal.approx"`) to be too short. Schmee et al. (1985) provide tables for exact confidence intervals for sample sizes up to $N = 100$ that were created based on Monte Carlo simulation. Schmee et al. (1985) state that these tables should work well for Type I censored data as well.

Shumway et al. (1989) evaluated the coverage of 90% confidence intervals for the mean based on using a Box-Cox transformation to induce normality, computing the mle's based on the normal distribution, then computing the mean in the original scale. They considered three methods of

constructing confidence intervals: the delta method, the bootstrap, and the bias-corrected bootstrap. Shumway et al. (1989) used three parent distributions in their study: Normal(3,1), the square of this distribuiton, and the exponentiation of this distribution (i.e., a lognormal distribution). Based on sample sizes of 10 and 50 with a censoring level at the 10'th or 20'th percentile, Shumway et al. (1989) found that the delta method performed quite well and was superior to the bootstrap method.

Millard et al. (2015; in preparation) show that the coverage of the profile likelihood method is excellent.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bain, L.J., and M. Engelhardt. (1991). *Statistical Analysis of Reliability and Life-Testing Models*. Marcel Dekker, New York, 496pp.

Cohen, A.C. (1959). Simplified Estimators for the Normal Distribution When Samples are Singly Censored or Truncated. *Technometrics* **1**(3), 217–237.

Cohen, A.C. (1963). Progressively Censored Samples in Life Testing. *Technometrics* **5**, 327–339

Cohen, A.C. (1991). *Truncated and Censored Samples*. Marcel Dekker, New York, New York, 312pp.

Cox, D.R. (1970). *Analysis of Binary Data*. Chapman & Hall, London. 142pp.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

El-Shaarawi, A.H. (1989). Inferences About the Mean from Censored Water Quality Data. *Water Resources Research* **25**(4) 685–690.

El-Shaarawi, A.H., and D.M. Dolan. (1989). Maximum Likelihood Estimation of Water Quality Concentrations from Censored Data. *Canadian Journal of Fisheries and Aquatic Sciences* **46**, 1033–1039.

El-Shaarawi, A.H., and S.R. Esterby. (1992). Replacement of Censored Observations by a Constant: An Evaluation. *Water Research* **26**(6), 835–844.

El-Shaarawi, A.H., and A. Naderi. (1991). Statistical Inference from Multiply Censored Environmental Data. *Environmental Monitoring and Assessment* **17**, 339–347.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135–146.

Gleit, A. (1985). Estimation for Small Normal Data Sets with Detection Limits. *Environmental Science and Technology* **19**, 1201–1206.

Haas, C.N., and P.A. Scheff. (1990). Estimation of Averages in Truncated Samples. *Environmental Science and Technology* **24**(6), 912–919.

Hashimoto, L.K., and R.R. Trussell. (1983). Evaluating Water Quality Data Near the Detection Limit. Paper presented at the Advanced Technology Conference, American Water Works Association, Las Vegas, Nevada, June 5-9, 1983.

Helsel, D.R. (1990). Less than Obvious: Statistical Treatment of Data Below the Detection Limit. *Environmental Science and Technology* **24**(12), 1766–1774.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997–2004.

Hirsch, R.M., and J.R. Stedinger. (1987). Plotting Positions for Historical Floods and Their Precision. *Water Resources Research* **23**(4), 715–727.

Korn, L.R., and D.E. Tyler. (2001). Robust Estimation for Chemical Concentration Data Subject to Detection Limits. In Fernholz, L., S. Morgenthaler, and W. Stahel, eds. *Statistics in Genetics and in the Environmental Sciences*. Birkhauser Verlag, Basel, pp.41–63.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Michael, J.R., and W.R. Schucany. (1986). Analysis of Data from Censored Samples. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, 560pp, Chapter 11, 461–496.

Millard, S.P., P. Dixon, and N.K. Neerchal. (2014; in preparation). *Environmental Statistics with R*. CRC Press, Boca Raton, Florida.

Nelson, W. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, 634pp.

Newman, M.C., P.M. Dixon, B.B. Looney, and J.E. Pinder. (1989). Estimating Mean and Variance for Environmental Samples with Below Detection Limit Observations. *Water Resources Bulletin* **25**(4), 905–916.

Pettitt, A. N. (1983). Re-Weighted Least Squares Estimation with Censored and Grouped Data: An Application of the EM Algorithm. *Journal of the Royal Statistical Society, Series B* **47**, 253–260.

Regal, R. (1982). Applying Order Statistic Censored Normal Confidence Intervals to Time Censored Data. Unpublished manuscript, University of Minnesota, Duluth, Department of Mathematical Sciences.

Royston, P. (2007). Profile Likelihood for Estimation and Confdence Intervals. *The Stata Journal* **7**(3), pp. 376–387.

Saw, J.G. (1961b). The Bias of the Maximum Likelihood Estimators of Location and Scale Parameters Given a Type II Censored Normal Sample. *Biometrika* **48**, 448–451.

Schmee, J., D.Gladstein, and W. Nelson. (1985). Confidence Limits for Parameters of a Normal Distribution from Singly Censored Samples, Using Maximum Likelihood. *Technometrics* **27**(2) 119–128.

Schneider, H. (1986). *Truncated and Censored Samples from Normal Populations*. Marcel Dekker, New York, New York, 273pp.

Shumway, R.H., A.S. Azari, and P. Johnson. (1989). Estimating Mean Concentrations Under Transformations for Environmental Data With Detection Limits. *Technometrics* **31**(3), 347–356.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Stryhn, H., and J. Christensen. (2003). *Confidence Intervals by the Profile Likelihood Method, with Applications in Veterinary Epidemiology*. Contributed paper at ISVEE X (November 2003, Chile). https://gilvanguedes.com/wp-content/uploads/2019/05/Profile-Likelihood-CI.pdf.

Travis, C.C., and M.L. Land. (1990). Estimating the Mean of Data Sets with Nondetectable Values. *Environmental Science and Technology* **24**, 961–962.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. Chapter 15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Venzon, D.J., and S.H. Moolgavkar. (1988). A Method for Computing Profile-Likelihood-Based Confidence Intervals. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **37**(1), pp. 87–94.

See Also

Normal, enorm, estimateCensored.object.

Examples

```
# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and standard deviation using the MLE,
# Q-Q regression (also called parametric regression on order statistics
# or ROS; e.g., USEPA, 2009 and Helsel, 2012), and imputation with Q-Q
# regression (also called robust regression on order statistics or rROS).

# We will log-transform the original observations and then call
# enormCensored.  Alternatively, we could have more simply called
# elnormCensored.

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
```

```
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now estimate the mean and standard deviation on the log-scale
# using the MLE:
#---------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  enormCensored(log(Manganese.ppb), Censored))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              0.6931472 1.6094379
#
#Estimated Parameter(s):          mean = 2.215905
#                                 sd   = 1.356291
#
#Estimation Method:               MLE
#
#Data:                            log(Manganese.ppb)
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%

# Now compare the MLE with the estimators based on
# Q-Q regression and imputation with Q-Q regression
#-------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
```

```
  enormCensored(log(Manganese.ppb), Censored))$parameters
#    mean       sd
#2.215905 1.356291

with(EPA.09.Ex.15.1.manganese.df,
  enormCensored(log(Manganese.ppb), Censored,
  method = "ROS"))$parameters
#    mean       sd
#2.293742 1.283635

with(EPA.09.Ex.15.1.manganese.df,
  enormCensored(log(Manganese.ppb), Censored,
  method = "rROS"))$parameters
#    mean       sd
#2.298656 1.238104

#----------

# The method used to estimate quantiles for a Q-Q plot is
# determined by the argument prob.method.  For the functions
# enormCensored and elnormCensored, for any estimation
# method that involves Q-Q regression, the default value of
# prob.method is "hirsch-stedinger" and the default value for the
# plotting position constant is plot.pos.con=0.375.

# Both Helsel (2012) and USEPA (2009) also use the Hirsch-Stedinger
# probability method but set the plotting position constant to 0.

with(EPA.09.Ex.15.1.manganese.df,
  enormCensored(log(Manganese.ppb), Censored,
  method = "rROS", plot.pos.con = 0))$parameters
#    mean       sd
#2.277175 1.261431

#----------

# Using the same data as above, compute a confidence interval
# for the mean on the log-scale using the profile-likelihood
# method.

with(EPA.09.Ex.15.1.manganese.df,
  enormCensored(log(Manganese.ppb), Censored, ci = TRUE))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#----------------------------------------
#
#Assumed Distribution:            Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              0.6931472 1.6094379
#
```

```
#Estimated Parameter(s):          mean = 2.215905
#                                 sd   = 1.356291
#
#Estimation Method:               MLE
#
#Data:                            log(Manganese.ppb)
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Profile Likelihood
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 1.595062
#                                 UCL = 2.771197
```

---

enpar *Estimate Mean, Standard Deviation, and Standard Error Nonparametrically*

Description

Estimate the mean, standard deviation, and standard error of the mean nonparametrically given a sample of data, and optionally construct a confidence interval for the mean.

Usage

```
enpar(x, ci = FALSE, ci.method = "bootstrap", ci.type = "two-sided",
   conf.level = 0.95, pivot.statistic = "z", n.bootstraps = 1000, seed = NULL)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
ci	logical scalar indicating whether to compute a confidence interval for the mean. The default value is ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the mean. The possible values are "bootstrap" (based on bootstrapping; the default), and "normal.approx" (normal approximation). See the **DETAILS** section for more information. This argument is ignored if ci=FALSE.

`ci.type` character string indicating what kind of confidence interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`. This argument is ignored if `ci=FALSE`.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

`pivot.statistic` character string indicating which statistic to use for the confidence interval for the mean when `ci.method="normal.approx"`. Possible values are `"z"` (confidence interval based on the z-statistic; the default), and `"t"` (confidence interval based on the t-statistic). This argument is ignored if `ci=FALSE` or `ci.method="bootstrap"`.

`n.bootstraps` numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean. This argument is ignored if `ci=FALSE` or `ci.method="normal.approx"`.

`seed` integer supplied to the function `set.seed` and used when `ci=TRUE` and `ci.method="bootstrap"`. The default value is `seed=NULL`, in which case the current value of `.Random.seed` is used. This argument is ignored if `ci=FALSE` or `ci.method="normal.approx"`. This argument is necessary to create reproducible results for the bootstrapped confidence intervals (see the **EXAMPLES** section).

Details

Let $\underline{x} = (x_1, x_2, \ldots, x_N)$ denote a vector of N observations from some distribution with mean μ and standard deviation σ.

Estimation

Unbiased and consistent estimators of the mean and variance are:

$$\hat{\mu} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (1)$$

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (2)$$

A consistent (but not unbiased) estimate of the standard deviation is given by the square root of the estimated variance above:

$$\hat{\sigma} = s \quad (3)$$

It can be shown that the variance of the sample mean is given by:

$$\sigma^2_{\hat{\mu}} = \sigma^2_{\bar{x}} = \frac{\sigma^2}{n} \quad (4)$$

so the standard deviation of the sample mean (usually called the *standard error*) can be estimated by:

$$\hat{\sigma}_{\hat{\mu}} = \hat{\sigma}_{\bar{x}} = \frac{s}{\sqrt{n}} \quad (5)$$

Confidence Intervals

This section explains how confidence intervals for the mean μ are computed.

Normal Approximation (`ci.method="normal.approx"`)

This method constructs approximate $(1-\alpha)100\%$ confidence intervals for μ based on the assumption that the estimator of μ, i.e., the sample mean, is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$ confidence interval for μ is constructed as:

$$[\hat{\mu} - t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}},\ \hat{\mu} + t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\mu}}] \quad (6)$$

where $\hat{\mu}$ denotes the estimate of μ, $\hat{\sigma}_{\hat{\mu}}$ denotes the estimated asymptotic standard deviation of the estimator of μ, m denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

When `pivot.statistic="z"`, the p'th quantile from the standard normal distribution is used in place of the p'th quantile from Student's t-distribution.

Bootstrap and Bias-Corrected Bootstrap Approximation (`ci.method="bootstrap"`)

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1-\alpha)100\%$ confidence interval for the population mean μ, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size N from the observations in $\underline{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\underline{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample.
2. Estimate μ based on the bootstrap sample created in Step 1, using the same method that was used to estimate μ using the original observations in $\underline{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of μ based on the bootstrap sample will usually differ from the original estimate based on $\underline{x}$. For the bootstrap-t method (see below), this step also involves estimating the standard error of the estimate of the mean and computing the statistic $T = (\hat{\mu}_B - \hat{\mu})/\hat{\sigma}_{\hat{\mu}_B}$ where $\hat{\mu}$ denotes the estimate of the mean based on the original sample, and $\hat{\mu}_B$ and $\hat{\sigma}_{\hat{\mu}_B}$ denote the estimate of the mean and estimate of the standard error of the estimate of the mean based on the bootstrap sample.
3. Repeat Steps 1 and 2 B times, where B is some large number. For the function `enpar`, the number of bootstraps B is determined by the argument `n.bootstraps` (see the section **ARGUMENTS** above). The default value of `n.bootstraps` is `1000`.
4. Use the B estimated values of μ to compute the empirical cumulative distribution function of the estimator of μ or to compute the empirical cumulative distribution function of the statistic T (see `ecdfPlot`), and then create a confidence interval for μ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}),\ \hat{G}^{-1}(1-\frac{\alpha}{2})] \qquad (7)$$

where $\hat{G}(t)$ denotes the empirical cdf of $\hat{\mu}_B$ evaluated at t and thus $\hat{G}^{-1}(p)$ denotes the p'th empirical quantile of the distribution of $\hat{\mu}_B$, that is, the p'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha),\ \infty] \qquad (8)$$

and a one-sided upper confidence interval is computed as:

$$[-\infty,\ \hat{G}^{-1}(1-\alpha)] \qquad (9)$$

The function `enpar` calls the R function `quantile` to compute the empirical quantiles used in Equations (7)-(9).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of μ can be off by $k/\sqrt{N}$, where k is some constant. Efron and Tibshirani (1993, pp.184–188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of μ may be off by k/N instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1),\ \hat{G}^{-1}(\alpha_2)] \qquad (10)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1-\hat{a}(z_0+z_{\alpha/2})}] \qquad (11)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1-\hat{a}(z_0+z_{1-\alpha/2})}] \qquad (12)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\mu})] \qquad (13)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^3}{6[\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^2]^{3/2}} \qquad (14)$$

where the quantity $\hat{\mu}_{(i)}$ denotes the estimate of μ using all the values in $\underline{x}$ except the i'th one, and

$$\hat{\mu}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\hat{\mu_{(i)}} \qquad (15)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1),\ \infty] \qquad (16)$$

and a one-sided upper confidence interval is given by:

$$[-\infty,\ \hat{G}^{-1}(\alpha_2)] \qquad (17)$$

where α_1 and α_2 are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with α in Equations (11) and (12).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of μ with respect to the true value of μ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of μ does not depend on the value of μ, hence the acceleration constant is not really necessary.

For the bootstrap-t method, the two-sided confidence interval (Efron and Tibshirani, 1993, p.160) is computed as:

$$[\hat{\mu} - t_{1-\alpha/2}\hat{\sigma}_{\hat{\mu}},\ \hat{\mu} - t_{\alpha/2}\hat{\sigma}_{\hat{\mu}}] \qquad (18)$$

where $\hat{\mu}$ and $\hat{\sigma}_{\hat{\mu}}$ denote the estimate of the mean and standard error of the estimate of the mean based on the original sample, and t_p denotes the p'th empirical quantile of the bootstrap distribution of the statistic T. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{\mu} - t_{1-\alpha}\hat{\sigma}_{\hat{\mu}},\ \infty] \qquad (19)$$

and a one-sided upper confidence interval is computed as:

$$[-\infty,\ \hat{\mu} - t_{\alpha}\hat{\sigma}_{\hat{\mu}}] \qquad (20)$$

When `ci.method="bootstrap"`, the function `enpar` computes the percentile method, bias-corrected and accelerated method, and bootstrap-t bootstrap confidence intervals. The percentile method is transformation respecting, but not second-order accurate. The bootstrap-t method is second-order accurate, but not transformation respecting. The bias-corrected and accelerated method is both transformation respecting and second-order accurate (Efron and Tibshirani, 1993, p.188).

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Note

The function `enpar` is related to the companion function `enparCensored` for censored data. To estimate the median and compute a confidence interval, use `eqnpar`.

The result of the call to `enpar` with `ci.method="normal.approx"` and `pivot.statistic="t"` produces the same result as the call to `enorm` with `ci.param="mean"`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

See Also

`enparCensored`, `eqnpar`, `enorm`, `mean`, `sd`, `estimate.object`.

Examples

```
# The data frame ACE.13.TCE.df contains observations on
# Trichloroethylene (TCE) concentrations (mg/L) at
# 10 groundwater monitoring wells before and after remediation.
#
# Compute the mean concentration for each period along with
# a 95% bootstrap BCa confidence interval for the mean.
#
# NOTE: Use of the argument "seed" is necessary to reproduce this example.
#
# Before remediation: 21.6 [14.2, 30.1]
# After remediation:   3.6 [ 1.6,  5.7]

with(ACE.13.TCE.df,
  enpar(TCE.mg.per.L[Period=="Before"], ci = TRUE, seed = 476))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Parameter(s):          mean    = 21.62400
#                                 sd      = 13.51134
#                                 se.mean =  4.27266
#
#Estimation Method:               Sample Mean
#
#Data:                            TCE.mg.per.L[Period == "Before"]
#
#Sample Size:                     10
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Bootstrap
#
#Number of Bootstraps:            1000
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             Pct.LCL = 13.95560
#                                 Pct.UCL = 29.79510
#                                 BCa.LCL = 14.16080
#                                 BCa.UCL = 30.06848
#                                 t.LCL   = 12.41945
#                                 t.UCL   = 32.47306

#----------

with(ACE.13.TCE.df,
  enpar(TCE.mg.per.L[Period=="After"], ci = TRUE, seed = 543))
```

```
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Parameter(s):          mean    = 3.632900
#                                 sd      = 3.554419
#                                 se.mean = 1.124006
#
#Estimation Method:               Sample Mean
#
#Data:                            TCE.mg.per.L[Period == "After"]
#
#Sample Size:                     10
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Bootstrap
#
#Number of Bootstraps:            1000
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             Pct.LCL = 1.833843
#                                 Pct.UCL = 5.830230
#                                 BCa.LCL = 1.631655
#                                 BCa.UCL = 5.677514
#                                 t.LCL   = 1.683791
#                                 t.UCL   = 8.101829
```

---

enparCensored *Estimate Mean, Standard Deviation, and Standard Error Nonparametrically Based on Censored Data*

Description

Estimate the mean, standard deviation, and standard error of the mean nonparametrically given a sample of data from a positive-valued distribution that has been subjected to left- or right-censoring, and optionally construct a confidence interval for the mean.

Usage

```
enparCensored(x, censored, censoring.side = "left", correct.se = TRUE,
  restricted = FALSE, left.censored.min = "Censoring Level",
  right.censored.max = "Censoring Level", ci = FALSE,
  ci.method = "normal.approx", ci.type = "two-sided", conf.level = 0.95,
  pivot.statistic = "t", ci.sample.size = "Total", n.bootstraps = 1000,
  seed = NULL, warn = FALSE)
```

Arguments

x	numeric vector of positive-valued observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
correct.se	logical scalar indicating whether to multiply the estimated standard error by a factor to correct for bias. The default value is correct.se=TRUE. See the DETAILS section below.
restricted	logical scalar indicating whether to compute the restricted mean in the case when the smallest censored value is less than or equal to the smallest uncensored value (left-censored data) or the largest censored value is greater than or equal to the largest uncensored value (right-censored data). The default value is restricted=FALSE. See the DETAILS section for more information.
left.censored.min	Only relevant for the case when censoring.side="left", the smallest censored value is less than or equal to the smallest uncensored value, and restricted=TRUE. In this case, left.censored.min must be the character string "Censoring Level", or else a numeric scalar between 0 and the smallest censored value. The default value is left.censored.min="Censoring Level". See the DETAILS section for more information.
right.censored.max	Only relevant for the case when censoring.side="right", the largest censored value is greater than or equal to the largest uncensored value, and restricted=TRUE. In this case, right.censored.max must be the character string "Censoring Level", or else a numeric scalar greater than or equal to the largest censored value. The default value is right.censored.max="Censoring Level". See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the mean. The possible values are "normal.approx" (normal approximation; the default), and "bootstrap" (based on bootstrapping). See the **DETAILS** section for more information. This argument is ignored if ci=FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

`pivot.statistic`
: character string indicating which statistic to use for the confidence interval for the mean when `ci.method="normal.approx"`. Possible values are `"t"` (confidence interval based on the t-statistic; the default), and `"z"` (confidence interval based on the z-statistic). When `pivot.statistic="t"` you may supply the argument `ci.sample size` (see below). This argument is ignored if `ci=FALSE`.

`ci.sample.size`
: character string indicating what sample size to assume when computing the confidence interval for the mean when `ci.method="normal.approx"` and `pivot.statistic="t"`. Possible values are `ci.sample.size="Total"` (the total number of observations; the default), and `ci.sample.size="Uncensored"` (the number of uncensored observations). This argument is ignored if `ci=FALSE`, `ci.method="bootstrap"`, or `pivot.statistic="z"`.

`n.bootstraps`
: numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when `ci.type="bootstrap"`. This argument is ignored if `ci=FALSE` or `ci.method="normal.approx"`.

`seed`
: integer supplied to the function `set.seed` and used when `ci=TRUE` and `ci.method="bootstrap"`. The default value is `seed=NULL`, in which case the current value of `.Random.seed` is used. This argument is ignored if `ci=FALSE` or `ci.method="normal.approx"`. The `seed` argument is necessary in order to create reproducible results for the bootstrapped confidence intervals (see the **EXAMPLES** section).

`warn`
: logical scalar indicating whether to issue a notification in the case when a restricted mean will be estimated, but setting the smallest censored value(s) to an uncensored value (left-censored data) or setting the largest censored value(s) to an uncensored value (right-censored data) results in no censored values in the data. In this case, the function `enpar` is called.

Details

Let $\underline{x} = (x_1, x_2, \ldots, x_N)$ denote a vector of N observations from some positive-valued distribution with mean μ and standard deviation σ. Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k censoring levels

$$T_1, T_2, \ldots, T_k;\ k \geq 1 \qquad (1)$$

Let $y_1, y_2, \ldots, y_n$ denote the n ordered uncensored observations, and let $r_1, r_2, \ldots, r_n$ denote the order of these uncensored observations within the context of all the observations (censored and uncensored). For example, if the left-censored data are {<10, 14, 14, <15, 20}, then $y_1 = 14, y_2 = 14, y_3 = 20$, and $r_1 = 2, r_2 = 3, r_3 = 5$.

Let $y'_1, y'_2, \ldots, y'_p$ denote the p ordered *distinct* uncensored observations, let m_j denote the number of detects at y'_j ($j = 1, 2, \ldots, p$), and let r'_j denote the number of $x_i \leq y'_j$, i.e., the number of observations (censored and uncensored) less than or equal to y'_j ($j = 1, 2, \ldots, p$). For example, if the left-censored data are {<10, 14, 14, <15, 20}, then $y'_1 = 14, y'_2 = 20$, $m_1 = 2, m_2 = 1$, and $r'_1 = 3, r'_2 = 5$.

Estimation

This section explains how the mean μ, standard deviation σ, and standard error of the mean $\hat{\sigma}_{\hat{\mu}}$ are

estimated, as well as the restricted mean.

Estimating the Mean

It can be shown that the mean of a positive-valued distribution is equal to the area under the survival curve (Klein and Moeschberger, 2003, p.33):

$$\mu = \int_0^\infty [1 - F(t)]dt = \int_0^\infty S(t)dt \qquad (2)$$

where $F(t)$ denotes the cumulative distribution function evaluated at t and $S(t) = 1 - F(t)$ denotes the survival function evaluated at t. When the Kaplan-Meier estimator is used to construct the survival function, you can use the area under this curve to estimate the mean of the distribution, and the estimator can be as efficient or more efficient than parametric estimators of the mean (Meier, 2004; Helsel, 2012; Lee and Wang, 2003). Let $\hat{F}(t)$ denote the Kaplan-Meier estimator of the empirical cumulative distribution function (ecdf) evaluated at t, and let $\hat{S}(t) = 1 - \hat{F}(t)$ denote the estimated survival function evaluated at t. (See the help files for `ecdfPlotCensored` and `qqPlotCensored` for an explanation of how the Kaplan-Meier estimator of the ecdf is computed.)

The formula for the estimated mean is given by (Lee and Wang, 2003, p. 74):

$$\hat{\mu} = \sum_{i=1}^{n} \hat{S}(y_{i-1})(y_i - y_{i-1}) \qquad (3)$$

where $y_0 = 0$ and $\hat{S}(y_0) = 1$ by definition. It can be shown that this formula is eqivalent to:

$$\hat{\mu} = \sum_{i=1}^{n} y_i[\hat{F}(y_i) - \hat{F}(y_{i-1})] \qquad (4)$$

where $\hat{F}(y_0) = \hat{F}(0) = 0$ by definition, and this is equivalent to:

$$\hat{\mu} = \sum_{i=1}^{p} y'_i[\hat{F}(y'_i) - \hat{F}(y'_{i-1})] \qquad (5)$$

(USEPA, 2009, pp. 15–7 to 15–12; Beal, 2010; USEPA, 2022, pp. 128–129).

Estimating the Standard Deviation

The formula for the estimated standard deviation is:

$$\hat{\sigma} = \{\sum_{i=1}^{n}(y_i - \hat{\mu})^2[\hat{F}(y_i) - \hat{F}(y_{i-1})]\}^{1/2} \qquad (6)$$

which is equivalent to:

$$\hat{\sigma} = \{\sum_{i=1}^{p}(y'_i - \hat{\mu})^2[\hat{F}(y'_i) - \hat{F}(y'_{i-1})]\}^{1/2} \qquad (7)$$

(USEPA, 2009, p. 15-10; Beal, 2010).

Estimating the Standard Error of the Mean

For left-censored data, the formula for the estimated standard error of the mean is:

$$\hat{\sigma}_{\hat{\mu}} = [\sum_{i=j}^{p-1} A_j^2 \frac{m_{j+1}}{r'_{j+1}(r'_{j+1} - m_{j+1})}]^{1/2} \qquad (8)$$

where

$$A_j = \sum_{i=1}^{j} (y'_{i+1} - y'_i)\hat{F}(y'_i) \qquad (9)$$

(Beal, 2010; USEPA, 2022, pp. 128–129).

For rigth-censored data, the formula for the estimated standard error of the mean is:

$$\hat{\sigma}_{\hat{\mu}} = [\sum_{r=1}^{n-1} \frac{A_r^2}{(N-r)(N-r+1)}]^{1/2} \qquad (10)$$

where

$$A_r = \sum_{i=r}^{n-1} (y_{i+1} - y_i)\hat{S}(y_i) \qquad (11)$$

(Lee and Wang, 2003, p. 74).

Kaplan and Meier suggest using a bias correction of $n/(n-1)$ for the estimated variance of the mean (Lee and Wang, 2003, p.75):

$$\hat{\sigma}_{\hat{\mu},BC} = \sqrt{\frac{n}{n-1}}\ \hat{\sigma}_{\hat{\mu}} \qquad (12)$$

When `correct.se=TRUE` (the default), Equation (12) is used. Beal (2010), ProUCL 5.2.0 (USEPA, 2022), and the `kmms` function in the **STAND** package (Frome and Frome, 2015) all compute the bias-corrected estimate of the standard error of the mean as well.

Estimating the Restricted Mean

If the smallest value for left-censored data is censored and less than or equal to the smallest uncensored value, then the estimated mean will be biased high, and if the largest value for right-censored data is censored and greater than or equal to the largest uncensored value, then the estimated mean will be biased low. One solution to this problem is to instead estimate what is called the ***restricted mean*** (Miller, 1981; Lee and Wang, 2003, p. 74; Meier, 2004; Barker, 2009).

To compute the restricted mean (`restricted=TRUE`), for left-censored data, the smallest censored observation(s) are treated as observed, and set to the smallest censoring level (`left.censored.min="Censoring Level"`) or some other value less than the smallest censoring level and greater than 0, and then applying the above formulas. To compute the restricted mean for right-censored data, the largest censored observation(s) are treated as observed and set to the censoring level (`right.censored.max="Censoring Level"`) or some value greater than the largest censoring level.

ProUCL 5.2.0 (USEPA, 2022, pp. 128–129) and Beal (2010) do not compute the restricted mean in cases where it could be applied, whereas USEPA (2009, pp. 15–7 to 15–12) and the `kmms` function in Version 2.0 of the R package **STAND** (Frome and Frome, 2015) do compute the restricted mean

and set the smallest censored observation(s) equal to the censoring level (i.e., what `enparCensored` does when `restricted=TRUE` and `left.censored.min="Censoring Level"`).

To be consistent with ProUCL 5.2.0, by default the function `enparCensored` does not compute the restricted mean (i.e., `restricted=FALSE`). Note that there are instances where ProUCL 5.2.0 does not agree with `enparCensored`, and it is not clear why. It should be noted that when the restricted mean is computed, the number of uncensored observations increases because the smallest (left-censored) or largest (right-censored) censored observation(s) is/are set to a specified value and treated as uncensored. The `kmms` function in Version 2.0 of the **STAND** package (Frome and Frome, 2015) is inconsistent in how it treats the number of uncensored observations when computing estimates associated with the restricted mean. Although `kmms` sets the smallest censored observations to the observed censoring level and treats them as not censored, when it computes the bias correction factor for the standard error of the mean, it assumes those observations are still censored (see the EXAMPLES section below).

In the unusual case when a restricted mean will be estimated and setting the smallest censored value(s) to an uncensored value (left-censored data), or setting the largest censored value(s) to an uncensored value (right-censored data), results in no censored values in the data, the Kaplan-Meier estimate of the mean reduces to the sample mean, so the function `enpar` is called and, if `warn=TRUE`, a warning is returned.

Confidence Intervals

This section explains how confidence intervals for the mean μ are computed.

Normal Approximation (`ci.method="normal.approx"`)

This method constructs approximate $(1-\alpha)100\%$ confidence intervals for μ based on the assumption that the estimator of μ is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$ confidence interval for μ is constructed as:

$$[\hat{\mu} - t_{1-\alpha/2,v-1}\hat{\sigma}_{\hat{\mu}},\ \hat{\mu} + t_{1-\alpha/2,v-1}\hat{\sigma}_{\hat{\mu}}] \quad (13)$$

where $\hat{\mu}$ denotes the estimate of μ, $\hat{\sigma}_{\hat{\mu}}$ denotes the estimated asymptotic standard deviation of the estimator of μ, v denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument `ci.sample.size` determines the value of v. The possible values are the total number of observations, N (`ci.sample.size="Total"`), or the number of uncensored observations, n (`ci.sample.size="Uncensored"`). To be consistent with ProUCL 5.2.0, in `enparCensored` the default value is the total number of observations. The `kmms` function in the **STAND** package, on the other hand, uses the number of uncensored observations.

When `pivot.statistic="z"`, the p'th quantile from the standard normal distribution is used in place of the p'th quantile from Student's t-distribution.

Bootstrap and Bias-Corrected Bootstrap Approximation (`ci.method="bootstrap"`)

The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1-\alpha)100\%$ confidence interval for the population mean μ, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size N from the observations in $\underline{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\underline{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).
2. Estimate μ based on the bootstrap sample created in Step 1, using the same method that was used to estimate μ using the original observations in $\underline{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of μ based on the bootstrap sample will usually differ from the original estimate based on $\underline{x}$. For the bootstrap-t method (see below), this step also involves estimating the standard error of the estimate of the mean and computing the statistic $T = (\hat{\mu}_B - \hat{\mu})/\hat{\sigma}_{\hat{\mu}_B}$ where $\hat{\mu}$ denotes the estimate of the mean based on the original sample, and $\hat{\mu}_B$ and $\hat{\sigma}_{\hat{\mu}_B}$ denote the estimate of the mean and estimate of the standard error of the estimate of the mean based on the bootstrap sample.
3. Repeat Steps 1 and 2 B times, where B is some large number. For the function `enparCensored`, the number of bootstraps B is determined by the argument `n.bootstraps` (see the section **ARGUMENTS** above). The default value of `n.bootstraps` is `1000`.
4. Use the B estimated values of μ to compute the empirical cumulative distribution function of the estimator of μ or to compute the empirical cumulative distribution function of the statistic T (see `ecdfPlot`), and then create a confidence interval for μ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}),\ \hat{G}^{-1}(1-\frac{\alpha}{2})] \qquad (14)$$

where $\hat{G}(t)$ denotes the empirical cdf of $\hat{\mu}_B$ evaluated at t and thus $\hat{G}^{-1}(p)$ denotes the p'th empirical quantile of the distribution of $\hat{\mu}_B$, that is, the p'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha),\ \infty] \qquad (15)$$

and a one-sided upper confidence interval is computed as:

$$[-\infty,\ \hat{G}^{-1}(1-\alpha)] \qquad (16)$$

The function `enparCensored` calls the R function `quantile` to compute the empirical quantiles used in Equations (14)-(16).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of μ can be off by $k/\sqrt{N}$, where k is some constant. Efron and Tibshirani (1993, pp.184–188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of μ may be off by k/N instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1),\ \hat{G}^{-1}(\alpha_2)] \qquad (17)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}(z_0 + z_{\alpha/2})}] \quad (18)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}(z_0 + z_{1-\alpha/2})}] \quad (19)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\mu})] \quad (20)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^3}{6[\sum_{i=1}^{N}(\hat{\mu}_{(\cdot)} - \hat{\mu}_{(i)})^2]^{3/2}} \quad (21)$$

where the quantity $\hat{\mu}_{(i)}$ denotes the estimate of μ using all the values in $\underline{x}$ except the i'th one, and

$$\hat{\mu}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\hat{\mu_{(i)}} \quad (22)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1),\ \infty] \quad (23)$$

and a one-sided upper confidence interval is given by:

$$[-\infty,\ \hat{G}^{-1}(\alpha_2)] \quad (24)$$

where α_1 and α_2 are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with α in Equations (18) and (19).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of μ with respect to the true value of μ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of μ does not depend on the value of μ, hence the acceleration constant is not really necessary.

For the bootstrap-t method, the two-sided confidence interval (Efron and Tibshirani, 1993, p.160) is computed as:

$$[\hat{\mu} - t_{1-\alpha/2}\hat{\sigma}_{\hat{\mu}},\ \hat{\mu} - t_{\alpha/2}\hat{\sigma}_{\hat{\mu}}] \quad (25)$$

where $\hat{\mu}$ and $\hat{\sigma}_{\hat{\mu}}$ denote the estimate of the mean and standard error of the estimate of the mean based on the original sample, and t_p denotes the p'th empirical quantile of the bootstrap distribution of the statistic T. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{\mu} - t_{1-\alpha}\hat{\sigma}_{\hat{\mu}},\ \infty] \quad (26)$$

and a one-sided upper confidence interval is computed as:

$$[-\infty,\ \hat{\mu} - t_{\alpha}\hat{\sigma}_{\hat{\mu}}] \quad (27)$$

When `ci.method="bootstrap"`, the function `enparCensored` computes the percentile method, bias-corrected and accelerated method, and bootstrap-t bootstrap confidence intervals. The percentile method is transformation respecting, but not second-order accurate. The bootstrap-t method is second-order accurate, but not transformation respecting. The bias-corrected and accelerated method is both transformation respecting and second-order accurate (Efron and Tibshirani, 1993, p.188).

Value

a list of class "estimateCensored" containing the estimated parameters and other information. See estimateCensored.object for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation, rather than rely on a single point-estimate of the mean. Since confidence intervals and regions depend on the properties of the estimators for both the mean and standard deviation, the results of studies that simply evaluated the performance of the mean and standard deviation separately cannot be readily extrapolated to predict the performance of various methods of constructing confidence intervals and regions. Furthermore, for several of the methods that have been proposed to estimate the mean based on type I left-censored data, standard errors of the estimates are not available, hence it is not possible to construct confidence intervals (El-Shaarawi and Dolan, 1989).

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring. See, for example, Singh et al. (2006).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Barker, C. (2009). The Mean, Median, and Confidence Intervals of the Kaplan-Meier Survival Estimate – Computations and Applications. *The American Statistician* **63**(1), 78–80.

Beal, D. (2010). *A Macro for Calculating Summary Statistics on Left Censored Environmental Data Using the Kaplan-Meier Method*. Paper SDA-09, presented at Southeast SAS Users Group 2010, September 26-28, Savannah, GA. https://analytics.ncsu.edu/sesug/2010/SDA09.Beal.pdf.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

El-Shaarawi, A.H., and D.M. Dolan. (1989). Maximum Likelihood Estimation of Water Quality Concentrations from Censored Data. *Canadian Journal of Fisheries and Aquatic Sciences* **46**, 1033–1039.

Frome E.L., and D.P. Frome (2015). *STAND: Statistical Analysis of Non-Detects*. R package version 2.0, https://CRAN.R-project.org/package=STAND.

Gillespie, B.W., Q. Chen, H. Reichert, A. Franzblau, E. Hedgeman, J. Lepkowski, P. Adriaens, A. Demond, W. Luksemburg, and D.H. Garabrant. (2010). Estimating Population Distributions When Some Data Are Below a Limit of Detection by Using a Reverse Kaplan-Meier Estimator. *Epidemiology* **21**(4), S64–S70.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Irwin, J.O. (1949). The Standard Error of an Estimate of Expectation of Life, with Special Reference to Expectation of Tumourless Life in Experiments with Mice. *Journal of Hygiene* **47**, 188–189.

Kaplan, E.L., and P. Meier. (1958). Nonparametric Estimation From Incomplete Observations. *Journal of the American Statistical Association* **53**, 457-481.

Klein, J.P., and M.L. Moeschberger. (2003). *Survival Analysis: Techniques for Censored and Truncated Data, Second Edition*. Springer, New York, 537pp.

Lee, E.T., and J.W. Wang. (2003). *Statistical Methods for Survival Data Analysis, Third Edition*. John Wiley & Sons, Hoboken, New Jersey, 513pp.

Meier, P., T. Karrison, R. Chappell, and H. Xie. (2004). The Price of Kaplan-Meier. *Journal of the American Statistical Association* **99**(467), 890–896.

Miller, R.G. (1981). *Survival Analysis*. John Wiley and Sons, New York.

Nelson, W. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, 634pp.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2022). *ProUCL Version 5.2.0 Technical Guide: Statistical Software for Environmental Applications for Data Sets with and without Nondetect Observations*. Prepared by: Neptune and Company, Inc., 1435 Garrison Street, Suite 201, Lakewood, CO 80215. pp. 128–129, 143. https://www.epa.gov/land-research/proucl-software.

See Also

ppointsCensored, ecdfPlotCensored, qqPlotCensored,estimateCensored.object, enpar.

Examples

```
# Using the lead concentration data from soil samples shown in
# Beal (2010), compute the Kaplan-Meier estimators of the mean,
# standard deviation, and standard error of the mean, as well as
# a 95% upper confidence limit for the mean.  Compare these
# results to those given in Beal (2010), and also to the results
# produced by ProUCL 5.2.0.

# First look at the data:
#-----------------------

head(Beal.2010.Pb.df)
#  Pb.char  Pb Censored
#1      <1 1.0     TRUE
#2      <1 1.0     TRUE
#3       2 2.0    FALSE
#4     2.5 2.5    FALSE
#5     2.8 2.8    FALSE
#6      <3 3.0     TRUE

tail(Beal.2010.Pb.df)
#   Pb.char   Pb Censored
#24     <10   10     TRUE
#25      10   10    FALSE
#26      15   15    FALSE
#27      49   49    FALSE
#28     200  200    FALSE
#29    9060 9060    FALSE

# enparCensored Results:
#-----------------------
Beal.unrestricted <- with(Beal.2010.Pb.df,
  enparCensored(x = Pb, censored = Censored, ci = TRUE,
    ci.type = "upper"))

Beal.unrestricted

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#-------------------------------------------
#
#Assumed Distribution:            None
#
#Censoring Side:                  left
#
#Censoring Level(s):               1  3  4  6  9 10
#
#Estimated Parameter(s):          mean    =  325.3396
#                                 sd      = 1651.0950
#                                 se.mean =  315.0023
#
#Estimation Method:               Kaplan-Meier
```

```
#                                (Bias-corrected se.mean)
#
#Data:                           Pb
#
#Censoring Variable:             Censored
#
#Sample Size:                    29
#
#Percent Censored:               34.48276%
#
#Confidence Interval for:        mean
#
#Assumed Sample Size:            29
#
#Confidence Interval Method:     Normal Approximation
#                                (t Distribution)
#
#Confidence Interval Type:       upper
#
#Confidence Level:               95%
#
#Confidence Interval:            LCL =   0.0000
#                                UCL = 861.1996

c(Beal.unrestricted$parameters, Beal.unrestricted$interval$limits)
#     mean        sd   se.mean       LCL       UCL
# 325.3396 1651.0950  315.0023    0.0000  861.1996

# Beal (2010) published results:
#-------------------------------
#   Mean   Std. Dev.  SE of Mean
# 325.34     1651.09      315.00

# ProUCL 5.2.0 results:
#----------------------
#   Mean   Std. Dev.  SE of Mean  95% UCL
# 325.2      1651         315       861.1

#----------

# Now compute the restricted mean and associated quantities,
# and compare these results with those produced by the
# kmms() function in the STAND package.
#-------------------------------------------------------------

Beal.restricted <- with(Beal.2010.Pb.df,
  enparCensored(x = Pb, censored = Censored, restricted = TRUE,
    ci = TRUE, ci.type = "upper"))

Beal.restricted

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
```

```
#------------------------------------------
#
#Assumed Distribution:            None
#
#Censoring Side:                  left
#
#Censoring Level(s):               1  3  4  6  9 10
#
#Estimated Parameter(s):          mean    =  325.2011
#                                 sd      = 1651.1221
#                                 se.mean =  314.1774
#
#Estimation Method:               Kaplan-Meier (Restricted Mean)
#                                 Smallest censored value(s)
#                                   set to Censoring Level
#                                 (Bias-corrected se.mean)
#
#Data:                            Pb
#
#Censoring Variable:              Censored
#
#Sample Size:                     29
#
#Percent Censored:                34.48276%
#
#Confidence Interval for:         mean
#
#Assumed Sample Size:             29
#
#Confidence Interval Method:      Normal Approximation
#                                 (t Distribution)
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =   0.000
#                                 UCL = 859.658

c(Beal.restricted$parameters, Beal.restricted$interval$limits)
#     mean        sd   se.mean       LCL       UCL
# 325.2011 1651.1221  314.1774    0.0000  859.6580

# kmms() results:
#----------------
#  KM.mean    KM.LCL    KM.UCL     KM.se     gamma
# 325.2011 -221.0419  871.4440  315.0075    0.9500

# NOTE: as pointed out above, the kmms() function treats the
#       smallest censored observations (<1 and <1) as NOT
#       censored when computing the mean and uncorrected
#       standard error of the mean, but assumes these
#       observations ARE censored when computing the corrected
```

```
#       standard error of the mean.
#----------------------------------------------------------------

Beal.restricted$parameters["se.mean"] * sqrt((20/21)) * sqrt((19/18))
#  se.mean
# 315.0075

#==========

# Repeat the above example, estimating the unrestricted mean and
# computing an upper confidence limit based on the bootstrap
# instead of on the normal approximation with a t pivot statistic.
# Compare results to those from ProUCL 5.2.0.
# Note:  Setting the seed argument lets you reproduce this example.
#----------------------------------------------------------------

Beal.unrestricted.boot <- with(Beal.2010.Pb.df,
  enparCensored(x = Pb, censored = Censored, ci = TRUE,
    ci.type = "upper", ci.method = "bootstrap", seed = 923))

Beal.unrestricted.boot

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Censoring Side:                  left
#
#Censoring Level(s):               1  3  4  6  9 10
#
#Estimated Parameter(s):          mean    =  325.3396
#                                 sd      = 1651.0950
#                                 se.mean =  315.0023
#
#Estimation Method:               Kaplan-Meier
#                                 (Bias-corrected se.mean)
#
#Data:                            Pb
#
#Censoring Variable:              Censored
#
#Sample Size:                     29
#
#Percent Censored:                34.48276%
#
#Confidence Interval for:         mean
#
#Assumed Sample Size:             29
#
#Confidence Interval Method:      Bootstrap
#
```

```
#Number of Bootstraps:            1000
#
#Number of Bootstrap Samples
#With No Censored Values:         0
#
#Number of Times Bootstrap
#Repeated Because Too Few
#Uncensored Observations:         0
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             Pct.LCL =     0.0000
#                                 Pct.UCL =   948.7342
#                                 BCa.LCL =     0.0000
#                                 BCa.UCL =   942.6596
#                                 t.LCL   =     0.0000
#                                 t.UCL   = 62121.8909

c(Beal.unrestricted.boot$interval$limits)
#   Pct.LCL    Pct.UCL    BCa.LCL    BCa.UCL      t.LCL      t.UCL
#    0.0000   948.7342     0.0000   942.6596     0.0000 62121.8909

# ProUCL 5.2.0 results:
#----------------------
#   Pct.LCL    Pct.UCL    BCa.LCL    BCa.UCL      t.LCL      t.UCL
#    0.0000   944.3        0.0000   947.8        0.0000 62169

#==========

# Clean up
#---------
rm(Beal.unrestricted, Beal.restricted, Beal.unrestricted.boot)
```

`Environmental` *Atmospheric Environmental Conditions in New York City*

Description

Daily measurements of ozone concentration, wind speed, temperature, and solar radiation in New York City for 153 consecutive days between May 1 and September 30, 1973.

Usage

```
Environmental.df
Air.df
```

Format

The data frame `Environmental.df` has 153 observations on the following 4 variables.

`ozone` Average ozone concentration (of hourly measurements) of in parts per billion.

`radiation` Solar radiation (from 08:00 to 12:00) in langleys.

`temperature` Maximum daily temperature in degrees Fahrenheit.

`wind` Average wind speed (at 07:00 and 10:00) in miles per hour.

Row names are the dates the data were collected.

The data frame `Air.df` is the same as `Environmental.df` except that the column `ozone` is the cube root of average ozone concentration.

Details

Data on ozone (ppb), solar radiation (langleys), temperature (degrees Fahrenheit), and wind speed (mph) for 153 consecutive days between May 1 and September 30, 1973. These data are a superset of the data contained in the data frame `environmental` in the package **lattice**.

Source

Chambers et al. (1983), pp. 347-349.

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, 395pp.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

Cleveland, W.S. (1994). *The Elements of Graphing Data*. Revised Edition. Hobart Press, Summit, New Jersey, 297pp.

Examples

```
# Scatterplot matrix
pairs(Environmental.df)

pairs(Air.df)


# Time series plot for ozone
attach(Environmental.df)
dates <- as.Date(row.names(Environmental.df), format = "%m/%d/%Y")
plot(dates, ozone, type = "l",
    xlab = "Time (Year = 1973)", ylab = "Ozone (ppb)",
    main = "Time Series Plot of Daily Ozone Measures")
detach("Environmental.df")
rm(dates)
```

EPA.02d.Ex.2.ug.per.L.vec

Concentrations in Exhibit 2 of 2002d USEPA Guidance Document

Description

Concentrations (mug/L) from an exposure unit.

Usage

```
data(EPA.02d.Ex.2.ug.per.L.vec)
```

Format

a numeric vector of concentrations (mug/L)

Source

USEPA. (2002d). *Calculating Upper Confidence Limits for Exposure Point Concentrations at Hazardous Waste Sites*. OSWER 9285.6-10, December 2002. Office of Emergency and Remedial Response, U.S. Environmental Protection Agency, Washington, D.C., p. 9.

EPA.02d.Ex.4.mg.per.kg.vec

Concentrations in Exhibit 4 of 2002d USEPA Guidance Document

Description

Concentrations (mg/kg) from an exposure unit.

Usage

```
data(EPA.02d.Ex.4.mg.per.kg.vec)
```

Format

a numeric vector of concentrations (mg/kg)

Source

USEPA. (2002d). *Calculating Upper Confidence Limits for Exposure Point Concentrations at Hazardous Waste Sites*. OSWER 9285.6-10, December 2002. Office of Emergency and Remedial Response, U.S. Environmental Protection Agency, Washington, D.C., p. 11.

`EPA.02d.Ex.6.mg.per.kg.vec`

Concentrations in Exhibit 6 of 2002d USEPA Guidance Document

Description

Concentrations (mg/kg) from an exposure unit.

Usage

```
data(EPA.02d.Ex.6.mg.per.kg.vec)
```

Format

a numeric vector of concentrations (mg/kg)

Source

USEPA. (2002d). *Calculating Upper Confidence Limits for Exposure Point Concentrations at Hazardous Waste Sites*. OSWER 9285.6-10, December 2002. Office of Emergency and Remedial Response, U.S. Environmental Protection Agency, Washington, D.C., p. 13.

`EPA.02d.Ex.9.mg.per.L.vec`

Concentrations in Exhibit 9 of 2002d USEPA Guidance Document

Description

Concentrations (mg/L) from an exposure unit.

Usage

```
data(EPA.02d.Ex.9.mg.per.L.vec)
```

Format

a numeric vector of concentrations (mg/L)

Source

USEPA. (2002d). *Calculating Upper Confidence Limits for Exposure Point Concentrations at Hazardous Waste Sites*. OSWER 9285.6-10, December 2002. Office of Emergency and Remedial Response, U.S. Environmental Protection Agency, Washington, D.C., p. 16.

EPA.09.Ex.10.1.nickel.df

Nickel Concentrations from Example 10-1 of 2009 USEPA Guidance Document

Description

Nickel concentrations (ppb) from four wells (five observations per year for each well). The Guidance Document has the label "Year" instead of "Well"; corrected in Errata.

Usage

```
EPA.09.Ex.10.1.nickel.df
```

Format

A data frame with 20 observations on the following 3 variables.

`Month` a numeric vector indicating the month the sample was taken

`Well` a factor indicating the well number

`Nickel.ppb` a numeric vector of nickel concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery, Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C., p.10-12.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.11.1.arsenic.df

Arsenic Concentrations from Example 11-1 of 2009 USEPA Guidance Document

Description

Arsenic concentrations (ppb) at six wells (four observations per well).

Usage

```
EPA.09.Ex.11.1.arsenic.df
```

Format

A data frame with 24 observations on the following 3 variables.

`Arsenic.ppb` a numeric vector of arsenic concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.11-3.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.12.1.ccl4.df

Carbon Tetrachloride Concentrations from Example 12-1 of 2009 USEPA Guidance Document

---

Description

Carbon tetrachloride (CCL4) concentrations (ppb) at five background wells (four measures at each well).

Usage

```
EPA.09.Ex.12.1.ccl4.df
```

Format

A data frame with 20 observations on the following 2 variables.

`Well` a factor indicating the well number

`CCL4.ppb` a numeric vector of CCL4 concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.12-3.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.12.4.naphthalene.df

Naphthalene Concentrations from Example 12-4 of 2009 USEPA Guidance Document

Description

Naphthalene concentrations (ppb) at five background wells (five quarterly measures at each well).

Usage

```
EPA.09.Ex.12.4.naphthalene.df
```

Format

A data frame with 25 observations on the following 3 variables.

`Quarter` a numeric vector indicating the quarter the sample was taken

`Well` a factor indicating the well number

`Naphthalene.ppb` a numeric vector of naphthalene concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.12-12.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.13.1.iron.df

Iron Concentrations from Example 13-1 of 2009 USEPA Guidance Document

Description

Dissolved iron (Fe) concentrations (ppm) at six upgradient wells (four quarterly measures at each well).

Usage

```
EPA.09.Ex.13.1.iron.df
```

Format

A data frame with 24 observations on the following 4 variables.

`Month` a numeric vector indicating the month the sample was taken

`Year` a numeric vector indicating the year the sample was taken

`Well` a factor indicating the well number

`Iron.ppm` a numeric vector if iron concentrations (ppm)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.13-3.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.14.1.manganese.df` *Manganese Concentrations from Example 14-1 of 2009 USEPA Guidance Document*

Description

Manganese concentrations (ppm) at four background wells (eight quarterly measures at each well).

Usage

```
EPA.09.Ex.14.1.manganese.df
```

Format

A data frame with 32 observations on the following 3 variables.

`Quarter` a numeric vector indicating the quarter the sample was taken

`Well` a factor indicating the well number

`Manganese.ppm` a numeric vector of manganese concentrations (ppm)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.14-5.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.14.3.alkalinity.df

Alkalinity Measures from Example 14-3 of 2009 USEPA Guidance Document

Description

Alkalinity measures (mg/L) collected from leachate at a solid waste landfill during a four and a half year period.

Usage

```
EPA.09.Ex.14.3.alkalinity.df
```

Format

A data frame with 54 observations on the following 2 variables.

`Date` a Date object indicating the date of collection

`Alkalinity.mg.per.L` a numeric vector of alkalinity measures (mg/L)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.14-14.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.14.4.arsenic.df

Arsenic Concentrations from Example 14-4 of 2009 USEPA Guidance Document

Description

Sixteen quarterly measures of arsenic concentrations (ppb).

Usage

```
EPA.09.Ex.14.4.arsenic.df
```

Format

A data frame with 16 observations on the following 4 variables.

`Sample.Date` a factor indicating the month and year of collection

`Month` a factor indicating the month of collection

`Year` a factor indicating the year of collection

`Arsenic.ppb` a numeric vector of arsenic concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.14-18.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.14.8.df` *Analyte Concentrations from Example 14-8 of 2009 USEPA Guidance Document*

Description

Monthly unadjusted and adjusted analyte concentrations over a 3-year period. Adjusted concentrations are computed by subtracting the monthly mean and adding the overall mean.

Usage

```
EPA.09.Ex.14.8.df
```

Format

A data frame with 36 observations on the following 4 variables.

`Month` a factor indicating the month of collection

`Year` a numeric vector indicating the year of collection

`Unadj.Conc` a numeric vector of unadjusted concentrations

`Adj.Conc` a numeric vector adjusted concentrations

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.14-32.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.15.1.manganese.df

Manganese Concentrations from Example 15-1 of 2009 USEPA Guidance Document

---

Description

Manganese concentrations (ppb) at five background wells (five measures at each well).

Usage

```
EPA.09.Ex.15.1.manganese.df
```

Format

A data frame with 25 observations on the following 5 variables.

`Sample` a numeric vector indicating the sample number (1-5)

`Well` a factor indicating the well number

`Manganese.Orig.ppb` a character vector of the original manganese concentrations (ppb)

`Manganese.ppb` a numeric vector of manganese concentrations with non-detects coded to their detecion limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.15-10.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.16.1.sulfate.df`

Sulfate Concentrations from Example 16-1 of 2009 USEPA Guidance Document

---

Description

Sulfate concentrations (ppm) at one background well and one downgradient well (eight quarterly measures at each well).

Usage

```
EPA.09.Ex.16.1.sulfate.df
```

Format

A data frame with 16 observations on the following 4 variables.

`Month` a factor indicating the month of collection

`Year` a factor indicating the year of collection

`Well.type` a factor indicating the well type (background vs. downgradient)

`Sulfate.ppm` a numeric vector of sulfate concentrations (ppm)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.16-6.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.16.2.benzene.df`

Benzene Concentrations from Example 16-2 of 2009 USEPA Guidance Document

---

Description

Benzene concentrations (ppb) at one background and one downgradient well (eight monthly measures at each well).

Usage

```
EPA.09.Ex.16.2.benzene.df
```

Format

A data frame with 16 observations on the following 3 variables.

`Month` a factor indicating the month of collection

`Well.type` a factor indicating the well type (background vs. downgradient)

`Benzene.ppb` a numeric vector of benzene concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.16-9.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.16.4.copper.df

Copper Concentrations from Example 16-4 of 2009 USEPA Guidance Document

---

Description

Copper concentrations (ppb) at two background wells and one compliance well (six measures at each well).

Usage

```
EPA.09.Ex.16.4.copper.df
```

Format

A data frame with 18 observations on the following 4 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Copper.ppb` a numeric vector of copper concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.16-19.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.16.5.PCE.df *Tetrachloroethylene Concentrations from Example 16-5 of 2009 USEPA Guidance Document*

Description

Tetrachloroethylene (PCE) concentrations (ppb) at one background well and one compliance well.

Usage

```
EPA.09.Ex.16.5.PCE.df
```

Format

A data frame with 14 observations on the following 4 variables.

`Well.type` a factor with levels `Background` `Compliance`

`PCE.Orig.ppb` a character vector of original PCE concentrations (ppb)

`PCE.ppb` a numeric vector of PCE concentrations (ppb) with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.16-22.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.17.1.loglead.df

Log-transformed Lead Concentrations from Example 17-1 of 2009 USEPA Guidance Document

Description

Log-transformed lead concentrations (ppb) at two background and four compliance wells (four quarterly measures at each well).

Usage

```
EPA.09.Ex.17.1.loglead.df
```

Format

A data frame with 24 observations on the following 4 variables.

`Month` a factor indicating the month of collection; `1` = Jan, `2` = Apr, `3` = Jul, `4` = Oct

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`LogLead` a numeric vector of log-transformed lead concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-7.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.17.2.toluene.df

Toluene Concentrations from Example 17-2 of 2009 USEPA Guidance Document

Description

Toluene concentrations (ppb) at two background and three compliance wells (five monthly measures at each well).

Usage

```
EPA.09.Ex.17.2.toluene.df
```

Format

A data frame with 25 observations on the following 6 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Toluene.ppb.orig` a character vector of original toluene concentrations (ppb)

`Toluene.ppb` a numeric vector of toluene concentrations (ppb) with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-13.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.17.3.chrysene.df

Chrysene Concentrations from Example 17-3 of 2009 USEPA Guidance Document

---

Description

Chrysene concentrations (ppb) at two background and three compliance wells (four monthly measures at each well).

Usage

```
EPA.09.Ex.17.3.chrysene.df
```

Format

A data frame with 20 observations on the following 4 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Chrysene.ppb` a numeric vector of chrysene concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-17.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.17.3.log.chrysene.df`

Log-transformed Chrysene Concentrations from Example 17-3 of 2009 USEPA Guidance Document

---

Description

Log-transformed chrysene concentrations (ppb) at two background and three compliance wells (four monthly measures at each well).

Usage

```
EPA.09.Ex.17.3.log.chrysene.df
```

Format

A data frame with 20 observations on the following 4 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Log.Chrysene.ppb` a numeric vector of log-transformed chrysene concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-18.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.17.4.copper.df

Copper Concentrations from Example 17-4 of 2009 USEPA Guidance Document

---

Description

Copper concentrations (ppb) at three background and two compliance wells (eight monthly measures at the background wells, four monthly measures at the compliance wells).

Usage

```
EPA.09.Ex.17.4.copper.df
```

Format

A data frame with 40 observations on the following 6 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Copper.ppb.orig` a character vector of original copper concentrations (ppb)

`Copper.ppb` a numeric vector of copper concentrations with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-21.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.17.5.chloride.df

Chloride Concentrations from Example 17-5 of 2009 USEPA Guidance Document

---

Description

Chloride concentrations (ppm) collected over a five-year period at a solid waste landfill.

Usage

```
EPA.09.Ex.17.5.chloride.df
```

Format

A data frame with 19 observations on the following 4 variables.

`Date` a Date object indicating the date of collection

`Chloride.ppm` a numeric vector of chloride concentrations (ppm)

`Elapsed.Days` a numeric vector indicating the number of days since January 1, 2002

`Residuals` a numeric vector of residuals from a linear regression trend fit

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-26.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.17.6.sulfate.df

Sulfate Concentrations from Example 17-6 of 2009 USEPA Guidance Document

---

Description

Sulfate concentrations (ppm) collected over several years. The date of collection is simply indicated by month and year of collection. The column `Date` is a Date object where the day of the month has been arbitrarily set to 1.

Usage

```
EPA.09.Ex.17.6.sulfate.df
```

Format

A data frame with 23 observations on the following 6 variables.

`Sample.No` a numeric vector indicating the sample number

`Year` a numeric vector indicating the year of collection

`Month` a numeric vector indicating the month of collection

`Sampling.Date` a numeric vector indicating the year and month of collection

`Date` a Date object indicating the date of collection, where the day of the month is arbitrarily set to 1

`Sulfate.ppm` a numeric vector of sulfate concentrations (ppm)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-33.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.17.7.sodium.df`

Sodium Concentrations from Example 17-7 of 2009 USEPA Guidance Document

---

Description

Sodium concentrations (ppm) collected over several years. The sample dates are recorded as the year of collection (2-digit format) plus a fractional part indicating when during the year the sample was collected.

Usage

```
EPA.09.Ex.17.7.sodium.df
```

Format

A data frame with 10 observations on the following 2 variables.

`Year` a numeric vector indicating the year of collection (a fractional number)

`Sodium.ppm` a numeric vector of sodium concentrations (ppm)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-36.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.18.1.arsenic.df

Arsenic Concentrations from Example 18-1 of 2009 USEPA Guidance Document

Description

Arsenic concentrations (ppb) in a single well at a solid waste landfill. Four observations per year over four years. Years 1-3 are the background period and Year 4 is the compliance period.

Usage

```
EPA.09.Ex.18.1.arsenic.df
```

Format

A data frame with 16 observations on the following 3 variables.

`Year` a factor indicating the year of collection

`Sampling.Period` a factor indicating the sampling period (background vs. compliance)

`Arsenic.ppb` a numeric vector of arsenic concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.18-10.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.18.2.chrysene.df

Chrysene Concentrations from Example 18-2 of 2009 USEPA Guidance Document

Description

Chrysene concentrations (ppb) at two background wells and one compliance well (four monthly measures at each well).

Usage

```
EPA.09.Ex.18.2.chrysene.df
```

Format

A data frame with 12 observations on the following 4 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Chrysene.ppb` a numeric vector of chrysene concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.18-15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.18.3.TCE.df` *Trichloroethylene Concentrations from Example 18-3 of 2009 USEPA Guidance Document*

Description

Trichloroethylene (TCE) concentrations (ppb) at three background wells and one compliance well. Six monthly measures at each background well, three monthly measures at the compliance well.

Usage

```
EPA.09.Ex.18.3.TCE.df
```

Format

A data frame with 24 observations on the following 6 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`TCE.ppb.orig` a character vector of original TCE concentrations (ppb)

`TCE.ppb` a numeric vector of TCE concentrations (ppb) with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.18-19.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.18.4.xylene.df

Xylene Concentrations from Example 18-4 of 2009 USEPA Guidance Document

---

Description

Xylene concentrations (ppb) at three background wells and one compliance well. Eight monthly measures at each complaince well; three monthly measures at the compliance well.

Usage

```
EPA.09.Ex.18.4.xylene.df
```

Format

A data frame with 32 observations on the following 6 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Xylene.ppb.orig` a character vector of original xylene concentrations (ppb)

`Xylene.ppb` a numeric vector of xylene concentrations (ppb) with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.18-22.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.19.1.sulfate.df`
Sulfate Concentrations from Example 19-1 of 2009 USEPA Guidance Document

---

Description

Sulfate concentrations (mg/L) at four background wells.

Usage

```
EPA.09.Ex.19.1.sulfate.df
```

Format

A data frame with 25 observations on the following 7 variables.

`Well` a factor indicating the well number

`Month` a numeric vector indicating the month of collection

`Day` a numeric vector indicating the day of the month of collection

`Year` a numeric vector indicating the year of collection

`Date` a Date object indicating the date of collection

`Sulfate.mg.per.l` a numeric vector of sulfate concentrations (mg/L)

`log.Sulfate.mg.per.l` a numeric vector of log-transformed sulfate concentrations (mg/L)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.19-17.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.19.2.chloride.df

Chloride Concentrations from Example 19-2 of 2009 USEPA Guidance Document

Description

Chloride concentrations (mg/L) at 10 compliance wells at a solid waste landfill. One year of quarterly measures at each well.

Usage

```
EPA.09.Ex.19.2.chloride.df
```

Format

A data frame with 40 observations on the following 2 variables.

`Well` a factor indicating the well number

`Chloride.mg.per.l` a numeric vector of chloride concentrations (mg/L)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.19-19.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.19.5.mercury.df

Mercury Concentrations from Example 19-5 of 2009 USEPA Guidance Document

Description

Mercury concentrations (ppb) at four background and two compliance wells.

Usage

```
EPA.09.Ex.19.5.mercury.df
```

Format

A data frame with 36 observations on the following 6 variables.

`Event` a factor indicating the time of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

`Mercury.ppb.orig` a character vector of original mercury concentrations (ppb)

`Mercury.ppb` a numeric vector of mercury concentrations (ppb) with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.19-33.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.20.1.nickel.df

Nickel Concentrations from Example 20-1 of 2009 USEPA Guidance Document

---

Description

Nickel concentrations (ppb) at a single well. Eight monthly measures during the background period and eight monthly measures during the compliance period.

Usage

```
EPA.09.Ex.20.1.nickel.df
```

Format

A data frame with 16 observations on the following 4 variables.

`Month` a factor indicating the month of collection

`Year` a factor indicating the year of collection

`Period` a factor indicating the period (baseline vs. compliance)

`Nickel.ppb` a numeric vector of nickel concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.20-4.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.21.1.aldicarb.df

Aldicarb Concentrations from Example 21-1 of 2009 USEPA Guidance Document

---

Description

Aldicarb concentrations (ppb) at three compliance wells (four monthly measures at each well).

Usage

```
EPA.09.Ex.21.1.aldicarb.df
```

Format

A data frame with 12 observations on the following 3 variables.

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Aldicarb.ppb` a numeric vector of aldicarb concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-4.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.21.2.benzene.df`

Benzene Concentrations from Example 21-2 of 2009 USEPA Guidance Document

---

Description

Benzene concentrations (ppb) collected at a landfill that previously handled smelter waste and is now undergoing remediation efforts.

Usage

```
EPA.09.Ex.21.2.benzene.df
```

Format

A data frame with 8 observations on the following 4 variables.

`Month` a numeric vector indicating the month of collection

`Benzene.ppb.orig` a character vector of original benzene concentrations (ppb)

`Benzene.ppb` a numeric vector of benzene concentrations (ppb) with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-7.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

`EPA.09.Ex.21.5.beryllium.df`

Beryllium Concentrations from Example 21-5 of 2009 USEPA Guidance Document

---

Description

Beryllium concentrations (ppb) at one well (four years of quarterly measures).

Usage

```
data(EPA.09.Ex.21.5.beryllium.df)
```

Format

A data frame with 16 observations on the following 3 variables.

`Year` a factor indicating the year of collection

`Quarter` a factor indicating the quarter of collection

`Beryllium.ppb` a numeric vector of beryllium concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-18.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.21.6.nitrate.df

Nitrate Concentrations from Example 21-6 of 2009 USEPA Guidance Document

---

Description

Nitrate concentrations (mg/L) at a well used for drinking water.

Usage

```
EPA.09.Ex.21.6.nitrate.df
```

Format

A data frame with 12 observations on the following 5 variables.

`Sampling.Date` a character vector indicating the sampling date

`Date` a Date object indicating the sampling date

`Nitrate.mg.per.l.orig` a character vector of original nitrate concentrations (mg/L)

`Nitrate.mg.per.l` a numeric vector of nitrate concentrations (mg/L) with nondetects set to their detection limit

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-22.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.21.7.TCE.df *Trichloroethylene Concentrations from Example 21-7 of 2009 USEPA Guidance Document*

Description

Trichloroethylene (TCE) concentrations (ppb) at a site undergoing remediation.

Usage

```
EPA.09.Ex.21.7.TCE.df
```

Format

A data frame with 10 observations on the following 2 variables.

`Month` a numeric vector indicating the month of collection

`TCE.ppb` a numeric vector of TCE concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-26.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.22.1.VC.df *Vinyl Chloride Concentrations from Example 22-1 of 2009 USEPA Guidance Document*

Description

Vinyl Chloride (VC) concentrations (ppb) during detection monitoring for two compliance wells. Four years of quarterly measures at each well. Compliance monitoring began with Year 2 of the sampling record.

Usage

```
EPA.09.Ex.22.1.VC.df
```

Format

A data frame with 32 observations on the following 5 variables.

`Year` a factor indicating the year of collection

`Quarter` a factor indicating the quarter of collection

`Period` a factor indicating the period (background vs. compliance)

`Well` a factor indicating the well number

`VC.ppb` a numeric vector of VC concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.22-6.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.22.2.Specific.Conductance.df *Specific Conductance from Example 22-2 of 2009 USEPA Guidance Document*

Description

Specific conductance (μmho) collected over several years at two wells at a hazardous waste facility.

Usage

```
EPA.09.Ex.22.2.Specific.Conductance.df
```

Format

A data frame with 43 observations on the following 3 variables.

`Well` a factor indicating the well number

`Date` a Date object indicating the date of collection

`Specific.Conductance.umho` a numeric vector of specific conductance (μmho)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.22-11.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.6.3.sulfate.df

Sulfate Concentrations from Example 6-3 of 2009 USEPA Guidance Document

---

Description

Sulfate concentrations (ppm) at two background wells (five quarterly measures at each well).

Usage

```
EPA.09.Ex.6.3.sulfate.df
```

Format

A data frame with 10 observations on the following 4 variables.

`Month` a numeric vector indicating the month the observations was taken

`Year` a numeric vector indicating the year the observation was taken

`Well` a factor indicating the well number

`Sulfate.ppm` a numeric vector of sulfate concentrations (ppm)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.6-20.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Ex.7.1.arsenic.df

Arsenic concentrations from Example 7.1 of 2009 USEPA Guidance Document

---

Description

Arsenic concentrations (μg/L) at a single well, consisting of: 8 historical observations, 4 future observations for Case 1, and 4 future observations for Case 2.

Usage

```
EPA.09.Ex.7.1.arsenic.df
```

Format

A data frame with 16 observations on the following 2 variables.

`Data.Source` a factor with levels `Historical`, `Case.1`, `Case.2`

`Arsenic.ug.per.l` a numeric vector of arsenic concentrations (μg/L)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.7-26.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Table.9.1.TCE.df

Trichloroethene concentrations in Table 9.1 of 2009 USEPA Guidance Document

Description

Time series of trichloroethene (TCE) concentrations (mg/L) taken at 2 separate wells. Some observations are annotated with a data qualifier of `U` (nondetect) or `J` (estimated detected concentration).

Usage

```
EPA.09.Table.9.1.TCE.df
```

Format

A data frame with 30 observations on the following 5 variables.

`Date.Collected` a factor indicating the date of collection

`Date` a Date object indicating the date of collection

`Well` a factor indicating the well number

`TCE.mg.per.L` a numeric vector indicating the TCE concnetrations (mg/L)

`Data.Qualifier` a factor indicating the data qualifier

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.9-3.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Table.9.3.df

Arsenic, Mercury and Strontium Concentrations in Table 9-3 of 2009 USEPA Guidance Document

Description

Arsenic, mercury, and strontium concentrations (mg/L) from a single well collected approximately quarterly. Nondetects are indicated by the data qualifier `U`.

Usage

```
EPA.09.Table.9.3.df
```

Format

A data frame with 15 observations on the following 8 variables.

`Date.Collected` a factor indicating the date of collection

`Date` a Date object indicating the date of collection

`Arsenic.mg.per.L` a numeric vector of arsenic concentrations (mg/L)

`Arsenic.Data.Qualifier` a factor indicating the data qualifier for arsenic

`Mercury.mg.per.L` a numeric vector of mercury concentrations (mg/L)

`Mercury.Data.Qualifier` a factor indicating the data qualifier for mercury

`Strontium.mg.per.L` a numeric vector of strontium concentrations

`Strontium.Data.Qualifier` a factor indicating the data qualifier for strontium

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.9-13.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.09.Table.9.4.nickel.vec

Nickel Concentrations in Table 9-4 of 2009 USEPA Guidance Document

---

Description

Nickel concentrations (ppb) from a single well.

Usage

```
EPA.09.Table.9.4.nickel.vec
```

Format

a numeric vector of nickel concentrations (ppb)

Source

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.9-18.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

EPA.89b.aldicarb1.df *Aldicarb Concentrations from 1989 USEPA Guidance Document*

Description

Aldicarb concentrations (ppb) at three compliance wells (four monthly samples at each well).

Usage

```
EPA.89b.aldicarb1.df
```

Format

A data frame with 12 observations on the following 3 variables.

`Aldicarb` Aldicarb concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.6-4.

EPA.89b.aldicarb2.df *Aldicarb Concentrations from 1989 USEPA Guidance Document*

Description

Aldicarb concentrations (ppm) at three compliance wells (four monthly samples at each well).

Usage

```
EPA.89b.aldicarb2.df
```

Format

A data frame with 12 observations on the following 3 variables.

`Aldicarb` Aldicarb concentrations (ppm)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.6-13.

EPA.89b.benzene.df *Benzene Concentrations from 1989 USEPA Guidance Document*

Description

Benzene concentrations (ppm) at one background and five compliance wells (four monthly samples for each well).

Usage

```
EPA.89b.benzene.df
```

Format

A data frame with 24 observations on the following 6 variables.

`Benzene.orig` a character vector of the original observations

`Benzene` a numeric vector with <1 observations coded as 1

`Censored` a logical vector indicating which observations are censored

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.5-18.

EPA.89b.cadmium.df *Cadmium Concentrations from 1989 USEPA Guidance Document*

Description

Cadmium concentrations (mg/L) at one set of background and one set of compliance wells. Nondetects reported as "BDL". Detection limit not given.

Usage

```
EPA.89b.cadmium.df
```

Format

A data frame with 88 observations on the following 4 variables.

`Cadmium.orig` a character vector of the original cadmium observations (mg/L)

`Cadmium` a numeric vector with `BDL` coded as `0`

`Censored` a logical vector indicating which observations are censored

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.8-6.

EPA.89b.chlordane1.df *Chlordane Concentrations from 1989 USEPA Guidance Document*

Description

Chlordane concentrations (ppm) in 24 water samples. Two possible phases: dissolved (18 observations) and immiscible (6 observations).

Usage

```
EPA.89b.chlordane1.df
```

Format

A data frame with 24 observations on the following 2 variables.

`Chlordane` Chlordane concentrations (ppm)

`Phase` a factor indicating the phase (dissolved vs. immiscible)

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.4-8.

EPA.89b.chlordane2.df *Chlordane Concentrations from 1989 USEPA Guidance Document*

Description

Chlordane concentrations (ppb) at one background and one compliance well. Observations taken during four separate months over two years. Four replicates taken for each "month/year/well type" combination.

Usage

```
data(EPA.89b.chlordane2.df)
```

Format

A data frame with 32 observations on the following 5 variables.

`Chlordane` Chlordane concentration (ppb)

`Month` a factor indicating the month of collection

`Year` a numeric vector indicating the year of collection (85 or 86)

`Replicate` a factor indicating the replicate number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.5-27.

EPA.89b.edb.df *EDB Concentrations from 1989 USEPA Guidance Document*

Description

EDB concentrations (ppb) at three compliance wells (four monthly samples at each well).

Usage

```
EPA.89b.edb.df
```

Format

A data frame with 12 observations on the following 3 variables.

`EDB` EDB concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.6-6.

EPA.89b.lead.df *Lead Concentrations from 1989 USEPA Guidance Document*

Description

Lead concentrations (ppm) at two background and four compliance wells (four monthly samples for each well).

Usage

```
EPA.89b.lead.df
```

Format

A data frame with 24 observations on the following 4 variables.

`Lead` Lead concentrations (ppm)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.5-23.

EPA.89b.loglead.df *Log-transformed Lead Concentrations from 1989 USEPA Guidance Document*

Description

Log-transformed lead concentrations (μg/L) at two background and four compliance wells (four monthly samples for each well).

Usage

```
EPA.89b.loglead.df
```

Format

A data frame with 24 observations on the following 4 variables.

`LogLead` Natural logarithm of lead concentrations (μg/L)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.5-11.

EPA.89b.manganese.df *Manganese Concentrations from 1989 USEPA Guidance Document*

Description

Manganese concentrations at six monitoring wells (four monthly samples for each well).

Usage

```
EPA.89b.manganese.df
```

Format

A data frame with 24 observations on the following 3 variables.

`Manganese` Manganese concentrations

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.4-19.

EPA.89b.sulfate.df *Sulfate Concentrations from 1989 USEPA Guidance Document*

Description

Sulfate concentrations (mg/L). Nondetects reported as `<1450`.

Usage

```
data(EPA.89b.sulfate.df)
```

Format

A data frame with 24 observations on the following 3 variables.

`Sulfate.orig` a character vector of original sulfate concentration (mg/L)

`Sulfate` a numeric vector of sulfate concentations with `<1450` coded as `1450`

`Censored` a logical vector indicating which observations are censored

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.8-9.

EPA.89b.t29.df *T-29 Concentrations from 1989 USEPA Guidance Document*

Description

T-29 concentrations (ppm) at two compliance wells (four monthly samples at each well, four replicates within each month). Detection limit is not given.

Usage

```
EPA.89b.t29.df
```

Format

A data frame with 32 observations on the following 6 variables.

`T29.orig` a character vector of the original T-29 concentrations (ppm)

`T29` a numeric vector of T-29 concentrations with <? coded as `0`

`Censored` a logical vector indicating which observations are censored

`Month` a factor indicating the month of collection

`Replicate` a factor indicating the replicate number

`Well` a factor indicating the well number

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.6-10.

EPA.89b.toc.vec *Total Organic Carbon Concentrations from 1989 USEPA Guidance Document*

Description

Numeric vector containing total organic carbon (TOC) concentrations (mg/L).

Usage

```
EPA.89b.toc.vec
```

Format

A numeric vector with 19 elements containing TOC concentrations (mg/L).

Source

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.8-13.

EPA.92c.arsenic1.df *Arsenic Concentrations from 1992 USEPA Guidance Document*

Description

Arsenic concentrations (ppm) at six monitoring wells (four monthly samples for each well).

Usage

```
EPA.92c.arsenic1.df
```

Format

A data frame with 24 observations on the following 3 variables.

`Arsenic` Arsenic concentrations (ppm)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.21.

EPA.92c.arsenic2.df *Arsenic Concentrations from 1992 USEPA Guidance Document*

Description

Arsenic concentrations (ppb) at three background wells and one compliance well (six monthly samples for each well; first four missing at compliance well). Nondetects reported as <5.

Usage

```
EPA.92c.arsenic2.df
```

Format

A data frame with 24 observations on the following 6 variables.

`Arsenic.orig` a character vector of original arsenic concentrations (ppb)

`Arsenic` a numeric vector of arsenic concentrations with <5 coded as 5

`Censored` a logical vector indicating which observations are censored

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.60.

EPA.92c.arsenic3.df *Arsenic Concentrations from 1992 USEPA Guidance Document*

Description

Arsenic concentrations at one background and one compliance monitoring well. Three years of observations for background well, two years of observations for compliance well, four samples per year for each well.

Usage

```
EPA.92c.arsenic3.df
```

Format

A data frame with 20 observations on the following 3 variables.

`Arsenic` a numeric vector of arsenic concentrations

`Year` a factor indicating the year of collection

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C.

EPA.92c.benzene1.df *Benzene Concentrations from 1992 USEPA Guidance Document*

Description

Benzene concentrations (ppb) at six background wells (six monthly samples for each well). Non-detects reported as <2.

Usage

```
EPA.92c.benzene1.df
```

Format

A data frame with 36 observations on the following 5 variables.

`Benzene.orig` a character vector of original benzene concentrations (ppb)

`Benzene` a numeric vector of benzene concentrations with <2 coded as 2

`Censored` a logical vector indicating which observations are censored

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.36.

EPA.92c.benzene2.df *Benzene Concentrations from 1992 USEPA Guidance Document*

Description

Benzene concentrations (ppb) at one background and one compliance well. Four observations per month for each well. Background well sampled in months 1,2, and 3; compliance well sampled in months 4 and 5.

Usage

```
EPA.92c.benzene2.df
```

Format

A data frame with 20 observations on the following 3 variables.

`Benzene` a numeric vector of benzene concentrations (ppb)

`Month` a factor indicating the month of collection

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.56.

`EPA.92c.ccl4.df` *Carbon Tetrachloride Concentrations from 1992 USEPA Guidance Document*

Description

Carbon tetrachloride (CCL4) concentrations (ppb) at five wells (four monthly samples at each well).

Usage

```
EPA.92c.ccl4.df
```

Format

A data frame with 20 observations on the following 3 variables.

`CCL4` a numeric vector of carbon tetrachloride concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.80.

EPA.92c.chrysene.df *Chrysene Concentrations from 1992 USEPA Guidance Document*

Description

Chrysene concentrations (ppb) at five compliance wells (four monthly samples for each well).

Usage

```
EPA.92c.chrysene.df
```

Format

A data frame with 20 observations on the following 3 variables.

`Chrysene` a numeric vector of chrysene concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.52.

EPA.92c.copper1.df *Copper Concentrations from 1992 USEPA Guidance Document*

Description

Copper concentrations (ppb) at two background and one compliance wells (six monthly samples for each well).

Usage

```
EPA.92c.copper1.df
```

Format

A data frame with 18 observations on the following 4 variables.

`Copper` a numeric vector of copper concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.47.

EPA.92c.copper2.df *Copper Concentrations from 1992 USEPA Guidance Document*

Description

Copper concentrations (ppb) at three background and two compliance wells (eight monthly samples for each well; first four missing at compliance wells). Nondetects reported as <5.

Usage

```
EPA.92c.copper2.df
```

Format

A data frame with 40 observations on the following 6 variables.

`Copper.orig` a character vector of original copper concentrations (ppb)

`Copper` a numeric vector of copper concentrations with <5 coded as 5

`Censored` a logical vector indicating which observations are censored

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.55.

EPA.92c.lognickel1.df *Log-transformed Nickel Concentrations from 1992 USEPA Guidance Document*

Description

Log-transformed nickel concentrations (ppb) at four monitoring wells (five monthly samples for each well).

Usage

```
EPA.92c.lognickel1.df
```

Format

A data frame with 20 observations on the following 3 variables.

`LogNickel` a numeric vector of log-transformed nickel concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.15.

EPA.92c.nickel1.df *Nickel Concentrations from 1992 USEPA Guidance Document*

Description

Nickel concentrations (ppb) at four monitoring wells (five monthly samples for each well).

Usage

```
EPA.92c.nickel1.df
```

Format

A data frame with 20 observations on the following 3 variables.

`Nickel` a numeric vector of nickel concentrations (ppb)

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.7.

EPA.92c.nickel2.df *Nickel Concentrations from 1992 USEPA Guidance Document*

Description

Nickel concentrations (ppb) at a monitoring well (eight months of samples, two samples for each sampling occasion).

Usage

```
EPA.92c.nickel2.df
```

Format

A data frame with 16 observations on the following 3 variables.

`Nickel` a numeric vector of nickel concentrations (ppb)

`Month` a factor indicating the month of collection

`Sample` a factor indicating the sample (replicate) number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.78.

EPA.92c.toluene.df *Toluene Concentrations from 1992 USEPA Guidance Document*

Description

Toluene concentrations (ppb) at two background and three compliance wells (five monthly samples at each well). Nondetects reported as <5.

Usage

```
EPA.92c.toluene.df
```

Format

A data frame with 25 observations on the following 6 variables.

`Toluene.orig` a character vector of original toluene concentrations (ppb)

`Toluene` a numeric vector of toluene concentrations with <5 coded as 5

`Censored` a logical vector indicating which observations are censored

`Month` a factor indicating the month of collection

`Well` a factor indicating the well number

`Well.type` a factor indicating the well type (background vs. compliance)

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.43.

EPA.92c.zinc.df *Zinc Concentrations from 1992 USEPA Guidance Document*

Description

Zinc concentrations (ppb) at five background wells (eight samples for each well). Nondetects reported as <7.

Usage

```
EPA.92c.zinc.df
```

Format

A data frame with 40 observations on the following 5 variables.

`Zinc.orig` a character vector of original zinc concentrations (ppb)

`Zinc` a numeric vector of zinc concentrations with <7 coded as 7

`Censored` a logical vector indicating which observations are censored

`Sample` a factor indicating the sample number

`Well` a factor indicating the well number

Source

USEPA. (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C. p.30.

EPA.92d.chromium.df *Chromium Concentrations from 1992 USEPA Guidance Document*

Description

Chromium concentrations (mg/kg) in soil samples collected randomly over a Superfund site.

Usage

```
EPA.92d.chromium.df
```

Format

A data frame with 15 observations on the following variable.

`Cr` a numeric vector of chromium concentrations (mg/kg)

Source

USEPA. (1992d). *Supplemental Guidance to RAGS: Calculating the Concentration Term*. Publication 9285.7-081, May 1992. Intermittent Bulletin, Volume 1, Number 1. Office of Emergency and Remedial Response, Hazardous Site Evaluation Division, OS-230. Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

EPA.92d.chromium.vec *Chromium Concentrations from 1992 USEPA Guidance Document*

Description

Chromium concentrations (mg/kg) in soil samples collected randomly over a Superfund site.

Usage

```
EPA.92d.chromium.vec
```

Format

A numeric vector with 15 observations.

Source

USEPA. (1992d). *Supplemental Guidance to RAGS: Calculating the Concentration Term*. Publication 9285.7-081, May 1992. Intermittent Bulletin, Volume 1, Number 1. Office of Emergency and Remedial Response, Hazardous Site Evaluation Division, OS-230. Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

EPA.94b.lead.df *Lead Concentrations from 1994 USEPA Guidance Document*

Description

Lead concentrations (mg/Kg) in soil samples at a reference area and a cleanup area. Nondetects reported as <39. There are 14 observations for each area.

Usage

```
EPA.94b.lead.df
```

Format

A data frame with 28 observations on the following 4 variables.

`Lead.orig` a character vector of original lead concentrations (mg/Kg)

`Lead` a numeric vector of lead concentrations with <39 coded as 39

`Censored` a logical vector indicating which observations are censored

`Area` a factor indicating the area (cleanup vs. reference)

Source

USEPA. (1994b). *Statistical Methods for Evaluating the Attainment of Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid Media*. EPA/230-R-94-004. Office of Policy, Planning, and Evaluation, U.S. Environmental Protection Agency, Washington, D.C. pp.6.20–6.21.

EPA.94b.tccb.df *1,2,3,4-Tetrachlorobenzene Concentrations from 1994 USEPA Guidance Document*

Description

1,2,3,4-Tetrachlorobenzene (TcCB) concentrations (ppb) in soil samples at a reference area and a cleanup area. There are 47 observations for the reference area and 77 for the cleanup area. There is only one nondetect in the dataset (it's in the cleanup area), and it is reported as ND. Here it is assumed the nondetect is less than the smallest reported value, which is 0.09 ppb. Note that on page 6.23 of USEPA (1994b), a value of 25.5 for the Cleanup Unit was erroneously omitted.

Usage

```
EPA.94b.tccb.df
```

Format

A data frame with 124 observations on the following 4 variables.

`TcCB.orig` a character vector with the original tetrachlorobenzene concentrations (ppb)

`TcCB` a numeric vector of tetrachlorobenzene with `<0.99` coded as `0.99`

`Censored` a logical vector indicating which observations are censored

`Area` a factor indicating the area (cleanup vs. reference)

Source

USEPA. (1994b). *Statistical Methods for Evaluating the Attainment of Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid Media*. EPA/230-R-94-004. Office of Policy, Planning, and Evaluation, U.S. Environmental Protection Agency, Washington, D.C. pp.6.22-6.25.

`EPA.97.cadmium.111.df` *Calibration Data for Cadmium at Mass 111*

Description

Calibration data for cadmium at mass 111 (ng/L; method 1638 ICPMS) that appeared in Gibbons et al. (1997b) and were provided to them by the U.S. EPA.

Usage

```
EPA.97.cadmium.111.df
```

Format

A data frame with 35 observations on the following 2 variables.

`Cadmium` Observed concentation of cadmium (ng/L)

`Spike` "True" concentration of cadmium taken from a standard (ng/L)

Source

Gibbons, R.D., D.E. Coleman, and R.F. Maddalone. (1997b). Response to Comment on "An Alternative Minimum Level Definition for Analytical Quantification". *Environmental Science and Technology*, **31**(12), 3729–3731.

`epareto` *Estimate Parameters of a Pareto Distribution*

Description

Estimate the location and shape parameters of a Pareto distribution.

Usage

```
epareto(x, method = "mle", plot.pos.con = 0.375)
```

Arguments

`x`	numeric vector of observations.
`method`	character string specifying the method of estimation. Possible values are `"mle"` (maximum likelihood; the default), and `"lse"` (least-squares). See the DETAILS section for more information on these estimation methods.
`plot.pos.con`	numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the values of the empirical cdf. The default value is `plot.pos.con=0.375`. This argument is used only when `method="lse"`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a Pareto distribution with parameters `location`=η and `shape`=θ.

Maximum Likelihood Estimatation (`method="mle"`)
The maximum likelihood estimators (mle's) of η and θ are given by (Evans et al., 1993; p.122; Johnson et al., 1994, p.581):

$$\hat{\eta}_{mle} = x_{(1)} \quad (1)$$

$$\hat{\theta}_{mle} = n[\sum_{i=1}^{n} log(\frac{x_i}{\hat{\eta}_{mle}})]^{-1} \quad (2)$$

where $x_(1)$ denotes the first order statistic (i.e., the minimum value).

Least-Squares Estimation (`method="lse"`)
The least-squares estimators (lse's) of η and θ are derived as follows. Let X denote a Pareto random variable with parameters `location`=η and `shape`=θ. It can be shown that

$$log[1 - F(x)] = \theta log(\eta) - \theta log(x) \quad (3)$$

where F denotes the cumulative distribution function of X. Set

$$y_i = log[1 - \hat{F}(x_i)] \quad (4)$$

$$z_i = log(x_i) \quad (5)$$

where $\hat{F}(x)$ denotes the empirical cumulative distribution function evaluated at x. The least-squares estimates of η and θ are obtained by solving the regression equation

$$y_i = \beta_0 + \beta_1 z_i \quad (6)$$

and setting

$$\hat{\theta}_{lse} = -\hat{\beta}_1 \quad (7)$$

$$\hat{\eta}_{lse} = exp(\frac{\hat{\beta}_0}{\hat{\theta}_{lse}}) \quad (8)$$

(Johnson et al., 1994, p.580).

Value

a list of class `"estimate"` containing the estimated parameters and other information.
See `estimate.object` for details.

Note

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto's law, which states that the number of persons N having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where θ denotes Pareto's constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the "location" parameter η, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the r'th moment only exists if $r < \theta$.

The Pareto distribution is related to the exponential distribution and logistic distribution as follows. Let X denote a Pareto random variable with `location`=η and `shape`=θ. Then $log(X/\eta)$ has an exponential distribution with parameter `rate`=θ, and $-log\{[(X/\eta)^{\theta}] - 1\}$ has a logistic distribution with parameters `location`=0 and `scale`=1.

The Pareto distribution has a very long right-hand tail. It is often applied in the study of socioeconomic data, including the distribution of income, firm size, population, and stock price fluctuations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

Pareto.

Examples

```
# Generate 30 observations from a Pareto distribution with parameters
# location=1 and shape=1 then estimate the parameters.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpareto(30, location = 1, shape = 1)
epareto(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Pareto
#
#Estimated Parameter(s):          location = 1.009046
#                                 shape    = 1.079850
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     30

#----------

# Compare the results of using the least-squares estimators:

epareto(dat, method="lse")$parameters
#location    shape
#1.085924 1.144180

#----------

# Clean up
#---------

rm(dat)
```

---

epdfPlot *Plot Empirical Probability Density Function*

Description

Produces an empirical probability density function plot.

Usage

```
epdfPlot(x, discrete = FALSE, density.arg.list = NULL, plot.it = TRUE,
  add = FALSE, epdf.col = "black", epdf.lwd = 3 * par("cex"), epdf.lty = 1,
  curve.fill = FALSE, curve.fill.col = "cyan", ...,
  type = ifelse(discrete, "h", "l"), main = NULL, xlab = NULL, ylab = NULL,
  xlim = NULL, ylim = NULL)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
discrete	logical scalar indicating whether the assumed parent distribution of x is discrete (discrete=TRUE) or continuous (discrete=FALSE; the default).
density.arg.list	list with arguments to the density function. The default value is density.arg.list=NULL. This argument is ignored if discrete=TRUE.
plot.it	logical scalar indicating whether to produce a plot or add to the current plot (see add) on the current graphics device. The default value is plot.it=TRUE.
add	logical scalar indicating whether to add the empirical pdf to the current plot (add=TRUE) or generate a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.
epdf.col	a numeric scalar or character string determining the color of the empirical pdf line or points. The default value is epdf.col="black". See the entry for col in the help file for par for more information.
epdf.lwd	a numeric scalar determining the width of the empirical pdf line. The default value is epdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.
epdf.lty	a numeric scalar determining the line type of the empirical pdf line. The default value is ecdf.lty=1. See the entry for lty in the help file for par for more information.
curve.fill	a logical scalar indicating whether to fill in the area below the empirical pdf curve with the color specified by curve.fill.col. The default value is curve.fill=FALSE.
curve.fill.col	a numeric scalar or character string indicating what color to use to fill in the area below the empirical pdf curve. The default value is curve.fill.col="cyan". This argument is ignored if curve.fill=FALSE.
type, main, xlab, ylab, xlim, ylim, ...	additional graphical parameters (see lines and par). In particular, the argument type specifies the kind of line type. By default, the function epdfPlot plots histogram-like vertical lines (type="h") when discrete=TRUE, and plots a straight line between points (type="l") when discrete=FALSE. The user may override these defaults by supplying the graphics parameter type (type="h" for histogram-like vertical lines, type="l" for linear interpolation, type="p" for points only, etc.).

Details

When a distribution is discrete and can only take on a finite number of values, the empirical pdf plot is the same as the standard relative frequency histogram; that is, each bar of the histogram represents the proportion of the sample equal to that particular number (or category). When a distribution is continuous, the function `epdfPlot` calls the R function `density` to compute the estimated probability density at a number of evenly spaced points between the minimum and maximum values.

Value

`epdfPlot` invisibly returns a list with the following components:

`x`	numeric vector of ordered quantiles.
`f.x`	numeric vector of the associated estimated values of the pdf.

Note

An ***empirical probability density function (epdf) plot*** is a graphical tool that can be used in conjunction with other graphical tools such as histograms and boxplots to assess the characteristics of a set of data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA.

See the REFERENCES section in the help file for `density`.

See Also

Empirical, `pdfPlot`, `ecdfPlot`, `cdfPlot`, `cdfCompare`, `qqPlot`.

Examples

```
# Using Reference Area TcCB data in EPA.94b.tccb.df,
# create a histogram of the log-transformed observations,
# then superimpose the empirical pdf plot.

dev.new()
log.TcCB <- with(EPA.94b.tccb.df, log(TcCB[Area == "Reference"]))

hist(log.TcCB, freq = FALSE, xlim = c(-2, 1),
  col = "cyan", xlab = "log [ TcCB (ppb) ]",
  ylab = "Relative Frequency",
  main = "Reference Area TcCB with Empirical PDF")

epdfPlot(log.TcCB, add = TRUE)
```

```
#==========

# Generate 20 observations from a Poisson distribution with
# parameter lambda = 10, and plot the empirical PDF.

set.seed(875)
x <- rpois(20, lambda = 10)
dev.new()
epdfPlot(x, discrete = TRUE)

#==========

# Clean up
#---------
rm(log.TcCB, x)
graphics.off()
```

---

epois *Estimate Parameter of a Poisson Distribution*

Description

Estimate the mean of a Poisson distribution, and optionally construct a confidence interval for the mean.

Usage

```
epois(x, method = "mle/mme/mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "exact", conf.level = 0.95)
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Currently the only possible value is "mle/mme/mvue" (maximum likelihood/method of moments/minimum variance unbiased; the default). See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for the location or scale parameter. The default value is FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the location or scale parameter. Possible values are "exact" (the default), "pearson.hartley.approx" (Pearson-Hartley approximation), and "normal.approx" (normal approximation). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a Poisson distribution with parameter lambda=λ. It can be shown (e.g., Forbes et al., 2009) that if y is defined as:

$$y = \sum_{i=1}^{n} x_i \quad (1)$$

then y is an observation from a Poisson distribution with parameter lambda=$n\lambda$.

Estimation

The maximum likelihood, method of moments, and minimum variance unbiased estimator (mle/mme/mvue) of λ is given by:

$$\hat{\lambda} = \bar{x} \quad (2)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{y}{n} \quad (3)$$

Confidence Intervals

There are three possible ways to construct a confidence interval for λ: based on the exact distribution of the estimator of λ (ci.type="exact"), based on an approximation of Pearson and Hartley (ci.type="pearson.hartley.approx"), or based on the normal approximation (ci.type="normal.approx").

Exact Confidence Interval (ci.method="exact")

If ci.type="two-sided", an exact $(1-\alpha)100\%$ confidence interval for λ can be constructed as $[LCL, UCL]$, where the confidence limits are computed such that:

$$Pr[Y \geq y \| \lambda = LCL] = \frac{\alpha}{2} \quad (4)$$

$$Pr[Y \leq y \| \lambda = UCL] = \frac{\alpha}{2} \quad (5)$$

where y is defined in equation (1) and Y denotes a Poisson random variable with parameter lambda=$n\lambda$.

If ci.type="lower", $\alpha/2$ is replaced with α in equation (4) and UCL is set to ∞.

If ci.type="upper", $\alpha/2$ is replaced with α in equation (5) and LCL is set to 0.

Note that an exact upper confidence bound can be computed even when all observations are 0.

Pearson-Hartley Approximation (ci.method="pearson.hartley.approx")

For a two-sided $(1-\alpha)100\%$ confidence interval for λ, the Pearson and Hartley approximation (Zar, 2010, p.587; Pearson and Hartley, 1970, p.81) is given by:

$$[\frac{\chi^2_{2n\bar{x},\alpha/2}}{2n}, \frac{\chi^2_{2n\bar{x}+2,1-\alpha/2}}{2n}] \quad (6)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-square distribution with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

Normal Approximation (`ci.method="normal.approx"`) An approximate $(1-\alpha)100\%$ confidence interval for λ can be constructed assuming the distribution of the estimator of λ is approximately normally distributed. A two-sided confidence interval is constructed as:

$$[\hat{\lambda} - z_{1-\alpha/2}\hat{\sigma}_{\hat{\lambda}}, \hat{\lambda} + z_{1-\alpha/2}\hat{\sigma}_{\hat{\lambda}}] \quad (7)$$

where z_p is the p'th quantile of the standard normal distribution, and the quantity

$$\hat{\sigma}_{\hat{\lambda}} = \sqrt{\hat{\lambda}/n} \quad (8)$$

denotes the estimated asymptotic standard deviation of the estimator of λ.

One-sided confidence intervals are constructed in a similar manner.

Value

a list of class `"estimate"` containing the estimated parameters and other information.
See `estimate.object` for details.

Note

The Poisson distribution is named after Poisson, who derived this distribution as the limiting distribution of the binomial distribution with parameters `size`=N and `prob`=p, where N tends to infinity, p tends to 0, and Np stays constant.

In this context, the Poisson distribution was used by Bortkiewicz (1898) to model the number of deaths (per annum) from kicks by horses in Prussian Army Corps. In this case, p, the probability of death from this cause, was small, but the number of soldiers exposed to this risk, N, was large.

The Poisson distribution has been applied in a variety of fields, including quality control (modeling number of defects produced in a process), ecology (number of organisms per unit area), and queueing theory. Gibbons (1987b) used the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical concentration (in ppb).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Gibbons, R.D. (1987b). Statistical Models for the Analysis of Volatile Organic Compounds in Waste Disposal Sites. *Ground Water* **25**, 572-580.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 4.

Pearson, E.S., and H.O. Hartley, eds. (1970). *Biometrika Tables for Statisticians, Volume 1*. Cambridge Universtiy Press, New York, p.81.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, pp. 585–586.

See Also

Poisson.

Examples

```
# Generate 20 observations from a Poisson distribution with parameter
# lambda=2, then estimate the parameter and construct a 90% confidence
# interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpois(20, lambda = 2)
epois(dat, ci = TRUE, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Poisson
#
#Estimated Parameter(s):          lambda = 1.8
#
#Estimation Method:               mle/mme/mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         lambda
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                90%
#
#Confidence Interval:             LCL = 1.336558
#                                 UCL = 2.377037

#----------

# Compare the different ways of constructing confidence intervals for
# lambda using the same data as in the previous example:

epois(dat, ci = TRUE, ci.method = "pearson",
  conf.level = 0.9)$interval$limits
#     LCL      UCL
#1.336558 2.377037

epois(dat, ci = TRUE, ci.method = "normal.approx",
```

```
  conf.level = 0.9)$interval$limits
#     LCL      UCL
#1.306544 2.293456

#----------

# Clean up
#---------

rm(dat)
```

---

epoisCensored *Estimate Mean of a Poisson Distribution Based on Type I Censored Data*

Description

Estimate the mean of a Poisson distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for the mean.

Usage

```
epoisCensored(x, censored, method = "mle", censoring.side = "left",
  ci = FALSE, ci.method = "profile.likelihood", ci.type = "two-sided",
  conf.level = 0.95, n.bootstraps = 1000, pivot.statistic = "z",
  ci.sample.size = sum(!censored))
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
method	character string specifying the method of estimation. The possible values are: "mle" (maximum likelihood; the default), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
ci	logical scalar indicating whether to compute a confidence interval for the mean or variance. The default value is ci=FALSE.

`ci.method` character string indicating what method to use to construct the confidence interval for the mean. The possible values are `"profile.likelihood"` (profile likelihood; the default), `"normal.approx"` (normal approximation), and `"bootstrap"` (based on bootstrapping). See the DETAILS section for more information. This argument is ignored if `ci=FALSE`.

`ci.type` character string indicating what kind of confidence interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`. This argument is ignored if `ci=FALSE`.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

`n.bootstraps` numeric scalar indicating how many bootstraps to use to construct the confidence interval for the mean when `ci.type="bootstrap"`. This argument is ignored if `ci=FALSE` and/or `ci.method` does not equal `"bootstrap"`.

`pivot.statistic` character string indicating which pivot statistic to use in the construction of the confidence interval for the mean when `ci.method="normal.approx"` (see the DETAILS section). The possible values are `pivot.statistic="z"` (the default) and `pivot.statistic="t"`. When `pivot.statistic="t"` you may supply the argument `ci.sample size` (see below). The argument `pivot.statistic` is ignored if `ci=FALSE`.

`ci.sample.size` numeric scalar indicating what sample size to assume to construct the confidence interval for the mean if `pivot.statistic="t"` and `ci.method="normal.approx"`. The default value is the number of uncensored observations.

Details

If `x` or `censored` contain any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x}$ denote a vector of N observations from a Poisson distribution with mean `lambda=`λ.

Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_k;\ k \geq 1 \qquad (1)$$

For the case when $k \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $k = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j = 1, 2, \ldots, k$, so that

$$\sum_{i=1}^{k} c_j = c \qquad (2)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one,

the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations.

Estimation

Maximum Likelihood Estimation (`method="mle"`)
For Type I left censored data, the likelihood function is given by:

$$L(\lambda|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (3)$$

where f and F denote the probability density function (pdf) and cumulative distribution function (cdf) of the population (Cohen, 1963; Cohen, 1991, pp.6, 50). That is,

$$f(t) = \frac{e^{-\lambda}\lambda^t}{t!},\ x = 0, 1, 2, \ldots \qquad (4)$$

$$F(t) = \sum_{i=0}^{t} f(i) = \sum_{i=0}^{t} \frac{e^{-\lambda}\lambda^i}{i!} \qquad (5)$$

(Johnson et al., 1992, p.151). For left singly censored data, equation (3) simplifies to:

$$L(\lambda|\underline{x}) = \binom{N}{c} [F(T)]^c \prod_{i=c+1}^{n} f[x_{(i)}] \qquad (6)$$

Similarly, for Type I right censored data, the likelihood function is given by:

$$L(\lambda|\underline{x}) = \binom{N}{c_1 c_2 \ldots c_k n} \prod_{j=1}^{k} [1 - F(T_j)]^{c_j} \prod_{i \in \Omega} f[x_{(i)}] \qquad (7)$$

and for right singly censored data this simplifies to:

$$L(\lambda|\underline{x}) = \binom{N}{c} [1 - F(T)]^c \prod_{i=1}^{n} f[x_{(i)}] \qquad (8)$$

The maximum likelihood estimators are computed by maximizing the likelihood function. For right-censored data, taking the derivative of the log-likelihood function with respect to λ and setting this to 0 produces the following equation:

$$\bar{x} = \lambda\{1 - \sum_{j=1}^{K} \frac{c_j}{n} [\frac{f(T_j)}{1 - F(T_j)}]\} \qquad (9)$$

where

$$\bar{x} = \frac{1}{n} \sum i \in \Omega x_i \qquad (10)$$

Note that the quantity defined in equation (10) is simply the mean of the uncensored observations.

For left-censored data, taking the derivative of the log-likelihood function with respect to λ and setting this to 0 produces the following equation:

$$\bar{x} = \lambda\{1 + \sum_{j=1}^{K} \frac{c_j}{n}[\frac{f(T_j - 1)}{F(T_j - 1)}]\} \qquad (11)$$

The function `epoisCensored` computes the maximum likelihood estimator of λ by solving Equation (9) (right-censored data) or Equation (11) (left-censored data); it uses the sample mean of the uncensored observations as the initial value.

Setting Censored Observations to Half the Censoring Level (`method="half.cen.level"`)
This method is applicable only to left censored data. This method involves simply replacing all the censored observations with half their detection limit, and then computing the mean and standard deviation with the usual formulas (see `epois`).

This method is included only to allow comparison of this method to other methods. ***Setting left-censored observations to half the censoring level is not recommended***.

Confidence Intervals
This section explains how confidence intervals for the mean λ are computed.

Likelihood Profile (`ci.method="profile.likelihood"`)
This method was proposed by Cox (1970, p.88), and Venzon and Moolgavkar (1988) introduced an efficient method of computation. This method is also discussed by Stryhn and Christensen (2003) and Royston (2007). The idea behind this method is to invert the likelihood-ratio test to obtain a confidence interval for the mean λ. Equation (3) above shows the form of the likelihood function $L(\lambda|\underline{x})$ for multiply left-censored data, and Equation (7) shows the function for multiply right-censored data.

Following Stryhn and Christensen (2003), denote the maximum likelihood estimate of the mean by λ^*. The likelihood ratio test statistic (G^2) of the hypothesis $H_0 : \lambda = \lambda_0$ (where λ_0 is a fixed value) equals the drop in $2log(L)$ between the "full" model and the reduced model with λ fixed at μ_0, i.e.,

$$G^2 = 2\{log[L(\lambda^*)] - log[L(\lambda_0)]\} \qquad (11)$$

. Under the null hypothesis, the test statistic G^2 follows a chi-squared distribution with 1 degree of freedom.

A two-sided $(1 - \alpha)100\%$ confidence interval for the mean λ consists of all values of λ_0 for which the test is not significant at level $alpha$:

$$\lambda_0 : G^2 \leq \chi^2_{1,1-\alpha} \qquad (12)$$

where $\chi^2_{\nu,p}$ denotes the p'th quantile of the chi-squared distribution with ν degrees of freedom. One-sided lower and one-sided upper confidence intervals are computed in a similar fashion, except that the quantity $1 - \alpha$ in Equation (12) is replaced with $1 - 2\alpha$.

Normal Approximation (`ci.method="normal.approx"`)
This method constructs approximate $(1-\alpha)100\%$ confidence intervals for λ based on the assumption that the estimator of λ is approximately normally distributed. That is, a two-sided $(1-\alpha)100\%$ confidence interval for λ is constructed as:

$$[\hat{\lambda} - t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\lambda}},\ \hat{\lambda} + t_{1-\alpha/2,m-1}\hat{\sigma}_{\hat{\lambda}}] \quad (13)$$

where $\hat{\lambda}$ denotes the estimate of λ, $\hat{\sigma}_{\hat{\lambda}}$ denotes the estimated asymptotic standard deviation of the estimator of λ, m denotes the assumed sample size for the confidence interval, and $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribuiton with ν degrees of freedom. One-sided confidence intervals are computed in a similar fashion.

The argument `ci.sample.size` determines the value of m and by default is equal to the number of uncensored observations. This is simply an ad-hoc method of constructing confidence intervals and is not based on any published theoretical results.

When `pivot.statistic="z"`, the p'th quantile from the standard normal distribution is used in place of the p'th quantile from Student's t-distribution.

When λ is estimated with the maximum likelihood estimator (`method="mle"`), the variance of $\hat{\lambda}$ is estimated based on the inverse of the Fisher Information matrix. When λ is estimated using the half-censoring-level method (`method="half.cen.level"`), the variance of $\hat{\lambda}$ is estimated as:

$$\hat{\sigma}^2_{\hat{\lambda}} = \frac{\hat{\lambda}}{m} \quad (14)$$

where m denotes the assumed sample size (see above).

Bootstrap and Bias-Corrected Bootstrap Approximation (`ci.method="bootstrap"`)
The bootstrap is a nonparametric method of estimating the distribution (and associated distribution parameters and quantiles) of a sample statistic, regardless of the distribution of the population from which the sample was drawn. The bootstrap was introduced by Efron (1979) and a general reference is Efron and Tibshirani (1993).

In the context of deriving an approximate $(1-\alpha)100\%$ confidence interval for the population mean λ, the bootstrap can be broken down into the following steps:

1. Create a bootstrap sample by taking a random sample of size N from the observations in $\underline{x}$, where sampling is done with replacement. Note that because sampling is done with replacement, the same element of $\underline{x}$ can appear more than once in the bootstrap sample. Thus, the bootstrap sample will usually not look exactly like the original sample (e.g., the number of censored observations in the bootstrap sample will often differ from the number of censored observations in the original sample).
2. Estimate λ based on the bootstrap sample created in Step 1, using the same method that was used to estimate λ using the original observations in $\underline{x}$. Because the bootstrap sample usually does not match the original sample, the estimate of λ based on the bootstrap sample will usually differ from the original estimate based on $\underline{x}$.
3. Repeat Steps 1 and 2 B times, where B is some large number. For the function `epoisCensored`, the number of bootstraps B is determined by the argument `n.bootstraps` (see the section ARGUMENTS above). The default value of `n.bootstraps` is `1000`.
4. Use the B estimated values of λ to compute the empirical cumulative distribution function of this estimator of λ (see `ecdfPlot`), and then create a confidence interval for λ based on this estimated cdf.

The two-sided percentile interval (Efron and Tibshirani, 1993, p.170) is computed as:

$$[\hat{G}^{-1}(\frac{\alpha}{2}),\ \hat{G}^{-1}(1-\frac{\alpha}{2})] \qquad (15)$$

where $\hat{G}(t)$ denotes the empirical cdf evaluated at t and thus $\hat{G}^{-1}(p)$ denotes the p'th empirical quantile, that is, the p'th quantile associated with the empirical cdf. Similarly, a one-sided lower confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha),\ \infty] \qquad (16)$$

and a one-sided upper confidence interval is computed as:

$$[0,\ \hat{G}^{-1}(1-\alpha)] \qquad (17)$$

The function `epoisCensored` calls the R function `quantile` to compute the empirical quantiles used in Equations (15)-(17).

The percentile method bootstrap confidence interval is only first-order accurate (Efron and Tibshirani, 1993, pp.187-188), meaning that the probability that the confidence interval will contain the true value of λ can be off by $k/\sqrt{N}$, where $k$is some constant. Efron and Tibshirani (1993, pp.184-188) proposed a bias-corrected and accelerated interval that is second-order accurate, meaning that the probability that the confidence interval will contain the true value of λ may be off by k/N instead of $k/\sqrt{N}$. The two-sided bias-corrected and accelerated confidence interval is computed as:

$$[\hat{G}^{-1}(\alpha_1),\ \hat{G}^{-1}(\alpha_2)] \qquad (18)$$

where

$$\alpha_1 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1-\hat{a}(z_0+z_{\alpha/2})}] \qquad (19)$$

$$\alpha_2 = \Phi[\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1-\hat{a}(z_0+z_{1-\alpha/2})}] \qquad (20)$$

$$\hat{z}_0 = \Phi^{-1}[\hat{G}(\hat{\lambda})] \qquad (21)$$

$$\hat{a} = \frac{\sum_{i=1}^{N}(\hat{\lambda}_{(\cdot)} - \hat{\lambda}_{(i)})^3}{6[\sum_{i=1}^{N}(\hat{\lambda}_{(\cdot)} - \hat{\lambda}_{(i)})^2]^{3/2}} \qquad (22)$$

where the quantity $\hat{\lambda}_{(i)}$ denotes the estimate of λ using all the values in $\underline{x}$ except the i'th one, and

$$\hat{\lambda}(\cdot) = \frac{1}{N}\sum_{i=1}^{N}\hat{\lambda_{(i)}} \qquad (23)$$

A one-sided lower confidence interval is given by:

$$[\hat{G}^{-1}(\alpha_1),\ \infty] \qquad (24)$$

and a one-sided upper confidence interval is given by:

$$[0,\ \hat{G}^{-1}(\alpha_2)] \qquad (25)$$

where α_1 and α_2 are computed as for a two-sided confidence interval, except $\alpha/2$ is replaced with α in Equations (19) and (20).

The constant $\hat{z}_0$ incorporates the bias correction, and the constant $\hat{a}$ is the acceleration constant. The term "acceleration" refers to the rate of change of the standard error of the estimate of λ with respect to the true value of λ (Efron and Tibshirani, 1993, p.186). For a normal (Gaussian) distribution, the standard error of the estimate of λ does not depend on the value of λ, hence the acceleration constant is not really necessary.

When `ci.method="bootstrap"`, the function `epoisCensored` computes both the percentile method and bias-corrected and accelerated method bootstrap confidence intervals.

Value

a list of class `"estimateCensored"` containing the estimated parameters and other information. See `estimateCensored.object` for details.

Note

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation. Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators for parameters of a normal or lognormal distribution based on censored environmental data.

In practice, it is better to use a confidence interval for the mean or a joint confidence region for the mean and standard deviation (or coefficient of variation), rather than rely on a single point-estimate of the mean. Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and coefficient of variation of a Poisson distribution when data are subjected to single or multiple censoring.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Cohen, A.C. (1991). *Truncated and Censored Samples*. Marcel Dekker, New York, New York, 312pp.

Cox, D.R. (1970). *Analysis of Binary Data*. Chapman & Hall, London. 142pp.

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* **7**, 1–26.

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, 436pp.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). *Statistical Distributions, Fourth Edition*. John Wiley and Sons, Hoboken, NJ.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions, Second Edition*. John Wiley and Sons, New York, Chapter 4.

Millard, S.P., P. Dixon, and N.K. Neerchal. (2014; in preparation). *Environmental Statistics with R*. CRC Press, Boca Raton, Florida.

Nelson, W. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, 634pp.

Royston, P. (2007). Profile Likelihood for Estimation and Confdence Intervals. *The Stata Journal* **7**(3), pp. 376–387.

Stryhn, H., and J. Christensen. (2003). *Confidence Intervals by the Profile Likelihood Method, with Applications in Veterinary Epidemiology*. Contributed paper at ISVEE X (November 2003, Chile). https://gilvanguedes.com/wp-content/uploads/2019/05/Profile-Likelihood-CI.pdf.

Venzon, D.J., and S.H. Moolgavkar. (1988). A Method for Computing Profile-Likelihood-Based Confidence Intervals. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **37**(1), pp. 87–94.

See Also

Poisson, epois, estimateCensored.object.

Examples

```
# Generate 20 observations from a Poisson distribution with
# parameter lambda=10, and censor the values less than 10.
# Then generate 20 more observations from the same distribution
# and censor the values less than 20.  Then estimate the mean
# using the maximum likelihood method.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(300)
dat.1 <- rpois(20, lambda=10)
censored.1 <- dat.1 < 10
dat.1[censored.1] <- 10

dat.2 <- rpois(20, lambda=10)
censored.2 <- dat.2 < 20
dat.2[censored.2] <- 20

dat <- c(dat.1, dat.2)
censored <- c(censored.1, censored.2)

epoisCensored(dat, censored, ci = TRUE)

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#----------------------------------------
#
#Assumed Distribution:            Poisson
#
```

```
#Censoring Side:                 left
#
#Censoring Level(s):             10 20
#
#Estimated Parameter(s):         lambda = 11.05402
#
#Estimation Method:              MLE
#
#Data:                           dat
#
#Censoring Variable:             censored
#
#Sample Size:                    40
#
#Percent Censored:               65%
#
#Confidence Interval for:        lambda
#
#Confidence Interval Method:     Profile Likelihood
#
#Confidence Interval Type:       two-sided
#
#Confidence Level:               95%
#
#Confidence Interval:            LCL =  9.842894
#                                UCL = 12.846484

#----------

# Clean up
#---------
rm(dat.1, censored.1, dat.2, censored.2, dat, censored)
```

---

eqbeta *Estimate Quantiles of a Beta Distribution*

Description

Estimate quantiles of a beta distribution.

Usage

```
eqbeta(x, p = 0.5, method = "mle", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a beta distribution (e.g., ebeta). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method to use to estimate the shape and scale parameters of the distribution. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for ebeta for more information.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqbeta returns estimated quantiles as well as estimates of the shape1 and shape2 parameters.

Quantiles are estimated by 1) estimating the shape1 and shape2 parameters by calling ebeta, and then 2) calling the function qbeta and using the estimated values for shape1 and shape2.

Value

If x is a numeric vector, eqbeta returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqbeta returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The beta distribution takes real values between 0 and 1. Special cases of the beta are the Uniform[0,1] when shape1=1 and shape2=1, and the arcsin distribution when shape1=0.5 and shape2=0.5. The arcsin distribution appears in the theory of random walks. The beta distribution is used in Bayesian analyses as a conjugate to the binomial distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

ebeta, Beta, estimate.object.

Examples

```
# Generate 20 observations from a beta distribution with parameters
# shape1=2 and shape2=4, then estimate the parameters via
# maximum likelihood and estimate the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rbeta(20, shape1 = 2, shape2 = 4)
eqbeta(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Beta
#
#Estimated Parameter(s):          shape1 =  5.392221
#                                 shape2 = 11.823233
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 0.4592796
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     20

#----------
# Clean up

rm(dat)
```

`eqbinom` *Estimate Quantiles of a Binomial Distribution*

Description

Estimate quantiles of a binomial distribution.

Usage

```
eqbinom(x, size = NULL, p = 0.5, method = "mle/mme/mvue", digits = 0)
```

Arguments

`x`	numeric or logical vector of observations, or an object resulting from a call to an estimating function that assumes a binomial distribution (e.g., `ebinom`). If `x`

	is a vector of observations, then when `size` is not supplied, `x` must be a numeric vector of 0s ("failures") and 1s ("successes"), or else a logical vector of `FALSE` values ("failures") and `TRUE` values ("successes"). When `size` is supplied, `x` must be a non-negative integer containing the number of "successes" out of the number of trials indicated by `size`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`size`	positive integer indicating the of number of trials; `size` must be at least as large as the value of `x`.
`p`	numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. The default value is `p=0.5`.
`method`	character string specifying the method of estimation. The only possible value is `"mle/mme/mvue"` (maximum likelihood, method of moments, and minimum variance unbiased). See the DETAILS section of the help file for `ebinom` for more information.
`digits`	an integer indicating the number of decimal places to round to when printing out the value of `100*p`. The default value is `digits=0`.

Details

The function `eqbinom` returns estimated quantiles as well as estimates of the `prob` parameter.

Quantiles are estimated by 1) estimating the prob parameter by calling `ebinom`, and then 2) calling the function `qbinom` and using the estimated value for `prob`.

Value

If `x` is a numeric vector, `eqbinom` returns a list of class `"estimate"` containing the estimated quantile(s) and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `eqbinom` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as components called `quantiles` and `quantile.method`.

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", p, is the same on each trial. A binomial discrete random variable X is the number of "successes" in n independent trials. A special case of the binomial distribution occurs when $n = 1$, in which case X is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143). The binomial distribution is also used to compute an upper bound on the overall Type I error rate for deciding whether a facility or location is in compliance with some set standard. Assume the null hypothesis is that the facility is in compliance. If a test of hypothesis is conducted periodically over time to test compliance and/or several tests are performed during each time period, and the facility or location is always in compliance, and each single test has a Type I error rate of α, and the result of each test is independent of the result of any other test (usually not a reasonable assumption), then the number of times the facility is declared out of compliance when

in fact it is in compliance is a binomial random variable with probability of "success" $p = \alpha$ being the probability of being declared out of compliance (see USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Agresti, A., and B.A. Coull. (1998). Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions. *The American Statistician*, **52**(2), 119–126.

Agresti, A., and B. Caffo. (2000). Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures. *The American Statistician*, **54**(4), 280–288.

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapters 2 and 15.

Cochran, W.G. (1977). *Sampling Techniques*. John Wiley and Sons, New York, Chapter 3.

Fisher, R.A., and F. Yates. (1963). *Statistical Tables for Biological, Agricultural, and Medical Research*. 6th edition. Hafner, New York, 146pp.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. Second Edition. John Wiley and Sons, New York, Chapters 1-2.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, Chapter 11.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 3.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Newcombe, R.G. (1998a). Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods. *Statistics in Medicine*, **17**, 857–872.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL, Chapter 4.

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.6-38.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 24.

See Also

ebinom, Binomial, estimate.object.

Examples

```
# Generate 20 observations from a binomial distribution with
# parameters size=1 and prob=0.2, then estimate the 'prob'
# parameter and the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.

set.seed(251)
dat <- rbinom(20, size = 1, prob = 0.2)
eqbinom(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Binomial
#
#Estimated Parameter(s):          size = 20.0
#                                 prob =  0.1
#
#Estimation Method:               mle/mme/mvue for 'prob'
#
#Estimated Quantile(s):           90'th %ile = 4
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle/mme/mvue for 'prob' Estimators
#
#Data:                            dat
#
#Sample Size:                     20
#
#
#

#----------
# Clean up

rm(dat)
```

---

eqevd *Estimate Quantiles of an Extreme Value (Gumbel) Distribution*

Description

Estimate quantiles of an extreme value distribution.

Usage

```
eqevd(x, p = 0.5, method = "mle", pwme.method = "unbiased",
  plot.pos.cons = c(a = 0.35, b = 0), digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes an extreme value distribution (e.g., eevd). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method to use to estimate the location and scale parameters. Possible values are "mle" (maximum likelihood; the default), "mme" (methods of moments), "mmue" (method of moments based on the unbiased estimator of variance), and "pwme" (probability-weighted moments). See the DETAILS section of the help file for eevd for more information on these estimation methods.
pwme.method	character string specifying what method to use to compute the probability-weighted moments when method="pwme". The possible values are "ubiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section of the help file for eevd for more information. This argument is ignored if method is not equal to "pwme".
plot.pos.cons	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="pwme" and pwme.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section of the help file for eevd for more information. This argument is ignored if method is not equal to "pwme" or if pwme.method="ubiased".
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqevd returns estimated quantiles as well as estimates of the location and scale parameters.

Quantiles are estimated by 1) estimating the location and scale parameters by calling eevd, and then 2) calling the function qevd and using the estimated values for location and scale.

Value

If x is a numeric vector, eqevd returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqevd returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

There are three families of extreme value distributions. The one described here is the Type I, also called the Gumbel extreme value distribution or simply Gumbel distribution. The name "extreme value" comes from the fact that this distribution is the limiting distribution (as n approaches infinity) of the greatest value among n independent random variables each having the same continuous distribution.

The Gumbel extreme value distribution is related to the exponential distribution as follows. Let Y be an exponential random variable with parameter rate=λ. Then $X = \eta - log(Y)$ has an extreme value distribution with parameters location=η and scale=$1/\lambda$.

The distribution described above and assumed by eevd is the *largest* extreme value distribution. The smallest extreme value distribution is the limiting distribution (as n approaches infinity) of the smallest value among n independent random variables each having the same continuous distribution. If X has a largest extreme value distribution with parameters location=η and scale=θ, then $Y = -X$ has a smallest extreme value distribution with parameters location=$-\eta$ and scale=θ. The smallest extreme value distribution is related to the Weibull distribution as follows. Let Y be a Weibull random variable with parameters shape=β and scale=α. Then $X = log(Y)$ has a smallest extreme value distribution with parameters location=$log(\alpha)$ and scale=$1/\beta$.

The extreme value distribution has been used extensively to model the distribution of streamflow, flooding, rainfall, temperature, wind speed, and other meteorological variables, as well as material strength and life data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Castillo, E. (1988). *Extreme Value Theory in Engineering*. Academic Press, New York, pp.184–198.

Downton, F. (1966). Linear Estimates of Parameters in the Extreme Value Distribution. *Technometrics* **8**(1), 3–17.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Greenwood, J.A., J.M. Landwehr, N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments: Definition and Relation to Parameters of Several Distributions Expressible in Inverse Form. *Water Resources Research* **15**(5), 1049–1054.

Hosking, J.R.M., J.R. Wallis, and E.F. Wood. (1985). Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments. *Technometrics* **27**(3), 251–261.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

Landwehr, J.M., N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments Compared With Some Traditional Techniques in Estimating Gumbel Parameters and Quantiles. *Water Resources Research* **15**(5), 1055–1064.

Tiago de Oliveira, J. (1963). Decision Results for the Parameters of the Extreme Value (Gumbel) Distribution Based on the Mean and Standard Deviation. *Trabajos de Estadistica* **14**, 61–81.

See Also

eevd, Extreme Value Distribution, estimate.object.

Examples

```
# Generate 20 observations from an extreme value distribution with
# parameters location=2 and scale=1, then estimate the parameters
# and estimate the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- revd(20, location = 2)
eqevd(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Extreme Value
#
#Estimated Parameter(s):          location = 1.9684093
#                                 scale    = 0.7481955
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 3.652124
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     20

#----------

# Clean up
#---------
rm(dat)
```

---

eqexp *Estimate Quantiles of an Exponential Distribution*

Description

Estimate quantiles of an exponential distribution.

Usage

```
eqexp(x, p = 0.5, method = "mle/mme", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes an exponential distribution (e.g., eexp). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method to use to estimate the rate parameter. Currently the only possible value is "mle/mme" (maximum likelihood/method of moments; the default). See the DETAILS section of the help file for eexp for more information.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqexp returns estimated quantiles as well as the estimate of the rate parameter.

Quantiles are estimated by 1) estimating the rate parameter by calling eexp, and then 2) calling the function qexp and using the estimated value for rate.

Value

If x is a numeric vector, eqexp returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqexp returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The exponential distribution is a special case of the gamma distribution, and takes on positive real values. A major use of the exponential distribution is in life testing where it is used to model the lifetime of a product, part, person, etc.

The exponential distribution is the only continuous distribution with a "lack of memory" property. That is, if the lifetime of a part follows the exponential distribution, then the distribution of the time until failure is the same as the distribution of the time until failure given that the part has survived to time t.

The exponential distribution is related to the double exponential (also called Laplace) distribution, and to the extreme value distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

eexp, Exponential, estimate.object.

Examples

```
# Generate 20 observations from an exponential distribution with parameter
# rate=2, then estimate the parameter and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rexp(20, rate = 2)
eqexp(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Exponential
#
#Estimated Parameter(s):          rate = 2.260587
#
#Estimation Method:               mle/mme
#
#Estimated Quantile(s):           90'th %ile = 1.018578
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle/mme Estimators
#
#Data:                            dat
#
#Sample Size:                     20
#

#----------

# Clean up
#---------
rm(dat)
```

---

eqgamma *Estimate Quantiles of a Gamma Distribution*

Description

Estimate quantiles of a gamma distribution, and optionally construct a confidence interval for a quantile.

Usage

```
eqgamma(x, p = 0.5, method = "mle", ci = FALSE,
  ci.type = "two-sided", conf.level = 0.95,
  normal.approx.transform = "kulkarni.powar", digits = 0)

eqgammaAlt(x, p = 0.5, method = "mle", ci = FALSE,
  ci.type = "two-sided", conf.level = 0.95,
  normal.approx.transform = "kulkarni.powar", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a gamma distribution (e.g., egamma or egammaAlt). If ci=TRUE then x must be a numeric vector of observations. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.
method	character string specifying the method to use to estimate the shape and scale parameters of the distribution. The possible values are "mle" (maximum likelihood; the default), "bcmle" (bias-corrected mle), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for egamma for more information.
ci	logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.
ci.type	character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
normal.approx.transform	character string indicating which power transformation to use. Possible values are "kulkarni.powar" (the default), "cube.root", and "fourth.root". See the DETAILS section for more informaiton. This argument is ignored if ci=FALSE.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqgamma returns estimated quantiles as well as estimates of the shape and scale parameters. The function eqgammaAlt returns estimated quantiles as well as estimates of the mean and coefficient of variation.

Quantiles are estimated by 1) estimating the shape and scale parameters by calling egamma, and then 2) calling the function qgamma and using the estimated values for shape and scale.

The confidence interval for a quantile is computed by:

1. using a power transformation on the original data to induce approximate normality,
2. using eqnorm to compute the confidence interval, and then
3. back-transforming the interval to create a confidence interval on the original scale.

This is similar to what is done to create tolerance intervals for a gamma distribuiton (Krishnamoorthy et al., 2008), and there is a one-to-one relationship between confidence intervals for a quantile and tolerance intervals (see the DETAILS section of the help file for eqnorm). The value normal.approx.transform="cube.root" uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value normal.approx.transform="fourth.root" uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value normal.approx.transform="kulkarni.powar" uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010). The "optimum" power r is determined by:

$$\begin{aligned} r &= -0.0705 - 0.178\, shape + 0.475\, \sqrt{shape} \quad \text{if } shape \leq 1.5 \\ r &= 0.246 \quad \text{if } shape > 1.5 \end{aligned}$$

where $shape$ denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power r, for the functions eqgamma and eqgammaAlt the power r is based on whatever estimate of shape is used (e.g., method="mle", method="bcmle", etc.).

Value

If x is a numeric vector, eqgamma and eqgammaAlt return a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqgamma and eqgammaAlt return a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method. In addition, if ci=TRUE, the returned list contains a component called interval containing the confidence interval information. If x already has a component called interval, this component is replaced with the confidence interval information.

Note

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter a tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribuiton instead.

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hawkins, D. M., and R.A.J. Wixley. (1986). A Note on the Transformation of Chi-Squared Variables to Normality. *The American Statistician*, **40**, 296–298.

Johnson, N.L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, Chapter 17.

Krishnamoorthy K., T. Mathew, and S. Mukherjee. (2008). Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability. *Technometrics*, **50**(1), 69–78.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Kulkarni, H.V., and S.K. Powar. (2010). A New Method for Interval Estimation of the Mean of the Gamma Distribution. *Lifetime Data Analysis*, **16**, 431–447.

Singh, A., A.K. Singh, and R.J. Iaci. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Wilson, E.B., and M.M. Hilferty. (1931). The Distribution of Chi-Squares. *Proceedings of the National Academy of Sciences*, **17**, 684–688.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

egamma, GammaDist, estimate.object, eqnorm, tolIntGamma.

Examples

```
# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then estimate the 90th percentile and create
# a one-sided upper 95% confidence interval for that percentile.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
eqgamma(dat, p = 0.9, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 2.203862
#                                 scale = 2.174928
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 9.113446
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         90'th %ile
#
#Confidence Interval Method:      Exact using
#                                 Kulkarni & Powar (2010)
#                                 transformation to Normality
#                                 based on mle of 'shape'
```

```
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =  0.00000
#                                 UCL = 13.79733

#----------
# Compare these results with the true 90'th percentile:

qgamma(p = 0.9, shape = 3, scale = 2)
#[1] 10.64464

#----------

# Using the same data as in the previous example, use egammaAlt
# to estimate the mean and cv based on the bias-corrected
# estimate of shape, and use the cube-root transformation to
# normality.

eqgammaAlt(dat, p = 0.9, method = "bcmle", ci = TRUE,
  ci.type = "upper", normal.approx.transform = "cube.root")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          mean = 4.7932408
#                                 cv   = 0.7242165
#
#Estimation Method:               bcmle of 'shape'
#
#Estimated Quantile(s):           90'th %ile = 9.428
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 bcmle of 'shape'
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         90'th %ile
#
#Confidence Interval Method:      Exact using
#                                 Wilson & Hilferty (1931) cube-root
#                                 transformation to Normality
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
```

```
#Confidence Interval:             LCL =  0.00000
#                                 UCL = 12.89643

#----------

# Clean up
rm(dat)

#--------------------------------------------------------------------

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and
# 95% confidence  using chrysene data and assuming a lognormal
# distribution.  Here we will use the same chrysene data but assume a
# gamma distribution.

# A beta-content upper tolerance limit with 95% coverage and
# 95% confidence is equivalent to the 95% upper confidence limit for
# the 95th percentile.

attach(EPA.09.Ex.17.3.chrysene.df)
Chrysene <- Chrysene.ppb[Well.type == "Background"]
eqgamma(Chrysene, p = 0.95, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 2.806929
#                                 scale = 5.286026
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           95'th %ile = 31.74348
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            Chrysene
#
#Sample Size:                     8
#
#Confidence Interval for:         95'th %ile
#
#Confidence Interval Method:      Exact using
#                                 Kulkarni & Powar (2010)
#                                 transformation to Normality
#                                 based on mle of 'shape'
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
```

```
#
#Confidence Interval:            LCL =  0.00000
#                                UCL = 69.32425

#----------
# Clean up

rm(Chrysene)
detach("EPA.09.Ex.17.3.chrysene.df")
```

---

eqgeom *Estimate Quantiles of a Geometric Distribution*

Description

Estimate quantiles of a geometric distribution.

Usage

```
eqgeom(x, p = 0.5, method = "mle/mme", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a geometric distribution (e.g., egeom). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method to use to estimate the probability parameter. Possible values are "mle/mme" (maximum likelihood and method of moments; the default) and "mvue" (minimum variance unbiased). You cannot use method="mvue" if length(x)=1. See the DETAILS section of the help file for egeom for more information on these estimation methods.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqgeom returns estimated quantiles as well as the estimate of the rate parameter.

Quantiles are estimated by 1) estimating the probability parameter by calling egeom, and then 2) calling the function qgeom and using the estimated value for the probability parameter.

Value

If x is a numeric vector, eqgeom returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqgeom returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The geometric distribution with parameter prob=p is a special case of the negative binomial distribution with parameters size=1 and prob=p.

The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 5.

See Also

egeom, Geometric, enbinom, NegBinomial, estimate.object.

Examples

```
# Generate an observation from a geometric distribution with parameter
# prob=0.2, then estimate the parameter prob and the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rgeom(1, prob = 0.2)
dat
#[1] 4

eqgeom(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Geometric
#
```

```
#Estimated Parameter(s):          prob = 0.2
#
#Estimation Method:               mle/mme
#
#Estimated Quantile(s):           90'th %ile = 10
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle/mme Estimators
#
#Data:                            dat
#
#Sample Size:                     1

#----------

# Clean up
#---------
rm(dat)
```

---

eqgevd *Estimate Quantiles of a Generalized Extreme Value Distribution*

Description

Estimate quantiles of a generalized extreme value distribution.

Usage

```
eqgevd(x, p = 0.5, method = "mle", pwme.method = "unbiased",
  tsoe.method = "med", plot.pos.cons = c(a = 0.35, b = 0), digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a generalized extreme value distribution (e.g., egevd). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method to use to estimate the location, scale, and threshold parameters. Possible values are "mle" (maximum likelihood; the default), "pwme" (probability-weighted moments), and "tsoe" (two-stage order-statistics estimator of Castillo and Hadi (1994)). See the DETAILS section of the help file for egevd for more information on these estimation methods.
pwme.method	character string specifying what method to use to compute the probability-weighted moments when method="pwme". The possible values are "ubiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section of the help file for

egevd for more information. This argument is ignored if method is not equal to "pwme".

tsoe.method — character string specifying the robust function to apply in the second stage of the two-stage order-statistics estimator when method="tsoe". Possible values are "med" (median; the default), and "lms" (least median of squares). See the DETAILS section of the help file for egevd for more information on these estimation methods. This argument is ignored if method is not equal to "tsoe".

plot.pos.cons — numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="pwme" and pwme.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section of the help file for egevd for more information. This argument is used only if method="tsoe", or if both method="pwme" and pwme.method="plotting.position".

digits — an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqgevd returns estimated quantiles as well as estimates of the location, scale and threshold parameters.

Quantiles are estimated by 1) estimating the location, scale, and threshold parameters by calling egevd, and then 2) calling the function qgevd and using the estimated values for location, scale, and threshold.

Value

If x is a numeric vector, eqevd returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqevd returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930's to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter $\kappa = 0$, the GEVD reduces to the Type I extreme value (Gumbel) distribution. (The function zTestGevdShape allows you to test the null hypothesis $H_0 : \kappa = 0$.) When $\kappa > 0$, the GEVD is the same as the

Type II extreme value distribution, and when $\kappa < 0$ it is the same as the Type III extreme value distribution.

Hosking et al. (1985) compare the asymptotic and small-sample statistical properties of the PWME with the MLE and Jenkinson's (1969) method of sextiles. Castillo and Hadi (1994) compare the small-sample statistical properties of the MLE, PWME, and TSOE. Hosking and Wallis (1995) compare the small-sample properties of unbaised L-moment estimators vs. plotting-position L-moment estimators. (PWMEs can be written as linear combinations of L-moments and thus have equivalent statistical properties.) Hosking and Wallis (1995) conclude that unbiased estimators should be used for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Castillo, E., and A. Hadi. (1994). Parameter and Quantile Estimation for the Generalized Extreme-Value Distribution. *Environmetrics* **5**, 417–432.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Greenwood, J.A., J.M. Landwehr, N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments: Definition and Relation to Parameters of Several Distributions Expressible in Inverse Form. *Water Resources Research* **15**(5), 1049–1054.

Hosking, J.R.M. (1984). Testing Whether the Shape Parameter is Zero in the Generalized Extreme-Value Distribution. *Biometrika* **71**(2), 367–374.

Hosking, J.R.M. (1985). Algorithm AS 215: Maximum-Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Applied Statistics* **34**(3), 301–310.

Hosking, J.R.M., J.R. Wallis, and E.F. Wood. (1985). Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments. *Technometrics* **27**(3), 251–261.

Jenkinson, A.F. (1969). Statistics of Extremes. *Technical Note 98*, World Meteorological Office, Geneva.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

Landwehr, J.M., N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments Compared With Some Traditional Techniques in Estimating Gumbel Parameters and Quantiles. *Water Resources Research* **15**(5), 1055–1064.

Macleod, A.J. (1989). Remark AS R76: A Remark on Algorithm AS 215: Maximum Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Applied Statistics* **38**(1), 198–199.

Prescott, P., and A.T. Walden. (1980). Maximum Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Biometrika* **67**(3), 723–724.

Prescott, P., and A.T. Walden. (1983). Maximum Likelihood Estimation of the Three-Parameter Generalized Extreme-Value Distribution from Censored Samples. *Journal of Statistical Computing and Simulation* **16**, 241–250.

See Also

egevd, Generalized Extreme Value Distribution, Extreme Value Distribution, eevd, estimate.object.

Examples

```
# Generate 20 observations from a generalized extreme value distribution
# with parameters location=2, scale=1, and shape=0.2, then compute the
# MLEs of location, shape,and threshold, and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(498)
dat <- rgevd(20, location = 2, scale = 1, shape = 0.2)
eqgevd(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Generalized Extreme Value
#
#Estimated Parameter(s):          location = 1.6144631
#                                 scale    = 0.9867007
#                                 shape    = 0.2632493
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 3.289912
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     20

#----------

# Clean up
#---------
rm(dat)
```

---

eqhyper *Estimate Quantiles of a Hypergeometric Distribution*

Description

Estimate quantiles of a hypergeometric distribution.

Usage

```
eqhyper(x, m = NULL, total = NULL, k = NULL, p = 0.5, method = "mle", digits = 0)
```

Arguments

x	non-negative integer indicating the number of white balls out of a sample of size k drawn without replacement from the urn, or an object resulting from a call to an estimating function that assumes a hypergeometric distribution (e.g., ehyper). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
m	non-negative integer indicating the number of white balls in the urn. You must supply m or total, but not both. Missing values (NAs) are not allowed.
total	positive integer indicating the total number of balls in the urn (i.e., m+n). You must supply m or total, but not both. Missing values (NAs) are not allowed.
k	positive integer indicating the number of balls drawn without replacement from the urn. Missing values (NAs) are not allowed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method of estimating the parameters of the hypergeometric distribution. Possible values are "mle" (maximum likelihood; the default) and "mvue" (minimum variance unbiased). The mvue method is only available when you are estimating m (i.e., when you supply the argument total). See the DETAILS section of the help file for ehyper for more information on these estimation methods.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqhyper returns estimated quantiles as well as estimates of the hypergeometric distribution parameters.

Quantiles are estimated by 1) estimating the distribution parameters by calling ehyper, and then 2) calling the function qhyper and using the estimated values for the distribution parameters.

Value

If x is a numeric vector, eqhyper returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqhyper returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The hypergeometric distribution can be described by an urn model with M white balls and N black balls. If K balls are drawn *with* replacement, then the number of white balls in the sample of size

K follows a binomial distribution with parameters size=K and prob=$M/(M+N)$. If K balls are drawn *without* replacement, then the number of white balls in the sample of size K follows a hypergeometric distribution with parameters m=M, n=N, and k=K.

The name "hypergeometric" comes from the fact that the probabilities associated with this distribution can be written as successive terms in the expansion of a function of a Gaussian hypergeometric series.

The hypergeometric distribution is applied in a variety of fields, including quality control and estimation of animal population size. It is also the distribution used to compute probabilities for Fishers's exact test for a 2x2 contingency table.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 6.

See Also

ehyper, Hypergeometric, estimate.object.

Examples

```
# Generate an observation from a hypergeometric distribution with
# parameters m=10, n=30, and k=5, then estimate the parameter m, and
# the 80'th percentile.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter actually estimated is m; once m is estimated,
# n is computed by subtracting the estimated value of m (8 in this example)
# from the given of value of m+n (40 in this example).  The parameters
# n and k are shown in the output in order to provide information on
# all of the parameters associated with the hypergeometric distribution.

set.seed(250)
dat <- rhyper(nn = 1, m = 10, n = 30, k = 5)
dat
#[1] 1

eqhyper(dat, total = 40, k = 5, p = 0.8)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Hypergeometric
#
#Estimated Parameter(s):          m =  8
```

```
#                                n = 32
#                                k =  5
#
#Estimation Method:              mle for 'm'
#
#Estimated Quantile(s):          80'th %ile = 2
#
#Quantile Estimation Method:     Quantile(s) Based on
#                                mle for 'm' Estimators
#
#Data:                           dat
#
#Sample Size:                    1

#----------

# Clean up
#---------
rm(dat)
```

---

eqlnorm *Estimate Quantiles of a Lognormal Distribution*

Description

Estimate quantiles of a lognormal distribution, and optionally construct a confidence interval for a quantile.

Usage

```
eqlnorm(x, p = 0.5, method = "qmle", ci = FALSE,
  ci.method = "exact", ci.type = "two-sided", conf.level = 0.95,
  digits = 0)
```

Arguments

x	a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., elnorm, elnormCensored). You *cannot* use objects resulting from a call to estimating functions that use the alternative parameterization such as elnormAlt. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.
method	character string indicating what method to use to estimate the quantile(s). The possible values are "qmle" (quasi maximum likelihood; the default) and "mvue" (minimum variance unbiased). The method "mvue" is available only

when p=0.5 (i.e., when you are estimating the median). See the DETAILS section for more information.

ci — logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.

ci.method — character string indicating what method to use to construct the confidence interval for the quantile. The possible values are "exact" (exact method; the default) and "normal.approx" (normal approximation). See the DETAILS section for more information.

ci.type — character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level — a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

digits — an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Quantiles and their associated confidence intervals are constructed by calling the function eqnorm using the log-transformed data and then exponentiating the quantiles and confidence limits.

In the special case when p=0.5 and method="mvue", the estimated median is computed using the method given in Gilbert (1987, p.172) and Bradu and Mundlak (1970).

Value

If x is a numeric vector, eqlnorm returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqlnorm returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method. In addition, if ci=TRUE, the returned list contains a component called interval containing the confidence interval information. If x already has a component called interval, this component is replaced with the confidence interval information.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Bradu, D., and Y. Mundlak. (1970). Estimation in Lognormal Linear Models. *Journal of the American Statistical Association* **65**, 198-211.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, pp.88-90.

Johnson, N.L., and B.L. Welch. (1940). Applications of the Non-Central t-Distribution. *Biometrika* **31**, 362-389.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Stedinger, J. (1983). Confidence Intervals for Design Events. *Journal of Hydraulic Engineering* **109**(1), 13-27.

Stedinger, J.R., R.M. Vogel, and E. Foufoula-Georgiou. (1993). Frequency Analysis of Extreme Events. In: Maidment, D.R., ed. *Handbook of Hydrology*. McGraw-Hill, New York, Chapter 18, pp.29-30.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

eqnorm, Lognormal, elnorm, estimate.object.

Examples

```
# Generate 20 observations from a lognormal distribution with
# parameters meanlog=3 and sdlog=0.5, then estimate the 90th
# percentile and create a one-sided upper 95% confidence interval
# for that percentile.
```

```
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(47)
dat <- rlnorm(20, meanlog = 3, sdlog = 0.5)
eqlnorm(dat, p = 0.9, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = 2.9482139
#                                 sdlog   = 0.4553215
#
#Estimation Method:               mvue
#
#Estimated Quantile(s):           90'th %ile = 34.18312
#
#Quantile Estimation Method:      qmle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         90'th %ile
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =  0.00000
#                                 UCL = 45.84008

#----------
# Compare these results with the true 90'th percentile:

qlnorm(p = 0.9, meanlog = 3, sdlog = 0.5)
#[1] 38.1214

#----------

# Clean up
rm(dat)

#--------------------------------------------------------------------

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal
# distribution.
```

```
# A beta-content upper tolerance limit with 95% coverage and 95%
# confidence is equivalent to the 95% upper confidence limit for the
# 95th percentile.

attach(EPA.09.Ex.17.3.chrysene.df)
Chrysene <- Chrysene.ppb[Well.type == "Background"]
eqlnorm(Chrysene, p = 0.95, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = 2.5085773
#                                 sdlog   = 0.6279479
#
#Estimation Method:               mvue
#
#Estimated Quantile(s):           95'th %ile = 34.51727
#
#Quantile Estimation Method:      qmle
#
#Data:                            Chrysene
#
#Sample Size:                     8
#
#Confidence Interval for:         95'th %ile
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =  0.0000
#                                 UCL = 90.9247

#----------
# Clean up

rm(Chrysene)
detach("EPA.09.Ex.17.3.chrysene.df")
```

---

eqlnorm3 *Estimate Quantiles of a Three-Parameter Lognormal Distribution*

Description

Estimate quantiles of a three-parameter lognormal distribution.

Usage

```
eqlnorm3(x, p = 0.5, method = "lmle", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a three-parameter lognormal distribution (e.g., elnorm3). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.
method	character string specifying the method of estimating the distribution parameters. Possible values are "lmle" (local maximum likelihood; the default), "mme" (method of moments), "mmue" (method of moments using an unbaised estimate of variance), "mmme" (modified method of moments due to Cohen and Whitten (1980)), "zero.skew" (zero-skewness estimator due to Griffiths (1980)), and "royston.skew" (estimator based on Royston's (1992b) index of skewness). See the DETAILS section of the help file for elnorm3 for more information on these estimation methods.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Quantiles are estimated by 1) estimating the distribution parameters by calling elnorm3, and then 2) calling the function qlnorm3 and using the estimated distribution parameters.

Value

If x is a numeric vector, eqlnorm3 returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqlnorm3 returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The problem of estimating the parameters of a three-parameter lognormal distribution has been extensively discussed by Aitchison and Brown (1957, Chapter 6), Calitz (1973), Cohen (1951), Cohen (1988), Cohen and Whitten (1980), Cohen et al. (1985), Griffiths (1980), Harter and Moore (1966), Hill (1963), and Royston (1992b). Stedinger (1980) and Hoshi et al. (1984) discuss fitting the three-parameter lognormal distribution to hydrologic data.

The global maximum likelihood estimates are inadmissible. In the past, several researchers have found that the local maximum likelihood estimates (lmle's) occasionally fail because of convergence

problems, but they were not using the likelihood profile and reparameterization of Griffiths (1980). Cohen (1988) recommends the modified methods of moments estimators over lmle's because they are easy to compute, they are unbiased with respect to μ and σ^2 (the mean and standard deviation on the log-scale), their variances are minimal or near minimal, and they do not suffer from regularity problems.

Because the distribution of the lmle of the threshold parameter γ is far from normal for moderate sample sizes (Griffiths, 1980), it is questionable whether confidence intervals for γ or the median based on asymptotic variances and covariances will perform well. Cohen and Whitten (1980) and Cohen et al. (1985), however, found that the asymptotic variances and covariances are reasonably close to corresponding simulated variances and covariances for the modified method of moments estimators (`method="mmme"`). In a simulation study (5000 monte carlo trials), Royston (1992b) found that the coverage of confidence intervals for γ based on the likelihood profile (`ci.method="likelihood.profile"`) was very close the nominal level (94.1% for a nominal level of 95%), although not symmetric. Royston (1992b) also found that the coverage of confidence intervals for γ based on the skewness method (`ci.method="skewness"`) was also very close (95.4%) and symmetric.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special references to its uses in economics)*. Cambridge University Press, London, Chapter 5.

Calitz, F. (1973). Maximum Likelihood Estimation of the Parameters of the Three-Parameter Lognormal Distribution–a Reconsideration. *Australian Journal of Statistics* **15**(3), 185–190.

Cohen, A.C. (1951). Estimating Parameters of Logarithmic-Normal Distributions by Maximum Likelihood. *Journal of the American Statistical Association* **46**, 206–212.

Cohen, A.C. (1988). Three-Parameter Estimation. In Crow, E.L., and K. Shimizu, eds. *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, Chapter 4.

Cohen, A.C., and B.J. Whitten. (1980). Estimation in the Three-Parameter Lognormal Distribution. *Journal of the American Statistical Association* **75**, 399–404.

Cohen, A.C., B.J. Whitten, and Y. Ding. (1985). Modified Moment Estimation for the Three-Parameter Lognormal Distribution. *Journal of Quality Technology* **17**, 92–99.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, Chapter 2.

Griffiths, D.A. (1980). Interval Estimation for the Three-Parameter Lognormal Distribution via the Likelihood Function. *Applied Statistics* **29**, 58–68.

Harter, H.L., and A.H. Moore. (1966). Local-Maximum-Likelihood Estimation of the Parameters of Three-Parameter Lognormal Populations from Complete and Censored Samples. *Journal of the American Statistical Association* **61**, 842–851.

Heyde, C.C. (1963). On a Property of the Lognormal Distribution. *Journal of the Royal Statistical Society, Series B* **25**, 392–393.

Hill, .B.M. (1963). The Three-Parameter Lognormal Distribution and Bayesian Analysis of a Point-Source Epidemic. *Journal of the American Statistical Association* **58**, 72–84.

Hoshi, K., J.R. Stedinger, and J. Burges. (1984). Estimation of Log-Normal Quantiles: Monte Carlo Results and First-Order Approximations. *Journal of Hydrology* **71**, 1–30.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Royston, J.P. (1992b). Estimation, Reference Ranges and Goodness of Fit for the Three-Parameter Log-Normal Distribution. *Statistics in Medicine* **11**, 897–912.

Stedinger, J.R. (1980). Fitting Lognormal Distributions to Hydrologic Data. *Water Resources Research* **16**(3), 481–490.

See Also

elnorm3, Lognormal3, Lognormal, LognormalAlt, Normal.

Examples

```
# Generate 20 observations from a 3-parameter lognormal distribution
# with parameters meanlog=1.5, sdlog=1, and threshold=10, then use
# Cohen and Whitten's (1980) modified moments estimators to estimate
# the parameters, and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnorm3(20, meanlog = 1.5, sdlog = 1, threshold = 10)
eqlnorm3(dat, method = "mmme", p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            3-Parameter Lognormal
#
#Estimated Parameter(s):          meanlog   = 1.5206664
#                                 sdlog     = 0.5330974
#                                 threshold = 9.6620403
#
#Estimation Method:               mmme
#
#Estimated Quantile(s):           90'th %ile = 18.72194
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mmme Estimators
#
#Data:                            dat
#
#Sample Size:                     20

# Clean up
#---------
rm(dat)
```

eqlnormCensored *Estimate Quantiles of a Lognormal Distribution Based on Type I Censored Data*

Description

Estimate quantiles of a lognormal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for a quantile.

Usage

```
eqlnormCensored(x, censored, censoring.side = "left", p = 0.5, method = "mle",
  ci = FALSE, ci.method = "exact.for.complete", ci.type = "two-sided",
  conf.level = 0.95, digits = 0, nmc = 1000, seed = NULL)
```

Arguments

x	a numeric vector of positive observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.
method	character string specifying the method of estimating the mean and standard deviation on the log-scale. For singly censored data, the possible values are: "mle" (maximum likelihood; the default), "bcmle" (bias-corrected maximum likelihood), "qq.reg" (quantile-quantile regression), "qq.reg.w.cen.level" (quantile-quantile regression including the censoring level), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), "impute.w.qq.reg.w.cen.level" (moment estimation based on imputation using the qq.reg.w.cen.level method), "impute.w.mle" (moment estimation based on imputation using the mle), "iterative.impute.w.qq.reg" (moment estimation based on iterative imputation using the qq.reg method), "m.est" (robust M-estimation), and

"half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

For multiply censored data, the possible values are:
"mle" (maximum likelihood; the default),
"qq.reg" (quantile-quantile regression),
"impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), and
"half.cen.level" (moment estimation based on setting the censored observations to half the censoring level).

See the DETAILS section for more information.

ci — logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.

ci.method — character string indicating what method to use to construct the confidence interval for the quantile. The possible values are:
"exact.for.complete" (exact method for complete (uncensored) data; the default),
"gpq" (method based on generalized pivotal quantities), and
"normal.approx" (normal approximation).
See the DETAILS section for more information. This argument is ignored if ci=FALSE.

ci.type — character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

conf.level — a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.

digits — an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

nmc — numeric scalar indicating the number of Monte Carlo simulations to run when ci=TRUE and ci.method="gpq". The default is nmc=1000.

seed — integer supplied to the function set.seed and used when ci=TRUE and ci.method="gpq". The default value is seed=NULL, in which case the current value of .Random.seed is used.

Details

Quantiles and their associated confidence intervals are constructed by calling the function eqnormCensored using the log-transformed data and then exponentiating the quantiles and confidence limits.

Value

eqlnormCensored returns a list of class "estimateCensored" containing the estimated quantile(s) and other information. See estimateCensored.object for details.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for a percentile, rather than rely on a single point-estimate of percentile. Confidence intervals for percentiles of a normal distribution depend on the properties of the estimators for both the mean and standard deviation.

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring (see, for example, Singh et al., 2006). Studies to evaluate the performance of a confidence interval for a percentile include: Caudill et al. (2007), Hewett and Ganner (2007), Kroll and Stedinger (1996), and Serasinghe (2010).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Caudill, S.P., L.-Y. Wong, W.E. Turner, R. Lee, A. Henderson, D. G. Patterson Jr. (2007). Percentile Estimation Using Variable Censored Data. *Chemosphere* **68**, 169–180.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, pp.132-136.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, pp.88-90.

Hewett, P., and G.H. Ganser. (2007). A Comparison of Several Methods for Analyzing Censored Data. *Annals of Occupational Hygiene* **51**(7), 611–632.

Johnson, N.L., and B.L. Welch. (1940). Applications of the Non-Central t-Distribution. *Biometrika* **31**, 362-389.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Kroll, C.N., and J.R. Stedinger. (1996). Estimation of Moments and Quantiles Using Censored Data. *Water Resources Research* **32**(4), 1005–1012.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Odeh, R.E., and D.B. Owen. (1980). *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*. Marcel Dekker, New York.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Serasinghe, S.K. (2010). *A Simulation Comparison of Parametric and Nonparametric Estimators of Quantiles from Right Censored Data*. A Report submitted in partial fulfillment of the requirements for the degree Master of Science, Department of Statistics, College of Arts and Sciences, Kansas State University, Manhattan, Kansas.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Stedinger, J. (1983). Confidence Intervals for Design Events. *Journal of Hydraulic Engineering* **109**(1), 13-27.

Stedinger, J.R., R.M. Vogel, and E. Foufoula-Georgiou. (1993). Frequency Analysis of Extreme Events. In: Maidment, D.R., ed. *Handbook of Hydrology*. McGraw-Hill, New York, Chapter 18, pp.29-30.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wald, A., and J. Wolfowitz. (1946). Tolerance Limits for a Normal Distribution. *Annals of Mathematical Statistics* **17**, 208-215.

See Also

eqnormCensored, enormCensored, tolIntNormCensored, elnormCensored, Lognormal, estimateCensored.object.

Examples

```
# Generate 15 observations from a lognormal distribution with
# parameters meanlog=3 and sdlog=0.5, and censor observations less than 10.
# Then generate 15 more observations from this distribution and  censor
# observations less than 9.
# Then estimate the 90th percentile and create a one-sided upper 95%
# confidence interval for that percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(47)

x.1 <- rlnorm(15, meanlog = 3, sdlog = 0.5)
sort(x.1)
# [1]  8.051717  9.651611 11.671282 12.271247 12.664108 17.446124
# [7] 17.707301 20.238069 20.487219 21.025510 21.208197 22.036554
#[13] 25.710773 28.661973 54.453557

censored.1 <- x.1 < 10
x.1[censored.1] <- 10

x.2 <- rlnorm(15, meanlog = 3, sdlog = 0.5)
sort(x.2)
# [1]  6.289074  7.511164  8.988267  9.179006 12.869408 14.130081
# [7] 16.941937 17.060513 19.287572 19.682126 20.363893 22.750203
#[13] 24.744306 28.089325 37.792873

censored.2 <- x.2 < 9
x.2[censored.2] <- 9

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)
```

```
eqlnormCensored(x, censored, p = 0.9, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#----------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):               9 10
#
#Estimated Parameter(s):          meanlog = 2.8099300
#                                 sdlog   = 0.5137151
#
#Estimation Method:               MLE
#
#Estimated Quantile(s):           90'th %ile = 32.08159
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 MLE Estimators
#
#Data:                            x
#
#Censoring Variable:              censored
#
#Sample Size:                     30
#
#Percent Censored:                16.66667%
#
#Confidence Interval for:         90'th %ile
#
#Assumed Sample Size:             30
#
#Confidence Interval Method:      Exact for
#                                 Complete Data
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =  0.00000
#                                 UCL = 41.38716

#----------

# Compare these results with the true 90'th percentile:

qlnorm(p = 0.9, meanlog = 3, sd = 0.5)
#[1] 38.1214

#----------
```

```
# Clean up
rm(x.1, censored.1, x.2, censored.2, x, censored)

#--------------------------------------------------------------------

# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and standard deviation using the MLE,
# and then construct an upper 95% confidence limit for the 90th percentile.

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                <2            2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now estimate the mean, standard deviation, and 90th percentile
# on the log-scale using the MLE, and construct an upper 95%
# confidence limit for the 90th percentile:
#---------------------------------------------------------------

with(EPA.09.Ex.15.1.manganese.df,
  eqlnormCensored(Manganese.ppb, Censored,
    p = 0.9, ci = TRUE, ci.type = "upper"))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#-------------------------------------------
#
```

```
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          meanlog = 2.215905
#                                 sdlog   = 1.356291
#
#Estimation Method:               MLE
#
#Estimated Quantile(s):           90'th %ile = 52.14674
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 MLE Estimators
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Confidence Interval for:         90'th %ile
#
#Assumed Sample Size:             25
#
#Confidence Interval Method:      Exact for
#                                 Complete Data
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =   0.0000
#                                 UCL = 110.9305
```

`eqlogis` *Estimate Quantiles of a Logistic Distribution*

Description

Estimate quantiles of a logistic distribution.

Usage

```
eqlogis(x, p = 0.5, method = "mle", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a logistic distribution (e.g., elogis). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method to use to estimate the distribution parameters. Possible values are "mle" (maximum likelihood; the default), "mme" (methods of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for elogis for more information.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqlogis returns estimated quantiles as well as estimates of the location and scale parameters.

Quantiles are estimated by 1) estimating the location and scale parameters by calling elogis, and then 2) calling the function qlogis and using the estimated values for location and scale.

Value

If x is a numeric vector, eqlogis returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqlogis returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The logistic distribution is defined on the real line and is unimodal and symmetric about its location parameter (the mean). It has longer tails than a normal (Gaussian) distribution. It is used to model growth curves and bioassay data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

elogis, Logistic, estimate.object.

Examples

```
# Generate 20 observations from a logistic distribution with
# parameters location=0 and scale=1, then estimate the parameters
# and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlogis(20)
eqlogis(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Logistic
#
#Estimated Parameter(s):          location = -0.2181845
#                                 scale    =  0.8152793
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 1.573167
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     20

#----------
# Clean up

rm(dat)
```

---

eqnbinom *Estimate Quantiles of a Negative Binomial Distribution*

Description

Estimate quantiles of a negative binomial distribution.

Usage

```
eqnbinom(x, size = NULL, p = 0.5, method = "mle/mme", digits = 0)
```

Arguments

x	vector of non-negative integers indicating the number of trials that took place *before* size "successes" occurred (the total number of trials that took place is x+1), or an object resulting from a call to an estimating function that assumes a negative binomial distribution (e.g., enbinom). If x is a vector of non-negative integers, then missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. If length(x)=n and n is greater than 1, it is assumed that x represents observations from n separate negative binomial experiments that all had the same probability of success (prob), but possibly different values of size.
size	vector of positive integers indicating the number of "successes" that must be observed before the trials are stopped. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The length of size must be 1 or else the same length as x.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method of estimating the probability parameter. Possible values are "mle/mme" (maximum likelihood and method of moments; the default) and "mvue" (minimum variance unbiased). You cannot use method="mvue" if the sum of the elements in size is 1. See the DETAILS section of the help file for enbinom for more information on these estimation methods.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqnbinom returns estimated quantiles as well as estimates of the prob parameter.

Quantiles are estimated by 1) estimating the prob parameter by calling enbinom, and then 2) calling the function qnbinom and using the estimated value for prob.

Value

If x is a numeric vector, eqnbinom returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqnbinom returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The negative binomial distribution has its roots in a gambling game where participants would bet on the number of tosses of a coin necessary to achieve a fixed number of heads. The negative binomial distribution has been applied in a wide variety of fields, including accident statistics, birth-and-death processes, and modeling spatial distributions of biological organisms.

The geometric distribution with parameter prob=p is a special case of the negative binomial distribution with parameters size=1 and prob=p.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 5.

See Also

enbinom, NegBinomial, egeom, Geometric, estimate.object.

Examples

```
# Generate an observation from a negative binomial distribution with
# parameters size=2 and prob=0.2, then estimate the parameter prob
# and the 90th percentile.
# Note: the call to set.seed simply allows you to reproduce this example.
# Also, the only parameter that is estimated is prob; the parameter
# size is supplied in the call to enbinom.  The parameter size is printed in
# order to show all of the parameters associated with the distribution.

set.seed(250)
dat <- rnbinom(1, size = 2, prob = 0.2)
dat
#[1] 5

eqnbinom(dat, size = 2, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Negative Binomial
#
#Estimated Parameter(s):          size = 2.0000000
#                                 prob = 0.2857143
#
#Estimation Method:               mle/mme for 'prob'
#
#Estimated Quantile(s):           90'th %ile = 11
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle/mme for 'prob' Estimators
#
#Data:                            dat, 2
#
#Sample Size:                     1
```

```
#----------
# Clean up

rm(dat)
```

---

eqnorm *Estimate Quantiles of a Normal Distribution*

Description

Estimate quantiles of a normal distribution, and optionally construct a confidence interval for a quantile.

Usage

```
eqnorm(x, p = 0.5, method = "qmle", ci = FALSE,
  ci.method = "exact", ci.type = "two-sided", conf.level = 0.95,
  digits = 0, warn = TRUE)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (i.e., enorm, enormCensored). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.
method	character string indicating what method to use to estimate the quantile(s). Currently the only possible value is method="qmle" (quasi maximum likelihood). See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the quantile. The possible values are "exact" (exact method; the default) and "normal.approx" (normal approximation). See the DETAILS section for more information.
ci.type	character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

warn — logical scalar indicating whether to warn in the case when ci=TRUE, ci.method="exact", and the supplied object x is of class "estimate" but did not use method="mvue" for estimation.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Quantiles are estimated by 1) estimating the mean and standard deviation parameters by calling enorm with method="mvue", and then 2) calling the function qnorm and using the estimated values for mean and standard deviation. This estimator of the p'th quantile is sometimes called the quasi-maximum likelihood estimator (qmle; Cohn et al., 1989) because if the maximum likelihood estimator of standard deviation were used in place of the minimum variaince unbiased one, then this estimator of the quantile would be the mle of the p'th quantile.

When ci=TRUE and ci.method="exact", the confidence interval for a quantile is computed by using the relationship between a confidence interval for a quantile and a tolerance interval. Specifically, it can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to an upper β-content tolerance interval with coverage $100p\%$ and confidence level $100(1-\alpha)\%$. Also, a lower confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to a lower β-content tolerance interval with coverage $100(1-p)\%$ and confidence level $100(1-\alpha)\%$. See the help file for tolIntNorm for information on tolerance intervals for a normal distribution.

When ci=TRUE and ci.method="normal.approx", the confidence interval for a quantile is computed by assuming the estimated quantile has an approximately normal distribution and using the asymptotic variance to construct the confidence interval (see Stedinger, 1983; Stedinger et al., 1993).

Value

If x is a numeric vector, eqnorm returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqnorm returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method. In addition, if ci=TRUE, the returned list contains a component called interval containing the confidence interval information. If x already has a component called interval, this component is replaced with the confidence interval information.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, pp.132-136.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, pp.88-90.

Johnson, N.L., and B.L. Welch. (1940). Applications of the Non-Central t-Distribution. *Biometrika* **31**, 362-389.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Stedinger, J. (1983). Confidence Intervals for Design Events. *Journal of Hydraulic Engineering* **109**(1), 13-27.

Stedinger, J.R., R.M. Vogel, and E. Foufoula-Georgiou. (1993). Frequency Analysis of Extreme Events. In: Maidment, D.R., ed. *Handbook of Hydrology*. McGraw-Hill, New York, Chapter 18, pp.29-30.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

enorm, tolIntNorm, Normal, estimate.object.

Examples

```
# Generate 20 observations from a normal distribution with
# parameters mean=10 and sd=2, then estimate the 90th
# percentile and create a one-sided upper 95% confidence interval
# for that percentile.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)
```

```
set.seed(47)
dat <- rnorm(20, mean = 10, sd = 2)
eqnorm(dat, p = 0.9, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 9.792856
#                                 sd   = 1.821286
#
#Estimation Method:               mvue
#
#Estimated Quantile(s):           90'th %ile = 12.12693
#
#Quantile Estimation Method:      qmle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         90'th %ile
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =     -Inf
#                                 UCL = 13.30064

#----------
# Compare these results with the true 90'th percentile:

qnorm(p = 0.9, mean = 10, sd = 2)
#[1] 12.56310

#----------

# Clean up
rm(dat)

#==========

# Example 21-4 of USEPA (2009, p. 21-13) shows how to construct a
# 99% lower confidence limit for the 95th percentile using chrysene
# data and assuming a lognormal distribution.  The data for this
# example are stored in EPA.09.Ex.21.1.aldicarb.df.

# The facility permit has established an ACL of 30 ppb that should not
# be exceeded more than 5% of the time.  Thus, if the lower confidence limit
```

```
# for the 95th percentile is greater than 30 ppb, the well is deemed to be
# out of compliance.

# Look at the data
#-----------------

head(EPA.09.Ex.21.1.aldicarb.df)
#  Month   Well Aldicarb.ppb
#1     1 Well.1         19.9
#2     2 Well.1         29.6
#3     3 Well.1         18.7
#4     4 Well.1         24.2
#5     1 Well.2         23.7
#6     2 Well.2         21.9

longToWide(EPA.09.Ex.21.1.aldicarb.df,
  "Aldicarb.ppb", "Month", "Well", paste.row.name = TRUE)
#        Well.1 Well.2 Well.3
#Month.1   19.9   23.7    5.6
#Month.2   29.6   21.9    3.3
#Month.3   18.7   26.9    2.3
#Month.4   24.2   26.1    6.9

# Estimate the 95th percentile and compute the lower
# 99% confidence limit for Well 1.
#---------------------------------------------------

with(EPA.09.Ex.21.1.aldicarb.df,
  eqnorm(Aldicarb.ppb[Well == "Well.1"], p = 0.95, ci = TRUE,
    ci.type = "lower", conf.level = 0.99))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 23.10000
#                                 sd   =  4.93491
#
#Estimation Method:               mvue
#
#Estimated Quantile(s):           95'th %ile = 31.2172
#
#Quantile Estimation Method:      qmle
#
#Data:                            Aldicarb.ppb[Well == "Well.1"]
#
#Sample Size:                     4
#
#Confidence Interval for:         95'th %ile
#
#Confidence Interval Method:      Exact
#
```

```
#Confidence Interval Type:        lower
#
#Confidence Level:                99%
#
#Confidence Interval:             LCL = 25.2855
#                                 UCL =     Inf


# Now compute the 99% lower confidence limit for each of the three
# wells all at once.
#-------------------------------------------------------------------

LCLs <- with(EPA.09.Ex.21.1.aldicarb.df,
  sapply(split(Aldicarb.ppb, Well),
    function(x) eqnorm(x, p = 0.95, method = "qmle", ci = TRUE,
    ci.type = "lower", conf.level = 0.99)$interval$limits["LCL"]))

round(LCLs, 2)
#Well.1.LCL Well.2.LCL Well.3.LCL
#     25.29      25.66       5.46

LCLs > 30
#Well.1.LCL Well.2.LCL Well.3.LCL
#     FALSE      FALSE      FALSE


# Clean up
#---------

rm(LCLs)


#==========

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence using chrysene data and assuming a lognormal
# distribution.

# A beta-content upper tolerance limit with 95% coverage and 95%
# confidence is equivalent to the 95% upper confidence limit for the
# 95th percentile.

# Here we will construct a 95% upper confidence limit for the 95th
# percentile based on the log-transformed data, then exponentiate the
# result to get the confidence limit on the original scale.  Note that
# it is easier to just use the function eqlnorm with the original data
# to achieve the same result.

attach(EPA.09.Ex.17.3.chrysene.df)
log.Chrysene <- log(Chrysene.ppb[Well.type == "Background"])
eqnorm(log.Chrysene, p = 0.95, ci = TRUE, ci.type = "upper")
```

```
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 2.5085773
#                                 sd   = 0.6279479
#
#Estimation Method:               mvue
#
#Estimated Quantile(s):           95'th %ile = 3.54146
#
#Quantile Estimation Method:      qmle
#
#Data:                            log.Chrysene
#
#Sample Size:                     8
#
#Confidence Interval for:         95'th %ile
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =     -Inf
#                                 UCL = 4.510032

exp(4.510032)
#[1] 90.92473

#----------
# Clean up

rm(log.Chrysene)
detach("EPA.09.Ex.17.3.chrysene.df")
```

`eqnormCensored` *Estimate Quantiles of a Normal Distribution Based on Type I Censored Data*

Description

Estimate quantiles of a normal distribution given a sample of data that has been subjected to Type I censoring, and optionally construct a confidence interval for a quantile.

Usage

```
eqnormCensored(x, censored, censoring.side = "left", p = 0.5, method = "mle",
```

```
    ci = FALSE, ci.method = "exact.for.complete", ci.type = "two-sided",
    conf.level = 0.95, digits = 0, nmc = 1000, seed = NULL)
```

Arguments

x	a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.
method	character string specifying the method of estimating the mean and standard deviation. For singly censored data, the possible values are: "mle" (maximum likelihood; the default), "bcmle" (bias-corrected maximum likelihood), "qq.reg" (quantile-quantile regression), "qq.reg.w.cen.level" (quantile-quantile regression including the censoring level), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), "impute.w.qq.reg.w.cen.level" (moment estimation based on imputation using the qq.reg.w.cen.level method), "impute.w.mle" (moment estimation based on imputation using the mle), "iterative.impute.w.qq.reg" (moment estimation based on iterative imputation using the qq.reg method), "m.est" (robust M-estimation), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level). For multiply censored data, the possible values are: "mle" (maximum likelihood; the default), "qq.reg" (quantile-quantile regression), "impute.w.qq.reg" (moment estimation based on imputation using the qq.reg method), and "half.cen.level" (moment estimation based on setting the censored observations to half the censoring level). See the DETAILS section for more information.

ci	logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the quantile. The possible values are: "exact.for.complete" (exact method for complete (i.e., uncensored) data; the default), "gpq" (method based on generalized pivotal quantities), and "normal.approx" (normal approximation). See the DETAILS section for more information. This argument is ignored if ci=FALSE.
ci.type	character string indicating what kind of confidence interval for the quantile to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.
nmc	numeric scalar indicating the number of Monte Carlo simulations to run when ci.method="gpq". The default is nmc=1000. This argument is ignored if ci=FALSE.
seed	integer supplied to the function set.seed and used when ci.method="gpq". The default value is seed=NULL, in which case the current value of .Random.seed is used. This argument is ignored when ci=FALSE.

Details

Estimating Quantiles

Quantiles are estimated by:

1. estimating the mean and standard deviation parameters by calling enormCensored, and then
2. calling the function qnorm and using the estimated values for the mean and standard deviation.

The estimated quantile thus depends on the method of estimating the mean and standard deviation.

Confidence Intervals for Quantiles

Exact Method When Data are Complete (ci.method="exact.for.complete")
When ci.method="exact.for.complete", the function eqnormCensored calls the function eqnorm, supplying it with the estimated mean and standard deviation, and setting the argument ci.method="exact". Thus, this is the exact method for computing a confidence interval for a quantile had the data been complete. Because the data have been subjected to Type I censoring, this method of constructing a confidence interval for the quantile is an approximation.

Normal Approximation (ci.method="normal.approx")
When ci.method="normal.approx", the function eqnormCensored calls the function eqnorm, supplying it with the estimated mean and standard deviation, and setting the argument

`ci.method="normal.approx"`. Thus, this is the normal approximation method for computing a confidence interval for a quantile had the data been complete. Because the data have been subjected to Type I censoring, this method of constructing a confidence interval for the quantile is an approximation both because of the normal approximation and because the estimates of the mean and standard devation are based on censored, instead of complete, data.

Generalized Pivotal Quantity (`ci.method="gpq"`)
When `ci.method="gpq"`, the function `eqnormCensored` uses the relationship between confidence intervals for quantiles and tolerance intervals and calls the function `tolIntNormCensored` with the argument `ti.method="gpq"` to construct the confidence interval. Specifically, it can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to an upper β-content tolerance interval with coverage $100p\%$ and confidence level $100(1-\alpha)\%$. Also, a lower confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to a lower β-content tolerance interval with coverage $100(1-p)\%$ and confidence level $100(1-\alpha)\%$.

Value

`eqnormCensored` returns a list of class `"estimateCensored"` containing the estimated quantile(s) and other information. See `estimateCensored.object` for details.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

A sample of data contains censored observations if some of the observations are reported only as being below or above some censoring level. In environmental data analysis, Type I left-censored data sets are common, with values being reported as "less than the detection limit" (e.g., Helsel, 2012). Data sets with only one censoring level are called *singly censored*; data sets with multiple censoring levels are called *multiply* or *progressively censored*.

Statistical methods for dealing with censored data sets have a long history in the field of survival analysis and life testing. More recently, researchers in the environmental field have proposed alternative methods of computing estimates and confidence intervals in addition to the classical ones such as maximum likelihood estimation.

Helsel (2012, Chapter 6) gives an excellent review of past studies of the properties of various estimators based on censored environmental data.

In practice, it is better to use a confidence interval for a percentile, rather than rely on a single point-estimate of percentile. Confidence intervals for percentiles of a normal distribution depend on the properties of the estimators for both the mean and standard deviation.

Few studies have been done to evaluate the performance of methods for constructing confidence intervals for the mean or joint confidence regions for the mean and standard deviation when data are subjected to single or multiple censoring (see, for example, Singh et al., 2006). Studies to

evaluate the performance of a confidence interval for a percentile include: Caudill et al. (2007), Hewett and Ganner (2007), Kroll and Stedinger (1996), and Serasinghe (2010).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Caudill, S.P., L.-Y. Wong, W.E. Turner, R. Lee, A. Henderson, D. G. Patterson Jr. (2007). Percentile Estimation Using Variable Censored Data. *Chemosphere* **68**, 169–180.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, pp.132-136.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, pp.88-90.

Hewett, P., and G.H. Ganser. (2007). A Comparison of Several Methods for Analyzing Censored Data. *Annals of Occupational Hygiene* **51**(7), 611–632.

Johnson, N.L., and B.L. Welch. (1940). Applications of the Non-Central t-Distribution. *Biometrika* **31**, 362-389.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Kroll, C.N., and J.R. Stedinger. (1996). Estimation of Moments and Quantiles Using Censored Data. *Water Resources Research* **32**(4), 1005–1012.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Odeh, R.E., and D.B. Owen. (1980). *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*. Marcel Dekker, New York.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Serasinghe, S.K. (2010). *A Simulation Comparison of Parametric and Nonparametric Estimators of Quantiles from Right Censored Data*. A Report submitted in partial fulfillment of the requirements for the degree Master of Science, Department of Statistics, College of Arts and Sciences, Kansas State University, Manhattan, Kansas.

Singh, A., R. Maichle, and S. Lee. (2006). *On the Computation of a 95% Upper Confidence Limit of the Unknown Population Mean Based Upon Data Sets with Below Detection Limit Observations*. EPA/600/R-06/022, March 2006. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Stedinger, J. (1983). Confidence Intervals for Design Events. *Journal of Hydraulic Engineering* **109**(1), 13-27.

Stedinger, J.R., R.M. Vogel, and E. Foufoula-Georgiou. (1993). Frequency Analysis of Extreme Events. In: Maidment, D.R., ed. *Handbook of Hydrology*. McGraw-Hill, New York, Chapter 18, pp.29-30.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wald, A., and J. Wolfowitz. (1946). Tolerance Limits for a Normal Distribution. *Annals of Mathematical Statistics* **17**, 208-215.

See Also

`enormCensored`, `tolIntNormCensored`, `Normal`, `estimateCensored.object`.

Examples

```
# Generate 15 observations from a normal distribution with
# parameters mean=10 and sd=2, and censor observations less than 8.
# Then generate 15 more observations from this distribution and  censor
# observations less than 7.
# Then estimate the 90th percentile and create a one-sided upper 95%
# confidence interval for that percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)
```

```
set.seed(47)

x.1 <- rnorm(15, mean = 10, sd = 2)
sort(x.1)
# [1]  6.343542  7.068499  7.828525  8.029036  8.155088  9.436470
# [7]  9.495908 10.030262 10.079205 10.182946 10.217551 10.370811
#[13] 10.987640 11.422285 13.989393
censored.1 <- x.1 < 8
x.1[censored.1] <- 8

x.2 <- rnorm(15, mean = 10, sd = 2)
sort(x.2)
# [1]  5.355255  6.065562  6.783680  6.867676  8.219412  8.593224
# [7]  9.319168  9.347066  9.837844  9.918844 10.055054 10.498296
#[13] 10.834382 11.341558 12.528482
censored.2 <- x.2 < 7
x.2[censored.2] <- 7

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)

eqnormCensored(x, censored, p = 0.9, ci = TRUE, ci.type = "upper")

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#------------------------------------------
#
#Assumed Distribution:            Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              7 8
#
#Estimated Parameter(s):          mean = 9.390624
#                                 sd   = 1.827156
#
#Estimation Method:               MLE
#
#Estimated Quantile(s):           90'th %ile = 11.73222
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 MLE Estimators
#
#Data:                            x
#
#Censoring Variable:              censored
#
#Sample Size:                     30
#
#Percent Censored:                16.66667%
#
#Confidence Interval for:         90'th %ile
#
```

```
#Assumed Sample Size:             30
#
#Confidence Interval Method:      Exact for
#                                 Complete Data
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =     -Inf
#                                 UCL = 12.63808

#----------

# Compare these results with the true 90'th percentile:

qnorm(p = 0.9, mean = 10, sd = 2)
#[1] 12.56310

#----------

# Clean up
rm(x.1, censored.1, x.2, censored.2, x, censored)

#==========

# Chapter 15 of USEPA (2009) gives several examples of estimating the mean
# and standard deviation of a lognormal distribution on the log-scale using
# manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will estimate the mean and standard deviation using the MLE,
# and then construct an upper 95% confidence limit for the 90th percentile.

# We will log-transform the original observations and then call
# eqnormCensored.  Alternatively, we could have more simply called
# eqlnormCensored.

# First look at the data:
#-----------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                <2           2.0     TRUE
```

```
longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now estimate the mean, standard deviation, and 90th percentile
# on the log-scale using the MLE, and construct an upper 95%
# confidence limit for the 90th percentile on the log-scale:
#------------------------------------------------------------------

est.list <- with(EPA.09.Ex.15.1.manganese.df,
  eqnormCensored(log(Manganese.ppb), Censored,
    p = 0.9, ci = TRUE, ci.type = "upper"))

est.list

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              0.6931472 1.6094379
#
#Estimated Parameter(s):          mean = 2.215905
#                                 sd   = 1.356291
#
#Estimation Method:               MLE
#
#Estimated Quantile(s):           90'th %ile = 3.954062
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 MLE Estimators
#
#Data:                            log(Manganese.ppb)
#
#Censoring Variable:              censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Confidence Interval for:         90'th %ile
#
```

```
#Assumed Sample Size:             25
#
#Confidence Interval Method:      Exact for
#                                 Complete Data
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL =     -Inf
#                                 UCL = 4.708904

# To estimate the 90th percentile on the original scale,
# we need to exponentiate the results
#-------------------------------------------------------
exp(est.list$quantiles)
#90'th %ile
#  52.14674

exp(est.list$interval$limits)
#     LCL      UCL
#  0.0000 110.9305

#----------

# Clean up
#---------
rm(est.list)
```

---

eqnpar *Estimate Quantiles of a Distribution Nonparametrically*

Description

Estimate quantiles of a distribution, and optionally create confidence intervals for them, without making any assumptions about the form of the distribution.

Usage

```
eqnpar(x, p = 0.5, type = 7, ci = FALSE, lcl.rank = NULL, ucl.rank = NULL,
  lb = -Inf, ub = Inf, ci.type = "two-sided",
  ci.method = "interpolate", digits = getOption("digits"),
  approx.conf.level = 0.95, min.coverage = TRUE, tol = 0)
```

Arguments

x	a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

p numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.

type an integer between 1 and 9 indicating which algorithm to use to estimate the quantile. The default value is type=7. See the help file for quantile for details.

ci logical scalar indicating whether to compute a confidence interval for the quantile. The default value is ci=FALSE.

lcl.rank, ucl.rank
positive integers indicating the ranks of the order statistics that are used for the lower and upper bounds of the confidence interval for the specified quantile. Both arguments must be integers between 1 and the number of non-missing values in x, and lcl.rank must be strictly less than ucl.rank. Setting values for lcl.rank and/or ucl.rank allows the user to bypass the automatic selection of order statistics. By default the value of these arguments is NULL, in which case order statistics are chosen based on the value of ci.type and ci.method. If only lcl.rank is supplied, a lower confidence interval is constructed. If only ucl.rank is supplied, an upper confidence interval is constructed. If both lcl.rank and ucl.rank are supplied, a two-sided confidence interval is constructed. These arguments are ignored if ci=FALSE.

lb, ub scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a confidence interval for a quantile from a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value. These arguments are ignored if ci=FALSE.

ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE, or lcl.rank and/or ucl.rank are supplied.

ci.method character string indicating the method to use to construct the confidence interval. The possible values are "interpolate" (the default), "exact", and "normal.approx". See the DETAILS section for more information on these methods. This argument is ignored if ci=FALSE, or lcl.rank and/or ucl.rank are supplied.

digits an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

approx.conf.level
a scalar between 0 and 1 indicating the desired confidence level of the confidence interval. The default value is 0.95. The true confidence level usually will not be exactly equal to approx.conf.level (see DETAILS). This argument is ignored if ci=FALSE, or lcl.rank and/or ucl.rank are supplied.

min.coverage for the case when ci.method="exact", a logical scalar indicating whether the confidence interval should have a minimum coverage at least as great as the value of the argument approx.conf.level. The default value is min.coverage=TRUE. This argument is ignored if ci=FALSE or ci.method is not equal to "exact".

tol for the case when ci.method="exact" and min.coverage=FALSE, a scalar between 0 and 1 indicating the maximum amount of coverage greater than the

value of approx.conf.level the user is willing to allow. The default value is tol=0.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Estimation

The function eqnpar calls the R function quantile to estimate quantiles.

Confidence Intervals

Let $x_1, x_2, \ldots, x_n$ denote a sample of n independent and identically distributed random variables from some arbitrary distribution. Furthermore, let $x_{(i)}$ denote the i'th order statistic for these n random variables. That is,

$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n)} \qquad (1)$$

Finally, let x_p denote the p'th quantile of the distribution, that is:

$$Pr(X < x_p) \leq p \qquad (2)$$

$$Pr(X \leq x_p) \geq p \qquad (3)$$

It can be shown (e.g., Conover, 1980, pp. 114-116) that for the i'th order statistic:

$$Pr[x_p < x_{(i)}] = F_{B(n,p)}[i-1];\ i = 1, 2, \ldots, n \qquad (4)$$

for a continuous distribution and

$$Pr[x_p < x_{(i)}] \leq F_{B(n,p)}[i-1];\ i = 1, 2, \ldots, n \qquad (5)$$

$$Pr[x_p \leq x_{(i)}] \geq F_{B(n,p)}[i-1];\ i = 1, 2, \ldots, n \qquad (6)$$

for a discrete distribution, where $F_{B(n,p)}[y]$ denotes the cumulative distribution function of a binomial random variable with parameters size=n and prob=p evaluated at y. These facts are used to construct confidence intervals for quantiles (see below).

Two-Sided Confidence Interval (ci.type="two-sided")

A two-sided nonparametric confidence interval for the p'th quantile is constructed as:

$$[x_{(r)}, x_{(s)}] \qquad (7)$$

where

$$1 \leq r \leq (n-1) \qquad (8)$$

$$2 \leq s \leq n \qquad (9)$$

$$r < s \qquad (10)$$

Note that the argument lcl.rank corresponds to r, and the argument ucl.rank corresponds to s.

This confidence interval has an associated confidence level that is at least as large as:

$$F_{B(n,p)}[s-1] - F_{B(n,p)}[r-1] \qquad (11)$$

for a discrete distribution and exactly equal to this value for a continuous distribution. This is because by Equations (4)-(6) above:

$$\begin{aligned} & Pr[x_{(r)} \leq x_p \leq x_{(s)}] \\ & = Pr[x_p \leq x_{(s)}] - Pr[x_p < x_{(r)}] \\ & \geq F_{B(n,p)}[s-1] - F_{B(n,p)}[r-1] \qquad (12) \end{aligned}$$

with equality if the distribution is continuous.

Exact Method (`ci.method="exact"`)
When `lcl.rank` (r) and `ucl.rank` (s) are not supplied by the user, and `ci.method="exact"`, r and s are initially chosen such that r is the smallest integer satisfying equation (13) below, and s is the largest integer satisfying equation (14) below:

$$F_{B(n,p)}[r-1] \geq \frac{\alpha}{2} \qquad (13)$$

$$F_{B(n,p)}[s-1] \leq 1 - \frac{\alpha}{2} \qquad (14)$$

where $\alpha = 1 - $`approx.conf.level`. The values of r and s are then each varied by ± 2 (with the restrictions $r \geq 1$, $s \leq n$, and $r < s$), and confidence levels computed for each of these combinations. If `min.coverage=TRUE`, the combination of r and s is selected that provides the closest coverage to `approx.conf.level`, with coverage greater than or equal to `approx.conf.level`. If `min.coverage=FALSE`, the combination of r and s is selected that provides the closest coverage to `approx.conf.level`, with coverage less than or equal to `approx.conf.level + tol`.

For this method, the confidence level associated with the confidence interval is exact if the underlying distribution is continuous.

Approximate Method (`ci.method="approx"`)
Here the term "Approximate" simply refers to the method of initially choosing the ranks for the lower and upper bounds. As for `ci.method="exact"`, the confidence level associated with the confidence interval is exact if the underlying distribution is continuous.

When `lcl.rank` (r) and `ucl.rank` (s) are not supplied by the user and `ci.method="normal.approx"`, r and s are initially chosen such that:

$$r = np - h \qquad (15)$$

$$s = np + h \qquad (16)$$

where

$$h = t_{n-1,1-\alpha/2}\sqrt{np(1-p)} \qquad (17)$$

and $t_{\nu,q}$ denotes the q'th quantile of Student's t-distribution with ν degrees of freedom, and $\alpha = 1 - $`approx.conf.level` (Conover, 1980, p. 112). With the restictions that $r \geq 1$ and $s \leq n$, r is rounded down to the nearest integer $r = r^* = floor(r)$ and s is rounded up to the nearest integer $s = s^* = ceiling(s)$. Again, with the restictions that $s \leq n$, if the confidence level using $s = s^* + 1$ and $r = r^*$ is less than or equal to `approx.conf.level`, then s is set to $s = s^* + 1$. Once this has been checked, with the restriction that $r \geq 1$, if the confidence level using the current value of s and $r = r^* - 1$ is less than or equal to `approx.conf.level`, then r is set to $r = r^* - 1$.

Interpolate Method (`ci.method="interpolate"`)
Let γ denote the desired confidence level associated with the confidence interval for the p'th quantile. Based on the work of Hettmansperger and Sheather (1986), Nyblom (1992) showed that if $[-\infty, x_{(w+1)}]$ is a one-sided upper confidence interval for the p'th quantile with associated confidence level γ_{w+1}, and $\gamma_{w+1} \geq \gamma \geq \gamma_w$, then the one-sided upper confidence interval

$$[-\infty, (1-\lambda)x_{(w)} + \lambda x_{(w+1)}] \qquad (18)$$

where

$$\lambda = \lambda(\beta, p, w, n)$$
$$= [1 + \frac{w(1-p)(\pi_{w+1} - \beta)}{(n-w)p(\beta - \pi_w)}]^{-1} \qquad (19)$$
$$\pi_w = F_{B(n,p)}[w-1] \qquad (20)$$
$$\beta = \gamma \qquad (21)$$

has associated confidence level approximately equal to γ for a wide range of distributions.

Thus, to construct an approximate two-sided confidence interval for the p'th quantile with confidence level γ, if $[x_{(r)}, x_{(s)}]$ has confidence level $\geq \gamma$ and $[x_{(r+1)}, x_{(s-1)}]$ has confidence level $\leq \gamma$, then the lower bound of the two-sided confidence interval is computed as:

$$(1-\lambda)x_{(r)} + \lambda x_{(r+1)} \qquad (21)$$

where

$$\beta = \alpha/2; \quad \alpha = 1 - \gamma \qquad (22)$$

and the upper bound of the two-sided confidence interval is computed as:

$$(1-\lambda)x_{(s-1)} + \lambda x_{(s)} \qquad (23)$$

where

$$\beta = 1 - \alpha/2 \qquad (24)$$

The values of r and s in Equations (21) and (23) are computed by using `ci.method="exact"` with the argument `min.coverage=TRUE`.

One-Sided Lower Confidence Interval (`ci.type="lower"`)
A one-sided lower nonparametric confidence interval for the p'th quantile is constructed as:

$$[x_{(r)}, ub] \qquad (25)$$

where ub denotes the value of the `ub` argument (the user-supplied upper bound).

Exact Method (`ci.method="exact"`)
When `lcl.rank` (r) is not supplied by the user, and `ci.method="exact"`, r is initially chosen such that it is the smallest integer satisfying the following equation:

$$F_{B(n,p)}[r-1] \geq \alpha \qquad (26)$$

where $\alpha = 1 - $`approx.conf.level`. The value of r is varied by ± 2 (with the restrictions $r \geq 1$ and $r \leq n$), and confidence levels computed for each of these combinations. If `min.coverage=TRUE`,

the value of r is selected that provides the closest coverage to `approx.conf.level`, with coverage greater than or equal to `approx.conf.level`. If `min.coverage=FALSE`, the value of r is selected that provides the closest coverage to `approx.conf.level`, with coverage less than or equal to `approx.conf.level + tol`.

For this method, the confidence level associated with the confidence interval is exact if the underlying distribution is continuous.

Approximate Method (`ci.method="approx"`)
When `lcl.rank` (r) is not supplied by the user and `ci.method="normal.approx"`, r is initially chosen such that

$$r = np - t_{n-1,1-\alpha}\sqrt{np(1-p)} \qquad (27)$$

With the restriction that $r \geq 1$ and $r \leq n$, if `p` is less than 0.5 then r is rounded up to the nearest integer, otherwise it is rounded down to the nearest integer. Denote this value by r^*. With the restriction that $r \geq 1$, if the confidence level using $r^* - 1$ is less than or equal to `approx.conf.level`, then r is set to $r = r^* - 1$.

Interpolate Method (`ci.method="interpolate"`)
Let γ denote the desired confidence level associated with the confidence interval for the p'th quantile. To construct an approximate one-sided lower confidence interval for the p'th quantile with confidence level γ, if $[x_{(r)}, ub]$ has confidence level $\geq \gamma$ and $[x_{(r+1)}, ub]$ has confidence level $\leq \gamma$, then the lower bound of the confidence interval is computed as:

$$(1-\lambda)x_{(r)} + \lambda x_{(r+1)} \qquad (28)$$

where

$$\beta = \alpha; \quad \alpha = 1 - \gamma \qquad (29)$$

The value of r in Equation (28) is computed by using `ci.method="exact"` with the arguments `ci.type="lower"` and `min.coverage=TRUE`.

One-Sided Upper Confidence Interval (`ci.type="upper"`)
A one-sided upper nonparametric confidence interval for the p'th quantile is constructed as:

$$[lb, x_{(s)}] \qquad (30)$$

where lb denotes the value of the `lb` argument (the user-supplied lower bound).

Exact Method (`ci.method="exact"`)
When `ucl.rank` (s) is not supplied by the user, and `ci.method="exact"`, s is initially chosen such that it is the largest integer satisfying the following equation:

$$F_{B(n,p)}[s-1] \leq 1 - \alpha \qquad (31)$$

where $\alpha = 1 - $`approx.conf.level`. The value of s is varied by ± 2 (with the restrictions $s \geq 1$ and $s \leq n$), and confidence levels computed for each of these combinations. If `min.coverage=TRUE`, the value of s is selected that provides the closest coverage to `approx.conf.level`, with coverage greater than or equal to `approx.conf.level`. If `min.coverage=FALSE`, the value of s is selected

that provides the closest coverage to approx.conf.level, with coverage less than or equal to approx.conf.level + tol.

For this method, the confidence level associated with the confidence interval is exact if the underlying distribution is continuous.

Approximate Method (ci.method="approx")
When ucl.rank (s) is not supplied by the user and ci.method="normal.approx", s is initially chosen such that

$$s = np + t_{n-1,1-\alpha}\sqrt{np(1-p)} \qquad (31)$$

With the restriction that $s \geq 1$ and $s \leq n$, if p is greater than 0.5 then s is rounded down to the nearest integer, otherwise it is rounded up to the nearest integer. Denote this value by s^*. With the restriction that $s \leq n$, if the confidence level using s^*+1 is less than or equal to approx.conf.level, then s is set to $s = s^* + 1$.

For this method, the confidence level associated with the confidence interval is exact if the underlying distribution is continuous.

Interpolate Method (ci.method="interpolate")
Let γ denote the desired confidence level associated with the confidence interval for the p'th quantile. To construct an approximate one-sided upper confidence interval for the p'th quantile with confidence level γ, if $[lb, x_{(s)}]$ has confidence level $\geq \gamma$ and $[lb, x_{(s-1)}]$ has confidence level $\leq \gamma$, then the upper bound of the confidence interval is computed as:

$$(1-\lambda)x_{(s-1)} + \lambda x_{(s)} \qquad (32)$$

where

$$\beta = \gamma \qquad (33)$$

The value of s in Equation (32) is computed by using ci.method="exact" with the arguments ci.type = "upper", and min.coverage=TRUE.

Note on Value of Confidence Level
Because of the discrete nature of order statistics, when ci.method="exact" or ci.method="normal.approx", the value of the confidence level returned by eqnpar will usually differ from the desired confidence level indicated by the value of the argument approx.conf.level. When ci.method="interpolate", eqnpar returns for the confidence level the value of the argument approx.conf.level. Nyblom (1992) and Hettmasperger and Sheather (1986) have shown that the Interpolate method produces confidence intervals with confidence levels quite close to the assumed confidence level for a wide range of distributions.

Value

a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) note that England has water quality limits based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. They also note that the U.S. EPA has specifications for air quality monitoring, aquatic standards on toxic chemicals, and maximum daily limits for industrial effluents that are all based on percentiles. Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

It can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to an upper β-content tolerance interval with coverage $100p\%$ and confidence level $100(1-\alpha)\%$. Also, a lower confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to a lower β-content tolerance interval with coverage $100(1-p)\%$ and confidence level $100(1-\alpha)\%$. See the help file for `tolIntNpar` for more information on nonparametric tolerance intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

The author is grateful to Michael H?hle, Department of Mathematics, Stockholm University (http://www2.math.su.se/~hoehle) for making me aware of the work of Nyblom (1992), and for suggesting improvements to the algorithm that was used in **EnvStats** Version 2.1.1 to construct a confidence interval when `ci.method="exact"`.

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, pp.132-136.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, pp.88-90.

Hettmansperger, T.P., and Sheather, S.J. (1986). Confidence Intervals Based on Interpolated Order Statistics. *Statistics & Probability Letters*, **4**, 75–79.

Nyblom, J. (1992). Note on Interpolated Order Statistics. *Statistics & Probability Letters*, **14**, 129–131.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`quantile`, `tolIntNpar`, Estimating Distribution Quantiles, Tolerance Intervals, `estimate.object`.

Examples

```
# The data frame ACE.13.TCE.df contains observations on
# Trichloroethylene (TCE) concentrations (mg/L) at
# 10 groundwater monitoring wells before and after remediation.
#
# Compute the median concentration for each period along with
# a 95% confidence interval for the median.
#
# Before remediation:  20.3 [8.8, 35.9]
# After  remediation:   2.5 [0.8,  5.9]

with(ACE.13.TCE.df,
  eqnpar(TCE.mg.per.L[Period=="Before"], ci = TRUE))

#Results of Distribution Parameter Estimation
#--------------------------------------------

#Assumed Distribution:            None

#Estimated Quantile(s):           Median = 20.3

#Quantile Estimation Method:      Nonparametric

#Data:                            TCE.mg.per.L[Period == "Before"]

#Sample Size:                     10

#Confidence Interval for:         50'th %ile

#Confidence Interval Method:      interpolate (Nyblom, 1992)

#Confidence Interval Type:        two-sided

#Confidence Level:                95%

#Confidence Limit Rank(s):        2 9
#                                 3 8

#Confidence Interval:             LCL =  8.804775
#                                 UCL = 35.874775

#----------

with(ACE.13.TCE.df, eqnpar(TCE.mg.per.L[Period=="After"], ci = TRUE))

#Results of Distribution Parameter Estimation
#--------------------------------------------

#Assumed Distribution:            None

#Estimated Quantile(s):           Median = 2.48
```

```
#Quantile Estimation Method:      Nonparametric

#Data:                            TCE.mg.per.L[Period == "After"]

#Sample Size:                     10

#Confidence Interval for:         50'th %ile

#Confidence Interval Method:      interpolate (Nyblom, 1992)

#Confidence Interval Type:        two-sided

#Confidence Level:                95%

#Confidence Limit Rank(s):        2 9
#                                 3 8

#Confidence Interval:             LCL = 0.7810901
#                                 UCL = 5.8763063

#==========

# Generate 20 observations from a cauchy distribution with parameters
# location=0, scale=1.  The true 75th percentile of this distribution is 1.
# Use eqnpar to estimate the 75th percentile and construct a 90% confidence interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rcauchy(20, location = 0, scale = 1)

#----------------------------------------------------------
# First, use the default method, ci.method="interpolate"
#----------------------------------------------------------

eqnpar(dat, p = 0.75, ci = TRUE, approx.conf.level = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           75'th %ile = 1.524903
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         75'th %ile
#
#Confidence Interval Method:      interpolate (Nyblom, 1992)
#
```

```
#Confidence Interval Type:        two-sided
#
#Confidence Level:                90%
#
#Confidence Limit Rank(s):        12 19
#                                 13 18
#
#Confidence Interval:             LCL = 0.8191423
#                                 UCL = 2.1215570

#----------

#------------------------------------------------------------
# Now use ci.method="exact".
# Note that the returned confidence level is greater than 90%.
#------------------------------------------------------------

eqnpar(dat, p = 0.75, ci = TRUE, approx.conf.level = 0.9,
  ci.method = "exact")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           75'th %ile = 1.524903
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         75'th %ile
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                93.47622%
#
#Confidence Limit Rank(s):        12 19
#
#Confidence Interval:             LCL = 0.7494692
#                                 UCL = 2.2156601

#----------

#------------------------------------------------------------
# Now use ci.method="exact" with min.coverage=FALSE.
# Note that the returned confidence level is less than 90%.
#------------------------------------------------------------
```

```
eqnpar(dat, p = 0.75, ci = TRUE, approx.conf.level = 0.9,
  ci.method = "exact", min.coverage = FALSE, )

#Results of Distribution Parameter Estimation
#-----------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           75'th %ile = 1.524903
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         75'th %ile
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                89.50169%
#
#Confidence Limit Rank(s):        13 20
#
#Confidence Interval:             LCL = 1.018038
#                                 UCL = 5.002399

#----------

#-------------------------------------------------------------
# Now supply our own bounds for the confidence interval.
# The first example above based on the Interpolate method
# used lcl.rank=12, ucl.rank=19 and lcl.rank=13, ucl.rank=18
# and interpolated between these two confidence intervals.
# Here we will specify lcl.rank=13 and ucl.rank=18.  The
# resulting confidence level is 81%.
#-------------------------------------------------------------

eqnpar(dat, p = 0.75, ci = TRUE, lcl.rank = 13, ucl.rank = 18)

#Results of Distribution Parameter Estimation
#-----------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           75'th %ile = 1.524903
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            dat
#
```

```
#Sample Size:                     20
#
#Confidence Interval for:         75'th %ile
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                80.69277%
#
#Confidence Limit Rank(s):        13 18
#
#Confidence Interval:             LCL = 1.018038
#                                 UCL = 2.071172

#----------

# Clean up
rm(dat)

#==========

# Modify Example 17-4 on page 17-21 of USEPA (2009).  This example uses
# copper concentrations (ppb) from 3 background wells to set an upper
# limit for 2 compliance wells.  Here we will attempt to compute an upper
# 95% confidence interval for the 95'th percentile of the distribution of
# copper concentrations in the background wells.
#
# The data are stored in EPA.92c.copper2.df.
#
# Note that even though these data are Type I left singly censored,
# it is still possible to compute an estimate of the 95'th percentile.

EPA.92c.copper2.df
#   Copper.orig Copper Censored Month Well  Well.type
#1           <5    5.0     TRUE     1    1 Background
#2           <5    5.0     TRUE     2    1 Background
#3          7.5    7.5    FALSE     3    1 Background
#...
#9          9.2    9.2    FALSE     1    2 Background
#10          <5    5.0     TRUE     2    2 Background
#11          <5    5.0     TRUE     3    2 Background
#...
#17          <5    5.0     TRUE     1    3 Background
#18         5.4    5.4    FALSE     2    3 Background
#19         6.7    6.7    FALSE     3    3 Background
#...
#29         6.2    6.2    FALSE     5    4 Compliance
#30          <5    5.0     TRUE     6    4 Compliance
#31         7.8    7.8    FALSE     7    4 Compliance
#...
#38          <5    5.0     TRUE     6    5 Compliance
#39         5.6    5.6    FALSE     7    5 Compliance
```

```
#40          <5    5.0     TRUE     8    5 Compliance

# Because of the small sample size of n=24 observations, it is not possible
# to create a nonparametric confidence interval for the 95th percentile
# that has an associated confidence level of 95%.  If we tried to do this,
# we would get an error message:

# with(EPA.92c.copper2.df,
#    eqnpar(Copper[Well.type=="Background"], p = 0.95, ci = TRUE, lb = 0,
#      ci.type = "upper", approx.conf.level = 0.95))
#
#Error in ci.qnpar.interpolate(x = c(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,  :
#  Minimum coverage of 0.95 is not possible with the given sample size.

# So instead, we will use ci.method="exact" with min.coverage=FALSE
# to construct the confidence interval.  Note that the associated
# confidence level is only 71%.

with(EPA.92c.copper2.df,
   eqnpar(Copper[Well.type=="Background"], p = 0.95, ci = TRUE,
      ci.method = "exact", min.coverage = FALSE,
      ci.type = "upper", lb = 0,
      approx.conf.level = 0.95))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           95'th %ile = 7.925
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            Copper[Well.type == "Background"]
#
#Sample Size:                     24
#
#Confidence Interval for:         95'th %ile
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                70.8011%
#
#Confidence Limit Rank(s):        NA 24
#
#Confidence Interval:             LCL = 0.0
#                                 UCL = 9.2

#----------

# For the above example, the true confidence level is 71% instead of 95%.
```

```
# This is a function of the small sample size.  In fact, as Example 17-4 on
# pages 17-21 of USEPA (2009) shows, the largest quantile for which you can
# construct a nonparametric confidence interval that will have associated
# confidence level of 95% is the 88'th percentile:

with(EPA.92c.copper2.df,
  eqnpar(Copper[Well.type=="Background"], p = 0.88, ci = TRUE,
    ci.type = "upper", lb = 0, ucl.rank = 24,
    approx.conf.level = 0.95))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           88'th %ile = 6.892
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            Copper[Well.type == "Background"]
#
#Sample Size:                     24
#
#Confidence Interval for:         88'th %ile
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        upper
#
#Confidence Level:                95.3486%
#
#Confidence Limit Rank(s):        NA 24
#
#Confidence Interval:             LCL = 0.0
#                                 UCL = 9.2

#==========

# Reproduce Example 21-6 on pages 21-21 to 21-22 of USEPA (2009).
# Use 12 measurements of nitrate (mg/L) at a well used for drinking water
# to determine with 95% confidence whether or not the infant-based, acute
# risk standard of 10 mg/L has been violated.  Assume that the risk
# standard represents an upper 95'th percentile limit on nitrate
# concentrations.  So what we need to do is construct a one-sided
# lower nonparametric confidence interval for the 95'th percentile
# that has associated confidence level of no more than 95%, and we will
# compare the lower confidence limit with the MCL of 10 mg/L.
#
# The data for this example are stored in EPA.09.Ex.21.6.nitrate.df.

# Look at the data:
#------------------
```

```
EPA.09.Ex.21.6.nitrate.df
#   Sampling.Date       Date Nitrate.mg.per.l.orig Nitrate.mg.per.l Censored
#1      7/28/1999 1999-07-28                  <5.0              5.0     TRUE
#2       9/3/1999 1999-09-03                  12.3             12.3    FALSE
#3     11/24/1999 1999-11-24                  <5.0              5.0     TRUE
#4       5/3/2000 2000-05-03                  <5.0              5.0     TRUE
#5      7/14/2000 2000-07-14                   8.1              8.1    FALSE
#6     10/31/2000 2000-10-31                  <5.0              5.0     TRUE
#7     12/14/2000 2000-12-14                    11             11.0    FALSE
#8      3/27/2001 2001-03-27                  35.1             35.1    FALSE
#9      6/13/2001 2001-06-13                  <5.0              5.0     TRUE
#10     9/16/2001 2001-09-16                  <5.0              5.0     TRUE
#11    11/26/2001 2001-11-26                   9.3              9.3    FALSE
#12     3/2/2002 2002-03-02                  10.3             10.3    FALSE

# Determine what order statistic to use for the lower confidence limit
# in order to achieve no more than 95% confidence.
#------------------------------------------------------------------------

conf.levels <- ciNparConfLevel(n = 12, p = 0.95, lcl.rank = 1:12,
  ci.type = "lower")
names(conf.levels) <- 1:12

round(conf.levels, 2)
#   1    2    3    4    5    6    7    8    9   10   11   12
#1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.98 0.88 0.54

# Using the 11'th largest observation for the lower confidence limit
# yields a confidence level of 88%.  Using the 10'th largest
# observation yields a confidence level of 98%.  The example in
# USEPA (2009) uses the 10'th largest observation.
#
# The 10'th largest observation is 11 mg/L which exceeds the
# MCL of 10 mg/L, so there is evidence of contamination.
#------------------------------------------------------------------

with(EPA.09.Ex.21.6.nitrate.df,
  eqnpar(Nitrate.mg.per.l, p = 0.95, ci = TRUE,
    ci.type = "lower", lcl.rank = 10))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Estimated Quantile(s):           95'th %ile = 22.56
#
#Quantile Estimation Method:      Nonparametric
#
#Data:                            Nitrate.mg.per.l
#
#Sample Size:                     12
#
```

```
#Confidence Interval for:         95'th %ile
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        lower
#
#Confidence Level:                98.04317%
#
#Confidence Limit Rank(s):        10 NA
#
#Confidence Interval:             LCL =  11
#                                 UCL = Inf

#==========

# Clean up
#---------

rm(conf.levels)
```

---

eqpareto *Estimate Quantiles of a Pareto Distribution*

Description

Estimate quantiles of a Pareto distribution.

Usage

```
eqpareto(x, p = 0.5, method = "mle", plot.pos.con = 0.375, digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a Pareto distribution (e.g., epareto). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method of estimating the distribution parameters. Possible values are "mle" (maximum likelihood; the default), and "lse" (least-squares). See the DETAILS section of the help file for epareto for more information on these estimation methods.
plot.pos.con	numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the values of the empirical cdf. The default value is plot.pos.con=0.375. This argument is used only when method="lse".
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqpareto returns estimated quantiles as well as estimates of the location and scale parameters.

Quantiles are estimated by 1) estimating the location and scale parameters by calling epareto, and then 2) calling the function qpareto and using the estimated values for location and scale.

Value

If x is a numeric vector, eqpareto returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqpareto returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto's law, which states that the number of persons N having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where θ denotes Pareto's constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the "location" parameter η, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the r'th moment only exists if $r < \theta$.

The Pareto distribution is related to the exponential distribution and logistic distribution as follows. Let X denote a Pareto random variable with location=η and shape=θ. Then $log(X/\eta)$ has an exponential distribution with parameter rate=θ, and $-log\{[(X/\eta)^{\theta}] - 1\}$ has a logistic distribution with parameters location=0 and scale=1.

The Pareto distribution has a very long right-hand tail. It is often applied in the study of socioeconomic data, including the distribution of income, firm size, population, and stock price fluctuations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

epareto, Pareto, estimate.object.

Examples

```
# Generate 30 observations from a Pareto distribution with
# parameters location=1 and shape=1 then estimate the parameters
# and the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpareto(30, location = 1, shape = 1)
eqpareto(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Pareto
#
#Estimated Parameter(s):          location = 1.009046
#                                 shape    = 1.079850
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 8.510708
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     30

#----------

# Clean up
#---------
rm(dat)
```

`eqpois` *Estimate Quantiles of a Poisson Distribution*

Description

Estimate quantiles of an Poisson distribution, and optionally contruct a confidence interval for a quantile.

Usage

```
eqpois(x, p = 0.5, method = "mle/mme/mvue", ci = FALSE, ci.method = "exact",
  ci.type = "two-sided", conf.level = 0.95, digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes an Poisson distribution (e.g., epois). If ci=TRUE then x must be a numeric vector of observations. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. When ci=TRUE, p must be a scalar. The default value is p=0.5.
method	character string specifying the method to use to estimate the mean. Currently the only possible value is "mle/mme/mvue" (maximum likelihood/method of moments/minimum variance unbiased; the default). See the DETAILS section of the help file for epois for more information.
ci	logical scalar indicating whether to compute a confidence interval for the specified quantile. The default value is ci=FALSE.
ci.method	character string indicating what method to use to construct the confidence interval for the quantile. The only possible value is "exact" (exact method; the default). See the DETAILS section for more information.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.
conf.level	a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is conf.level=0.95. This argument is ignored if ci=FALSE.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqpois returns estimated quantiles as well as the estimate of the mean parameter.

Estimation

Let X denote a Poisson random variable with parameter lambda=λ. Let $x_{p|\lambda}$ denote the p'th quantile of the distribution. That is,

$$Pr(X < x_{p|\lambda}) \leq p \leq Pr(X \leq x_{p|\lambda}) \quad (1)$$

Note that due to the discrete nature of the Poisson distribution, there will be several values of p associated with one value of X. For example, for $\lambda = 2$, the value 1 is the p'th quantile for any value of p between 0.14 and 0.406.

Let $\underline{x}$ denote a vector of n observations from a Poisson distribution with parameter lambda=λ. The p'th quantile is estimated as the p'th quantile from a Poisson distribution assuming the true value of λ is equal to the estimated value of λ. That is:

$$\hat{x}_{p|\lambda} = x_{p|\lambda=\hat{\lambda}} \quad (2)$$

where

$$\hat{\lambda} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (3)$$

Because the estimator in equation (3) is the maximum likelihood estimator of λ (see the help file for epois), the estimated quantile is the maximum likelihood estimator.

Quantiles are estimated by 1) estimating the mean parameter by calling epois, and then 2) calling the function qpois and using the estimated value for the mean parameter.

Confidence Intervals

It can be shown (e.g., Conover, 1980, pp.119-121) that an upper confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to an upper β-content tolerance interval with coverage $100p\%$ and confidence level $100(1-\alpha)\%$. Also, a lower confidence interval for the p'th quantile with confidence level $100(1-\alpha)\%$ is equivalent to a lower β-content tolerance interval with coverage $100(1-p)\%$ and confidence level $100(1-\alpha)\%$.

Thus, based on the theory of tolerance intervals for a Poisson distribution (see tolIntPois), if ci.type="upper", a one-sided upper $100(1-\alpha)\%$ confidence interval for the p'th quantile is constructed as:

$$[0, x_{p|\lambda=UCL}] \quad (4)$$

where UCL denotes the upper $100(1-\alpha)\%$ confidence limit for λ (see the help file for epois for information on how UCL is computed).

Similarly, if ci.type="lower", a one-sided lower $100(1-\alpha)\%$ confidence interval for the p'th quantile is constructed as:

$$[x_{p|\lambda=LCL}, \infty] \quad (5)$$

where LCL denotes the lower $100(1-\alpha)\%$ confidence limit for λ (see the help file for epois for information on how LCL is computed).

Finally, if ci.type="two-sided", a two-sided $100(1-\alpha)\%$ confidence interval for the p'th quantile is constructed as:

$$[x_{p|\lambda=LCL}, x_{p|\lambda=UCL}] \quad (6)$$

where LCL and UCL denote the two-sided lower and upper $100(1-\alpha)\%$ confidence limits for λ (see the help file for epois for information on how LCL and UCL are computed).

Value

If x is a numeric vector, eqpois returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqpois returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

Percentiles are sometimes used in environmental standards and regulations. For example, Berthouex and Brown (2002, p.71) state:

The U.S. EPA has specifications for air quality monitoring that are, in effect, percentile limitations. ... The U.S. EPA has provided guidance for setting aquatic standards on toxic chemicals that require estimating 99th percentiles and using this statistic to make important decisions about monitoring and compliance. They have also used the 99th percentile to establish maximum daily limits for industrial effluents (e.g., pulp and paper).

Given the importance of these quantities, it is essential to characterize the amount of uncertainty associated with the estimates of these quantities. This is done with confidence intervals.

The Poisson distribution is named after Poisson, who derived this distribution as the limiting distribution of the binomial distribution with parameters `size`=N and `prob`=p, where N tends to infinity, p tends to 0, and Np stays constant.

In this context, the Poisson distribution was used by Bortkiewicz (1898) to model the number of deaths (per annum) from kicks by horses in Prussian Army Corps. In this case, p, the probability of death from this cause, was small, but the number of soldiers exposed to this risk, N, was large.

The Poisson distribution has been applied in a variety of fields, including quality control (modeling number of defects produced in a process), ecology (number of organisms per unit area), and queueing theory. Gibbons (1987b) used the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical concentration (in ppb).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Berthouex, P.M., and I. Hau. (1991). Difficulties Related to Using Extreme Percentiles for Water Quality Regulations. *Research Journal of the Water Pollution Control Federation* **63**(6), 873–879.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York, Chapter 3.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Gibbons, R.D. (1987b). Statistical Models for the Analysis of Volatile Organic Compounds in Waste Disposal Sites. *Ground Water* **25**, 572-580.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 4.

Pearson, E.S., and H.O. Hartley, eds. (1970). *Biometrika Tables for Statisticians, Volume 1*. Cambridge Universtiy Press, New York, p.81.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`epois`, Poisson, `estimate.object`.

Examples

```
# Generate 20 observations from a Poisson distribution with parameter
# lambda=2.  The true 90'th percentile of this distribution is 4 (actually,
# 4 is the p'th quantile for any value of p between 0.86 and 0.947).
# Here we will use eqpois to estimate the 90'th percentile and construct a
# two-sided 95% confidence interval for this percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpois(20, lambda = 2)
eqpois(dat, p = 0.9, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Poisson
#
#Estimated Parameter(s):          lambda = 1.8
#
#Estimation Method:               mle/mme/mvue
#
#Estimated Quantile(s):           90'th %ile = 4
#
#Quantile Estimation Method:      mle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         90'th %ile
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 3
#                                 UCL = 5

# Clean up
#---------
rm(dat)
```

---

equnif *Estimate Quantiles of a Uniform Distribution*

Description

Estimate quantiles of a uniform distribution.

Usage

```
equnif(x, p = 0.5, method = "mle", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a uniform distribution (e.g., eunif). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method of estimating the distribution parameters. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for eunif for more information on these estimation methods.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function equnif returns estimated quantiles as well as estimates of the location and scale parameters.

Quantiles are estimated by 1) estimating the location and scale parameters by calling eunif, and then 2) calling the function qunif and using the estimated values for location and scale.

Value

If x is a numeric vector, equnif returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, equnif returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The uniform distribution (also called the rectangular distribution) with parameters min and max takes on values on the real line between min and max with equal probability. It has been used to represent the distribution of round-off errors in tabulated values. Another important application is that the distribution of the cumulative distribution function (cdf) of any kind of continuous random variable follows a uniform distribution with parameters min=0 and max=1.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

eunif, Uniform, estimate.object.

Examples

```
# Generate 20 observations from a uniform distribution with parameters
# min=-2 and max=3, then estimate the parameters via maximum likelihood
# and estimate the 90th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- runif(20, min = -2, max = 3)
equnif(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Uniform
#
#Estimated Parameter(s):          min = -1.574529
#                                 max =  2.837006
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 2.395852
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     20

#----------
# Clean up

rm(dat)
```

---

eqweibull *Estimate Quantiles of a Weibull Distribution*

Description

Estimate quantiles of a Weibull distribution.

Usage

```
eqweibull(x, p = 0.5, method = "mle", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a Weibull distribution (e.g., eweibull). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method of estimating the distribution parameters. Possible values are "mle" (maximum likelihood; the default), "mme" (methods of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for eweibull for more information.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqweibull returns estimated quantiles as well as estimates of the shape and scale parameters.

Quantiles are estimated by 1) estimating the shape and scale parameters by calling eweibull, and then 2) calling the function qweibull and using the estimated values for shape and scale.

Value

If x is a numeric vector, eqweibull returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqweibull returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The Weibull distribution is named after the Swedish physicist Waloddi Weibull, who used this distribution to model breaking strengths of materials. The Weibull distribution has been extensively applied in the fields of reliability and quality control.

The exponential distribution is a special case of the Weibull distribution: a Weibull random variable with parameters shape=1 and scale=β is equivalent to an exponential random variable with parameter rate=$1/\beta$.

The Weibull distribution is related to the Type I extreme value (Gumbel) distribution as follows: if X is a random variable from a Weibull distribution with parameters `shape`=α and `scale`=β, then

$$Y = -log(X) \quad (10)$$

is a random variable from an extreme value distribution with parameters `location`=$-log(\beta)$ and `scale`=$1/\alpha$.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

`eweibull`, Weibull, Exponential, EVD, `estimate.object`.

Examples

```
# Generate 20 observations from a Weibull distribution with parameters
# shape=2 and scale=3, then estimate the parameters via maximum likelihood,
# and estimate the 90'th percentile.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rweibull(20, shape = 2, scale = 3)
eqweibull(dat, p = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Weibull
#
#Estimated Parameter(s):          shape = 2.673098
#                                 scale = 3.047762
#
#Estimation Method:               mle
#
#Estimated Quantile(s):           90'th %ile = 4.163755
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mle Estimators
#
#Data:                            dat
#
#Sample Size:                     20
```

```
#----------

# Clean up
#---------
rm(dat)
```

`eqzmlnorm` *Estimate Quantiles of a Zero-Modified Lognormal (Delta) Distribution*

Description

Estimate quantiles of a zero-modified lognormal distribution or a zero-modified lognormal distribution (alternative parameterization).

Usage

```
eqzmlnorm(x, p = 0.5, method = "mvue", digits = 0)

eqzmlnormAlt(x, p = 0.5, method = "mvue", digits = 0)
```

Arguments

`x`	a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a zero-modified lognormal distribution. For `eqzmlnorm`, if `x` is an object, it must be the result of calling `ezmlnorm`, not `ezmlnormAlt`. For `eqzmlnormAlt`, if `x` is an object, it must be the result of calling `ezmlnormAlt`, not `ezmlnorm`. If `x` is a numeric vector, missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`p`	numeric vector of probabilities for which quantiles will be estimated. All values of `p` must be between 0 and 1. When `ci=TRUE`, `p` must be a scalar. The default value is `p=0.5`.
`method`	character string specifying the method of estimation. The only possible value is `"mvue"` (minimum variance unbiased; the default). See the DETAILS section of the help file for `ezmlnorm` for more information.
`digits`	an integer indicating the number of decimal places to round to when printing out the value of `100*p`. The default value is `digits=0`.

Details

The functions `eqzmlnorm` and `eqzmlnormAlt` return estimated quantiles as well as estimates of the distribution parameters.

Quantiles are estimated by:

1. estimating the distribution parameters by calling ezmlnorm or ezmlnormAlt, and then
2. calling the function qzmlnorm or qzmlnormAlt and using the estimated distribution parameters.

Value

If x is a numeric vector, eqzmlnorm and eqzmlnormAlt return a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqzmlnorm and eqzmlnormAlt return a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as "Below Detection Limit" (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J. (1955). On the Distribution of a Positive Random Variable Having a Discrete Probability Mass at the Origin. *Journal of the American Statistical Association* **50**, 901–908.

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special reference to its uses in economics)*. Cambridge University Press, London. pp.94-99.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, pp.47–51.

Gibbons, RD., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley and Sons, Hoboken, NJ.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135–146.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R*. Second Edition. John Wiley and Sons, Hoboken, NJ, Chapter 1.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, p.312.

Owen, W., and T. DeRouen. (1980). Estimation of the Mean for Lognormal Data Containing Zeros and Left-Censored Values, with Applications to the Measurement of Worker Exposure to Air Contaminants. *Biometrics* **36**, 707–719.

USEPA (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, Permits and State Programs Division, US Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

ezmlnorm, Zero-Modified Lognormal, ezmlnormAlt, Zero-Modified Lognormal (Alternative Parameterization), Zero-Modified Normal, Lognormal.

Examples

```
# Generate 100 observations from a zero-modified lognormal (delta)
# distribution with mean=2, cv=1, and p.zero=0.5, then estimate the
# parameters and also the 80'th and 90'th percentiles.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rzmlnormAlt(100, mean = 2, cv = 1, p.zero = 0.5)
eqzmlnormAlt(dat, p = c(0.8, 0.9))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Zero-Modified Lognormal (Delta)
#
#Estimated Parameter(s):          mean         = 1.9604561
#                                 cv           = 0.9169411
#                                 p.zero       = 0.4500000
#                                 mean.zmlnorm = 1.0782508
#                                 cv.zmlnorm   = 1.5307175
#
#Estimation Method:               mvue
#
#Estimated Quantile(s):           80'th %ile = 1.897451
#                                 90'th %ile = 2.937976
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mvue Estimators
#
#Data:                            dat
#
#Sample Size:                     100

#----------
```

```
# Compare the estimated quatiles with the true quantiles

qzmlnormAlt(mean = 2, cv = 1, p.zero = 0.5, p = c(0.8, 0.9))
#[1] 1.746299 2.849858

#----------

# Clean up
rm(dat)
```

---

eqzmnorm *Estimate Quantiles of a Zero-Modified Normal Distribution*

Description

Estimate quantiles of a zero-modified normal distribution.

Usage

```
eqzmnorm(x, p = 0.5, method = "mvue", digits = 0)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a zero-modified normal distribution (e.g., ezmnorm). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
p	numeric vector of probabilities for which quantiles will be estimated. All values of p must be between 0 and 1. The default value is p=0.5.
method	character string specifying the method of estimating the disribution parameters. Currently, the only possible value is "mvue" (minimum variance unbiased; the default). See the DETAILS section of the help file for ezmnorm for more information.
digits	an integer indicating the number of decimal places to round to when printing out the value of 100*p. The default value is digits=0.

Details

The function eqzmnorm returns estimated quantiles as well as estimates of the distribution parameters.

Quantiles are estimated by 1) estimating the distribution parameters by calling ezmnorm, and then 2) calling the function qzmnorm and using the estimated values for the distribution parameters.

Value

If x is a numeric vector, eqzmnorm returns a list of class "estimate" containing the estimated quantile(s) and other information. See estimate.object for details.

If x is the result of calling an estimation function, eqzmnorm returns a list whose class is the same as x. The list contains the same components as x, as well as components called quantiles and quantile.method.

Note

The zero-modified normal distribution is sometimes used to model chemical concentrations for which some observations are reported as "Below Detection Limit". See, for example USEPA (1992c, pp.27-34). In most cases, however, the zero-modified lognormal (delta) distribution will be more appropriate, since chemical concentrations are bounded below at 0 (e.g., Gilliom and Helsel, 1986; Owen and DeRouen, 1980).

Once you estimate the parameters of the zero-modified normal distribution, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with a confidence interval.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J. (1955). On the Distribution of a Positive Random Variable Having a Discrete Probability Mass at the Origin. *Journal of the American Statistical Association* **50**, 901–908.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135–146.

Owen, W., and T. DeRouen. (1980). Estimation of the Mean for Lognormal Data Containing Zeros and Left-Censored Values, with Applications to the Measurement of Worker Exposure to Air Contaminants. *Biometrics* **36**, 707–719.

USEPA (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, Permits and State Programs Division, US Environmental Protection Agency, Washington, D.C.

See Also

ZeroModifiedNormal, Normal, ezmlnorm, ZeroModifiedLognormal, estimate.object.

Examples

```
# Generate 100 observations from a zero-modified normal distribution
# with mean=4, sd=2, and p.zero=0.5, then estimate the parameters and
# the 80th and 90th percentiles.
# (Note: the call to set.seed simply allows you to reproduce this example.)
```

```
set.seed(250)
dat <- rzmnorm(100, mean = 4, sd = 2, p.zero = 0.5)
eqzmnorm(dat, p = c(0.8, 0.9))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Zero-Modified Normal
#
#Estimated Parameter(s):          mean        = 4.037732
#                                 sd          = 1.917004
#                                 p.zero      = 0.450000
#                                 mean.zmnorm = 2.220753
#                                 sd.zmnorm   = 2.465829
#
#Estimation Method:               mvue
#
#Estimated Quantile(s):           80'th %ile = 4.706298
#                                 90'th %ile = 5.779250
#
#Quantile Estimation Method:      Quantile(s) Based on
#                                 mvue Estimators
#
#Data:                            dat
#
#Sample Size:                     100

#----------

# Compare the estimated quantiles with the true quantiles

qzmnorm(mean = 4, sd = 2, p.zero = 0.5, p = c(0.8, 0.9))
#[1] 4.506694 5.683242

#----------

# Clean up
rm(dat)
```

---

errorBar *Plot Pointwise Error Bars*

Description

Plot pointwise error bars given their upper and lower limits.

The `errorBar` function is a modified version of the S function `error.bar`. The **EnvStats** function `errorBar` includes the additional arguments `draw.lower`, `draw.upper`, `gap.size`, `bar.ends.size`, and `col` to determine whether both the lower and upper error bars are drawn and to control the size of the gaps, the size of the bar ends, and the color of the bars.

Usage

```
errorBar(x, y = NULL, lower, upper, incr = TRUE, draw.lower = TRUE,
  draw.upper = TRUE, bar.ends = TRUE, gap = TRUE, add = FALSE,
  horizontal = FALSE, gap.size = 0.75, bar.ends.size = 1, col = 1,
  ..., xlab = deparse(substitute(x)), xlim, ylim)
```

Arguments

x, y	coordinates of points. The coordinates can be given by two vector arguments or by a single vector x. When both x and y are supplied and horizontal=FALSE (see below), x specifies where the groups will be plotted on the x-axis and y denotes the centers for each group that will be plotted on the y-axis. When both x and y are supplied and horizontal=TRUE (see below), y specifies where the groups will be plotted on the y-axis and x denotes the centers for each group that will be plotted on the x-axis. If a single numeric vector is given, then time(x) is plotted on the x-axis and x is plotted on the y-axis. Missing values (NAs) are allowed; points containing missing values are omitted from the plot.
lower	pointwise lower limits of the error bars. This may be a single number or a vector the same length as x and y. If incr=TRUE, then lower is expected to contain the lower half widths of the error bars. If incr=FALSE, then lower contains the coordinates of the lower limits in terms of x or y, depending on the orientation of the bars.
upper	pointwise upper limits of the error bars. This may be a single number or a vector the same length as x and y. If incr=TRUE, then upper is expected to contain the upper half widths of the error bars. If incr=FALSE, then upper contains the coordinates of the upper limits in terms of x or y, depending on the orientation of the bars. If upper is missing, the upper limits are drawn symmetric to the lower limits.
incr	logical scalar indicating whether the values in lower and upper represent increments. If incr=TRUE (the default), then lower and upper are assumed to represent half the widths of the error bars (increments).
draw.lower	logical scalar indicating whether to draw the lower error bar. The default is draw.lower=TRUE.
draw.upper	logical scalar indicating whether to draw the upper error bar. The default is draw.upper=TRUE.
bar.ends	logical scalar indicating whether flat bars should be drawn at the endpoints. The default is bar.ends=TRUE.
gap	logical scalar indicating whether gaps should be left around the points to emphasize their locations. The default is gap=TRUE.
add	logical scalar indicating whether error bars should be added to the current plot. If add=TRUE and a graphics device is open, the error bars are added to the plot in the open device and no axes are drawn. In this case, you should use the plot

parameters xlim and ylim to provide enough room for the error bars; otherwise, "Lines out of bounds" warnings are generated. If add=FALSE (the default), a new coordinate system is set up for the error bar plot and axes are drawn.

horizontal — logical scalar indicating whether the error bars should be oriented horizontally (horizontal=TRUE) or vertically (horizontal=FALSE; the default).

gap.size — numeric scalar controlling the width of the gap.

bar.ends.size — numeric scalar controlling the length of the bar ends.

col — numeric or character vector indicating the color(s) of the bars.

xlab, xlim, ylim, ... — additional graphical parameters (see par).

Details

errorBar creates a plot of y versus x with pointwise error bars.

Value

errorBar invisibly returns a list with the following components:

group.centers — numeric vector of values on the group axis (the x-axis unless horizontal=TRUE) indicating the centers of the groups.

group.stats — a matrix with the number of rows equal to the number of groups and three columns indicating the group location parameter (Center), the lower limit for the error bar (Lower), and the upper limit for the error bar (Upper).

Author(s)

Authors of S (for code for error.bar in S).

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Cleveland, W.S. (1994). *The Elements of Graphing Data*. Hobart Press, Summit, New Jersey.

See Also

plot, segments, pointwise, stripChart.

Examples

```
# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area.  These data are strored
# in the data frame EPA.94b.tccb.df.
#
# Using the log-transformed data, create
#
# 1. A dynamite plot (bar plot showing mean plus 1 SE)
```

```
#
# 2. A confidence interval plot.

TcCB.mat <- summaryStats(TcCB ~ Area, data = EPA.94b.tccb.df,
    se = TRUE, ci = TRUE)
Means <- TcCB.mat[, "Mean"]
SEs   <- TcCB.mat[, "SE"]
LCLs  <- TcCB.mat[, "95%.LCL"]
UCLs  <- TcCB.mat[, "95%.UCL"]

# Dynamite Plot
#--------------
dev.new()
group.centers <- barplot(Means, col = c("red", "blue"),
  ylim = range(0, Means, Means + SEs), ylab = "TcCB (ppb)",
  main = "Dynamite Plot for TcCB Data")
errorBar(x = as.vector(group.centers), y = Means,
  lower = SEs, draw.lower = FALSE, gap = FALSE,
  col = c("red", "blue"), add = TRUE)


# Confidence Interval Plot
#-------------------------
xlim <- par("usr")[1:2]
dev.new()
errorBar(x = as.vector(group.centers), y = Means,
  lower = LCLs, upper = UCLs, incr = FALSE, gap = FALSE,
  col = c("red", "blue"), xlim = xlim, xaxt = "n",
  xlab = "", ylab = "TcCB (ppb)",
  main = "Confidence Interval Plot for TcCB Data")
axis(1, at = group.centers, labels = dimnames(TcCB.mat)[[1]])


# Clean up
#---------
rm(TcCB.mat, Means, SEs, LCLs, UCLs, group.centers, xlim)
graphics.off()
```

`estimate.object` *S3 Class "estimate"*

Description

Objects of S3 class `"estimate"` are returned by any of the **EnvStats** functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of data assumed to come from that distribution.

Details

Objects of S3 class `"estimate"` are lists that contain information about the estimated distribution parameters, quantiles, and intervals. The names of the **EnvStats** functions that produce objects of

class "estimate" have the following forms:

Form of Function Name	Result
e*abb*	Parameter Estimation
eq*abb*	Quantile Estimation
predInt*Abb*	Prediction Interval
tolInt*Abb*	Tolerance Interval

where *abb* denotes the abbreviation of the name of a probability distribution (see the help file for Distribution.df for a list of available probability distributions and their abbreviations), and *Abb* denotes the same thing as *abb* except the first letter of the abbreviation for the probability distribution is capitalized.

See the help files Estimating Distribution Parameters and Estimating Distribution Quantiles for lists of functions that estimate distribution parameters and quantiles. See the help files Prediction Intervals and Tolerance Intervals for lists of functions that create prediction and tolerance intervals.

For example:

- The function enorm returns an object of class "estimate" (a list) with information about the estimated mean and standard deviation of the assumed normal (Gaussian) distribution, as well as an optional confidence interval for the mean.
- The function eqnorm returns a list of class "estimate" with information about the estimated mean and standard deviation of the assumed normal distribution, the estimated user-specified quantile(s), and an optional confidence interval for a single quantile.
- The function predIntNorm returns a list of class "estimate" with information about the estimated mean and standard deviation of the assumed normal distribution, along with a prediction interval for a user-specified number of future observations (or means, medians, or sums).
- The function tolIntNorm returns a list of class "estimate" with information about the estimated mean and standard deviation of the assumed normal distribution, along with a tolerance interval.

Value

Required Components

The following components must be included in a legitimate list of class "estimate".

distribution	character string indicating the name of the assumed distribution (this equals "Nonparametric") for nonparametric procedures).
sample.size	numeric scalar indicating the sample size used to estimate the parameters or quantiles.
data.name	character string indicating the name of the data object used to compute the estimated parameters or quantiles.
bad.obs	numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the estimation.

Optional Components

The following components may optionally be included in a legitimate list of class `"estimate"`.

`parameters`	(parametric estimation only) a numeric vector with a names attribute containing the names and values of the estimated distribution parameters.
`n.param.est`	(parametric estimation only) a scalar indicating the number of distribution parameters estimated.
`method`	(parametric estimation only) a character string indicating the method used to compute the estimated parameters.
`quantiles`	a numeric vector of estimated quantiles.
`quantile.method`	a character string indicating the method of quantile estimation.
`interval`	a list of class `"intervalEstimate"` containing information on a confidence, tolerance, or prediction interval.

All lists of class `"intervalEstimate"` contain the following component:

`name`	a character string inidicating the kind of interval. Possible values are: `"Confidence"`, `"Tolerance"`, or `"Prediction"`.

The number and names of the other components in a list of class `"intervalEstimate"` depends on the kind of interval it is. These components may include:

`parameter`	a character string indicating the parameter for which the interval is constructed (e.g., `"mean"`, `"95'th %ile"`, etc.).
`limits`	a numeric vector containing the lower and upper bounds of the interval.
`type`	the type of interval (i.e., `"two-sided"`, `"lower"`, or `"upper"`).
`method`	the method used to construct the interval (e.g., `"normal.approx"`).
`conf.level`	the confidence level associated with the interval.
`sample.size`	the sample size associated with the interval.
`dof`	(parametric intervals only) the degrees of freedom associated with the interval.
`limit.ranks`	(nonparametric intervals only) the rank(s) of the order statistic(s) used to construct the interval.
`m`	(prediction intervals only) the total number of future observations (`n.mean=1`, `n.median=1`, or `n.sum=1`) or averages (`n.mean>1`), medians (`n.median>1`), or sums (`n.sum>1`).
`k`	(prediction intervals only) the minimum number of future observations (`n.mean=1`, `n.median=1`, or `n.sum=1`), or averages (`n.mean>1`), medians (`n.median>1`) or sums (`n.sum>1`) out of the total `m` that the interval should contain.
`n.mean`	(prediction intervals only) the sample size associated with the future averages that should be contained in the interval.

`n.median` (prediction intervals only) the sample size associated with the future medians that should be contained in the interval.

`n.sum` (Poisson prediction intervals only) the sample size associated with the future sums that should be contained in the interval.

`rule` (simultaneous prediction intervals only) the rule used to construct the simultaneous prediction interval.

`delta.over.sigma`
(simultaneous prediction intervals only) numeric scalar indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation for both populations.

Methods

Generic functions that have methods for objects of class `"estimate"` include:
`print`.

Note

Since objects of class `"estimate"` are lists, you may extract their components with the `$` and `[[` operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

Estimating Distribution Parameters, Estimating Distribution Quantiles, `Distribution.df`, Prediction Intervals, Tolerance Intervals, `estimateCensored.object`.

Examples

```
# Create an object of class "estimate", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- rnorm(20, mean = 3, sd = 2)

estimate.obj <- enorm(dat, ci = TRUE)

mode(estimate.obj)
#[1] "list"

class(estimate.obj)
#[1] "estimate"
```

```
names(estimate.obj)
#[1] "distribution" "sample.size"  "parameters"
#[4] "n.param.est"  "method"       "data.name"
#[7] "bad.obs"      "interval"

names(estimate.obj$interval)
#[1] "name"        "parameter"   "limits"
#[4] "type"        "method"      "conf.level"
#[7] "sample.size" "dof"

estimate.obj

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 2.861160
#                                 sd   = 1.180226
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 2.308798
#                                 UCL = 3.413523

#----------

# Extract the confidence limits for the mean

estimate.obj$interval$limits
#     LCL      UCL
#2.308798 3.413523

#----------

# Clean up

rm(dat, estimate.obj)
```

`estimateCensored.object` *S3 Class "estimateCensored"*

Description

Objects of S3 class `"estimateCensored"` are returned by any of the **EnvStats** functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of ***censored*** data assumed to come from that distribution.

Details

Objects of S3 class `"estimateCensored"` are lists that contain information about the estimated distribution parameters, quantiles, and (if present) intervals, as well as the censoring side, censoring levels and percentage of censored observations. The names of the **EnvStats** functions that produce objects of class `"estimateCensored"` have the following forms:

Form of Function Name	Result
e*abb*Censored	Parameter Estimation
eq*abb*Censored	Quantile Estimation
predInt*Abb*Censored	Prediction Interval
tolInt*Abb*Censored	Tolerance Interval

where *abb* denotes the abbreviation of the name of a probability distribution (see the help file for `Distribution.df` for a list of available probability distributions and their abbreviations), and *Abb* denotes the same thing as *abb* except the first letter of the abbreviation for the probability distribution is capitalized.

See the sections **Estimating Distribution Parameters**, **Estimating Distribution Quantiles**, and **Prediction and Tolerance Intervals** in the help file EnvStats Functions for Censored Data for a list of functions that estimate distribution parameters, estimate distribution quantiles, create prediction intervals, or create tolerance intervals using censored data.

For example:

- The function `enormCensored` returns an object of class `"estimateCensored"` (a list) with information about the estimated mean and standard deviation of the assumed normal (Gaussian) distribution, information about the amount and side of censoring, and also an optional confidence interval for the mean.
- The function `eqnormCensored` returns a list of class `"estimateCensored"` with information about the estimated mean and standard deviation of the assumed normal distribution, information about the amount and side of censoring, the estimated user-specified quantile(s), and an optional confidence interval for a single quantile.
- The function `tolIntNormCensored` returns a list of class `"estimateCensored"` with information about the estimated mean and standard deviation of the assumed normal distribution, information about the amount and side of censoring, and the computed tolerance interval.

Value

Required Components

The following components must be included in a legitimate list of class "estimateCensored".

distribution	character string indicating the name of the assumed distribution (this equals "Nonparametric") for nonparametric procedures).
sample.size	numeric scalar indicating the sample size used to estimate the parameters or quantiles.
censoring.side	character string indicating whether the data are left- or right-censored.
censoring.levels	numeric scalar or vector indicating the censoring level(s).
percent.censored	numeric scalar indicating the percent of non-missing observations that are censored.
data.name	character string indicating the name of the data object used to compute the estimateCensored parameters or quantiles.
censoring.name	character string indicating the name of the data object used to identify which values are censored.
bad.obs	numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the estimation.

Optional Components

The following components may optionally be included in a legitimate list of class "estimateCensored".

parameters	(parametric estimation only) a numeric vector with a names attribute containing the names and values of the estimateCensored distribution parameters.
n.param.est	(parametric estimation only) a scalar indicating the number of distribution parameters estimateCensored.
method	(parametric estimation only) a character string indicating the method used to compute the estimateCensored parameters.
quantiles	a numeric vector of estimateCensored quantiles.
quantile.method	a character string indicating the method of quantile estimation.
interval	a list of class "intervalEstimate" containing information on a confidence, tolerance, or prediction interval.

All lists of class "intervalEstimateCensored" contain the following component:

name	a character string inidicating the kind of interval. Possible values are: "Confidence", "Tolerance", or "Prediction".

The number and names of the other components in a list of class "intervalEstimate" depends on the kind of interval it is. These components may include:

parameter	a character string indicating the parameter for which the interval is constructed (e.g., "mean", "95'th %ile", etc.).
limits	a numeric vector containing the lower and upper bounds of the interval.
type	the type of interval (i.e., "two-sided", "lower", or "upper").
method	the method used to construct the interval (e.g., "normal.approx").
conf.level	the confidence level associated with the interval.
sample.size	the sample size associated with the interval.
dof	(parametric intervals only) the degrees of freedom associated with the interval.
limit.ranks	(nonparametric intervals only) the rank(s) of the order statistic(s) used to construct the interval.
m	(prediction intervals only) the total number of future observations (n.mean=1, n.median=1, or n.sum=1) or averages (n.mean>1), medians (n.median>1), or sums (n.sum>1).
k	(prediction intervals only) the minimum number of future observations (n.mean=1, n.median=1, or n.sum=1), or averages (n.mean>1), medians (n.median>1) or sums (n.sum>1) out of the total m that the interval should contain.
n.mean	(prediction intervals only) the sample size associated with the future averages that should be contained in the interval.
n.median	(prediction intervals only) the sample size associated with the future medians that should be contained in the interval.
n.sum	(Poisson prediction intervals only) the sample size associated with the future sums that should be contained in the interval.
rule	(simultaneous prediction intervals only) the rule used to construct the simultaneous prediction interval.
delta.over.sigma	(simultaneous prediction intervals only) numeric scalar indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation for both populations.

Methods

Generic functions that have methods for objects of class "estimateCensored" include: print.

Note

Since objects of class "estimateCensored" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

EnvStats Functions for Censored Data, `Distribution.df`, `estimate.object`.

Examples

```
# Create an object of class "estimateCensored", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- rnorm(20, mean = 100, sd = 20)
censored <- dat < 90
dat[censored] <- 90

estimateCensored.obj <- enormCensored(dat, censored, ci = TRUE)

mode(estimateCensored.obj)
#[1] "list"

class(estimateCensored.obj)
#[1] "estimateCensored"

names(estimateCensored.obj)
# [1] "distribution"     "sample.size"      "censoring.side"   "censoring.levels"
# [5] "percent.censored" "parameters"       "n.param.est"      "method"
# [9] "data.name"        "censoring.name"   "bad.obs"          "interval"
#[13] "var.cov.params"

names(estimateCensored.obj$interval)
#[1] "name"       "parameter"  "limits"     "type"       "method"     "conf.level"


estimateCensored.obj

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              90
#
#Estimated Parameter(s):          mean = 96.52796
#                                 sd   = 14.62275
#
```

```
#Estimation Method:              MLE
#
#Data:                           dat
#
#Censoring Variable:             censored
#
#Sample Size:                    20
#
#Percent Censored:               25%
#
#Confidence Interval for:        mean
#
#Confidence Interval Method:     Profile Likelihood
#
#Confidence Interval Type:       two-sided
#
#Confidence Level:               95%
#
#Confidence Interval:            LCL =  88.82415
#                                UCL = 103.27604


#----------

# Extract the confidence limits for the mean

estimateCensored.obj$interval$limits
#     LCL      UCL
# 91.7801 103.7839

#----------

# Clean up

rm(dat, censored, estimateCensored.obj)
```

EulersConstant *Euler's Constant*

Description

Explanation of Euler's Constant.

Details

Euler's Constant, here denoted ϵ, is a real-valued number that can be defined in several ways. Johnson et al. (1992, p. 5) use the definition:

$$\epsilon = \lim_{n \to \infty} [1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} - log(n)]$$

and note that it can also be expressed as

$$\epsilon = -\Psi(1)$$

where $\Psi()$ is the digamma function (Johnson et al., 1992, p.8).

The value of Euler's Constant, to 10 decimal places, is 0.5772156649.

The expression for the mean of a Type I extreme value (Gumbel) distribution involves Euler's constant; hence Euler's constant is used to compute the method of moments estimators for this distribution (see eevd).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, pp.4-8.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

Extreme Value Distribution, eevd.

---

eunif *Estimate Parameters of a Uniform Distribution*

Description

Estimate the minimum and maximum parameters of a uniform distribution.

Usage

```
eunif(x, method = "mle")
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
method	character string specifying the method of estimation. The possible values are "mle" (maximum likelihood; the default), "mme" (method of moments), and "mmue" (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from an uniform distribution with parameters min=a and max=b. Also, let $x_{(i)}$ denote the i'th order statistic.

Estimation

Maximum Likelihood Estimation (method="mle")
The maximum likelihood estimators (mle's) of a and b are given by (Johnson et al, 1995, p.286):

$$\hat{a}_{mle} = x_{(1)} \quad (1)$$

$$\hat{b}_{mle} = x_{(n)} \quad (2)$$

Method of Moments Estimation (method="mme")
The method of moments estimators (mme's) of a and b are given by (Forbes et al., 2011):

$$\hat{a}_{mme} = \bar{x} - \sqrt{3}s_m \quad (3)$$

$$\hat{b}_{mme} = \bar{x} + \sqrt{3}s_m \quad (4)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (5)$$

$$s_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (6)$$

Method of Moments Estimation Based on the Unbiased Estimator of Variance (method="mmue")
The method of moments estimators based on the unbiased estimator of variance are exactly the same as the method of moments estimators given in equations (3-6) above, except that the method of moments estimator of variance in equation (6) is replaced with the unbiased estimator of variance:

$$\hat{a}_{mmue} = \bar{x} - \sqrt{3}s \quad (7)$$

$$\hat{b}_{mmue} = \bar{x} + \sqrt{3}s \quad (8)$$

where

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (9)$$

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

Note

The uniform distribution (also called the rectangular distribution) with parameters `min` and `max` takes on values on the real line between `min` and `max` with equal probability. It has been used to represent the distribution of round-off errors in tabulated values. Another important application is that the distribution of the cumulative distribution function (cdf) of any kind of continuous random variable follows a uniform distribution with parameters `min=0` and `max=1`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

Uniform, `estimate.object`.

Examples

```
# Generate 20 observations from a uniform distribution with parameters
# min=-2 and max=3, then estimate the parameters via maximum likelihood.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- runif(20, min = -2, max = 3)
eunif(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Uniform
#
#Estimated Parameter(s):          min = -1.574529
#                                 max =  2.837006
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20

#----------

# Compare the three methods of estimation:

eunif(dat, method = "mle")$parameters
```

```
#      min       max
#-1.574529  2.837006


eunif(dat, method = "mme")$parameters
#      min       max
#-1.988462  2.650737


eunif(dat, method = "mmue")$parameters
#      min       max
#-2.048721  2.710996

#----------

# Clean up
#---------
rm(dat)
```

EVD *The Extreme Value (Gumbel) Distribution*

Description

Density, distribution function, quantile function, and random generation for the (largest) extreme value distribution.

Usage

```
devd(x, location = 0, scale = 1)
pevd(q, location = 0, scale = 1)
qevd(p, location = 0, scale = 1)
revd(n, location = 0, scale = 1)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
location	vector of location parameters.
scale	vector of positive scale parameters.

Details

Let X be an extreme value random variable with parameters `location`=η and `scale`=θ. The density function of X is given by:

$$f(x;\eta,\theta) = \frac{1}{\theta}e^{-(x-\eta)/\theta}exp[-e^{-(x-\eta)/\theta}]$$

where $-\infty < x, \eta < \infty$ and $\theta > 0$.

The cumulative distribution function of X is given by:

$$F(x;\eta,\theta) = exp[-e^{-(x-\eta)/\theta}]$$

The p^{th} quantile of X is given by:

$$x_p = \eta - \theta log[-log(p)]$$

The mode, mean, variance, skew, and kurtosis of X are given by:

$$Mode(X) = \eta$$

$$E(X) = \eta + \epsilon\theta$$

$$Var(X) = \theta^2\pi^2/6$$

$$Skew(X) = \sqrt{\beta_1} = 1.139547$$

$$Kurtosis(X) = \beta_2 = 5.4$$

where ϵ denotes Euler's constant, which is equivalent to `-digamma(1)`.

Value

density (`devd`), probability (`pevd`), quantile (`qevd`), or random sample (`revd`) for the extreme value distribution with location parameter(s) determined by `location` and scale parameter(s) determined by `scale`.

Note

There are three families of extreme value distributions. The one described here is the Type I, also called the Gumbel extreme value distribution or simply Gumbel distribution. The name "extreme value" comes from the fact that this distribution is the limiting distribution (as n approaches infinity) of the greatest value among n independent random variables each having the same continuous distribution.

The Gumbel extreme value distribution is related to the exponential distribution as follows. Let Y be an exponential random variable with parameter `rate`=λ. Then $X = \eta - log(Y)$ has an extreme value distribution with parameters `location`=η and `scale`=$1/\lambda$.

The distribution described above and used by `devd`, `pevd`, `qevd`, and `revd` is the *largest* extreme value distribution. The smallest extreme value distribution is the limiting distribution (as n approaches infinity) of the smallest value among n independent random variables each having the same continuous distribution. If X has a largest extreme value distribution with parameters `location`=η and `scale`=θ, then $Y = -X$ has a smallest extreme value distribution with parameters

location=$-\eta$ and scale=θ. The smallest extreme value distribution is related to the Weibull distribution as follows. Let Y be a Weibull random variable with parameters shape=β and scale=α. Then $X = log(Y)$ has a smallest extreme value distribution with parameters location=$log(\alpha)$ and scale=$1/\beta$.

The extreme value distribution has been used extensively to model the distribution of streamflow, flooding, rainfall, temperature, wind speed, and other meteorological variables, as well as material strength and life data.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

eevd, GEVD, Probability Distributions and Random Numbers.

Examples

```
# Density of an extreme value distribution with location=0, scale=1,
# evaluated at 0.5:

devd(.5)
#[1] 0.3307043

#----------

# The cdf of an extreme value distribution with location=1, scale=2,
# evaluated at 0.5:

pevd(.5, 1, 2)
#[1] 0.2769203

#----------

# The 25'th percentile of an extreme value distribution with
# location=-2, scale=0.5:

qevd(.25, -2, 0.5)
#[1] -2.163317

#----------

# Random sample of 4 observations from an extreme value distribution with
# location=5, scale=2.
```

```
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
revd(4, 5, 2)
#[1] 9.070406 7.669139 4.511481 5.903675
```

`evNormOrdStats` *Expected Value of Order Statistics for Random Sample from Standard Normal Distribution*

Description

Compute the expected values of order statistics for a random sample from a standard normal distribution.

Usage

```
evNormOrdStats(n = 1,
  method = "royston", lower = -9, inc = 0.025, warn = TRUE,
  alpha = 3/8, nmc = 2000, seed = 47, approximate = NULL)

evNormOrdStatsScalar(r = 1, n = 1,
  method = "royston", lower = -9, inc = 0.025, warn = TRUE,
  alpha = 3/8, nmc = 2000, conf.level = 0.95, seed = 47, approximate = NULL)
```

Arguments

`n`	positive integer indicating the sample size.
`r`	positive integer between 1 and `n` specifying the order statistic for which to compute the expected value.
`method`	character string indicating what method to use. The possible values are: • `"royston"`. Method based on approximating the exact integral as given in Royston (1982). • `"blom"`. Method based on the approximation formula proposed by Blom (1958). • `"mc"`. Method based on Monte Carlo simulation. See the DETAILS section below.
`lower`	numeric scalar ≤ -9 defining the lower bound used for approximating the integral when `method="royston"`. The upper bound is automatically set to `-lower`. The default value is `lower=-9`.
`inc`	numeric scalar between `.Machine$double.eps` and `0.025` that determines the width of each subdivision used to approximate the integral when `method="royston"`. The default value is `inc=0.025`.
`warn`	logical scalar indicating whether to issue a warning when `method="royston"` and the sample size is greater than 2000. The default value is `warn=TRUE`.

`alpha`	numeric scalar between 0 and 0.5 that determines the constant used when `method="blom"`. The default value is `alpha=3/8`.
`nmc`	integer ≥ 100 denoting the number of Monte Carlo simulations to use when `method="mc"`. The default value is `nmc=2000`.
`conf.level`	numeric scalar between 0 and 1 denoting the confidence level of the confidence interval for the expected value of the normal order statistic when `method="mc"`. The default value is `conf.level=0.95`.
`seed`	integer between $-(2^31-1)$ and 2^31-1 specifying the argument to `set.seed` (the random number seed) when `method="mc"`. The default value is `seed=47`.
`approximate`	logical scalar included for backwards compatibility with versions of **EnvStats** prior to version 2.3.0. When `method` is not supplied and `approximate=FALSE`, `method` is set to `method="royston"`. When `method` is not supplied and `approximate=TRUE`, `method` is set to `method="blom"`. This argument is ignored if `method` is supplied and/or `approxmiate=NULL` (the default).

Details

Let $\underline{z} = z_1, z_2, \ldots, z_n$ denote a vector of n observations from a normal distribution with parameters `mean=0` and `sd=1`. That is, $\underline{z}$ denotes a vector of n observations from a *standard* normal distribution. Let $z_{(r)}$ denote the r'th order statistic of $\underline{z}$, for $r = 1, 2, \ldots, n$. The probability density function of $z_{(r)}$ is given by:

$$f_{r,n}(t) = \frac{n!}{(r-1)!(n-r)!}[\Phi(t)]^{r-1}[1-\Phi(t)]^{n-r}\phi(t) \qquad (1)$$

where Φ and ϕ denote the cumulative distribution function and probability density function of the standard normal distribution, respectively (Johnson et al., 1994, p.93). Thus, the expected value of $z_{(r)}$ is given by:

$$E(r,n) = E[z_{(r)}] = \int_{-\infty}^{\infty} t f_{r,n}(t)dt \qquad (2)$$

It can be shown that if n is odd, then

$$E[(n+1)/2, n] = 0 \qquad (3)$$

Also, for all values of n,

$$E(r,n) = -E(n-r+1,n) \qquad (4)$$

The function `evNormOrdStatsScalar` computes the value of $E(r,n)$ for user-specified values of r and n.

The function `evNormOrdStats` computes the values of $E(r,n)$ for all values of r (i.e., for $r = 1, 2, \ldots, n$) for a user-specified value of n.

Exact Method Based on Royston's Approximation to the Integral (`method="royston"`)

When `method="royston"`, the integral in Equation (2) above is approximated by computing the value of the integrand between the values of `lower` and `-lower` using increments of `inc`, then summing these values and multiplying by `inc`. In particular, the integrand is restructured as:

$$t\ f_{r,n}(t) = t\ exp\{log(n!) - log[(r-1)!] - log[(n-r)!] + (r-1)log[\Phi(t)] + (n-r)log[1-\Phi(t)] + log[\phi(t)]\} \quad (5)$$

By default, as per Royston (1982), the integrand is evaluated between -9 and 9 in increments of 0.025. The approximation is computed this way for values of r between 1 and $[n/2]$, where $[x]$ denotes the floor of x. If $r > [n/2]$, then the approximation is computed for $E(n - r + 1, n)$ and Equation (4) is used.

Note that Equation (1) in Royston (1982) differs from Equations (1) and (2) above because Royston's paper is based on the r^{th} *largest* value, not the r^{th} order statistic.

Royston (1982) states that this algorithm "is accurate to at least seven decimal places on a 36-bit machine," that it has been validated up to a sample size of $n = 2000$, and that the accuracy for $n > 2000$ may be improved by reducing the value of the argument `inc`. Note that making `inc` smaller will increase the computation time.

Approxmation Based on Blom's Method (`method="blom"`)

When `method="blom"`, the following approximation to $E(r, n)$, proposed by Blom (1958, pp. 68-75), is used:

$$E(r, n) \approx \Phi^{-1}(\frac{r - \alpha}{n - 2\alpha + 1}) \qquad (5)$$

By default, $\alpha = 3/8 = 0.375$. This approximation is quite accurate. For example, for $n \geq 2$, the approximation is accurate to the first decimal place, and for $n \geq 9$ it is accurate to the second decimal place.

Harter (1961) discusses appropriate values of α for various sample sizes n and values of r.

Approximation Based on Monte Carlo Simulation (`method="mc"`)

When `method="mc"`, Monte Carlo simulation is used to estmate the expected value of the r^{th} order statistic. That is, $N =$ `nmc` trials are run in which, for each trial, a random sample of n standard normal observations is generated and the r^{th} order statistic is computed. Then, the average value of this order statistic over all N trials is computed, along with a confidence interval for the expected value, assuming an approximately normal distribution for the mean of the order statistic (the confidence interval is computed by supplying the simulated values of the r^{th} order statistic to the function `enorm`).

NOTE: This method has not been optimized for large sample sizes n (i.e., large values of the argument `n`) and/or a large number of Monte Carlo trials N (i.e., large values of the argument `nmc`) and may take a long time to execute in these cases.

Value

For `evNormOrdStats`: a numeric vector of length `n` containing the expected values of all the order statistics for a random sample of `n` standard normal deviates.

For `evNormOrdStatsScalar`: a numeric scalar containing the expected value of the `r`'th order statistic from a random sample of `n` standard normal deviates. When `method="mc"`, the returned object also has a `cont.int` attribute that contains the 95 and a `nmc` attribute indicating the number of Monte Carlo trials run.

Note

The expected values of normal order statistics are used to construct normal quantile-quantile (Q-Q) plots (see `qqPlot`) and to compute goodness-of-fit statistics (see `gofTest`). Usually, however,

approximations are used instead of exact values. The functions evNormOrdStats and evNormOrdStatsScalar have been included mainly because evNormOrdStatsScalar is called by elnorm3 and predIntNparSimultaneousTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Blom, G. (1958). *Statistical Estimates and Transformed Beta Variables*. John Wiley and Sons, New York.

Harter, H. L. (1961). *Expected Values of Normal Order Statistics* **48**, 151–165.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, pp. 93–99.

Royston, J.P. (1982). Algorithm AS 177. Expected Normal Order Statistics (Exact and Approximate). *Applied Statistics* **31**, 161–165.

See Also

Normal, ppoints, elnorm3, predIntNparSimultaneousTestPower, gofTest, qqPlot.

Examples

```
# Compute the expected value of the minimum for a random sample of size 10
# from a standard normal distribution:

# Based on method="royston"
#--------------------------
evNormOrdStatsScalar(r = 1, n = 10)
#[1] -1.538753


# Based on method="blom"
#-----------------------
evNormOrdStatsScalar(r = 1, n = 10, method = "blom")
#[1] -1.546635


# Based on method="mc" with 10,000 Monte Carlo trials
#----------------------------------------------------
evNormOrdStatsScalar(r = 1, n = 10, method = "mc", nmc = 10000)
#[1] -1.544318
#attr(,"confint")
#   95%LCL    95%UCL
#-1.555838 -1.532797
#attr(,"nmc")
#[1] 10000

#====================
```

```
# Compute the expected values of all of the order statistics
# for a random sample of size 10 from a standard normal distribution
# based on Royston's (1982) method:
#-----------------------------------------------------------------

evNormOrdStats(10)
#[1] -1.5387527 -1.0013570 -0.6560591 -0.3757647 -0.1226678
#[6]  0.1226678  0.3757647  0.6560591  1.0013570  1.5387527


# Compare the above with Blom (1958) scores:
#-------------------------------------------

evNormOrdStats(10, method = "blom")
#[1] -1.5466353 -1.0004905 -0.6554235 -0.3754618 -0.1225808
#[6]  0.1225808  0.3754618  0.6554235  1.0004905  1.5466353
```

`eweibull` *Estimate Parameters of a Weibull Distribution*

Description

Estimate the shape and scale parameters of a Weibull distribution.

Usage

```
eweibull(x, method = "mle")
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`method`	character string specifying the method of estimation. Possible values are `"mle"` (maximum likelihood; the default), `"mme"` (methods of moments), and `"mmue"` (method of moments based on the unbiased estimator of variance). See the DETAILS section for more information on these estimation methods.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from an Weibull distribution with parameters `shape`=α and `scale`=β.

Estimation

Maximum Likelihood Estimation (`method="mle"`)
The maximum likelihood estimators (mle's) of α and β are the solutions of the simultaneous equations (Forbes et al., 2011):

$$\hat{\alpha}_{mle} = \frac{n}{\{(1/\hat{\beta}_{mle})^{\hat{\alpha}_{mle}} \sum_{i=1}^{n}[x_i^{\hat{\alpha}_{mle}} log(x_i)]\} - \sum_{i=1}^{n} log(x_i)} \quad (1)$$

$$\hat{\beta}_{mle} = [\frac{1}{n}\sum_{i=1}^{n} x_i^{\hat{\alpha}_{mle}}]^{1/\hat{\alpha}_{mle}} \quad (2)$$

Method of Moments Estimation (`method="mme"`)
The method of moments estimator (mme) of α is computed by solving the equation:

$$\frac{s}{\bar{x}} = \{\frac{\Gamma[(\hat{\alpha}_{mme}+2)/\hat{\alpha}_{mme}]}{\{\Gamma[(\hat{\alpha}_{mme}+1)/\hat{\alpha}_{mme}]\}^2} - 1\}^{1/2} \quad (3)$$

and the method of moments estimator (mme) of β is then computed as:

$$\hat{\beta}_{mme} = \frac{\bar{x}}{\Gamma[(\hat{\alpha}_{mme}+1)/\hat{\alpha}_{mme}]} \quad (4)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (5)$$

$$s_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (6)$$

and $\Gamma()$ denotes the gamma function.

Method of Moments Estimation Based on the Unbiased Estimator of Variance (`method="mmue"`)
The method of moments estimators based on the unbiased estimator of variance are exactly the same as the method of moments estimators given in equations (3-6) above, except that the method of moments estimator of variance in equation (6) is replaced with the unbiased estimator of variance:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad (7)$$

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

Note

The Weibull distribution is named after the Swedish physicist Waloddi Weibull, who used this distribution to model breaking strengths of materials. The Weibull distribution has been extensively applied in the fields of reliability and quality control.

The exponential distribution is a special case of the Weibull distribution: a Weibull random variable with parameters shape=1 and scale=β is equivalent to an exponential random variable with parameter rate=$1/\beta$.

The Weibull distribution is related to the Type I extreme value (Gumbel) distribution as follows: if X is a random variable from a Weibull distribution with parameters shape=α and scale=β, then

$$Y = -log(X) \quad (10)$$

is a random variable from an extreme value distribution with parameters location=$-log(\beta)$ and scale=$1/\alpha$.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

Weibull, Exponential, EVD, estimate.object.

Examples

```
# Generate 20 observations from a Weibull distribution with parameters
# shape=2 and scale=3, then estimate the parameters via maximum likelihood.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rweibull(20, shape = 2, scale = 3)
eweibull(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Weibull
#
#Estimated Parameter(s):          shape = 2.673098
#                                 scale = 3.047762
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20

#----------
```

```
# Use the same data as in previous example, and compute the method of
# moments estimators based on the unbiased estimator of variance:

eweibull(dat, method = "mmue")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Weibull
#
#Estimated Parameter(s):          shape = 2.528377
#                                 scale = 3.052507
#
#Estimation Method:               mmue
#
#Data:                            dat
#
#Sample Size:                     20

#----------

# Clean up
#---------
rm(dat)
```

`ezmlnorm` *Estimate Parameters of a Zero-Modified Lognormal (Delta) Distribution*

Description

Estimate the parameters of a zero-modified lognormal distribution or a zero-modified lognormal distribution (alternative parameterization), and optionally construct a confidence interval for the mean.

Usage

```
ezmlnorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "normal.approx", conf.level = 0.95)

ezmlnormAlt(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "normal.approx", conf.level = 0.95)
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.

`method` character string specifying the method of estimation. The only possible value is `"mvue"` (minimum variance unbiased; the default). See the DETAILS section for more information on this estimation method.

`ci` logical scalar indicating whether to compute a confidence interval for the mean. The default value is `FALSE`. If `ci=TRUE` and there are less than three non-missing observations in `x`, or if all observations are zeros, a warning will be issued and no confidence interval will be computed.

`ci.type` character string indicating what kind of confidence interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`. This argument is ignored if `ci=FALSE`.

`ci.method` character string indicating what method to use to construct the confidence interval for the mean. The only possible value is `"normal.approx"` (the default). See the DETAILS section for more information. This argument is ignored if `ci=FALSE`.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a zero-modified lognormal distribution with parameters `meanlog=`μ, `sdlog=`σ, and `p.zero=`p. Alternatively, let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a zero-modified lognormal distribution (alternative parameterization) with parameters `mean=`θ, `cv=`τ, and `p.zero=`p.

Let r denote the number of observations in $\underline{x}$ that are equal to 0, and order the observations so that $x_1, x_2, \ldots, x_r$ denote the r zero observations and $x_{r+1}, x_{r+2}, \ldots, x_n$ denote the $n - r$ non-zero observations.

Note that θ is *not* the mean of the zero-modified lognormal distribution; it is the mean of the lognormal part of the distribution. Similarly, τ is *not* the coefficient of variation of the zero-modified lognormal distribution; it is the coefficient of variation of the lognormal part of the distribution.

Let γ, δ, and ϕ denote the mean, standard deviation, and coefficient of variation of the overall zero-modified lognormal (delta) distribution. Let η denote the standard deviation of the lognormal part of the distribution, so that $\eta = \theta\tau$. Aitchison (1955) shows that:

$$\gamma = (1 - p)\theta \quad (1)$$

$$\delta^2 = (1 - p)\eta^2 + p(1 - p)\theta^2 \quad (2)$$

so that

$$\phi = \frac{\delta}{\gamma} = \frac{\sqrt{\tau^2 + p}}{\sqrt{1 - p}} \quad (3)$$

Estimation

Minimum Variance Unbiased Estimation (`method="mvue"`)
Aitchison (1955) shows that the minimum variance unbiased estimators (mvue's) of γ and δ are:

$$\hat{\gamma}_{mvue} = \begin{cases} (1-\frac{r}{n})e^{\bar{y}}g_{n-r-1}(\frac{s^2}{2}) & \text{if } r < n-1, \\ x_n/n & \text{if } r = n-1, \\ 0 & \text{if } r = n \end{cases} \quad (4)$$

$$\hat{\delta}^2_{mvue} = \begin{cases} (1-\frac{r}{n})e^{2\bar{y}}\{g_{n-r-1}(2s^2) - \frac{n-r-1}{n-1}g_{n-r-1}[\frac{(n-r-2)s^2}{n-r-1}]\} & \text{if } r < n-1, \\ x_n^2/n & \text{if } r = n-1, \\ 0 & \text{if } r = n \end{cases} \quad (5)$$

where

$$y_i = log(x_i),\ r = r+1, r+2, \ldots, n \quad (6)$$

$$\bar{y} = \frac{1}{n-r}\sum_{i=r+1}^{n} y_i \quad (7)$$

$$s^2 = \frac{1}{n-r-1}\sum_{i=r+1}^{n}(y_i - \bar{y})^2 \quad (8)$$

$$g_m(z) = \sum_{i=0}^{\infty}\frac{m^i(m+2i)}{m(m+2)\cdots(m+2i)}(\frac{m}{m+1})^i(\frac{z^i}{i!}) \quad (9)$$

Note that when $r = n-1$ or $r = n$, the estimator of γ is simply the sample mean for all observations (including zero values), and the estimator for δ^2 is simply the sample variance for all observations.

The expected value and asymptotic variance of the mvue of γ are (Aitchison and Brown, 1957, p.99; Owen and DeRouen, 1980):

$$E(\hat{\gamma}_{mvue}) = \gamma \quad (10)$$

$$AVar(\hat{\gamma}_{mvue}) = \frac{1}{n}exp(2\mu+\sigma^2)(1-p)(p+\frac{2\sigma^2+\sigma^4}{2}) \quad (11)$$

Confidence Intervals

Based on Normal Approximation (`ci.method="normal.approx"`)
An approximate $(1-\alpha)100\%$ confidence interval for γ is constructed based on the assumption that the estimator of γ is approximately normally distributed. Thus, an approximate two-sided $(1-\alpha)100\%$ confidence interval for γ is constructed as:

$$[\hat{\gamma}_{mvue} - t_{n-2,1-\alpha/2}\hat{\sigma}_{\hat{\gamma}},\ \hat{\gamma}_{mvue} + t_{n-2,1-\alpha/2}\hat{\sigma}_{\hat{\gamma}}] \quad (12)$$

where $t_{\nu,p}$ is the p'th quantile of Student's t-distribution with ν degrees of freedom, and the quantity $\hat{\sigma}_{\hat{\gamma}}$ is the estimated standard deviation of the mvue of γ, and is computed by replacing the values of μ, σ, and p in equation (11) above with their estimated values and taking the square root.

Note that there must be at least 3 non-missing observations ($n \geq 3$) and at least one observation must be non-zero ($r \leq n-1$) in order to construct a confidence interval.

One-sided confidence intervals are computed in a similar fashion.

Value

a list of class "estimate" containing the estimated parameters and other information. See estimate.object for details.

For the function ezmlnorm, the component called parameters is a numeric vector with the following estimated parameters:

Parameter Name	Explanation
meanlog	mean of the log of the lognormal part of the distribution.
sdlog	standard deviation of the log of the lognormal part of the distribution.
p.zero	probability that an observation will be 0.
mean.zmlnorm	mean of the overall zero-modified lognormal (delta) distribution.
sd.zmlnorm	standard deviation of the overall zero-modified lognormal (delta) distribution.

For the function ezmlnormAlt, the component called parameters is a numeric vector with the following estimated parameters:

Parameter Name	Explanation
mean	mean of the lognormal part of the distribution.
cv	coefficient of variation of the lognormal part of the distribution.
p.zero	probability that an observation will be 0.
mean.zmlnorm	mean of the overall zero-modified lognormal (delta) distribution.
sd.zmlnorm	standard deviation of the overall zero-modified lognormal (delta) distribution.

Note

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as "Below Detection Limit" (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J. (1955). On the Distribution of a Positive Random Variable Having a Discrete Probability Mass at the Origin. *Journal of the American Statistical Association* **50**, 901–908.

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special reference to its uses in economics)*. Cambridge University Press, London. pp.94-99.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, pp.47–51.

Gibbons, RD., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley and Sons, Hoboken, NJ.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135–146.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R*. Second Edition. John Wiley and Sons, Hoboken, NJ, Chapter 1.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, p.312.

Owen, W., and T. DeRouen. (1980). Estimation of the Mean for Lognormal Data Containing Zeros and Left-Censored Values, with Applications to the Measurement of Worker Exposure to Air Contaminants. *Biometrics* **36**, 707–719.

USEPA (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, Permits and State Programs Division, US Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

Zero-Modified Lognormal, Zero-Modified Normal, Lognormal.

Examples

```
# Generate 100 observations from a zero-modified lognormal (delta)
# distribution with mean=2, cv=1, and p.zero=0.5, then estimate the
# parameters. According to equations (1) and (3) above, the overall mean
# is mean.zmlnorm=1 and the overall cv is cv.zmlnorm=sqrt(3).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rzmlnormAlt(100, mean = 2, cv = 1, p.zero = 0.5)
ezmlnormAlt(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Zero-Modified Lognormal (Delta)
#
#Estimated Parameter(s):          mean         = 1.9604561
#                                 cv           = 0.9169411
#                                 p.zero       = 0.4500000
#                                 mean.zmlnorm = 1.0782508
```

```
#                                cv.zmlnorm   = 1.5307175
#
#Estimation Method:              mvue
#
#Data:                           dat
#
#Sample Size:                    100
#
#Confidence Interval for:        mean.zmlnorm
#
#Confidence Interval Method:     Normal Approximation
#                                (t Distribution)
#
#Confidence Interval Type:       two-sided
#
#Confidence Level:               95%
#
#Confidence Interval:            LCL = 0.748134
#                                UCL = 1.408368

#----------

# Clean up
rm(dat)
```

---

ezmnorm *Estimate Parameters of a Zero-Modified Normal Distribution*

Description

Estimate the mean and standard deviation of a zero-modified normal distribution, and optionally construct a confidence interval for the mean.

Usage

```
ezmnorm(x, method = "mvue", ci = FALSE, ci.type = "two-sided",
  ci.method = "normal.approx", conf.level = 0.95)
```

Arguments

x	numeric vector of observations.
method	character string specifying the method of estimation. Currently, the only possible value is "mvue" (minimum variance unbiased; the default). See the DETAILS section for more information.
ci	logical scalar indicating whether to compute a confidence interval for the mean. The default value is FALSE.
ci.type	character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper". This argument is ignored if ci=FALSE.

`ci.method` character string indicating what method to use to construct the confidence interval for the mean. Currently the only possible value is `"normal.approx"` (the default). See the DETAILS section for more information.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the confidence interval. The default value is `conf.level=0.95`. This argument is ignored if `ci=FALSE`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n observations from a zero-modified normal distribution with parameters `mean`=μ, `sd`=σ, and `p.zero`=p. Let r denote the number of observations in $\underline{x}$ that are equal to 0, and order the observations so that $x_1, x_2, \ldots, x_r$ denote the r zero observations, and $x_{r+1}, x_{r+2}, \ldots, x_n$ denote the $n - r$ non-zero observations.

Note that μ is *not* the mean of the zero-modified normal distribution; it is the mean of the normal part of the distribution. Similarly, σ is *not* the standard deviation of the zero-modified normal distribution; it is the standard deviation of the normal part of the distribution.

Let γ and δ denote the mean and standard deviation of the overall zero-modified normal distribution. Aitchison (1955) shows that:

$$\gamma = (1 - p)\mu \quad (1)$$

$$\delta^2 = (1 - p)\sigma^2 + p(1 - p)\mu^2 \quad (2)$$

Estimation

Minimum Variance Unbiased Estimation (`method="mvue"`)
Aitchison (1955) shows that the minimum variance unbiased estimators (mvue's) of γ and δ are:

$$\hat{\gamma}_{mvue} = \bar{x} \quad (3)$$

$$\hat{\delta}^2_{mvue} = \begin{cases} \frac{n-r-1}{n-1}(s^*)^2 + \frac{r}{n}(\frac{n-r}{n-1})(\bar{x}^*)^2 & \text{if } r < n-1, \\ x_n^2/n & \text{if } r = n-1, \\ 0 & \text{if } r = n \end{cases} \quad (4)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (5)$$

$$\bar{x}^* = \frac{1}{n-r}\sum_{i=r+1}^{n} x_i \quad (6)$$

$$(s^*)^2 = \frac{1}{n-r-1}\sum_{i=r+1}^{n} (x_i - \bar{x}^*)^2 \quad (7)$$

Note that the quantity in equation (5) is the sample mean of all observations (including 0 values), the quantity in equation (6) is the sample mean of all non-zero observations, and the quantity in equation (7) is the sample variance of all non-zero observations. Also note that for $r = n - 1$ or $r = n$, the estimator of δ^2 is the sample variance for all observations (including 0 values).

Confidence Intervals

Based on Normal Approximation (`ci.method="normal.approx"`)
An approximate $(1-\alpha)100\%$ confidence interval for γ is constructed based on the assumption that the estimator of γ is approximately normally distributed. Aitchison (1955) shows that

$$Var(\hat{\gamma}_{mvue}) = Var(\bar{x}) = \frac{\delta^2}{n} \quad (8)$$

Thus, an approximate two-sided $(1-\alpha)100\%$ confidence interval for γ is constructed as:

$$[\hat{\gamma}_{mvue} - t_{n-2,1-\alpha/2}\frac{\hat{\delta}_{mvue}}{\sqrt{n}},\ \hat{\gamma}_{mvue} + t_{n-2,1-\alpha/2}\frac{\hat{\delta}_{mvue}}{\sqrt{n}}] \quad (9)$$

where $t_{\nu,p}$ is the p'th quantile of Student's t-distribution with ν degrees of freedom.

One-sided confidence intervals are computed in a similar fashion.

Value

a list of class `"estimate"` containing the estimated parameters and other information. See `estimate.object` for details.

The component called `parameters` is a numeric vector with the following estimated parameters:

Parameter Name	Explanation
`mean`	mean of the normal (Gaussian) part of the distribution.
`sd`	standard deviation of the normal (Gaussian) part of the distribution.
`p.zero`	probability that an observation will be 0.
`mean.zmnorm`	mean of the overall zero-modified normal distribution.
`sd.zmnorm`	standard deviation of the overall normal distribution.

Note

The zero-modified normal distribution is sometimes used to model chemical concentrations for which some observations are reported as "Below Detection Limit". See, for example USEPA (1992c, pp.27-34). In most cases, however, the zero-modified lognormal (delta) distribution will be more appropriate, since chemical concentrations are bounded below at 0 (e.g., Gilliom and Helsel, 1986; Owen and DeRouen, 1980).

Once you estimate the parameters of the zero-modified normal distribution, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with a confidence interval.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see `qqPlotCensored`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J. (1955). On the Distribution of a Positive Random Variable Having a Discrete Probability Mass at the Origin. *Journal of the American Statistical Association* **50**, 901–908.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135–146.

Owen, W., and T. DeRouen. (1980). Estimation of the Mean for Lognormal Data Containing Zeros and Left-Censored Values, with Applications to the Measurement of Worker Exposure to Air Contaminants. *Biometrics* **36**, 707–719.

USEPA (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, Permits and State Programs Division, US Environmental Protection Agency, Washington, D.C.

See Also

ZeroModifiedNormal, Normal, ezmlnorm, ZeroModifiedLognormal, estimate.object.

Examples

```
# Generate 100 observations from a zero-modified normal distribution
# with mean=4, sd=2, and p.zero=0.5, then estimate the parameters.
# According to equations (1) and (2) above, the overall mean is
# mean.zmnorm=2 and the overall standard deviation is sd.zmnorm=sqrt(6).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rzmnorm(100, mean = 4, sd = 2, p.zero = 0.5)
ezmnorm(dat, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Zero-Modified Normal
#
#Estimated Parameter(s):          mean        = 4.037732
#                                 sd          = 1.917004
#                                 p.zero      = 0.450000
#                                 mean.zmnorm = 2.220753
#                                 sd.zmnorm   = 2.465829
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     100
#
#Confidence Interval for:         mean.zmnorm
```

```
#
#Confidence Interval Method:      Normal Approximation
#                                 (t Distribution)
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 1.731417
#                                 UCL = 2.710088

#----------

# Following Example 9 on page 34 of USEPA (1992c), compute an
# estimate of the mean of the zinc data, assuming a
# zero-modified normal distribution. The data are stored in
# EPA.92c.zinc.df.

head(EPA.92c.zinc.df)
#  Zinc.orig  Zinc Censored Sample Well
#1        <7  7.00     TRUE      1    1
#2     11.41 11.41    FALSE      2    1
#3        <7  7.00     TRUE      3    1
#4        <7  7.00     TRUE      4    1
#5        <7  7.00     TRUE      5    1
#6     10.00 10.00    FALSE      6    1

New.Zinc <- EPA.92c.zinc.df$Zinc
New.Zinc[EPA.92c.zinc.df$Censored] <- 0
ezmnorm(New.Zinc, ci = TRUE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Zero-Modified Normal
#
#Estimated Parameter(s):          mean        = 11.891000
#                                 sd          =  1.594523
#                                 p.zero      =  0.500000
#                                 mean.zmnorm =  5.945500
#                                 sd.zmnorm   =  6.123235
#
#Estimation Method:               mvue
#
#Data:                            New.Zinc
#
#Sample Size:                     40
#
#Confidence Interval for:         mean.zmnorm
#
#Confidence Interval Method:      Normal Approximation
#                                 (t Distribution)
#
```

```
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 3.985545
#                                 UCL = 7.905455

#----------

# Clean up
rm(dat, New.Zinc)
```

FcnsByCat *EnvStats Functions Listed by Category*

Description

Hyperlink list of **EnvStats** functions by category.

Details

- Calibration
- Censored Data
- Data Transformations
- Estimating Distribution Parameters
- Estimating Distribution Quantiles
- Goodness-of-Fit Tests
- Hypothesis Tests
- Monte Carlo Simulation and Risk Assessment
- Plotting Probability Distributions
- Plotting Using ggplot2
- Power and Sample Size Calculations
- Prediction Intervals
- Printing and Plotting Methods
- Probability Distributions and Random Numbers
- Summary Statistics
- Tolerance Intervals
- Trend Analysis

`FcnsByCatCalibration` *EnvStats Functions for Calibration*

Description

The **EnvStats** functions listed below are useful for performing calibration and inverse prediction to determine the concentration of a chemical based on a machine signal.

Details

Function Name	*Description*
`anovaPE`	Compute lack-of-fit and pure error ANOVA table for a linear model.
`calibrate`	Fit a calibration line or curve.
`detectionLimitCalibrate`	Determine detection limit based on using a calibration line (or curve) and inverse regression.
`inversePredictCalibrate`	Predict concentration using a calibration line (or curve) and inverse regression.
`pointwise`	Pointwise confidence limits for predictions.
`predict.lm`	Predict method for linear model fits.

`FcnsByCatCensoredData` *EnvStats Functions for Censored Data*

Description

The **EnvStats** functions listed below are useful for dealing with Type I censored data.

Details

Data Transformations

Function Name	*Description*
`boxcoxCensored`	Compute values of an objective for Box-Cox Power transformations, or compute optimal transformation, for Type I censored data.
`print.boxcoxCensored`	Print an object of class `"boxcoxCensored"`.
`plot.boxcoxCensored`	Plot an object of class `"boxcoxCensored"`.

Estimating Distribution Parameters

Function Name	*Description*
egammaCensored	Estimate shape and scale parameters for a gamma distribution based on Type I censored data.
egammaAltCensored	Estimate mean and CV for a gamma distribution based on Type I censored data.
elnormCensored	Estimate parameters for a lognormal distribution (log-scale) based on Type I censored data.
elnormAltCensored	Estimate parameters for a lognormal distribution (original scale) based on Type I censored data.
enormCensored	Estimate parameters for a Normal distribution based on Type I censored data.
epoisCensored	Estimate parameter for a Poisson distribution based on Type I censored data.
enparCensored	Estimate the mean and standard deviation nonparametrically.
gpqCiNormSinglyCensored	Generate the generalized pivotal quantity used to construct a confidence interval for the mean of a Normal distribution based on Type I singly censored data.
gpqCiNormMultiplyCensored	Generate the generalized pivotal quantity used to construct a confidence interval for the mean of a Normal distribution based on Type I multiply censored data.
print.estimateCensored	Print an object of class "estimateCensored".

Estimating Distribution Quantiles

Function Name	*Description*
eqlnormCensored	Estimate quantiles of a Lognormal distribution (log-scale) based on Type I censored data, and optionally construct a confidence interval for a quantile.
eqnormCensored	Estimate quantiles of a Normal distribution based on Type I censored data, and optionally construct a confidence interval for a quantile.

All of the functions for computing quantiles (and associated confidence intervals) for complete (uncensored) data are listed in the help file Estimating Distribution Quantiles. All of these functions,

with the exception of eqnpar, will accept an object of class "estimateCensored". Thus, you may estimate quantiles (and construct *approximate* confidence intervals) for any distribution for which:

1. There exists a function to estimate distribution parameters using censored data (see the section **Estimating Distribution Parameters** above).
2. There exists a function to estimate quantiles for that distribution based on complete data (see the help file Estimating Distribution Quantiles).

Nonparametric estimates of quantiles (and associated confidence intervals) can be constructed from censored data as long as the order statistics used in the results are above all left-censored observations or below all right-censored observations. See the help file for eqnpar for more information and examples.

Goodness-of-Fit Tests

Function Name	*Description*
gofTestCensored	Perform a goodness-of-fit test based on Type I left- or right-censored data.
print.gofCensored	Print an object of class "gofCensored".
plot.gofCensored	Plot an object of class "gofCensored".

Hypothesis Tests

Function Name	*Description*
twoSampleLinearRankTestCensored	Perform two-sample linear rank tests based on censored data.
print.htestCensored	Printing method for object of class "htestCensored".

Plotting Probability Distributions

Function Name	*Description*
cdfCompareCensored	Plot two cumulative distribution functions based on Type I censored data.
ecdfPlotCensored	Plot an empirical cumulative distribution function based on Type I censored data.
ppointsCensored	Compute plotting positions for Type I censored data.
qqPlotCensored	Produce quantile-quantile (Q-Q) plots, also called probability plots, based on Type I censored data.

Prediction and Tolerance Intervals

Function Name	*Description*
gpqTolIntNormSinglyCensored	Generate the generalized pivotal quantity used to construct a tolerance interval for a Normal distribution based on Type I singly censored data.
gpqTolIntNormMultiplyCensored	Generate the generalized pivotal quantity used to construct a tolerance interval for a Normal distribution based on Type I multiply censored data.
tolIntLnormCensored	Tolerance interval for a lognormal distribution (log-scale) based on Type I censored data.
tolIntNormCensored	Tolerance interval for a Normal distribution based on Type I censored data.

All of the functions for computing prediction and tolerance intervals for complete (uncensored) data are listed in the help files Prediction Intervals and Tolerance Intervals. All of these functions, with the exceptions of predIntNpar and tolIntNpar, will accept an object of class "estimateCensored". Thus, you may construct *approximate* prediction or tolerance intervals for any distribution for which:

1. There exists a function to estimate distribution parameters using censored data (see the section **Estimating Distribution Parameters** above).
2. There exists a function to create a prediction or tolerance interval for that distribution based on complete data (see the help files Prediction Intervals and Tolerance Intervals).

Nonparametric prediction and tolerance intervals can be constructed from censored data as long as the order statistics used in the results are above all left-censored observations or below all right-censored observations. See the help files for predIntNpar, predIntNparSimultaneous, and tolIntNpar for more information and examples.

FcnsByCatDataTrans *EnvStats Functions for Data Transformations*

Description

The **EnvStats** functions listed below are useful for deciding on data transformations.

Details

Function Name	*Description*
boxcox	Compute values of an objective for Box-Cox transformations, or compute optimal transformation based on raw observations or residuals from a linear model.
boxcoxTransform	Apply a Box-Cox Power transformation to a set of data.
plot.boxcox	Plotting method for an object of class "boxcox".
plot.boxcoxLm	Plotting method for an object of class "boxcoxLm".
print.boxcox	Printing method for an object of class "boxcox".
print.boxcoxLm	Printing method for an object of class "boxcoxLm".

FcnsByCatEstDistParams

EnvStats Functions for Estimating Distribution Parameters

Description

The **EnvStats** functions listed below are useful for estimating distribution parameters and optionally constructing confidence intervals.

Details

Function Name	*Description*
ebeta	Estimate parameters of a Beta distribution
ebinom	Estimate parameter of a Binomial distribution
eexp	Estimate parameter of an Exponential distribution
eevd	Estimate parameters of an Extreme Value distribution
egamma	Estimate shape and scale parameters of a Gamma distribution
egammaAlt	Estimate mean and CV parameters of a Gamma distribution
egevd	Estimate parameters of a Generalized Extreme Value distribution
egeom	Estimate parameter of a Geometric distribution
ehyper	Estimate parameter of a Hypergeometric distribution
elogis	Estimate parameters of a Logistic distribution
elnorm	Estimate parameters of a Lognormal distribution (log-scale)
elnormAlt	Estimate parameters of a Lognormal distribution (original scale)
elnorm3	Estimate parameters of a Three-Parameter Lognormal distribution
enbinom	Estimate parameter of a Negative Binomial distribution
enorm	Estimate parameters of a Normal distribution
enpar	Estimate Mean, Standard Deviation, and Standard Error Nonparametrically
epareto	Estimate parameters of a Pareto distribution
epois	Estimate parameter of a Poisson distribution
eunif	Estimate parameters of a Uniform distribution

eweibull	Estimate parameters of a Weibull distribution
ezmlnorm	Estimate parameters of a Zero-Modified Lognormal (Delta) distribution (log-Scale)
ezmlnormAlt	Estimate parameters of a Zero-Modified Lognormal (Delta) distribution (original Scale)
ezmnorm	Estimate parameters of a Zero-Modified Normal distribution

---

FcnsByCatEstDistQuants

EnvStats Functions for Estimating Distribution Quantiles

Description

The **EnvStats** functions listed below are useful for estimating distribution quantiles and, for some functions, optionally constructing confidence intervals for a quantile.

Details

Function Name	*Description*
eqbeta	Estimate quantiles of a Beta distribution.
eqbinom	Estimate quantiles of a Binomial distribution.
eqexp	Estimate quantiles of an Exponential distribution.
eqevd	Estimate quantiles of an Extreme Value distribution.
eqgamma	Estimate quantiles of a Gamma distribution using the Shape and Scale Parameterization, and optionally construct a confidence interval for a quantile.
eqgammaAlt	Estimate quantiles of a Gamma distribution using the mean and CV Parameterization, and optionally construct a confidence interval for a quantile.
eqgevd	Estimate quantiles of a Generalized Extreme Value distribution.
eqgeom	Estimate quantiles of a Geometric distribution.
eqhyper	Estimate quantiles of a Hypergeometric distribution.
eqlogis	Estimate quantiles of a Logistic distribution.
eqlnorm	Estimate quantiles of a Lognormal distribution (log-scale), and optionally construct a confidence interval for a quantile.
eqlnorm3	Estimate quantiles of a Three-Parameter Lognormal distribution.
eqnbinom	Estimate quantiles of a Negative Binomial distribution.
eqnorm	Estimate quantiles of a Normal distribution, and optionally construct a confidence interval for a quantile.
eqpareto	Estimate quantiles of a Pareto distribution.
eqpois	Estimate quantiles of a Poisson distribution, and optionally construct a confidence interval for a quantile.
equnif	Estimate quantiles of a Uniform distribution.

eqweibull	Estimate quantiles of a Weibull distribution.
eqzmlnorm	Estimate quantiles of a Zero-Modified Lognormal (Delta) distribution (log-scale).
eqzmlnormAlt	Estimate quantiles of a Zero-Modified Lognormal (Delta) distribution (original scale).
eqzmnorm	Estimate quantiles of a Zero-Modified Normal distribution.

FcnsByCatGOFTests *EnvStats Functions for Goodness-of-Fit Tests*

Description

The **EnvStats** functions listed below are useful for performing goodness-of-fit tests for user-specified probability distributions.

Details

Goodness-of-Fit Tests

Function Name	*Description*
gofTest	Perform a goodness-of-fit test for a specified probability distribution. The resulting object is of class "gof" unless the test is the two-sample Kolmogorov-Smirnov test, in which case the resulting object is of class "gofTwoSample".
plot.gof	S3 class method for plotting an object of class "gof".
print.gof	S3 class method for printing an object of class "gof".
plot.gofTwoSample	S3 class method for plotting an object of class "gofTwoSample".
print.gofTwoSample	S3 class method for printing an object of class "gofTwoSample".
gofGroupTest	Perform a goodness-of-fit test to determine whether data in a set of groups appear to all come from the same probability distribution (with possibly different parameters for each group). The resulting object is of class "gofGroup".
plot.gofGroup	S3 class method for plotting an object of class "gofGroup".
print.gofGroup	S3 class method for printing an object of class "gofGroup".

Tests for Outliers

Function Name	*Description*
rosnerTest	Perform Rosner's test for outliers assuming a normal (Gaussian) distribution.
print.gofOutlier	S3 class method for printing an object of class "gofOutlier".

Choose a Distribution

Function Name	*Description*
distChoose	Choose best fitting distribution based on goodness-of-fit tests.
print.distChoose	S3 class method for printing an object of class "distChoose".

FcnsByCatHypothTests *EnvStats Functions for Hypothesis Tests*

Description

The **EnvStats** functions listed below are useful for performing hypothesis tests not already built into R. See Power and Sample Size Calculations for a list of functions you can use to perform power and sample size calculations based on various hypothesis tests.

Details

For goodness-of-fit tests, see Goodness-of-Fit Tests.

Function Name	*Description*
chenTTest	Chen's modified one-sided t-test for skewed distributions.
kendallTrendTest	Nonparametric test for monotonic trend based on Kendall's tau statistic (and optional confidence interval for slope).
kendallSeasonalTrendTest	Nonparametric test for monotonic trend within each season based on Kendall's tau statistic (and optional confidence interval for slope).
oneSamplePermutationTest	Fisher's one-sample randomization (permutation) test for location.
quantileTest	Two-sample rank test to detect a shift in a proportion of the "treated" population.
quantileTestPValue	Compute p-value associated with a specified combination of m, n, r and k for the quantile test. Useful for determining r and k for a given significance level α.
serialCorrelationTest	Test for the presence of serial correlation.
signTest	One- or paired-sample sign test on the median.
twoSampleLinearRankTest	Two-sample linear rank test to detect a shift in the "treated" population.
twoSamplePermutationTestLocation	Two-sample or paired-sample randomization (permutation) test for location.

twoSamplePermutationTestProportion	Randomization (permutation) test to compare two proportions (Fisher's exact test).
varTest	One-sample test on variance or two-sample test to compare variances.
varGroupTest	Test for homogeneity of variance among two or more groups.
zTestGevdShape	Estimate the shape parameter of a Generalized Extreme Value distribution and test the null hypothesis that the true value is equal to 0.

FcnsByCatMCandRisk *EnvStats Functions for Monte Carlo Simulation and Risk Assessment*

Description

The **EnvStats** functions listed below are useful for performing Monte Carlo simulations and risk assessment.

Details

Function Name	*Description*
Empirical	Empirical distribution based on a set of observations.
simulateVector	Simulate a vector of random numbers from a specified theoretical probability distribution or empirical probability distribution using either Latin hypercube sampling or simple random sampling.
simulateMvMatrix	Simulate a multivariate matrix of random numbers from specified theoretical probability distributions and/or empirical probability distributions based on a specified rank correlation matrix, using either Latin hypercube sampling or simple random sampling.

FcnsByCatPlotProbDists

EnvStats Functions for Plotting Probability Distributions

Description

The **EnvStats** functions listed below are useful for plotting probability distributions.

Details

Function Name	*Description*
cdfCompare	Plot two cumulative distribution functions with the same x-axis in order to compare them.
cdfPlot	Plot a cumulative distribution function.
ecdfPlot	Plot empirical cumulative distribution function.
epdfPlot	Plot empirical probability density function.
pdfPlot	Plot probability density function.
qqPlot	Produce a quantile-quantile (Q-Q) plot, also called a probability plot.
qqPlotGestalt	Plot several Q-Q plots from the same distribution in order to develop a Gestalt of Q-Q plots for that distribution.

FcnsByCatPlotUsingggplot2

EnvStats Functions for Creating Plots Using the ggplot2 Package

---

Description

The **EnvStats** functions listed below are useful for creating plots with the **ggplot2** package.

Details

Function Name	*Description*
geom_stripchart	Adaptation of the **EnvStats** function stripChart, used to create a strip plot using functions from the package **ggplot2**.
stat_n_text	Add text indicating the sample size to a **ggplot2** plot.
stat_mean_sd_text	Add text indicating the mean and standard deviation to a **ggplot2** plot.
stat_median_iqr_text	Add text indicating the median and interquartile range to a **ggplot2** plot.
stat_test_text	Add text indicating the results of a hypothesis test comparing groups to a **ggplot2** plot.

FcnsByCatPower *EnvStats Functions for Power and Sample Size Calculations*

Description

The **EnvStats** functions listed below are useful for power and sample size calculations.

Details

Confidence Intervals

Function Name	*Description*
ciTableProp	Confidence intervals for binomial proportion, or difference between two proportions, following Bacchetti (2010)
ciBinomHalfWidth	Compute the half-width of a confidence interval for a Binomial proportion or the difference between two proportions.
ciBinomN	Compute the sample size necessary to achieve a specified half-width of a confidence interval for a Binomial proportion or the difference between two proportions.
plotCiBinomDesign	Create plots for a sampling design based on a confidence interval for a Binomial proportion or the difference between two proportions.
ciTableMean	Confidence intervals for mean of normal distribution, or difference between two means, following Bacchetti (2010)
ciNormHalfWidth	Compute the half-width of a confidence interval for the mean of a Normal distribution or the difference between two means.
ciNormN	Compute the sample size necessary to achieve a specified half-width of a confidence interval for the mean of a Normal distribution or the difference between two means.
plotCiNormDesign	Create plots for a sampling design based on a confidence interval for the mean of a Normal distribution or the difference between two means.
ciNparConfLevel	Compute the confidence level associated with a nonparametric confidence interval for a percentile.
ciNparN	Compute the sample size necessary to achieve a specified confidence level for a nonparametric confidence interval for a percentile.
plotCiNparDesign	Create plots for a sampling design based on a nonparametric confidence interval for a percentile.

Hypothesis Tests

Function Name	*Description*
aovN	Compute the sample sizes necessary to achieve a specified power for a one-way fixed-effects analysis of variance test.
aovPower	Compute the power of a one-way fixed-effects analysis of

	variance test.
plotAovDesign	Create plots for a sampling design based on a one-way analysis of variance.
propTestN	Compute the sample size necessary to achieve a specified power for a one- or two-sample proportion test.
propTestPower	Compute the power of a one- or two-sample proportion test.
propTestMdd	Compute the minimal detectable difference associated with a one- or two-sample proportion test.
plotPropTestDesign	Create plots involving sample size, power, difference, and significance level for a one- or two-sample proportion test.
tTestAlpha	Compute the Type I Error associated with specified values for for power, sample size(s), and scaled MDD for a one- or two-sample t-test.
tTestN	Compute the sample size necessary to achieve a specified power for a one- or two-sample t-test.
tTestPower	Compute the power of a one- or two-sample t-test.
tTestScaledMdd	Compute the scaled minimal detectable difference associated with a one- or two-sample t-test.
plotTTestDesign	Create plots for a sampling design based on a one- or two-sample t-test.
tTestLnormAltN	Compute the sample size necessary to achieve a specified power for a one- or two-sample t-test, assuming lognormal data.
tTestLnormAltPower	Compute the power of a one- or two-sample t-test, assuming lognormal data.
tTestLnormAltRatioOfMeans	Compute the minimal or maximal detectable ratio of means associated with a one- or two-sample t-test, assuming lognormal data.
plotTTestLnormAltDesign	Create plots for a sampling design based on a one- or two-sample t-test, assuming lognormal data.
linearTrendTestN	Compute the sample size necessary to achieve a specified power for a t-test for linear trend.
linearTrendTestPower	Compute the power of a t-test for linear trend.
linearTrendTestScaledMds	Compute the scaled minimal detectable slope for a t-test for linear trend.
plotLinearTrendTestDesign	Create plots for a sampling design based on a t-test for linear trend.

Prediction Intervals

Normal Distribution Prediction Intervals

Function Name	*Description*
predIntNormHalfWidth	Compute the half-width of a prediction interval for a normal distribution.
predIntNormK	Compute the required value of K for a prediction interval for a Normal distribution.

predIntNormN	Compute the sample size necessary to achieve a specified half-width for a prediction interval for a Normal distribution.
plotPredIntNormDesign	Create plots for a sampling design based on the width of a prediction interval for a Normal distribution.
predIntNormTestPower	Compute the probability that at least one future observation (or mean) falls outside a prediction interval for a Normal distribution.
plotPredIntNormTestPowerCurve	Create plots for a sampling design based on a prediction interval for a Normal distribution.
predIntNormSimultaneousTestPower	Compute the probability that at least one set of future observations (or means) violates the given rule based on a simultaneous prediction interval for a Normal distribution.
plotPredIntNormSimultaneousTestPowerCurve	Create plots for a sampling design based on a simultaneous prediction interval for a Normal distribution.

Lognormal Distribution Prediction Intervals

Function Name	*Description*
predIntLnormAltTestPower	Compute the probability that at least one future observation (or geometric mean) falls outside a prediction interval for a lognormal distribution.
plotPredIntLnormAltTestPowerCurve	Create plots for a sampling design based on a prediction interval for a lognormal distribution.
predIntLnormAltSimultaneousTestPower	Compute the probability that at least one set of future observations (or geometric means) violates the given rule based on a simultaneous prediction interval for a lognormal distribution.
plotPredIntLnormAltSimultaneousTestPowerCurve	Create plots for a sampling design based on a simultaneous prediction interval for a lognormal distribution.

Nonparametric Prediction Intervals

Function Name	*Description*
predIntNparConfLevel	Compute the confidence level associated with a nonparametric prediction interval.
predIntNparN	Compute the required sample size to achieve a specified confidence level for a nonparametric prediction interval.
plotPredIntNparDesign	Create plots for a sampling design based on the confidence level and sample size of a nonparametric prediction interval.
predIntNparSimultaneousConfLevel	Compute the confidence level associated with a simultaneous nonparametric prediction interval.
predIntNparSimultaneousN	Compute the required sample size for a simultaneous nonparametric prediction interval.
plotPredIntNparSimultaneousDesign	Create plots for a sampling design based on a simultaneous nonparametric prediction interval.
predIntNparSimultaneousTestPower	Compute the probability that at least one set of future observations violates the given rule based on a nonparametric simultaneous prediction interval.
plotPredIntNparSimultaneousTestPowerCurve	Create plots for a sampling design based on a simultaneous nonparametric prediction interval.

Tolerance Intervals

Function Name	*Description*
tolIntNormHalfWidth	Compute the half-width of a tolerance interval for a normal distribution.
tolIntNormK	Compute the required value of K for a tolerance interval for a Normal distribution.
tolIntNormN	Compute the sample size necessary to achieve a specified half-width for a tolerance interval for a Normal distribution.
plotTolIntNormDesign	Create plots for a sampling design based on a tolerance interval for a Normal distribution.
tolIntNparConfLevel	Compute the confidence level associated with a nonparametric tolerance interval for a specified sample size and coverage.
tolIntNparCoverage	Compute the coverage associated with a nonparametric tolerance interval for a specified sample size and confidence level.
tolIntNparN	Compute the sample size required for a nonparametric

	tolerance interval with a specified coverage and confidence level.
plotTolIntNparDesign	Create plots for a sampling design based on a nonparametric tolerance interval.

FcnsByCatPredInts *EnvStats Functions for Prediction Intervals*

Description

The **EnvStats** functions listed below are useful for computing prediction intervals and simultaneous prediction intervals. See Power and Sample Size for a list of functions useful for computing power and sample size for a design based on a prediction interval width, or a design based on a hypothesis test for future observations falling outside of a prediciton interval.

Details

Function Name	*Description*
predIntGamma, predIntGammaAlt	Prediction interval for the next k observations or next set of k means for a Gamma distribution.
predIntGammaSimultaneous, predIntGammaAltSimultaneous	Construct a simultaneous prediction interval for the next r sampling occasions based on a Gamma distribution.
predIntLnorm, predIntLnormAlt	Prediction interval for the next k observations or geometric means from a Lognormal distribution.
predIntLnormSimultaneous, predIntLnormAltSimultaneous	Construct a simultaneous prediction interval for the next r sampling occasions based on a Lognormal distribution.
predIntNorm	Prediction interval for the next k observations or means from a Normal (Gaussian) distribution.
predIntNormK	Compute the value of K for a prediction interval for a Normal distribution.
predIntNormSimultaneous	Construct a simultaneous prediction interval for the next r sampling occasions based on a Normal distribution.
predIntNormSimultaneousK	Compute the value of K for a simultaneous prediction interval for the next r sampling occasions based on a Normal distribution.
predIntNpar	Nonparametric prediction interval for the next k

	of K observations.
predIntNparSimultaneous	Construct a nonparametric simultaneous prediction interval for the next r sampling occasions.
predIntPois	Prediction interval for the next k observations or sums from a Poisson distribution.

FcnsByCatPrintPlot *EnvStats Functions for Printing and Plotting Objects of Various S3 Classes*

Description

The **EnvStats** functions listed below are printing and plotting methods for various S3 classes.

Details

Printing Methods

Function Name	*Description*
print.boxcox	Print an object that inherits from class "boxcox".
print.boxcoxCensored	Print an object that inherits from class "boxcoxCensored".
print.boxcoxLm	Print an object that inherits from class "boxcoxLm".
print.estimate	Print an object that inherits from class "estimate".
print.estimateCensored	Print an object that inherits from class "estimateCensored".
print.gof	Print an object that inherits from class "gof".
print.gofCensored	Print an object that inherits from class "gofCensored".
print.gofGroup	Print an object that inherits from class "gofGroup".
print.gofTwoSample	Print an object that inherits from class "gofTwoSample".
print.htest	Print an object that inherits from class "htest".
print.htestCensored	Print an object that inherits from class "htestCensored".
print.permutationTest	Print an object that inherits from class "permutationTest".
print.summaryStats	Print an object that inherits from class "summaryStats".

Plotting Methods

Function Name	*Description*
plot.boxcox	Plot an object that inherits from class "boxcox".
plot.boxcoxCensored	Plot an object that inherits from class "boxcoxCensored".
plot.boxcoxLm	Plot an object that inherits from class "boxcoxLm".
plot.gof	Plot an object that inherits from class "gof".
plot.gofCensored	Plot an object that inherits from class "gofCensored".
plot.gofGroup	Plot an object that inherits from class "gofGroup".
plot.gofTwoSample	Plot an object that inherits from class "gofTwoSample".
plot.permutationTest	Plot an object that inherits from class "permutationTest".

FcnsByCatProbDists *EnvStats Probability Distributions and Random Numbers*

Description

Listed below are all of the probability distributions available in R and **EnvStats**. Distributions with a description in **bold** are new ones that are part of **EnvStats**. For each distribution, there are functions for generating: values for the probability density function, values for the cumulative distribution function, quantiles, and random numbers.

The data frame Distribution.df contains information about all of these probability distributions.

Details

Distribution Abbreviation	*Description*
beta	Beta distribution.
binom	Binomial distribution.
cauchy	Cauchy distribution.
chi	**Chi distribution**.
chisq	Chi-squared distribution.
exp	Exponential distribution.
evd	**Extreme value distribution**.
f	F-distribution.
gamma	Gamma distribution.
gammAlt	**Gamma distribution parameterized with mean and CV**.
gevd	**Generalized extreme value distribution**.
geom	Geometric distribution.
hyper	Hypergeometric distribution.
logis	Logistic distribution.

lnorm	Lognormal distribution.
lnormAlt	**Lognormal distribution parameterized with mean and CV**.
lnormMix	**Mixture of two lognormal distributions**.
lnormMixAlt	**Mixture of two lognormal distributions parameterized by their means and CVs**.
lnorm3	**Three-parameter lognormal distribution**.
lnormTrunc	**Truncated lognormal distribution**.
lnormTruncAlt	**Truncated lognormal distribution parameterized by mean and CV**.
nbinom	Negative binomial distribution.
norm	Normal distribution.
normMix	**Mixture of two normal distributions**.
normTrunc	**Truncated normal distribution**.
pareto	**Pareto distribution**.
pois	Poisson distribution.
t	Student's t-distribution.
tri	**Triangular distribution**.
unif	Uniform distribution.
weibull	Weibull distribution.
wilcox	Wilcoxon rank sum distribution.
zmlnorm	**Zero-modified lognormal (delta) distribution**.
zmlnormAlt	**Zero-modified lognormal (delta) distribution parameterized with mean and CV**.
zmnorm	**Zero-modified normal distribution**.

In addition, the functions evNormOrdStats and evNormOrdStatsScalar compute expected values of order statistics from a standard normal distribution.

FcnsByCatSumStats *EnvStats Functions for Summary Statistics and Plots*

Description

The **EnvStats** functions listed below create summary statistics and plots.

Details

Summary Statistics

R comes with several functions for computing summary statistics, including mean, var, median, range, quantile, and summary. The following functions in **EnvStats** complement these R functions.

Function Name	*Description*
cv	Coefficient of variation
geoMean	Geometric mean

geoSD	Geometric standard deviation
iqr	Interquartile range
kurtosis	Kurtosis
lMoment	L-moments
pwMoment	Probability-weighted moments
skewness	Skew
summaryFull	Extensive summary statistics
summaryStats	Summary statistics

Summary Plots

R comes with several functions for creating plots to summarize data, including hist, barplot, boxplot, dotchart, stripchart, and numerous others.

The help file Plotting Probability Distributions lists several **EnvStats** functions useful for producing summary plots as well.

In addition, the **EnvStats** function stripChart is a modification of stripchart that allows you to include summary statistics on the plot itself.

Finally, the help file Plotting Using ggplot2 lists several **EnvStats** functions for adding information to plots produced with the ggplot function, including the function geom_stripchart, which is an adaptation of the **EnvStats** function stripChart.

FcnsByCatTolInts *EnvStats Functions for Tolerance Intervals*

Description

The **EnvStats** functions listed below are useful for computing tolerance intervals. See Power and Sample Size for a list of functions useful for computing power and sample size for a design based on a tolerance interval width.

Details

Function Name	*Description*
tolIntGamma, tolIntGammaAlt	Tolerance interval for a Gamma distribution.
tolIntLnorm, tolIntLnormAlt	Tolerance interval for a lognormal distribution.
tolIntNorm	Tolerance interval for a Normal (Gaussian) distribution.
tolIntNormK	Compute the constant K for a Normal (Gaussian) tolerance interval.
tolIntNpar	Nonparametric tolerance interval.

tolIntPois Tolerance interval for a Poisson distribution.

FcnsByCatTrend *EnvStats Functions for Trend Analysis*

Description

See Hypothesis Tests.

GammaAlt *The Gamma Distribution (Alternative Parameterization)*

Description

Density, distribution function, quantile function, and random generation for the gamma distribution with parameters mean and cv.

Usage

```
dgammaAlt(x, mean, cv = 1, log = FALSE)
pgammaAlt(q, mean, cv = 1, lower.tail = TRUE, log.p = FALSE)
qgammaAlt(p, mean, cv = 1, lower.tail = TRUE, log.p = FALSE)
rgammaAlt(n, mean, cv = 1)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
mean	vector of (positive) means of the distribution of the random variable.
cv	vector of (positive) coefficients of variation of the random variable.
log, log.p	logical; if TRUE, probabilities/densities p are returned as $log(p)$.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Details

Let X be a random variable with a gamma distribution with parameters `shape`=α and `scale`=β. The relationship between these parameters and the mean (`mean`=μ) and coefficient of variation (`cv`=τ) of this distribution is given by:

$$\alpha = \tau^{-2} \quad (1)$$

$$\beta = \mu/\alpha \quad (2)$$

$$\mu = \alpha\beta \quad (3)$$

$$\tau = \alpha^{-1/2} \quad (4)$$

Thus, the functions `dgammaAlt`, `pgammaAlt`, `qgammaAlt`, and `rgammaAlt` call the R functions `dgamma`, `pgamma`, `qgamma`, and `rgamma`, respectively, using the values for the `shape` and `scale` parameters given by: `shape <- cv^-2`, `scale <- mean/shape`.

Value

`dgammaAlt` gives the density, `pgammaAlt` gives the distribution function, `qgammaAlt` gives the quantile function, and `rgammaAlt` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

Note

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distribution. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter α tends to infinity or the cv parameter τ tends to 0.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) discourage using the assumption of a lognormal distribution for some types of environmental data and recommend instead assessing whether the data appear to fit a gamma distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). *Statistical Distributions, Fourth Edition*. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Singh, A., A.K. Singh, and R.J. Iaci. (2002). Estimation of the Exposure Point Concentration Term Using a Gamma Distribution. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

See Also

GammaDist, `egammaAlt`, Probability Distributions and Random Numbers.

Examples

```
# Density of a gamma distribution with parameters mean=10 and cv=2,
# evaluated at 7:

dgammaAlt(7, mean = 10, cv = 2)
#[1] 0.02139335

#----------

# The cdf of a gamma distribution with parameters mean=10 and cv=2,
# evaluated at 12:

pgammaAlt(12, mean = 10, cv = 2)
#[1] 0.7713307

#----------

# The 25'th percentile of a gamma distribution with parameters
# mean=10 and cv=2:

qgammaAlt(0.25, mean = 10, cv = 2)
#[1] 0.1056871

#----------

# A random sample of 4 numbers from a gamma distribution with
# parameters mean=10 and cv=2.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(10)
rgammaAlt(4, mean = 10, cv = 2)
#[1] 3.772004230 1.889028078 0.002987823 8.179824976
```

---

geoMean *Geometric Mean*

Description

Compute the sample geometric mean.

Usage

```
geoMean(x, na.rm = FALSE)
```

Arguments

x	numeric vector of observations.
na.rm	logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.

Details

If x contains any non-positive values (values less than or equal to 0), geoMean returns NA and issues a warning.

Let $\underline{x}$ denote a vector of n observations from some distribution. The sample geometric mean is a measure of central tendency. It is defined as:

$$\bar{x}_G = \sqrt[n]{x_1 x_2 \ldots x_n} = [\prod_{i=1}^{n} x_i]^{1/n} \qquad (1)$$

that is, it is the n'th root of the product of all n observations.

An equivalent way to define the geometric mean is by:

$$\bar{x}_G = exp[\frac{1}{n}\sum_{i=1}^{n} log(x_i)] = e^{\bar{y}} \qquad (2)$$

where

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad (3)$$

$$y_i = log(x_i), \quad i = 1, 2, \ldots, n \qquad (4)$$

That is, the sample geometric mean is antilog of the sample mean of the log-transformed observations.

The geometric mean is only defined for positive observations. It can be shown that the geometric mean is less than or equal to the sample arithmetic mean with equality only when all of the observations are the same value.

Value

A numeric scalar – the sample geometric mean.

Note

The geometric mean is sometimes used to average ratios and percent changes (Zar, 2010). For the lognormal distribution, the geometric mean is the maximum likelihood estimator of the ***median*** of the distribution, although it is sometimes used incorrectly to estimate the mean of the distribution (see the NOTE section in the help file for elnormAlt).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Taylor, J.K. (1990). *Statistical Techniques for Data Analysis*. Lewis Publishers, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

geoSD, summaryFull, Summary Statistics, mean, median.

Examples

```
# Generate 20 observations from a lognormal distribution with parameters
# mean=10 and cv=2, and compute the mean, median, and geometric mean.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormAlt(20, mean = 10, cv = 2)

mean(dat)
#[1] 5.339273

median(dat)
#[1] 3.692091

geoMean(dat)
#[1] 4.095127

#----------
# Clean up
rm(dat)
```

---

geom_stripchart *1-D Scatter Plots with Confidence Intervals Using ggplot2*

Description

geom_stripchart is an adaptation of the **EnvStats** function stripChart and is used to create a strip plot using functions from the package ggplot2. geom_stripchart produces one-dimensional scatter plots (also called dot plots), along with text indicating sample size and estimates of location (mean or median) and scale (standard deviation or interquartile range), as well as confidence intervals for the population location parameter, and results of a hypothesis test comparing group locations.

Usage

```
geom_stripchart(..., seed = 47, paired = FALSE, paired.lines = paired,
  group = NULL, x.nudge = if (paired && paired.lines) c(-0.3, 0.3) else 0.3,
  text.box = FALSE, location = "mean", ci = "normal", digits = 1,
  digit.type = "round", nsmall = ifelse(digit.type == "round", digits, 0),
  jitter.params = list(), point.params = list(), line.params = list(),
  location.params = list(), errorbar.params = list(),
  n.text = TRUE, n.text.box = text.box, n.text.params = list(),
  location.scale.text = TRUE, location.scale.text.box = text.box,
  location.scale.text.params = list(), test.text = FALSE,
  test.text.box = text.box,
  test = ifelse(location == "mean", "parametric", "nonparametric"),
  test.text.params = list())
```

Arguments

...	Arguments that can be passed on layer, as well as to the geoms and stats used to create the strip chart, including geom_jitter, geom_point, geom_line, stat_summary, geom_errorbar, stat_n_text, stat_mean_sd_text, stat_median_iqr_text, and stat_test_text.
seed	For the case of non-paired data, the argument seed is passed to the R function set.seed. Because jittering depends on the R random number generator, using the same value of seed each time the same data are plotted with geom_stripchart ensures that the resulting plot is the same. The default value is set.seed=47.
paired	For the case of two groups, a logical scalar indicating whether the data should be considered to be paired. The default value is paired=FALSE. When paired=TRUE, you must also supply the argument group (see below) to indicate which variable to use to identify the pairing (e.g., a variable that indicates which observations were taken before treatment and which observations were taken after treatment). NOTE: if the argument test.text.params is supplied and it includes a component named paired, the value of that component is overriden by the value of the argument paired.
paired.lines	For the case when there are two groups and the observations are paired (i.e., paired=TRUE), a logical scalar indicating whether to draw lines between the paired observations. The default value is the value of paired.
group	For the case when there are two groups and the observations are paired (i.e., paired=TRUE), a character string indicating the variable that identifies the pair-

ing (e.g., a variable that indicates which observations were taken before treatment and which observations were taken after treatment).

`x.nudge` A numeric scalar indicating the amount to move the estimates of location and confidence interval lines on the x-axis to the right or left of the actual x value. This is used to avoid overlapping with the actual data. For the case of two groups and paired observations, the default value is `x.nudge=c(-0.3, 0.3)`, which draws estimates of location and confidence interval lines to the left by 0.3 for the first group and to the right by 0.3 for the second group. Otherwise, the default value is `x.nudge=0.3`.

`text.box` A logical scalar indicating whether to surround text indicating sample size, location/scale estimates, and test results with text boxes (i.e., whether to use `link[ggplot2]{geom_label}` instead of `geom_text`). This is a convenience argument and is simply a way to simultaneously set the values of `n.text.box`, `location.scale.text.box`, and `test.text.box`. You can override this argument for sample size, location/scale, and/or test results by supplying the arguments `n.text.box`, `location.scale.text.box`, and `test.text.box` individually (see below).

`location` A character string indicating whether to display the mean for each group (`location="mean"`, the default) or the median for each group (`location="median"`).

`ci` For the case when `location="mean"`, a character string indicating what kind of confidence interval to plot for each group. Possible values are `ci="normal"` (standard confidence interval based on Student's t-distribution; the default), or `ci="boot"` (confidence interval based on the bootstrap).

NOTE: For the case when `location="median"`, quantiles are plotted. By default, the 25'th and 75'th percentiles are plotted, but this can be changed via including a component called `conf.int` in the argument `error.bar.params`. See the help file for `smedian.hilow` for details.

`digits` Integer indicating the number of digits to use for displaying text indicating the location and scale estimates and, for the case of one or two groups, the number of digits to use for displaying text indicating the confidence interval associated with the test of hypothesis. When `digit.type="round"` (see below) the argument `digits` indicates the number of digits to round to, and when `digit.type="signif"` the argument `digits` indicates the number of significant digits to display. The default value is `digits=1`.

For location/scale estimates, you can override the value of this argument by including a component named `digits` in the argument `location.scale.text.params`, and for confidence interval limits, you can override the value of this argument by including a component named `location.digits` in the argument `test.text.params`.

`digit.type` Character string indicating whether the `digits` argument (see above) refers to significant digits (`digit.type="signif"`), or how many decimal places to round to (`digit.type="round"`, the default).

For location/scale estimates, you can override the value of this argument by including a component named `digit.type` in the argument `location.scale.text.params`, and for confidence interval limits, you can override the value of this argument by including a component named `location.digit.type` in the argument `test.text.params`.

`nsmall` Integer passed to the function `format` indicating the the minimum number of digits to the right of the decimal point for the computed estimates of location and

scale and the confidence interval associated with the test of hypothesis. The default value is nsmall=digits when digit.type="round" and nsmall=0 when digit.type="signif". When nsmall is greater than 0, all computed estimates of location and scale and all confidence interval limits will have the same number of digits to the right of the decimal point (including, possibly, trailing zeros). To omit trailing zeros, set nsmall=0.

jitter.params — A list containing arguments to the function geom_jitter that will be used to plot the points. The default value is an empty list: jitter.params=list(), in which case the default values for geom_jitter are used, except for the following modifications: pch=1, width=0.15, height=0.

This argument is ignored when there are two groups and both paired=TRUE and paired.lines=TRUE (see the explanation for point.params and line.params below).

point.params — For the case when there are two groups and both paired=TRUE and paired.lines=TRUE, this is a list containing arguments to the function geom_point that will be used to plot the points. The default value is an empty list: point.params=list(), in which case the default values for geom_point are used, except for the following modification: pch=1.

line.params — For the case when there are two groups and both paired=TRUE and paired.lines=TRUE, this is a list containing arguments to the function geom_line that will be used to draw lines between the paired observations. The default value is an empty list: line.params=list(), in which case the default values for geom_line are used, except for the following modification: color="gray".

location.params

A list containing arguments to the function stat_summary that will be used to plot the estimates of location, i.e., stat_summary is called with fun=location and geom="point" where location is the location argument to geom_stripchart described above. The default value is an empty list: location.params=list(), in which case the default values for stat_summary are used, except for the following modifications: size=2, position=position_nudge(x = x.nudge)), where x.nudge is the x.nudge argument to geom_stripchart described above.

errorbar.params

A list containing arguments to the function stat_summary that will be used to plot the confidence interval, i.e., stat_summary is called with fun.data=fun.data and geom="errorbar", where fun.data has the value "mean_cl_normal" or "mean_cl_boot" depending of the value of the argument ci described above, or, in the case when location="median" fun.data has the value "median_hilow". The default value is an empty list: errorbar.params=list(), in which case the default values for stat_summary are used, except for the following modifications:
fun.args = list(conf.int = ifelse(location == "mean", 0.95, 0.5)), size = 0.75, width = 0.075, position = position_nudge(x = x.nudge)), where x.nudge is the x.nudge argument to geom_stripchart described above.

n.text — A logical scalar indicating whether to display the sample size for each group. The default is n.text=TRUE.

n.text.box — A logical scalar indicating whether to surround the text indicating the sample size for each group with a text box (i.e., whether to use geom_label instead of

geom_text). The default is n.text.box=text.box (see above for the argument text.box).

n.text.params A list containing arguments to the function stat_n_text that will be used to display text indicating the sample size for each group. The default value is an empty list: n.text.params=list(), in which case the default values for stat_n_text are used.

location.scale.text
A logical scalar indicating whether to display text indicating the location and scale for each group. The default is location.scale.text=TRUE.

location.scale.text.box
A logical scalar indicating whether to surround the text indicating the location and scale for each group with a text box (i.e., whether to use geom_label instead of geom_text). The default is location.scale.text.box=text.box (see above for the argument text.box).

location.scale.text.params
A list containing arguments to the function stat_mean_sd_text (when location="mean") or arguments to the function stat_median_iqr_text (when location="median") that will be used to display text indicating the location and scale for each group. The default value is an empty list: location.scale.text.params=list(), in which case the default values for stat_mean_sd_text or stat_mean_sd_text are used.

test.text A logical scalar indicating whether to display the results of the hypthesis test comparing groups. The default is test.text=FALSE.

test.text.box A logical scalar indicating whether to surround the text indicating the results of the hypothesis test comparing groups with a text box (i.e., whether to use geom_label instead of geom_text). The default is test.text.box=text.box (see above for the argument text.box).

test A character string indicating whether to use a standard parametric test (test="parametric", the default when location="mean") or nonparametric test (test="nonparametric", the default when location="median") to compare groups.

test.text.params
A list containing arguments to the function stat_test_text that will be used to display text indicating the results of the hypothesis test to compare groups. The default value is an empty list: test.text.params=list(), in which case the default values for stat_test_text are used, with the exception that the value of the argument test in the call to stat_test_text is determined by the value of the argument test supplied to geom_stripchart (see above).

Details

See the vignette **Extending ggplot2** at https://cran.r-project.org/package=ggplot2/vignettes/extending-ggplot2.html and Chapter 12 of Wickham (2016) for information on how to create a new geom.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis (Use R!)*. Second Edition. Springer.

See Also

stat_n_text, stat_mean_sd_text, stat_median_iqr_text, stat_test_text, geom_jitter, geom_point, geom_line, stat_summary, geom_text, geom_label.

Examples

```
  # First, load and attach the ggplot2 package.
  #-------------------------------------------

  library(ggplot2)

  #==========

  #---------------------
  # 3 Independent Groups
  #---------------------

  # Example 1:

  # Using the built-in data frame mtcars,
  # create a stipchart of miles per gallon vs. number of cylinders
  # using different colors for each level of the number of cylinders.
  #-------------------------------------------------------------------

  p <- ggplot(mtcars, aes(x = factor(cyl), y = mpg, color = factor(cyl)))

  p + geom_stripchart() +
    labs(x = "Number of Cylinders", y = "Miles per Gallon")

  #==========
## Not run:
  # Example 2:

  # Repeat Example 1, but include the results of the
  # standard parametric analysis of variance.
  #-----------------------------------------------

  dev.new()
  p + geom_stripchart(test.text = TRUE) +
    labs(x = "Number of Cylinders", y = "Miles per Gallon")

  #==========

  # Example 3:

  # Using Example 2, show explicitly the layering
  # process that geom_stripchart is using.
```

```
#
# This plot should look identical to the previous one.
#---------------------------------------------------

set.seed(47)
dev.new()
p + theme(legend.position = "none") +
  geom_jitter(pch = 1, width = 0.15, height = 0) +
  stat_summary(fun = "mean", geom = "point",
    size = 2, position = position_nudge(x = 0.3)) +
  stat_summary(fun.data = "mean_cl_normal", geom = "errorbar",
    size = 0.75, width = 0.075, position = position_nudge(x = 0.3)) +
  stat_n_text() +
  stat_mean_sd_text() +
  stat_test_text() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 4:

# Repeat Example 2, but put all text in a text box.
#--------------------------------------------------

dev.new()
p + geom_stripchart(text.box = TRUE, test.text = TRUE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 5:

# Repeat Example 2, but put just the test results
# in a text box.
#-----------------------------------------------

dev.new()
p + geom_stripchart(test.text = TRUE, test.text.box = TRUE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 6:

# Repeat Example 2, but:
# 1) plot the median and IQR instead of the mean and the 95
# 2) show text for the median and IQR, and
# 3) use the nonparametric test to compare groups.
#
# Note that following what the ggplot2 stat_summary function
# does when you specify a "confidence interval" for the
# median (i.e., when you call stat_summary with the arguments
# geom="errorbar" and fun.data="median_hilow"), the displayed
```

```
# error bars show intervals based on estimated quuantiles.
# By default, stat_summary with the arguments
# geom="errorbar" and fun.data="median_hilow" displays
# error bars using the 2.5'th and 97.5'th percentiles.
# The function geom_stripchart, however, by default
# displays error bars using the 25'th and 75'th percentiles
# (see the explanation for the argument ci above).
#-----------------------------------------------------------

dev.new()
p + geom_stripchart(location = "median", test.text = TRUE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Clean up
#---------

graphics.off()
rm(p)

#========================================

#---------------------
# 2 Independent Groups
#---------------------

# Example 7:

# Repeat Example 2, but use only the groups with
# 4 and 8 cylinders.
#-----------------------------------------------

dev.new()
p <- ggplot(subset(mtcars, cyl %in% c(4, 8)),
  aes(x = factor(cyl), y = mpg, color = cyl))

p + geom_stripchart(test.text = TRUE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 8:

# Repeat Example 7, but
# 1) facet by transmission type
# 2) make the text smaller
# 3) put the text for the test results in a text box
#    and make them blue.

dev.new()
p + geom_stripchart(test.text = TRUE, test.text.box = TRUE,
      n.text.params = list(size = 3),
```

```
      location.scale.text.params = list(size = 3),
      test.text.params = list(size = 3, color = "blue")) +
  facet_wrap(~ am, labeller = label_both) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Clean up
#---------

graphics.off()
rm(p)

#========================================

#---------------------
# 2 Independent Groups
#---------------------

# Example 9:

# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area.  These data are strored
# in the data frame EPA.94b.tccb.df.
#
# First create one-dimensional scatterplots to compare the
# TcCB concentrations between the areas and use a nonparametric
# test to test for a difference between areas.

dev.new()
p <- ggplot(EPA.94b.tccb.df, aes(x = Area, y = TcCB, color = Area))

p + geom_stripchart(location = "median", test.text = TRUE) +
  labs(y = "TcCB (ppb)")

#==========

# Example 10:

# Now log-transform the TcCB data and use a parametric test
# to compare the areas.

dev.new()
p <- ggplot(EPA.94b.tccb.df, aes(x = Area, y = log10(TcCB), color = Area))

p + geom_stripchart(test.text = TRUE) +
  labs(y = "log10[ TcCB (ppb) ]")

#==========

# Example 11:
```

```
# Repeat Example 10, but allow the variances to differ
# between Areas.
#-----------------------------------------------------

dev.new()
p + geom_stripchart(test.text = TRUE,
      test.text.params = list(test.arg.list = list(var.equal=FALSE))) +
  labs(y = "log10[ TcCB (ppb) ]")

#==========

# Clean up
#---------

graphics.off()
rm(p)

#=======================================

#--------------------
# Paired Observations
#--------------------

# Example 12:

# The data frame ACE.13.TCE.df contians paired observations of
# trichloroethylene (TCE; mg/L) at 10 groundwater monitoring wells
# before and after remediation.
#
# Create one-dimensional scatterplots to compare TCE concentrations
# before and after remediation and use a paired t-test to
# test for a difference between periods.

ACE.13.TCE.df
#   TCE.mg.per.L Well Period
#1        20.900    1 Before
#2         9.170    2 Before
#3         5.960    3 Before
#...      ......   .. ......
#18        0.520    8  After
#19        3.060    9  After
#20        1.900   10  After

dev.new()
p <- ggplot(ACE.13.TCE.df, aes(x = Period, y = TCE.mg.per.L, color = Period))

p + geom_stripchart(paired = TRUE, group = "Well", test.text = TRUE) +
  labs(y = "TCE (mg/L)")

#==========

# Example 13:
```

```
# Repeat Example 11, but use a one-sided alternative since
# remediation should decrease TCE concentration.
#-----------------------------------------------------------

dev.new()
p + geom_stripchart(paired = TRUE, group = "Well", test.text = TRUE,
  test.text.params = list(test.arg.list = list(alternative="less"))) +
  labs(y = "TCE (mg/L)")

#==========

# Clean up
#---------

graphics.off()
rm(p)

#========================================

#----------------------------------------
# Paired Observations, Nonparametric Test
#----------------------------------------

# Example 14:

# The data frame Helsel.Hirsch.02.Mayfly.df contains paired counts
# of mayfly nymphs above and below industrial outfalls in 12 streams.
#
# Create one-dimensional scatterplots to compare the
# counts between locations and use a nonparametric test
# to compare counts above and below the outfalls.

Helsel.Hirsch.02.Mayfly.df
#   Mayfly.Count Stream Location
#1            12      1    Above
#2            15      2    Above
#3            11      3    Above
#...         ...     ..    .....
#22           60     10    Below
#23           53     11    Below
#24          124     12    Below

dev.new()
p <- ggplot(Helsel.Hirsch.02.Mayfly.df,
  aes(x = Location, y = Mayfly.Count, color = Location))

p + geom_stripchart(location = "median", paired = TRUE,
    group = "Stream", test.text = TRUE) +
  labs(y = "Number of Mayfly Nymphs")

#==========
```

```
  # Clean up
  #---------

  graphics.off()
  rm(p)

## End(Not run)
```

---

geoSD *Geometric Standard Deviation.*

Description

Compute the sample geometric standard deviation.

Usage

```
geoSD(x, na.rm = FALSE, sqrt.unbiased = TRUE)
```

Arguments

x	numeric vector of observations.
na.rm	logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.
sqrt.unbiased	logical scalar specifying what method to use to compute the sample standard deviation of the log-transformed observations. If sqrt.unbiased=TRUE (the default), the square root of the unbiased estimator of variance is used, otherwise the method of moments estimator of standard deviation is used. See the DETAILS section for more information.

Details

If x contains any non-positive values (values less than or equal to 0), geoMean returns NA and issues a warning.

Let $\underline{x}$ denote a vector of n observations from some distribution. The sample geometric standard deviation is a measure of variability. It is defined as:

$$s_G = exp(s_y) \qquad (1)$$

where

$$s_y = [\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2]^{1/2} \qquad (2)$$

$$y_i = log(x_i), \;\; i = 1, 2, \ldots, n \qquad (3)$$

That is, the sample geometric standard deviation is the antilog of the sample standard deviation of the log-transformed observations.

The sample standard deviation of the log-transformed observations shown in Equation (2) is the square root of the unbiased estimator of variance. (Note that this estimator of standard deviation is not an unbiased estimator.) Sometimes, the square root of the method of moments estimator of variance is used instead:

$$s_y = [\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2]^{1/2} \qquad (4)$$

This is the estimator used in Equation (1) when `sqrt.unbiased=FALSE`.

Value

A numeric scalar – the sample geometric standard deviation.

Note

The geometric standard deviation is only defined for positive observations. It is usually computed only for observations that are assumed to have come from a lognormal distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY.

Leidel, N.A., K.A. Busch, and J.R. Lynch. (1977). *Occupational Exposure Sampling Strategy Manual*. U.S. Department of Health, Education, and Welfare, Public Health Service, Center for Disease Control, National Institute for Occupational Safety and Health, Cincinnati, Ohio 45226, January, 1977, pp.102–103.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Taylor, J.K. (1990). *Statistical Techniques for Data Analysis*. Lewis Publishers, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`geoMean`, Lognormal, `elnorm`, `summaryFull`, `Summary Statistics`.

Examples

```
# Generate 2000 observations from a lognormal distribution with parameters
# mean=10 and cv=1, which implies the standard deviation (on the original
# scale) is 10.  Compute the mean, geometric mean, standard deviation,
# and geometric standard deviation.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
```

```
dat <- rlnormAlt(2000, mean = 10, cv = 1)

mean(dat)
#[1] 10.23417

geoMean(dat)
#[1] 7.160154

sd(dat)
#[1] 9.786493

geoSD(dat)
#[1] 2.334358

#----------
# Clean up
rm(dat)
```

GEVD *The Generalized Extreme Value Distribution*

Description

Density, distribution function, quantile function, and random generation for the generalized extreme value distribution.

Usage

```
dgevd(x, location = 0, scale = 1, shape = 0)
pgevd(q, location = 0, scale = 1, shape = 0)
qgevd(p, location = 0, scale = 1, shape = 0)
rgevd(n, location = 0, scale = 1, shape = 0)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
location	vector of location parameters.
scale	vector of positive scale parameters.
shape	vector of shape parameters.

Details

Let X be a generalized extreme value random variable with parameters `location`=η, `scale`=θ, and `shape`=κ. When the shape parameter $\kappa = 0$, the generalized extreme value distribution reduces to the extreme value distribution. When the shape parameter $\kappa \neq 0$, the cumulative distribution function of X is given by:

$$F(x; \eta, \theta, \kappa) = exp\{-[1 - \kappa(x - \eta)/\theta]^{1/\kappa}\}$$

where $-\infty < \eta, \kappa < \infty$ and $\theta > 0$. When $\kappa > 0$, the range of x is:

$$-\infty < x \leq \eta + \theta/\kappa$$

and when $\kappa < 0$ the range of x is:

$$\eta + \theta/\kappa \leq x < \infty$$

The p^th quantile of X is given by:

$$x_p = \eta + \frac{\theta\{1 - [-log(p)]^\kappa\}}{\kappa}$$

Value

density (`devd`), probability (`pevd`), quantile (`qevd`), or random sample (`revd`) for the generalized extreme value distribution with location parameter(s) determined by `location`, scale parameter(s) determined by `scale`, and shape parameter(s) determined by `shape`.

Note

Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930's to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter $\kappa = 0$, the GEVD reduces to the Type I extreme value (Gumbel) distribution. (The function `zTestGevdShape` allows you to test the null hypothesis that the shape parameter is equal to 0.) When $\kappa > 0$, the GEVD is the same as the Type II extreme value distribution, and when $\kappa < 0$ it is the same as the Type III extreme value distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Jenkinson, A.F. (1955). The Frequency Distribution of the Annual Maximum (or Minimum) of Meteorological Events. *Quarterly Journal of the Royal Meteorological Society*, **81**, 158–171.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

egevd, zTestGevdShape, EVD, Probability Distributions and Random Numbers.

Examples

```
# Density of a generalized extreme value distribution with
# location=0, scale=1, and shape=0, evaluated at 0.5:

dgevd(.5)
#[1] 0.3307043

#----------

# The cdf of a generalized extreme value distribution with
# location=1, scale=2, and shape=0.25, evaluated at 0.5:

pgevd(.5, 1, 2, 0.25)
#[1] 0.2795905

#----------

# The 90'th percentile of a generalized extreme value distribution with
# location=-2, scale=0.5, and shape=-0.25:

qgevd(.9, -2, 0.5, -0.25)
#[1] -0.4895683

#----------

# Random sample of 4 observations from a generalized extreme value
# distribution with location=5, scale=2, and shape=1.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rgevd(4, 5, 2, 1)
#[1] 6.738692 6.473457 4.446649 5.727085
```

Gibbons.et.al.09.Alkilinity.vec

Alkilinity Data from Gibbons et al. (2009)

Description

Alkilinity concentrations (mg/L) in groundwater.

Usage

```
data(Gibbons.et.al.09.Alkilinity.vec)
```

Format

A numeric vector with 27 elements.

Source

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley & Sons, Hoboken. Table 5.5, p. 107.

Gibbons.et.al.09.Vinyl.Chloride.vec *Vinyl Chloride Data from Gibbons et al. (2009)*

Description

Vinyl chloride concentrations (mug/L) in groundwater from upgradient background monitoring wells.

Usage

```
data(Gibbons.et.al.09.Vinyl.Chloride.vec)
```

Format

A numeric vector with 34 elements.

Source

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley & Sons, Hoboken. Table 4.3, p. 87.

---

gof.object *S3 Class "gof"*

Description

Objects of S3 class `"gof"` are returned by the **EnvStats** function `gofTest` when just the `x` argument is supplied.

Details

Objects of S3 class `"gof"` are lists that contain information about the assumed distribution, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

Required Components

The following components must be included in a legitimate list of class "gof".

distribution — a character string indicating the name of the assumed distribution (see Distribution.df).

dist.abb — a character string containing the abbreviated name of the distribution (see Distribution.df).

distribution.parameters — a numeric vector with a names attribute containing the names and values of the estimated or user-supplied distribution parameters associated with the assumed distribution.

n.param.est — a scalar indicating the number of distribution parameters estimated prior to performing the goodness-of-fit test. The value of this component will be 0 if the parameters were supplied by the user.

estimation.method — a character string indicating the method used to compute the estimated parameters. The value of this component will depend on the available estimation methods (see Distribution.df). The value of this component will be NULL if the parameters were supplied by the user.

statistic — a numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.

sample.size — a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.

parameters — numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.

z.value — (except when test="chisq" or test="ks") numeric scalar containing the z-value associated with the goodness-of-fit statistic.

p.value — numeric scalar containing the p-value associated with the goodness-of-fit statistic.

alternative — character string indicating the alternative hypothesis.

method — character string indicating the name of the goodness-of-fit test (e.g., "Shapiro-Wilk GOF").

data — numeric vector containing the data actually used for the goodness-of-fit test (i.e., the original data without any missing or infinite values).

data.name — character string indicating the name of the data object used for the goodness-of-fit test.

bad.obs — numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the goodness-of-fit test.

NOTE: when the function gofTest is called with both arguments x and y and test="ks", it returns an object of class "gofTwoSample". No specific parametric distribution is assumed, so the value of the component distribution is "Equal" and the following components are omitted: dist.abb, distribution.parameters, n.param.est, estimation.method, and z.value.

Optional Components

The following components are included in the result of calling gofTest with the argument test="chisq" and may be used by the function plot.gof:

cut.points	numeric vector containing the cutpoints used to define the cells.
counts	numeric vector containing the observed number of counts for each cell.
expected	numeric vector containing the expected number of counts for each cell.
X2.components	numeric vector containing the contribution of each cell to the chi-square statistic.

Methods

Generic functions that have methods for objects of class "gof" include:
print, plot.

Note

Since objects of class "gof" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

gofTest, print.gof, plot.gof, Goodness-of-Fit Tests, Distribution.df, gofCensored.object.

Examples

```
# Create an object of class "gof", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- rnorm(20, mean = 3, sd = 2)

gof.obj <- gofTest(dat)

mode(gof.obj)
#[1] "list"

class(gof.obj)
#[1] "gof"

names(gof.obj)
```

```
# [1] "distribution"            "dist.abb"
# [3] "distribution.parameters" "n.param.est"
# [5] "estimation.method"       "statistic"
# [7] "sample.size"             "parameters"
# [9] "z.value"                 "p.value"
#[11] "alternative"             "method"
#[13] "data"                    "data.name"
#[15] "bad.obs"

gof.obj

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#
#Hypothesized Distribution:       Normal
#
#Estimated Parameter(s):          mean = 2.861160
#                                 sd   = 1.180226
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Test Statistic:                  W = 0.9640724
#
#Test Statistic Parameter:        n = 20
#
#P-value:                         0.6279872
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Normal Distribution.

#==========

# Extract the p-value
#--------------------

gof.obj$p.value
#[1] 0.6279872

#==========

# Plot the results of the test
#-----------------------------

dev.new()
plot(gof.obj)

#==========
```

```
# Clean up
#---------
rm(dat, gof.obj)
graphics.off()
```

---

gofCensored.object *S3 Class "gofCensored"*

Description

Objects of S3 class "gofCensored" are returned by the **EnvStats** function gofTestCensored.

Details

Objects of S3 class "gofCensored" are lists that contain information about the assumed distribution, the amount of censoring, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

Required Components

The following components must be included in a legitimate list of class "gofCensored".

distribution — a character string indicating the name of the assumed distribution (see Distribution.df).

dist.abb — a character string containing the abbreviated name of the distribution (see Distribution.df).

distribution.parameters — a numeric vector with a names attribute containing the names and values of the estimated or user-supplied distribution parameters associated with the assumed distribution.

n.param.est — a scalar indicating the number of distribution parameters estimated prior to performing the goodness-of-fit test. The value of this component will be 0 if the parameters were supplied by the user.

estimation.method — a character string indicating the method used to compute the estimated parameters. The value of this component will depend on the available estimation methods (see Distribution.df). The value of this component will be NULL if the parameters were supplied by the user.

statistic — a numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.

sample.size — a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.

censoring.side — character string indicating whether the data are left- or right-censored.

`censoring.levels`
: numeric scalar or vector indicating the censoring level(s).

`percent.censored`
: numeric scalar indicating the percent of non-missing observations that are censored.

`parameters`
: numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the `statistic` component.

`z.value`
: (except when `test="chisq"` or `test="ks"`) numeric scalar containing the z-value associated with the goodness-of-fit statistic.

`p.value`
: numeric scalar containing the p-value associated with the goodness-of-fit statistic.

`alternative`
: character string indicating the alternative hypothesis.

`method`
: character string indicating the name of the goodness-of-fit test (e.g., `"Shapiro-Wilk GOF"`).

`data.name`
: character string indicating the name of the data object used for the goodness-of-fit test.

`censored`
: logical vector indicating which observations are censored.

`censoring.name`
: character string indicating the name of the object used to indicate the censoring.

Optional Components

The following components are included when the argument `keep.data` is set to `TRUE` in the call to the function producing the object of class `"gofCensored"`.

`data`
: numeric vector containing the data actually used for the goodness-of-fit test (i.e., the original data without any missing or infinite values).

`censored`
: logical vector indicating the censoring status of the data actually used for the goodness-of-fit test.

The following component is included when the data object contains missing (`NA`), undefined (`NaN`) and/or infinite (`Inf`, `-Inf`) values.

`bad.obs`
: numeric scalar indicating the number of missing (`NA`), undefined (`NaN`) and/or infinite (`Inf`, `-Inf`) values that were removed from the data object prior to performing the goodness-of-fit test.

Methods

Generic functions that have methods for objects of class `"gofCensored"` include:
`print`, `plot`.

Note

Since objects of class `"gofCensored"` are lists, you may extract their components with the `$` and `[[` operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

gofTestCensored, print.gofCensored, plot.gofCensored, Censored Data, Goodness-of-Fit Tests, Distribution.df, gof.object.

Examples

```
# Create an object of class "gofCensored", then print it out.
#-----------------------------------------------------------

gofCensored.obj <- with(EPA.09.Ex.15.1.manganese.df,
  gofTestCensored(Manganese.ppb, Censored, test = "sf"))

mode(gofCensored.obj)
#[1] "list"

class(gofCensored.obj)
#[1] "gofCensored"

names(gofCensored.obj)
# [1] "distribution"            "dist.abb"
# [3] "distribution.parameters" "n.param.est"
# [5] "estimation.method"       "statistic"
# [7] "sample.size"             "censoring.side"
# [9] "censoring.levels"        "percent.censored"
#[11] "parameters"              "z.value"
#[13] "p.value"                 "alternative"
#[15] "method"                  "data"
#[17] "data.name"               "censored"
#[19] "censoring.name"          "bad.obs"

gofCensored.obj

#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#-------------------------------
#
#Test Method:                     Shapiro-Francia GOF
#                                 (Multiply Censored Data)
#
#Hypothesized Distribution:       Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          mean = 15.23508
#                                 sd   = 30.62812
#
```

```
#Estimation Method:              MLE
#
#Data:                           Manganese.ppb
#
#Censoring Variable:             Censored
#
#Sample Size:                    25
#
#Percent Censored:               24%
#
#Test Statistic:                 W = 0.8368016
#
#Test Statistic Parameters:      N     = 25.00
#                                DELTA =  0.24
#
#P-value:                        0.004662658
#
#Alternative Hypothesis:         True cdf does not equal the
#                                Normal Distribution.

#==========

# Extract the p-value
#--------------------

gofCensored.obj$p.value
#[1] 0.004662658

#==========

# Plot the results of the test
#-----------------------------

dev.new()
plot(gofCensored.obj)

#==========

# Clean up
#---------
rm(gofCensored.obj)
graphics.off()
```

`gofGroup.object` *S3 Class "gofGroup"*

Description

Objects of S3 class `"gofGroup"` are returned by the **EnvStats** function `gofGroupTest`.

Details

Objects of S3 class "gofGroup" are lists that contain information about the assumed distribution, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

Required Components

The following components must be included in a legitimate list of class "gofGroup".

distribution	a character string indicating the name of the assumed distribution (see Distribution.df).
dist.abb	a character string containing the abbreviated name of the distribution (see Distribution.df).
statistic	a numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.
sample.size	a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.
parameters	numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.
p.value	numeric scalar containing the p-value associated with the goodness-of-fit statistic.
alternative	character string indicating the alternative hypothesis.
method	character string indicating the name of the goodness-of-fit test (e.g., "Wilk-Shapiro GOF (Normal Scores)").
data.name	character string indicating the name of the data object used for the goodness-of-fit test.
grouping.variable	character string indicating the name of the variable defining the groups.
bad.obs	numeric vector indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from each group and the grouping variable prior to performing the goodness-of-fit test.
n.groups	numeric scalar containing the number of groups.
group.names	character vector containing the levels of the grouping variable, i.e., the names of each of the groups.
group.scores	numeric vector containing the individual statistics for each group.

Optional Component

The following component is included when gofGroupTest is called with a formula for the first argument and a data argument.

parent.of.data	character string indicating the name of the object supplied in the data argument.

Methods

Generic functions that have methods for objects of class "gofGroup" include:
print, plot.

Note

Since objects of class "gofGroup" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

gofGroupTest, print.gofGroup, plot.gofGroup, Goodness-of-Fit Tests, Distribution.df.

Examples

```
# Create an object of class "gofGroup", then print it out.

# Example 10-4 of USEPA (2009, page 10-20) gives an example of
# simultaneously testing the assumption of normality for nickel
# concentrations (ppb) in groundwater collected at 4 monitoring
# wells over 5 months.  The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

gofGroup.obj <- gofGroupTest(Nickel.ppb ~ Well,
  data = EPA.09.Ex.10.1.nickel.df)

mode(gofGroup.obj)
#[1] "list"

class(gofGroup.obj)
#[1] "gofGroup"

names(gofGroup.obj)
# [1] "distribution"      "dist.abb"          "statistic"
# [4] "sample.size"       "parameters"        "p.value"
# [7] "alternative"       "method"            "data.name"
#[10] "grouping.variable" "parent.of.data"    "bad.obs"
#[13] "n.groups"          "group.names"       "group.scores"

gofGroup.obj
#Results of Group Goodness-of-Fit Test
#--------------------------------------
#
#Test Method:                     Wilk-Shapiro GOF (Normal Scores)
#
#Hypothesized Distribution:       Normal
#
#Data:                            Nickel.ppb
#
#Grouping Variable:               Well
#
#Data Source:                     EPA.09.Ex.10.1.nickel.df
#
```

```
#Number of Groups:                4
#
#Sample Sizes:                    Well.1 = 5
#                                 Well.2 = 5
#                                 Well.3 = 5
#                                 Well.4 = 5
#
#Test Statistic:                  z (G) = -3.658696
#
#P-values for
#Individual Tests:                Well.1 = 0.03510747
#                                 Well.2 = 0.02385344
#                                 Well.3 = 0.01120775
#                                 Well.4 = 0.10681461
#
#P-value for
#Group Test:                      0.0001267509
#
#Alternative Hypothesis:          At least one group
#                                 does not come from a
#                                 Normal Distribution.

#==========

# Extract the p-values
#---------------------

gofGroup.obj$p.value
#      Well.1       Well.2       Well.3       Well.4        z (G)
#0.0351074733 0.0238534406 0.0112077511 0.1068146088 0.0001267509

#==========

# Plot the results of the test
#-----------------------------

dev.new()
plot(gofGroup.obj)

#==========

# Clean up
#---------
rm(gofGroup.obj)
graphics.off()
```

---

gofGroupTest *Goodness-of-Fit Test for a Specified Probability Distribution for Groups*

Description

Perform a goodness-of-fit test to determine whether data in a set of groups appear to all come from the same probability distribution (with possibly different parameters for each group).

Usage

```
gofGroupTest(object, ...)

## S3 method for class 'formula'
gofGroupTest(object, data = NULL, subset,
  na.action = na.pass, ...)

## Default S3 method:
gofGroupTest(object, group, test = "sw",
  distribution = "norm", est.arg.list = NULL, n.classes = NULL,
  cut.points = NULL, param.list = NULL,
  estimate.params = ifelse(is.null(param.list), TRUE, FALSE),
  n.param.est = NULL, correct = NULL, digits = .Options$digits,
  exact = NULL, ws.method = "normal scores",
  data.name = NULL, group.name = NULL, parent.of.data = NULL,
  subset.expression = NULL, ...)

## S3 method for class 'data.frame'
gofGroupTest(object, ...)

## S3 method for class 'matrix'
gofGroupTest(object, ...)

## S3 method for class 'list'
gofGroupTest(object, ...)
```

Arguments

object	an object containing data for 2 or more groups to be compared to the hypothesized distribution specified by distribution. In the default method, the argument object must be a numeric vector. When object is a data frame, all columns must be numeric. When object is a matrix, it must be a numeric matrix. When object is a list, all components must be numeric vectors. In the formula method, a symbolic specification of the form y ~ g can be given, indicating the observations in the vector y are to be grouped according to the levels of the factor g. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
data	when object is a formula, data specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which summaryStats is called.

`subset` when `object` is a formula, `subset` specifies an optional vector specifying a subset of observations to be used.

`na.action` when `object` is a formula, `na.action` specifies a function which indicates what should happen when the data contain `NA`s. The default is `na.pass`.

`group` when `object` is a numeric vector, `group` is a factor or character vector indicating which group each observation belongs to. When `object` is a matrix or data frame this argument is ignored and the columns define the groups. When `object` is a list this argument is ignored and the components define the groups. When `object` is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.

`test` character string defining which goodness-of-fit test to perform on each group. Possible values are: `"sw"` (Shapiro-Wilk; the default), `"sf"` (Shapiro-Francia), `"ppcc"` (Probability Plot Correlation Coefficient), `"skew"` (Zero-skew), `"chisq"` (Chi-squared), `"ks"` (Kolmogorov-Smirnov), and `"ws"` (Wilk-Shapiro test for Uniform [0, 1] distribution).

`distribution` a character string denoting the distribution abbreviation. See the help file for `Distribution.df` for a list of distributions and their abbreviations. The default value is `distribution="norm"` (Normal distribution).

When `test="sw"`, `test="sf"`, or `test="ppcc"`, any continuous distribuiton is allowed (e.g., `"norm"` (normal), `"lnorm"` (lognormal), `"gamma"` (gamma), etc.), as well as mixed distributions involving the normal distribution (i.e., `"zmnorm"` (zero-modified normal), `"zmlnorm"` (zero-modified lognormal (delta)), and `"zmlnorm.alt"` (zero-modified lognormal with alternative parameterization)).

When `test="skew"`, only the values `"norm"` (normal), `"lnorm"` (lognormal), `"lnorm.alt"` (lognormal with alternative parameterization), `"zmnorm"` (zero-modified normal), `"zmlnorm"` (zero-modified lognormal (delta)), and `"zmlnorm.alt"` (zero-modified lognormal with alternative parameterization) are allowed.

When `test="ks"`, any continuous distribution is allowed.

When `test="chisq"`, any distribuiton is allowed.

When `test="ws"`, this argument is ignored.

`est.arg.list` a list of arguments to be passed to the function estimating the distribution parameters for each group of observations. For example, if `test="sw"` and `distribution="gamma"`, setting `est.arg.list=list(method="bcmle")` indicates using the bias-corrected maximum-likelihood estimators of shape and scale (see the help file for `egamma`. See the help file Estimating Distribution Parameters for a list of estimating functions. The default value is `est.arg.list=NULL` so that all default values for the estimating function are used. This argument is ignored if `estimate.params=FALSE`.

When `test="sw"`, `test="sf"`, `test="ppcc"`, or `test="skew"`, and you are testing for some form of normality (i.e., Normal, Lognormal, Three-Parameter Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta)), the estimated parameters are provided in the output merely for information, and the choice of the method of estimation has no effect on the goodness-of-fit test statistics or p-values.

When test="ks", and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="chisq" and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="ws", this argument is ignored.

n.classes — for the case when test="chisq", the number of cells into which the observations within each group are to be allocated. If the argument cut.points is supplied, then n.classes is set to length(cut.points)-1. The default value is
ceiling(2* (length(x)^(2/5))) and is recommended by Moore (1986).

cut.points — for the case when test="chisq", a vector of cutpoints that defines the cells for each group of observations. The element x[i] is allocated to cell j if cut.points[j] < x[i] $\leq$ cut.points[j+1]. If x[i] is less than or equal to the first cutpoint or greater than the last cutpoint, then x[i] is treated as missing. If the hypothesized distribution is discrete, cut.points must be supplied. The default value is cut.points=NULL, in which case the cutpoints are determined by n.classes equi-probable intervals.

param.list — for the case when test="ks" or test="chisq", a list with values for the parameters of the specified distribution. See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. The default value is NULL, which forces estimation of the distribution parameters. This argument is ignored if estimate.params=TRUE.

estimate.params — for the case when test="ks" or test="chisq", a logical scalar indicating whether to perform the goodness-of-fit test based on estimating the distribution parameters (estimate.params=TRUE) or using the user-supplied distribution parameters specified by param.list (estimate.params=FALSE). The default value of estimate.params is TRUE if param.list=NULL, otherwise it is FALSE.

n.param.est — for the case when test="ks" or test="chisq", an integer indicating the number of parameters estimated from the data.
If estimate.params=TRUE, the default value is the number of parameters associated with the distribution specified by distribution (e.g., 2 for a normal distribution). If estimate.params=FALSE, the default value is n.param.est=0.

correct — for the case when test="chisq", a logical scalar indicating whether to use the continuity correction. The default value is correct=FALSE unless n.classes=2.

digits — a scalar indicating how many significant digits to print out for the parameters associated with the hypothesized distribution. The default value is .Options$digits.

exact — for the case when test="ks", exact=NULL by default, but can be set to a logical scalar indicating whether an exact p-value should be computed. See the help file for ks.test for more information.

`ws.method` character string indicating which method to use when performing the Wilk-Shapiro test for a Uniform [0,1] distribution on the p-values from the goodness-of-fit tests on each group. Possible values are `ws.method="normal scores"` (the default) or `ws.method="chi-square scores"`. See the subsection *Wilk-Shapiro goodness-of-fit test for Uniform [0, 1] distribution* under the DETAILS section of the help file for `gofTest` for more information.

NOTE: In the case where you are testing whether each group comes from a Uniform [0,1] distribution (i.e., when you set `test="ws"`), the argument `ws.method` determines which score types are used for each individual test of the groups as well.

`data.name` character string indicating the name of the data used for the goodness-of-fit tests. The default value is `data.name=deparse(substitute(object))`.

`group.name` character string indicating the name of the data used to create the groups. The default value is `group.name=deparse(substitute(group))`.

`parent.of.data` character string indicating the source of the data used for the goodness-of-fit tests.

`subset.expression` character string indicating the expression used to subset the data.

`...` additional arguments affecting the goodness-of-fit test.

Details

The function `gofGroupTest` performs a goodness-of-fit test for each group of data by calling the function `gofTest`. Using the p-values from these goodness-of-fit tests, it then calls the function `gofTest` with the argument `test="ws"` to test whether the p-values appear to come from a Uniform [0,1] distribution.

Value

a list of class `"gofGroup"` containing the results of the group goodness-of-fit test. Objects of class `"gofGroup"` have special printing and plotting methods. See the help file for `gofGroup.object` for details.

Note

The Wilk-Shapiro (1968) tests for a Uniform [0, 1] distribution were introduced in the context of testing whether several independent samples all come from normal distributions, with possibly different means and variances. The function `gofGroupTest` extends this idea to allow you to test whether several independent samples come from the same distribution (e.g., gamma, extreme value, etc.), with possibly different parameters.

Examples of simultaneously assessing whether several groups come from the same distribution are given in USEPA (2009) and Gibbons et al. (2009).

In practice, almost any goodness-of-fit test will *not* reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test *will* reject the null hypothesis if the number of observations is very large, since "real" data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution of "real" data is close enough to some theoretical distribution that fairly accurate results may be provided by

assuming that particular theoretical distribution. One way to asses the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see qqPlot).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.17-17.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wilk, M.B., and S.S. Shapiro. (1968). The Joint Assessment of Normality of Several Independent Samples. *Technometrics*, **10**(4), 825-839.

See Also

gofTest, gofGroup.object, print.gofGroup, plot.gofGroup, qqPlot.

Examples

```
# Example 10-4 of USEPA (2009, page 10-20) gives an example of
# simultaneously testing the assumption of normality for nickel
# concentrations (ppb) in groundwater collected at 4 monitoring
# wells over 5 months.  The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

EPA.09.Ex.10.1.nickel.df
#   Month   Well Nickel.ppb
#1      1 Well.1       58.8
#2      3 Well.1        1.0
#3      6 Well.1      262.0
#4      8 Well.1       56.0
#5     10 Well.1        8.7
#6      1 Well.2       19.0
#7      3 Well.2       81.5
#8      6 Well.2      331.0
#9      8 Well.2       14.0
#10    10 Well.2       64.4
#11     1 Well.3       39.0
#12     3 Well.3      151.0
#13     6 Well.3       27.0
#14     8 Well.3       21.4
#15    10 Well.3      578.0
```

```
#16     1 Well.4        3.1
#17     3 Well.4      942.0
#18     6 Well.4       85.6
#19     8 Well.4       10.0
#20    10 Well.4      637.0


# Test for a normal distribution at each well:
#---------------------------------------------

gofGroup.list <- gofGroupTest(Nickel.ppb ~ Well,
  data = EPA.09.Ex.10.1.nickel.df)

gofGroup.list

#Results of Group Goodness-of-Fit Test
#-------------------------------------
#
#Test Method:                     Wilk-Shapiro GOF (Normal Scores)
#
#Hypothesized Distribution:       Normal
#
#Data:                            Nickel.ppb
#
#Grouping Variable:               Well
#
#Data Source:                     EPA.09.Ex.10.1.nickel.df
#
#Number of Groups:                4
#
#Sample Sizes:                    Well.1 = 5
#                                 Well.2 = 5
#                                 Well.3 = 5
#                                 Well.4 = 5
#
#Test Statistic:                  z (G) = -3.658696
#
#P-values for
#Individual Tests:                Well.1 = 0.03510747
#                                 Well.2 = 0.02385344
#                                 Well.3 = 0.01120775
#                                 Well.4 = 0.10681461
#
#P-value for
#Group Test:                      0.0001267509
#
#Alternative Hypothesis:          At least one group
#                                 does not come from a
#                                 Normal Distribution.

dev.new()
plot(gofGroup.list)
```

```
#----------

# Test for a lognormal distribution at each well:
#-----------------------------------------------

gofGroupTest(Nickel.ppb ~ Well, data = EPA.09.Ex.10.1.nickel.df,
  dist = "lnorm")

#Results of Group Goodness-of-Fit Test
#-------------------------------------
#
#Test Method:                     Wilk-Shapiro GOF (Normal Scores)
#
#Hypothesized Distribution:       Lognormal
#
#Data:                            Nickel.ppb
#
#Grouping Variable:               Well
#
#Data Source:                     EPA.09.Ex.10.1.nickel.df
#
#Number of Groups:                4
#
#Sample Sizes:                    Well.1 = 5
#                                 Well.2 = 5
#                                 Well.3 = 5
#                                 Well.4 = 5
#
#Test Statistic:                  z (G) = 0.2401720
#
#P-values for
#Individual Tests:                Well.1 = 0.6898164
#                                 Well.2 = 0.6700394
#                                 Well.3 = 0.3208299
#                                 Well.4 = 0.5041375
#
#P-value for
#Group Test:                      0.5949015
#
#Alternative Hypothesis:          At least one group
#                                 does not come from a
#                                 Lognormal Distribution.

#----------
# Clean up
rm(gofGroup.list)
graphics.off()
```

---

gofOutlier.object *S3 Class "gofOutlier"*

Description

Objects of S3 class "gofOutlier" are returned by the **EnvStats** function rosnerTest.

Details

Objects of S3 class "gofOutlier" are lists that contain information about the assumed distribution, the test statistics, the Type I error level, and the number of outliers detected.

Value

Required Components

The following components must be included in a legitimate list of class "gofOutlier".

distribution	a character string indicating the name of the assumed distribution (see Distribution.df).
statistic	a numeric vector with a names attribute containing the names and values of the outlier test statistic for each outlier tested.
sample.size	a numeric scalar containing the number of non-missing observations in the sample used for the outlier test.
parameters	numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.
alpha	numeric scalar indicating the Type I error level.
crit.value	numeric vector containing the critical values associated with the test for each outlier.
alternative	character string indicating the alternative hypothesis.
method	character string indicating the name of the outlier test.
data	numeric vector containing the data actually used for the outlier test (i.e., the original data without any missing or infinite values).
data.name	character string indicating the name of the data object used for the goodness-of-fit test.
all.stats	data frame containing all of the results of the test.

Optional Components

The following component is included when the data object contains missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values.

bad.obs	numeric scalar indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data object prior to performing the test for outliers.

Methods

Generic functions that have methods for objects of class "gofOutlier" include:
print.

Note

Since objects of class "gofOutlier" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

rosnerTest, print.gofOutlier, Goodness-of-Fit Tests.

Examples

```
# Create an object of class "gofOutlier", then print it out.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- c(rnorm(30, mean = 3, sd = 2), rnorm(3, mean = 10, sd = 1))

gofOutlier.obj <- rosnerTest(dat, k = 4)

mode(gofOutlier.obj)
#[1] "list"

class(gofOutlier.obj)
#[1] "gofOutlier"

names(gofOutlier.obj)
# [1] "distribution" "statistic"    "sample.size"  "parameters"
# [5] "alpha"        "crit.value"   "n.outliers"   "alternative"
# [9] "method"       "data"         "data.name"    "bad.obs"
#[13] "all.stats"

gofOutlier.obj

#Results of Outlier Test
#-------------------------
#
#Test Method:                     Rosner's Test for Outliers
#
#Hypothesized Distribution:       Normal
#
#Data:                            dat
#
#Sample Size:                     33
#
#Test Statistics:                 R.1 = 2.848514
#                                 R.2 = 3.086875
#                                 R.3 = 3.033044
```

```
#                                 R.4 = 2.380235
#
#Test Statistic Parameter:        k = 4
#
#Alternative Hypothesis:          Up to 4 observations are not
#                                 from the same Distribution.
#
#Type I Error:                    5%
#
#Number of Outliers Detected:     3
#
#  i   Mean.i     SD.i      Value Obs.Num    R.i+1 lambda.i+1 Outlier
#1 0 3.549744 2.531011 10.7593656      33 2.848514   2.951949    TRUE
#2 1 3.324444 2.209872 10.1460427      31 3.086875   2.938048    TRUE
#3 2 3.104392 1.856109  8.7340527      32 3.033044   2.923571    TRUE
#4 3 2.916737 1.560335 -0.7972275      25 2.380235   2.908473   FALSE

#==========

# Extract the data frame with all the test results
#-------------------------------------------------

gofOutlier.obj$all.stats
#  i   Mean.i     SD.i      Value Obs.Num    R.i+1 lambda.i+1 Outlier
#1 0 3.549744 2.531011 10.7593656      33 2.848514   2.951949    TRUE
#2 1 3.324444 2.209872 10.1460427      31 3.086875   2.938048    TRUE
#3 2 3.104392 1.856109  8.7340527      32 3.033044   2.923571    TRUE
#4 3 2.916737 1.560335 -0.7972275      25 2.380235   2.908473   FALSE

#==========

# Clean up
#---------
rm(dat, gofOutlier.obj)
```

`gofTest` *Goodness-of-Fit Test*

Description

Perform a goodness-of-fit test to determine whether a data set appears to come from a specified probability distribution or if two data sets appear to come from the same distribution.

Usage

```
gofTest(y, ...)

## S3 method for class 'formula'
gofTest(y, data = NULL, subset,
  na.action = na.pass, ...)
```

```
## Default S3 method:
gofTest(y, x = NULL,
  test = ifelse(is.null(x), "sw", "ks"),
  distribution = "norm", est.arg.list = NULL,
  alternative = "two.sided", n.classes = NULL,
  cut.points = NULL, param.list = NULL,
  estimate.params = ifelse(is.null(param.list), TRUE, FALSE),
  n.param.est = NULL, correct = NULL, digits = .Options$digits,
  exact = NULL, ws.method = "normal scores", warn = TRUE, keep.data = TRUE,
  data.name = NULL, data.name.x = NULL, parent.of.data = NULL,
  subset.expression = NULL, ...)
```

Arguments

y	an object containing data for the goodness-of-fit test. In the default method, the argument y must be numeric vector of observations. In the formula method, y must be a formula of the form y ~ 1 or y ~ x. The form y ~ 1 indicates use the observations in the vector y for a one-sample goodness-of-fit test. The form y ~ x is only relevant to the case of the two-sample Kolmogorov-Smirnov test (test="ks") and indicates use the observations in the vector y as the second sample and use the observations in the vector x as the first sample. Note that for the formula method, x and y must be the same length but this is not a requirement of the test and you can use vectors of different lengths via the default method. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
data	specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which gofTest is called.
subset	specifies an optional vector specifying a subset of observations to be used.
na.action	specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.
x	numeric vector of values for the first sample in the case of a two-sample Kolmogorov-Smirnov goodness-of-fit test (test="ks"). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
test	character string defining which goodness-of-fit test to perform. Possible values are: • "sw". Shapiro-Wilk; the default when x is NOT supplied. • "sf". Shapiro-Francia. • "ppcc". Probability Plot Correlation Coefficient. • "ad". Anderson-Darling. • "cmv". Cramer-von Mises. • "lillie". Lilliefor. • "skew". Zero-skew. • "chisq". Chi-squared.

- "ks". Kolmogorov-Smirnov; the default when x IS supplied.
- "ws". Wilk-Shapiro test for Uniform [0, 1] distribution.
- "proucl.ad.gamma". Anderson-Darling test for a gamma distribution using ProUCL critical values.
- "proucl.ks.gamma". Kolmogorov-Smirnov test for a gamma distribution using ProUCL critical values.

When the argument x is supplied, you must set test="ks", which is what gofTest does by default.

distribution a character string denoting the distribution abbreviation. See the help file for Distribution.df for a list of distributions and their abbreviations. The default value is distribution="norm" (Normal distribution).

When test="sw", test="sf", or test="ppcc", any continuous distribuiton is allowed (e.g., "norm" (normal), "lnorm" (lognormal), "gamma" (gamma), etc.), as well as mixed distributions involving the normal distribution (i.e., "zmnorm" (zero-modified normal), "zmlnorm" (zero-modified lognormal (delta)), and "zmlnormAlt" (zero-modified lognormal with alternative parameterization)).

When test="ad", test="cvm", test="lillie", or test="skew", only the values "norm" (normal), "lnorm" (lognormal), "lnormAlt" (lognormal with alternative parameterization), "zmnorm" (zero-modified normal), "zmlnorm" (zero-modified lognormal (delta)), and "zmlnormAlt" (zero-modified lognormal with alternative parameterization) are allowed.

When test="ks", any continuous distribution is allowed.

When test="chisq", any distribuiton is allowed.

When test="ws", this argument is ignored.

When test="proucl.ad.gamma" or test="proucl.ks.gamma", you must set distribution="gamma" or distribution="gammaAlt".

est.arg.list a list of arguments to be passed to the function estimating the distribution parameters. For example, if test="sw" and distribution="gamma", setting est.arg.list=list(method="bcmle") indicates using the bias-corrected maximum-likelihood estimators of shape and scale (see the help file for egamma). See the help file Estimating Distribution Parameters for a list of estimating functions. The default value is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE.

When test="sw", test="sf", test="ppcc", test="ad", test="cvm", test="lillie", or test="skew", and you are testing for some form of normality (i.e., Normal, Lognormal, Three-Parameter Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta)), the estimated parameters are provided in the output merely for information, and the choice of the method of estimation has no effect on the goodness-of-fit test statistic or p-value.

When test="ks", x is not supplied, and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="chisq" and estimate.params=TRUE, the estimated parameters are used to specify the null hypothesis of which distribution the data are assumed to come from.

When test="ws", test="proucl.ad.gamma", or test="proucl.ks.gamma", this argument is ignored.

alternative — for the case when test="ks", test="skew", or test="ws", character string specifying the alternative hypothesis. When test="ks" or test="skew", the possible values are "two-sided" (the default), "greater", or "less". When test="ws", the possible values are "greater" (the default), or "less". See the DETAILS section of the help file for ks.test for more explanation of the meaning of this argument.

n.classes — for the case when test="chisq", the number of cells into which the observations are to be allocated. If the argument cut.points is supplied, then n.classes is set to length(cut.points)-1. The default value is ceiling(2* (length(x)^(2/5))) and is recommended by Moore (1986).

cut.points — for the case when test="chisq", a vector of cutpoints that defines the cells. The element x[i] is allocated to cell j if cut.points[j] < x[i] $\leq$ cut.points[j+1]. If x[i] is less than or equal to the first cutpoint or greater than the last cutpoint, then x[i] is treated as missing. If the hypothesized distribution is discrete, cut.points must be supplied. The default value is cut.points=NULL, in which case the cutpoints are determined by n.classes equi-probable intervals.

param.list — for the case when test="ks" and x is not supplied, or when test="chisq", a list with values for the parameters of the specified distribution. See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. The default value is param.list=NULL, which forces estimation of the distribution parameters. This argument is ignored if estimate.params=TRUE.

estimate.params — for the case when test="ks" and x is not supplied, or when test="chisq", a logical scalar indicating whether to perform the goodness-of-fit test based on estimating the distribution parameters (estimate.params=TRUE) or using the user-supplied distribution parameters specified by param.list (estimate.params=FALSE). The default value of estimate.params is TRUE if param.list=NULL, otherwise it is FALSE.

n.param.est — for the case when test="ks" and x is not supplied, or when test="chisq", an integer indicating the number of parameters estimated from the data. If estimate.params=TRUE, the default value is the number of parameters associated with the distribution specified by distribution (e.g., 2 for a normal distribution). If estimate.params=FALSE, the default value is n.param.est=0.

correct — for the case when test="chisq", a logical scalar indicating whether to use the continuity correction. The default value is correct=FALSE unless n.classes=2.

digits — for the case when test="ks" and x is not supplied, or when test="chisq", and param.list is supplied, a scalar indicating how many significant digits to print out for the parameters associated with the hypothesized distribution. The default value is .Options$digits.

exact — for the case when test="ks", exact=NULL by default, but can be set to a logical scalar indicating whether an exact p-value should be computed. See the help file for ks.test for more information.

ws.method	for the case when test="ws", this argument specifies whether to perform the test based on normal scores (ws.method="normal scores", the default) or chi-square scores (ws.method="chi-square scores"). See the DETAILS section for more information.
warn	logical scalar indicating whether to print a warning message when observations with NAs, NaNs, or Infs in y or x are removed. The default value is warn=TRUE.
keep.data	logical scalar indicating whether to return the data used for the goodness-of-fit test. The default value is keep.data=TRUE.
data.name	character string indicating the name of the data used for argument y.
data.name.x	character string indicating the name of the data used for argument x.
parent.of.data	character string indicating the source of the data used for the goodness-of-fit test.
subset.expression	character string indicating the expression used to subset the data.
...	additional arguments affecting the goodness-of-fit test.

Details

- **Shapiro-Wilk Goodness-of-Fit Test** (test="sw").

  The Shapiro-Wilk goodness-of-fit test (Shapiro and Wilk, 1965; Royston, 1992a) is one of the most commonly used goodness-of-fit tests for normality. You can use it to test the following hypothesized distributions: Normal, Lognormal, Three-Parameter Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta). **In addition, you can also use it to test the null hypothesis of any continuous distribution that is available** (see the help file for Distribution.df, and see explanation below).

  Shapiro-Wilk W-Statistic and P-Value for Testing Normality

  Let X denote a random variable with cumulative distribution function (cdf) F. Suppose we want to test the null hypothesis that F is the cdf of a normal (Gaussian) distribution with some arbitrary mean μ and standard deviation σ against the alternative hypothesis that F is the cdf of some other distribution. The table below shows the random variable for which F is the assumed cdf, given the value of the argument distribution.

Value of distribution	Distribution Name	Random Variable for which F is the cdf
"norm"	Normal	X
"lnorm"	Lognormal (Log-space)	$log(X)$
"lnormAlt"	Lognormal (Untransformed)	$log(X)$
"lnorm3"	Three-Parameter Lognormal	$log(X - \gamma)$
"zmnorm"	Zero-Modified Normal	$X \mid X > 0$
"zmlnorm"	Zero-Modified Lognormal (Log-space)	$log(X) \mid X > 0$
"zmlnormAlt"	Zero-Modified Lognormal (Untransformed)	$log(X) \mid X > 0$

  Note that for the three-parameter lognormal distribution, the symbol γ denotes the threshold parameter.

Let $\underline{x} = (x_1, x_2, \ldots, x_n)$ denote the vector of n ***ordered*** observations assumed to come from a normal distribution.

The Shapiro-Wilk W-Statistic
Shapiro and Wilk (1965) introduced the following statistic to test the null hypothesis that F is the cdf of a normal distribution:

$$W = \frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad (1)$$

where the quantity a_i is the i'th element of the vector $\underline{a}$ defined by:

$$\underline{a} = \frac{\underline{m}^T V^{-1}}{[\underline{m}^T V^{-1} V^{-1} \underline{m}]^{1/2}} \qquad (2)$$

where T denotes the transpose operator, and $\underline{m}$ is the vector of expected values and V is the variance-covariance matrix of the order statistics of a random sample of size n from a standard normal distribution. That is, the values of $\underline{a}$ are the expected values of the standard normal order statistics weighted by their variance-covariance matrix, and normalized so that

$$\underline{a}^T \underline{a} = 1 \qquad (3)$$

It can be shown that the coefficients $\underline{a}$ are antisymmetric, that is,

$$a_i = -a_{n-i+1} \qquad (4)$$

and for odd n,

$$a_{(n+1)/2} = 0 \qquad (5)$$

Now because

$$\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i = 0 \qquad (6)$$

and

$$\sum_{i=1}^{n}(a_i - \bar{a})^2 = \sum_{i=1}^{n} a_i^2 = \underline{a}^T \underline{a} = 1 \qquad (7)$$

the W-statistic in Equation (1) is the same as the square of the sample product-moment correlation between the vectors $\underline{a}$ and $\underline{x}$:

$$W = r(\underline{a}, \underline{x})^2 \qquad (8)$$

where

$$r(\underline{x}, \underline{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{[\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2]^{1/2}} \qquad (9)$$

(see the R help file for `cor`).
The Shapiro-Wilk W-statistic is also simply the ratio of two estimators of variance, and can be rewritten as

$$W = \frac{\hat{\sigma}^2_{BLUE}}{\hat{\sigma}^2_{MVUE}} \qquad (10)$$

where the numerator is the square of the best linear unbiased estimate (BLUE) of the standard deviation, and the denominator is the minimum variance unbiased estimator (MVUE) of the variance:

$$\hat{\sigma}_{BLUE} = \frac{\sum_{i=1}^{n} a_i x_i}{\sqrt{n-1}} \qquad (11)$$

$$\hat{\sigma}^2_{MVUE} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1} \qquad (12)$$

Small values of W indicate the null hypothesis is probably not true. Shapiro and Wilk (1965) computed the values of the coefficients $\underline{a}$ and the percentage points for W (based on smoothing the empirical null distribution of W) for sample sizes up to 50. Computation of the W-statistic for larger sample sizes can be cumbersome, since computation of the coefficients $\underline{a}$ requires storage of at least $n + [n(n+1)/2]$ reals followed by $n \times n$ matrix inversion (Royston, 1992a).

The Shapiro-Francia W'-Statistic

Shapiro and Francia (1972) introduced a modification of the W-test that depends only on the expected values of the order statistics ($\underline{m}$) and not on the variance-covariance matrix (V):

$$W' = \frac{(\sum_{i=1}^{n} b_i x_i)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \qquad (13)$$

where the quantity b_i is the i'th element of the vector $\underline{b}$ defined by:

$$\underline{b} = \frac{\underline{m}}{[\underline{m}^T \underline{m}]^{1/2}} \qquad (14)$$

Several authors, including Ryan and Joiner (1973), Filliben (1975), and Weisberg and Bingham (1975), note that the W'-statistic is intuitively appealing because it is the squared Pearson correlation coefficient associated with a normal probability plot. That is, it is the squared correlation between the ordered sample values $\underline{x}$ and the expected normal order statistics $\underline{m}$:

$$W' = r(\underline{b}, \underline{x})^2 = r(\underline{m}, \underline{x})^2 \qquad (15)$$

Shapiro and Francia (1972) present a table of empirical percentage points for W' based on a Monte Carlo simulation. It can be shown that the asymptotic null distributions of W and W' are identical, but convergence is very slow (Verrill and Johnson, 1988).

The Weisberg-Bingham Approximation to the W'-Statistic

Weisberg and Bingham (1975) introduced an approximation of the Shapiro-Francia W'-statistic that is easier to compute. They suggested using Blom scores (Blom, 1958, pp.68–75) to approximate the element of $\underline{m}$:

$$\tilde{W}' = \frac{(\sum_{i=1}^{n} c_i x_i)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \qquad (16)$$

where the quantity c_i is the i'th element of the vector $\underline{c}$ defined by:

$$\underline{c} = \frac{\tilde{\underline{m}}}{[\tilde{\underline{m}}^T \tilde{\underline{m}}]^{1/2}} \qquad (17)$$

and

$$\tilde{m}_i = \Phi^{-1}[\frac{i - (3/8)}{n + (1/4)}] \qquad (18)$$

and Φ denotes the standard normal cdf. That is, the values of the elements of $\underline{m}$ in Equation (14) are replaced with their estimates based on the usual plotting positions for a normal distribution.

Royston's Approximation to the Shapiro-Wilk W-Test

Royston (1992a) presents an approximation for the coefficients $\underline{a}$ necessary to compute the Shapiro-Wilk W-statistic, and also a transformation of the W-statistic that has approximately a standard normal distribution under the null hypothesis.

Noting that, up to a constant, the components of $\underline{b}$ in Equation (14) and $\underline{c}$ in Equation (17) differ from those of $\underline{a}$ in Equation (2) mainly in the first and last two components, Royston (1992a) used the approximation $\underline{c}$ as the basis for approximating $\underline{a}$ using polynomial (quintic) regression analysis. For $4 \leq n \leq 1000$, the approximation gave the following equations for the last two (and hence first two) components of $\underline{a}$:

$$\tilde{a}_n = c_n + 0.221157y - 0.147981y^2 - 2.071190y^3 + 4.434685y^4 - 2.706056y^5 \qquad (19)$$

$$\tilde{a}_{n-1} = c_{n-1} + 0.042981y - 0.293762y^2 - 1.752461y^3 + 5.682633y^4 - 3.582633y^5 \qquad (20)$$

where

$$y = \sqrt{n} \qquad (21)$$

The other components are computed as:

$$\tilde{a}_i = \frac{\tilde{m}_i}{\sqrt{\eta}} \qquad (22)$$

for $i = 2, \ldots, n-1$ if $n \leq 5$, or $i = 3, \ldots, n-2$ if $n > 5$, where

$$\eta = \frac{\underline{\tilde{m}}^T \underline{\tilde{m}} - 2\tilde{m}_n^2}{1 - 2\tilde{a}_n^2} \qquad (23)$$

if $n \leq 5$, and

$$\eta = \frac{\underline{\tilde{m}}^T \underline{\tilde{m}} - 2\tilde{m}_n^2 - 2\tilde{m}_{n-1}^2}{1 - 2\tilde{a}_n^2 - 2\tilde{a}_{n-1}^2} \qquad (24)$$

if $n > 5$.

Royston (1992a) found his approximation to $\underline{a}$ to be accurate to at least ± 1 in the third decimal place over all values of i and selected values of n, and also found that critical percentage points of W based on his approximation agreed closely with the exact critical percentage points calculated by Verrill and Johnson (1988).

Transformation of the Null Distribution of W to Normality

In order to compute a p-value associated with a particular value of W, Royston (1992a) approximated the distribution of $(1 - W)$ by a three-parameter lognormal distribution for $4 \leq n \leq 11$, and the upper half of the distribution of $(1 - W)$ by a two-parameter lognormal distribution for $12 \leq n \leq 2000$. Setting

$$z = \frac{w - \mu}{\sigma} \qquad (25)$$

the p-value associated with W is given by:

$$p = 1 - \Phi(z) \qquad (26)$$

For $4 \leq n \leq 11$, the quantities necessary to compute z are given by:

$$w = -log[\gamma - log(1 - W)] \quad (27)$$

$$\gamma = -2.273 + 0.459n \quad (28)$$

$$\mu = 0.5440 - 0.39978n + 0.025054n^2 - 0.000671n^3 \quad (29)$$

$$\sigma = exp(1.3822 - 0.77857n + 0.062767n^2 - 0.0020322n^3) \quad (30)$$

For $12 \leq n \leq 2000$, the quantities necessary to compute z are given by:

$$w = log(1 - W) \quad (31)$$

$$\gamma = log(n) \quad (32)$$

$$\mu = -1.5861 - 0.31082y - 0.083751y^2 + 0.00038915y^3 \quad (33)$$

$$\sigma = exp(-0.4803 - 0.082676y + 0.0030302y^2) \quad (34)$$

For the last approximation when $12 \leq n \leq 2000$, Royston (1992a) claims this approximation is actually valid for sample sizes up to $n = 5000$.

Modification for the Three-Parameter Lognormal Distribution

When `distribution="lnorm3"`, the function `gofTest` assumes the vector $\underline{x}$ is a random sample from a three-parameter lognormal distribution. It estimates the threshold parameter via the zero-skewness method (see `elnorm3`), and then performs the Shapiro-Wilk goodness-of-fit test for normality on $log(x - \hat{\gamma})$ where $\hat{\gamma}$ is the estimated threshold parmater. Because the threshold parameter has to be estimated, however, the p-value associated with the computed z-statistic will tend to be conservative (larger than it should be under the null hypothesis). Royston (1992b) proposed the following transformation of the z-statistic:

$$z' = \frac{z - \mu_z}{\sigma_z} \quad (35)$$

where for $5 \leq n \leq 11$,

$$\mu_z = -3.8267 + 2.8242u - 0.63673u^2 - 0.020815v \quad (36)$$

$$\sigma_z = -4.9914 + 8.6724u - 4.27905u^2 + 0.70350u^3 - 0.013431v \quad (37)$$

and for $12 \leq n \leq 2000$,

$$\mu_z = -3.7796 + 2.4038u - 0.6675u^2 - 0.082863u^3 - 0.0037935u^4 - 0.027027v - 0.0019887vu \quad (38)$$

$$\sigma_z = 2.1924 - 1.0957u + 0.33737u^2 - 0.043201u^3 + 0.0019974u^4 - 0.0053312vu \quad (39)$$

where

$$u = log(n) \quad (40)$$

$$v = u(\hat{\sigma} - \hat{\sigma}^2) \quad (41)$$

$$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \quad (42)$$

$$y_i = log(x_i - \hat{\gamma}) \quad (43)$$

and γ denotes the threshold parameter. The p-value associated with this test is then given by:

$$p = 1 - \Phi(z') \qquad (44)$$

Testing Goodness-of-Fit for Any Continuous Distribution
The function `gofTest` extends the Shapiro-Wilk test to test for goodness-of-fit for any continuous distribution by using the idea of Chen and Balakrishnan (1995), who proposed a general purpose approximate goodness-of-fit test based on the Cramer-von Mises or Anderson-Darling goodness-of-fit tests for normality. The function `gofTest` modifies the approach of Chen and Balakrishnan (1995) by using the same first 2 steps, and then applying the Shapiro-Wilk test:

1. Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote the vector of n ***ordered*** observations. Compute cumulative probabilities for each x_i based on the cumulative distribution function for the hypothesized distribution. That is, compute $p_i = F(x_i, \hat{\theta})$ where $F(x, \theta)$ denotes the hypothesized cumulative distribution function with parameter(s) θ, and $\hat{\theta}$ denotes the estimated parameter(s).
2. Compute standard normal deviates based on the computed cumulative probabilities: $y_i = \Phi^{-1}(p_i)$
3. Perform the Shapiro-Wilk goodness-of-fit test on the y_i's.

- **Shapiro-Francia Goodness-of-Fit Test** (`test="sf"`).

The Shapiro-Francia goodness-of-fit test (Shapiro and Francia, 1972; Weisberg and Bingham, 1975; Royston, 1992c) is also one of the most commonly used goodness-of-fit tests for normality. You can use it to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta). In addition, you can also use it to test the null hypothesis of any continuous distribution that is available (see the help file for `Distribution.df`). See the section *Testing Goodness-of-Fit for Any Continuous Distribution* above for an explanation of how this is done.

Royston's Transformation of the Shapiro-Francia W'-Statistic to Normality
Equation (13) above gives the formula for the Shapiro-Francia W'-statistic, and Equation (16) above gives the formula for Weisberg-Bingham approximation to the W'-statistic (denoted $\tilde{W}'$). Royston (1992c) presents an algorithm to transform the $\tilde{W}'$-statistic so that its null distribution is approximately a standard normal. For $5 \leq n \leq 5000$, Royston (1992c) approximates the distribution of $(1 - \tilde{W}')$ by a lognormal distribution. Setting

$$z = \frac{w - \mu}{\sigma} \qquad (45)$$

the p-value associated with $\tilde{W}'$ is given by:

$$p = 1 - \Phi(z) \qquad (46)$$

The quantities necessary to compute z are given by:

$$w = log(1 - \tilde{W}') \qquad (47)$$

$$\nu = log(n) \qquad (48)$$

$$u = log(\nu) - \nu \qquad (49)$$

$$\mu = -1.2725 + 1.0521u \qquad (50)$$

$$v = log(\nu) + \frac{2}{\nu} \qquad (51)$$

$$\sigma = 1.0308 - 0.26758v \qquad (52)$$

Testing Goodness-of-Fit for Any Continuous Distribution

The function `gofTest` extends the Shapiro-Francia test to test for goodness-of-fit for any continuous distribution by using the idea of Chen and Balakrishnan (1995), who proposed a general purpose approximate goodness-of-fit test based on the Cramer-von Mises or Anderson-Darling goodness-of-fit tests for normality. The function `gofTest` modifies the approach of Chen and Balakrishnan (1995) by using the same first 2 steps, and then applying the Shapiro-Francia test:

1. Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote the vector of n ***ordered*** observations. Compute cumulative probabilities for each x_i based on the cumulative distribution function for the hypothesized distribution. That is, compute $p_i = F(x_i, \hat{\theta})$ where $F(x, \theta)$ denotes the hypothesized cumulative distribution function with parameter(s) θ, and $\hat{\theta}$ denotes the estimated parameter(s).
2. Compute standard normal deviates based on the computed cumulative probabilities: $y_i = \Phi^{-1}(p_i)$
3. Perform the Shapiro-Francia goodness-of-fit test on the y_i's.

- **Probability Plot Correlation Coefficient (PPCC) Goodness-of-Fit Test** (`test="ppcc"`).

  The PPPCC goodness-of-fit test (Filliben, 1975; Looney and Gulledge, 1985) can be used to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta). In addition, you can also use it to test the null hypothesis of any continuous distribution that is available (see the help file for `Distribution.df`). The function `gofTest` computes the PPCC test statistic using Blom plotting positions.

  Filliben (1975) proposed using the correlation coefficient r from a normal probability plot to perform a goodness-of-fit test for normality, and he provided a table of critical values for r under the for samples sizes between 3 and 100. Vogel (1986) provided an additional table for sample sizes between 100 and 10,000.

  Looney and Gulledge (1985) investigated the characteristics of Filliben's probability plot correlation coefficient (PPCC) test using the plotting position formulas given in Filliben (1975), as well as three other plotting position formulas: Hazen plotting positions, Weibull plotting positions, and Blom plotting positions (see the help file for `qqPlot` for an explanation of these plotting positions). They concluded that the PPCC test based on Blom plotting positions performs slightly better than tests based on other plotting positions, and they provide a table of empirical percentage points for the distribution of r based on Blom plotting positions.

  The function `gofTest` computes the PPCC test statistic r using Blom plotting positions. It can be shown that the square of this statistic is equivalent to the Weisberg-Bingham Approximation to the Shapiro-Francia W'-Test (Weisberg and Bingham, 1975; Royston, 1993). Thus the PPCC goodness-of-fit test is equivalent to the Shapiro-Francia goodness-of-fit test.

- **Anderson-Darling Goodness-of-Fit Test** (`test="ad"`).

  The Anderson-Darling goodness-of-fit test (Stephens, 1986a; Thode, 2002) can be used to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta).

  When `test="ad"`, the function `gofTest` calls the function `ad.test` in the package **nortest**. Documentation from that package is as follows:

The Anderson-Darling test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is:

$$A = -n - \frac{1}{n}\sum_{i=1}^{n}[2i-1][ln(p_{(i)}) + ln(1-p_{(n-i+1)})]$$

where $p_{(i)} = \Phi([x_{(i)} - \bar{x}]/s)$. Here, Φ is the cumulative distribution function of the standard normal distribution, and $\bar{x}$ and s are mean and standard deviation of the data values. The p-value is computed from the modified statistic $Z = A(1.0 + 0.75/n + 2.25/n^2)$ according to Table 4.9 in Stephens [(1986a)].

- **Cramer-von Mises Goodness-of-Fit Test** (`test="cvm"`).

  The Cramer-von Mises goodness-of-fit test (Stephens, 1986a; Thode, 2002) can be used to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta).

  When `test="cvm"`, the function `gofTest` calls the function `cvm.test` in the package **nortest**. Documentation from that package is as follows:

  The Cramer-von Mises test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is:

$$W = \frac{1}{12n} + \sum_{i=1}^{n}\left(p_{(i)} - \frac{2i-1}{2n}\right)^2$$

  where $p_{(i)} = \Phi([x_{(i)} - \bar{x}]/s)$. Here, Φ is the cumulative distribution function of the standard normal distribution, and $\bar{x}$ and s are mean and standard deviation of the data values. The p-value is computed from the modified statistic $Z = W(1.0 + 0.75/n)$ according to Table 4.9 in Stephens [(1986a)].

- **Lilliefors Goodness-of-Fit Test** (`test="lillie"`).

  The Lilliefors goodness-of-fit test (Stephens, 1974; Dallal and Wilkinson, 1986; Thode, 2002) can be used to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta).

  When `test="lillie"`, the function `gofTest` calls the function `lillie.test` in the package **nortest**. Documentation from that package is as follows:

  The Lilliefors (Kolmogorov-Smirnov) test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is the maximal absolute difference between empirical and hypothetical cumulative distribution function. It may be computed as $D = max\{D^+, D^-\}$ with

$$D^+ = \max_{i=1,\ldots,n}\{i/n - p_{(i)}\}, \quad D^- = \max_{i=1,\ldots,n}\{p_{(i)} - (i-1)/n\}$$

  where $p_{(i)} = \Phi([x_{(i)} - \bar{x}]/s)$. Here, Φ is the cumulative distribution function of the standard normal distribution, and $\bar{x}$ and s are mean and standard deviation of the data values. The p-value is computed from the Dallal-Wilkinson (1986) formula, which is claimed to be only reliable when the p-value is smaller than 0.1. If the Dallal-Wilkinson p-value turns out to be greater than 0.1, then the p-value is computed from the distribution of the modified statistic $Z = D(\sqrt{n} - 0.01 + 0.85/\sqrt{n})$, see Stephens (1974), the actual p-value formula being obtained by a simulation and approximation process.

- **Zero-Skew Goodness-of-Fit Test** (`test="skew"`).

  The Zero-skew goodness-of-fit test (D'Agostino, 1970) can be used to test the following hypothesized distributions: Normal, Lognormal, Zero-Modified Normal, or Zero-Modified Lognormal (Delta).

  When `test="skew"`, the function `gofTest` tests the null hypothesis that the skew of the distribution is 0:

$$H_0 : \sqrt{\beta}_1 = 0 \qquad (53)$$

  where

$$\sqrt{\beta}_1 = \frac{\mu_3}{\mu_2^{3/2}} \qquad (54)$$

  and the quantity μ_r denotes the r'th moment about the mean (also called the r'th central moment). The quantity $\sqrt{\beta_1}$ is called the coefficient of skewness, and is estimated by:

$$\sqrt{b}_1 = \frac{m_3}{m_2^{3/2}} \qquad (55)$$

  where

$$m_r = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^r \qquad (56)$$

  denotes the r'th sample central moment.

  The possible alternative hypotheses are:

$$H_a : \sqrt{\beta}_1 \neq 0 \qquad (57)$$

$$H_a : \sqrt{\beta}_1 < 0 \qquad (58)$$

$$H_a : \sqrt{\beta}_1 > 0 \qquad (59)$$

  which correspond to `alternative="two-sided"`, `alternative="less"`, and `alternative="greater"`, respectively.

  To test the null hypothesis of zero skew, D'Agostino (1970) derived an approximation to the distribution of $\sqrt{b_1}$ under the null hypothesis of zero-skew, assuming the observations comprise a random sample from a normal (Gaussian) distribution. Based on D'Agostino's approximation, the statistic Z shown below is assumed to follow a standard normal distribution and is used to compute the p-value associated with the test of H_0:

$$Z = \delta \ log\{\frac{Y}{\alpha} + [(\frac{Y}{\alpha})^2 + 1]^{1/2}\} \qquad (60)$$

  where

$$Y = \sqrt{b_1}[\frac{(n+1)(n+3)}{6(n-2)}]^{1/2} \qquad (61)$$

$$\beta_2 = \frac{3(n^2 + 27n - 70)(n+1)(n+3)}{(n-2)(n+5)(n+7)(n+9)} \qquad (62)$$

$$W^2 = -1 + \sqrt{2\beta_2 - 2} \qquad (63)$$

$$\delta = 1/\sqrt{log(W)} \qquad (64)$$

$$\alpha = [2/(W^2 - 1)]^{1/2} \qquad (65)$$

When the sample size n is at least 150, a simpler approximation may be used in which Y in Equation (61) is assumed to follow a standard normal distribution and is used to compute the p-value associated with the hypothesis test.

- **Kolmogorov-Smirnov Goodness-of-Fit Test** (`test="ks"`).

  When `test="ks"`, the function `gofTest` calls the R function `ks.test` to compute the test statistic and p-value. Note that for the one-sample case, the distribution parameters should be pre-specified and not estimated from the data, and if the distribution parameters are estimated from the data you will receive a warning that this test is very conservative (Type I error smaller than assumed; high Type II error) in this case.

- **ProUCL Kolmogorov-Smirnov Goodness-of-Fit Test for Gamma** (`test="proucl.ks.gamma"`).

  When `test="proucl.ks.gamma"`, the function `gofTest` calls the R function `ks.test` to compute the Kolmogorov-Smirnov test statistic based on the maximum likelihood estimates of the shape and scale parameters (see `egamma`). The p-value is computed based on the simulated critical values given in `ProUCL.Crit.Vals.for.KS.Test.for.Gamma.array` (USEPA, 2015). The sample size must be between 5 and 1000, and the value of the maximum likelihood estimate of the shape parameter must be between 0.025 and 50. The critical value for the test statistic is computed using the simulated critical values and linear interpolation.

- **ProUCL Anderson-Darling Goodness-of-Fit Test for Gamma** (`test="proucl.ad.gamma"`).

  When `test="proucl.ad.gamma"`, the function `gofTest` computes the Anderson-Darling test statistic (Stephens, 1986a, p.101) based on the maximum likelihood estimates of the shape and scale parameters (see `egamma`). The p-value is computed based on the simulated critical values given in `ProUCL.Crit.Vals.for.AD.Test.for.Gamma.array` (USEPA, 2015). The sample size must be between 5 and 1000, and the value of the maximum likelihood estimate of the shape parameter must be between 0.025 and 50. The critical value for the test statistic is computed using the simulated critical values and linear interpolation.

- **Chi-Squared Goodness-of-Fit Test** (`test="chisq"`).

  The method used by `gofTest` is a modification of what is used for `chisq.test`. If the hypothesized distribution function is completely specified, the degrees of freedom are $m - 1$ where m denotes the number of classes. If any parameters are estimated, the degrees of freedom depend on the method of estimation. The function `gofTest` follows the convention of computing degrees of freedom as $m - 1 - k$, where k is the number of parameters estimated. It can be shown that if the parameters are estimated by maximum likelihood, the degrees of freedom are bounded between $m - 1$ and $m - 1 - k$. Therefore, especially when the sample size is small, it is important to compare the test statistic to the chi-squared distribution with both $m - 1$ and $m - 1 - k$ degrees of freedom. See Kendall and Stuart (1991, Chapter 30) for a more complete discussion.

  The distribution theory of chi-square statistics is a large sample theory. The expected cell counts are assumed to be at least moderately large. As a rule of thumb, each should be at least 5. Although authors have found this rule to be conservative (especially when the class probabilities are not too different from each other), the user should regard p-values with caution when expected cell counts are small.

- **Wilk-Shapiro Goodness-of-Fit Test for Uniform [0, 1] Distribution** (`test="ws"`).
  Wilk and Shapiro (1968) suggested this test in the context of jointly testing several independent samples for normality simultaneously. If $p_1, p_2, \ldots, p_n$ denote the p-values associated with the test for normality of n independent samples, then under the null hypothesis that all n samples come from a normal distribution, the p-values are a random sample of n observations from a Uniform [0,1] distribution, that is a Uniform distribution with minimum 0 and maximum 1. Wilk and Shapiro (1968) suggested two different methods for testing whether the p-values come from a Uniform [0, 1] distribution:

  – *Test Based on Normal Scores.* Under the null hypothesis, the normal scores

  $$\Phi^{-1}(p_1), \Phi^{-1}(p_2), \ldots, \Phi^{-1}(p_n)$$

  are a random sample of n observations from a standard normal distribution. Wilk and Shapiro (1968) denote the i'th normal score by

  $$G_i = \Phi^{-1}(p_i) \qquad (66)$$

  and note that under the null hypothesis, the quantity G defined as

  $$G = \frac{1}{\sqrt{n}} \sum_{1}^{n} G_i \qquad (67)$$

  has a standard normal distribution. Wilk and Shapiro (1968) were interested in the alternative hypothesis that some of the n independent samples did not come from a normal distribution and hence would be associated with smaller p-values than expected under the null hypothesis, which translates to the alternative that the cdf for the distribution of the p-values is greater than the cdf of a Uniform [0, 1] distribution (`alternative="greater"`). In terms of the test statistic G, this alternative hypothesis would tend to make G smaller than expected, so the p-value is given by $\Phi(G)$. For the one-sided lower alternative that the cdf for the distribution of p-values is less than the cdf for a Uniform [0, 1] distribution, the p-value is given by

  $$p = 1 - \Phi(G) \qquad (68)$$

  .

  – *Test Based on Chi-Square Scores.* Under the null hypothesis, the chi-square scores

  $$-2\,log(p_1), -2\,log(p_2), \ldots, -2\,log(p_n)$$

  are a random sample of n observations from a chi-square distribution with 2 degrees of freedom (Fisher, 1950). Wilk and Shapiro (1968) denote the i'th chi-square score by

  $$C_i = -2\,log(p_i) \qquad (69)$$

  and note that under the null hypothesis, the quantity C defined as

  $$C = \sum_{1}^{n} C_i \qquad (70)$$

  has a chi-square distribution with $2n$ degrees of freedom. Wilk and Shapiro (1968) were interested in the alternative hypothesis that some of the n independent samples did not

come from a normal distribution and hence would be associated with smaller p-values than expected under the null hypothesis, which translates to the alternative that the cdf for the distribution of the p-values is greater than the cdf of a Uniform [0, 1] distribution (`alternative="greater"`). In terms of the test statistic C, this alternative hypothesis would tend to make C larger than expected, so the p-value is given by

$$p = 1 - F_{2n}(C) \qquad (71)$$

where $F_2 n$ denotes the cumulative distribution function of the chi-square distribution with $2n$ degrees of freedom. For the one-sided lower alternative that the cdf for the distribution of p-values is less than the cdf for a Uniform [0, 1] distribution, the p-value is given by

$$p = F_{2n}(C) \qquad (72)$$

Value

a list of class `"gof"` containing the results of the goodness-of-fit test, unless the two-sample Kolmogorov-Smirnov test is used, in which case the value is a list of class `"gofTwoSample"`. Objects of class `"gof"` and `"gofTwoSample"` have special printing and plotting methods. See the help files for `gof.object` and `gofTwoSample.object` for details.

Note

The Shapiro-Wilk test (Shapiro and Wilk, 1965) and the Shapiro-Francia test (Shapiro and Francia, 1972) are probably the two most commonly used hypothesis tests to test departures from normality. The Shapiro-Wilk test is most powerful at detecting short-tailed (platykurtic) and skewed distributions, and least powerful against symmetric, moderately long-tailed (leptokurtic) distributions. Conversely, the Shapiro-Francia test is more powerful against symmetric long-tailed distributions and less powerful against short-tailed distributions (Royston, 1992b; 1993). In general, the Shapiro-Wilk and Shapiro-Francia tests outperform the Anderson-Darling test, which in turn outperforms the Cramer-von Mises test, which in turn outperforms the Lilliefors test (Stephens, 1986a; Razali and Wah, 2011; Romao et al., 2010).

The zero-skew goodness-of-fit test for normality is one of several tests that have been proposed to test the assumption of a normal distribution (D'Agostino, 1986b). This test has been included mainly because it is called by `elnorm3`. Ususally, the Shapiro-Wilk or Shapiro-Francia test is preferred to this test, unless the direction of the alternative to normality (e.g., positive skew) is known (D'Agostino, 1986b, pp. 405–406).

Kolmogorov (1933) introduced a goodness-of-fit test to test the hypothesis that a random sample of n observations **x** comes from a specific hypothesized distribution with cumulative distribution function H. This test is now usually called the one-sample Kolmogorov-Smirnov goodness-of-fit test. Smirnov (1939) introduced a goodness-of-fit test to test the hypothesis that a random sample of n observations **x** comes from the same distribution as a random sample of m observations **y**. This test is now usually called the two-sample Kolmogorov-Smirnov goodness-of-fit test. Both tests are based on the maximum vertical distance between two cumulative distribution functions. For the one-sample problem with a small sample size, the Kolmogorov-Smirnov test may be preferred over the chi-squared goodness-of-fit test since the KS-test is exact, while the chi-squared test is based on an asymptotic approximation.

The chi-squared test, introduced by Pearson in 1900, is the oldest and best known goodness-of-fit test. The idea is to reduce the goodness-of-fit problem to a multinomial setting by comparing

the observed cell counts with their expected values under the null hypothesis. Grouping the data sacrifices information, especially if the hypothesized distribution is continuous. On the other hand, chi-squared tests can be be applied to any type of variable: continuous, discrete, or a combination of these.

The Wilk-Shapiro (1968) tests for a Uniform [0, 1] distribution were introduced in the context of testing whether several independent samples all come from normal distributions, with possibly different means and variances. The function `gofGroupTest` extends this idea to allow you to test whether several independent samples come from the same distribution (e.g., gamma, extreme value, etc.), with possibly different parameters.

In practice, almost any goodness-of-fit test will *not* reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test *will* reject the null hypothesis if the number of observations is very large, since "real" data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution of "real" data is close enough to some theoretical distribution that fairly accurate results may be provided by assuming that particular theoretical distribution. One way to asses the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see `qqPlot`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

Juergen Gross and Uwe Ligges for the Anderson-Darling, Carmer-von Mises, and Lilliefors tests called from the package **nortest**.

References

Birnbaum, Z.W., and F.H. Tingey. (1951). One-Sided Confidence Contours for Probability Distribution Functions. *Annals of Mathematical Statistics* **22**, 592-596.

Blom, G. (1958). *Statistical Estimates and Transformed Beta Variables*. John Wiley and Sons, New York.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Dallal, G.E., and L. Wilkinson. (1986). An Analytic Approximation to the Distribution of Lilliefor's Test for Normality. *The American Statistician* **40**, 294-296.

D'Agostino, R.B. (1970). Transformation to Normality of the Null Distribution of $g1$. *Biometrika* **57**, 679-681.

D'Agostino, R.B. (1971). An Omnibus Test of Normality for Moderate and Large Size Samples. *Biometrika* **58**, 341-348.

D'Agostino, R.B. (1986b). Tests for the Normal Distribution. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York.

D'Agostino, R.B., and E.S. Pearson (1973). Tests for Departures from Normality. Empirical Results for the Distributions of $b2$ and $\sqrt{b1}$. *Biometrika* **60**(3), 613-622.

D'Agostino, R.B., and G.L. Tietjen (1973). Approaches to the Null Distribution of $\sqrt{b1}$. *Biometrika* **60**(1), 169-173.

Fisher, R.A. (1950). *Statistical Methods for Research Workers*. 11'th Edition. Hafner Publishing Company, New York, pp.99-100.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Kendall, M.G., and A. Stuart. (1991). *The Advanced Theory of Statistics, Volume 2: Inference and Relationship*. Fifth Edition. Oxford University Press, New York.

Kim, P.J., and R.I. Jennrich. (1973). Tables of the Exact Sampling Distribution of the Two Sample Kolmogorov-Smirnov Criterion. In Harter, H.L., and D.B. Owen, eds. *Selected Tables in Mathematical Statistics, Vol. 1*. American Mathematical Society, Providence, Rhode Island, pp.79-170.

Kolmogorov, A.N. (1933). Sulla determinazione empirica di una legge di distribuzione. *Giornale dell' Istituto Italiano degle Attuari* **4**, 83-91.

Marsaglia, G., W.W. Tsang, and J. Wang. (2003). Evaluating Kolmogorov's distribution. *Journal of Statistical Software*, **8**(18). doi:10.18637/jss.v008.i18.

Moore, D.S. (1986). Tests of Chi-Squared Type. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, pp.63-95.

Pomeranz, J. (1973). Exact Cumulative Distribution of the Kolmogorov-Smirnov Statistic for Small Samples (Algorithm 487). *Collected Algorithms from ACM* ??, ???-???.

Razali, N.M., and Y.B. Wah. (2011). Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors, and Anderson-Darling Tests. *Journal of Statistical Modeling and Analytics* **2**(1), 21–33.

Romao, X., Delgado, R., and A. Costa. (2010). An Empirical Power Comparison of Univariate Goodness-of-Fit Tests for Normality. *Journal of Statistical Computation and Simulation* **80**(5), 545–591.

Royston, J.P. (1992a). Approximating the Shapiro-Wilk W-Test for Non-Normality. *Statistics and Computing* **2**, 117-119.

Royston, J.P. (1992b). Estimation, Reference Ranges and Goodness of Fit for the Three-Parameter Log-Normal Distribution. *Statistics in Medicine* **11**, 897-912.

Royston, J.P. (1992c). A Pocket-Calculator Algorithm for the Shapiro-Francia Test of Non-Normality: An Application to Medicine. *Statistics in Medicine* **12**, 181-184.

Royston, P. (1993). A Toolkit for Testing for Non-Normality in Complete and Censored Samples. *The Statistician* **42**, 37-43.

Ryan, T., and B. Joiner. (1973). *Normal Probability Plots and Tests for Normality*. Technical Report, Pennsylvania State University, Department of Statistics.

Shapiro, S.S., and R.S. Francia. (1972). An Approximate Analysis of Variance Test for Normality. *Journal of the American Statistical Association* **67**(337), 215-219.

Shapiro, S.S., and M.B. Wilk. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika* **52**, 591-611.

Smirnov, N.V. (1939). Estimate of Deviation Between Empirical Distribution Functions in Two Independent Samples. *Bulletin Moscow University* **2**(2), 3-16.

Smirnov, N.V. (1948). Table for Estimating the Goodness of Fit of Empirical Distributions. *Annals of Mathematical Statistics* **19**, 279-281.

Stephens, M.A. (1970). Use of the Kolmogorov-Smirnov, Cramer-von Mises and Related Statistics Without Extensive Tables. *Journal of the Royal Statistical Society, Series B*, **32**, 115-122.

Stephens, M.A. (1974). EDF Statistics for Goodness of Fit and Some Comparisons. *Journal of the American Statistical Association* **69**, 730-737.

Stephens, M.A. (1986a). Tests Based on EDF Statistics. In D'Agostino, R. B., and M.A. Stevens, eds. *Goodness-of-Fit Techniques*. Marcel Dekker, New York.

Thode Jr., H.C. (2002). *Testing for Normality*. Marcel Dekker, New York.

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C.

Verrill, S., and R.A. Johnson. (1987). The Asymptotic Equivalence of Some Modified Shapiro-Wilk Statistics – Complete and Censored Sample Cases. *The Annals of Statistics* **15**(1), 413-419.

Verrill, S., and R.A. Johnson. (1988). Tables and Large-Sample Distribution Theory for Censored-Data Correlation Statistics for Testing Normality. *Journal of the American Statistical Association* **83**, 1192-1197.

Weisberg, S., and C. Bingham. (1975). An Approximate Analysis of Variance Test for Non-Normality Suitable for Machine Calculation. *Technometrics* **17**, 133-134.

Wilk, M.B., and S.S. Shapiro. (1968). The Joint Assessment of Normality of Several Independent Samples. *Technometrics*, **10**(4), 825-839.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

rosnerTest, gof.object, print.gof, plot.gof, shapiro.test, ks.test, chisq.test, Normal, Lognormal, Lognormal3, Zero-Modified Normal, Zero-Modified Lognormal (Delta), enorm, elnorm, elnormAlt, elnorm3, ezmnorm, ezmlnorm, ezmlnormAlt, qqPlot.

Examples

```
# Generate 20 observations from a gamma distribution with
# parameters shape = 2 and scale = 3 then run various
# goodness-of-fit tests.
# (Note:  the call to set.seed lets you reproduce this example.)

set.seed(47)
dat <- rgamma(20, shape = 2, scale = 3)

# Shapiro-Wilk generalized goodness-of-fit test
#----------------------------------------------
gof.list <- gofTest(dat, distribution = "gamma")
gof.list

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Shapiro-Wilk GOF Based on
#                                 Chen & Balakrisnan (1995)
#
#Hypothesized Distribution:       Gamma
#
#Estimated Parameter(s):          shape = 1.909462
#                                 scale = 4.056819
#
#Estimation Method:               mle
```

```
#
#Data:                              dat
#
#Sample Size:                       20
#
#Test Statistic:                    W = 0.9834958
#
#Test Statistic Parameter:          n = 20
#
#P-value:                           0.970903
#
#Alternative Hypothesis:            True cdf does not equal the
#                                   Gamma Distribution.

dev.new()
plot(gof.list)

#----------

# Redo the example above, but use the bias-corrected mle

gofTest(dat, distribution = "gamma",
  est.arg.list = list(method = "bcmle"))

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                       Shapiro-Wilk GOF Based on
#                                   Chen & Balakrisnan (1995)
#
#Hypothesized Distribution:         Gamma
#
#Estimated Parameter(s):            shape = 1.656376
#                                   scale = 4.676680
#
#Estimation Method:                 bcmle
#
#Data:                              dat
#
#Sample Size:                       20
#
#Test Statistic:                    W = 0.9834346
#
#Test Statistic Parameter:          n = 20
#
#P-value:                           0.9704046
#
#Alternative Hypothesis:            True cdf does not equal the
#                                   Gamma Distribution.

#----------

# Komogorov-Smirnov goodness-of-fit test (pre-specified parameters)
```

```
#----------------------------------------------------------------

gofTest(dat, test = "ks", distribution = "gamma",
  param.list = list(shape = 2, scale = 3))

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Kolmogorov-Smirnov GOF
#
#Hypothesized Distribution:       Gamma(shape = 2, scale = 3)
#
#Data:                            dat
#
#Sample Size:                     20
#
#Test Statistic:                  ks = 0.2313878
#
#Test Statistic Parameter:        n = 20
#
#P-value:                         0.2005083
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Gamma(shape = 2, scale = 3)
#                                 Distribution.

#----------

# ProUCL Version of Komogorov-Smirnov goodness-of-fit test
# for a Gamma Distribution (estimated parameters)
#-----------------------------------------------------------

gofTest(dat, test = "proucl.ks.gamma", distribution = "gamma")

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     ProUCL Kolmogorov-Smirnov Gamma GOF
#
#Hypothesized Distribution:       Gamma
#
#Estimated Parameter(s):          shape = 1.909462
#                                 scale = 4.056819
#
#Estimation Method:               MLE
#
#Data:                            dat
#
#Sample Size:                     20
#
#Test Statistic:                  D = 0.0988692
#
#Test Statistic Parameter:        n = 20
```

```
#
#Critical Values:                 D.0.01 = 0.228
#                                 D.0.05 = 0.196
#                                 D.0.10 = 0.180
#
#P-value:                         >= 0.10
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Gamma Distribution.

#----------

# Chi-squared goodness-of-fit test (estimated parameters)
#-------------------------------------------------------

gofTest(dat, test = "chisq", distribution = "gamma", n.classes = 4)

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Chi-square GOF
#
#Hypothesized Distribution:       Gamma
#
#Estimated Parameter(s):          shape = 1.909462
#                                 scale = 4.056819
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Test Statistic:                  Chi-square = 1.2
#
#Test Statistic Parameter:        df = 1
#
#P-value:                         0.2733217
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Gamma Distribution.

#----------
# Clean up

rm(dat, gof.list)
graphics.off()

#--------------------------------------------------------------------

# Example 10-2 of USEPA (2009, page 10-14) gives an example of
# using the Shapiro-Wilk test to test the assumption of normality
# for nickel concentrations (ppb) in groundwater collected over
```

```
# 4 years.  The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

EPA.09.Ex.10.1.nickel.df
#   Month   Well Nickel.ppb
#1      1 Well.1       58.8
#2      3 Well.1        1.0
#3      6 Well.1      262.0
#4      8 Well.1       56.0
#5     10 Well.1        8.7
#6      1 Well.2       19.0
#7      3 Well.2       81.5
#8      6 Well.2      331.0
#9      8 Well.2       14.0
#10    10 Well.2       64.4
#11     1 Well.3       39.0
#12     3 Well.3      151.0
#13     6 Well.3       27.0
#14     8 Well.3       21.4
#15    10 Well.3      578.0
#16     1 Well.4        3.1
#17     3 Well.4      942.0
#18     6 Well.4       85.6
#19     8 Well.4       10.0
#20    10 Well.4      637.0

# Test for a normal distribution:
#-------------------------------

gof.list <- gofTest(Nickel.ppb ~ 1, data = EPA.09.Ex.10.1.nickel.df)
gof.list

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#
#Hypothesized Distribution:       Normal
#
#Estimated Parameter(s):          mean = 169.5250
#                                 sd   = 259.7175
#
#Estimation Method:               mvue
#
#Data:                            Nickel.ppb
#
#Data Source:                     EPA.09.Ex.10.1.nickel.df
#
#Sample Size:                     20
#
#Test Statistic:                  W = 0.6788888
#
#Test Statistic Parameter:        n = 20
```

```
#
#P-value:                          2.17927e-05
#
#Alternative Hypothesis:           True cdf does not equal the
#                                  Normal Distribution.

dev.new()
plot(gof.list)

#----------

# Test for a lognormal distribution:
#-----------------------------------

gofTest(Nickel.ppb ~ 1, data = EPA.09.Ex.10.1.nickel.df,
  dist = "lnorm")

#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method:                      Shapiro-Wilk GOF
#
#Hypothesized Distribution:        Lognormal
#
#Estimated Parameter(s):           meanlog = 3.918529
#                                  sdlog   = 1.801404
#
#Estimation Method:                mvue
#
#Data:                             Nickel.ppb
#
#Data Source:                      EPA.09.Ex.10.1.nickel.df
#
#Sample Size:                      20
#
#Test Statistic:                   W = 0.978946
#
#Test Statistic Parameter:         n = 20
#
#P-value:                          0.9197735
#
#Alternative Hypothesis:           True cdf does not equal the
#                                  Lognormal Distribution.

#----------

# Test for a lognormal distribution, but use the
# Mean and CV parameterization:
#-----------------------------------------------

gofTest(Nickel.ppb ~ 1, data = EPA.09.Ex.10.1.nickel.df,
  dist = "lnormAlt")
```

```
#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#
#Hypothesized Distribution:       Lognormal
#
#Estimated Parameter(s):          mean = 213.415628
#                                 cv   =   2.809377
#
#Estimation Method:               mvue
#
#Data:                            Nickel.ppb
#
#Data Source:                     EPA.09.Ex.10.1.nickel.df
#
#Sample Size:                     20
#
#Test Statistic:                  W = 0.978946
#
#Test Statistic Parameter:        n = 20
#
#P-value:                         0.9197735
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.

#----------
# Clean up

rm(gof.list)
graphics.off()

#--------------------------------------------------------------------------

# Generate 20 observations from a normal distribution with mean=3 and sd=2, and
# generate 10 observaions from a normal distribution with mean=2 and sd=2 then
# test whether these sets of observations come from the same distribution.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(300)
dat1 <- rnorm(20, mean = 3, sd = 2)
dat2 <- rnorm(10, mean = 1, sd = 2)
gofTest(x = dat1, y = dat2, test = "ks")

#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method:                     2-Sample K-S GOF
#
#Hypothesized Distribution:       Equal
#
#Data:                            x = dat1
```

```
#                                y = dat2
#
#Sample Sizes:                   n.x = 20
#                                n.y = 10
#
#Test Statistic:                 ks = 0.7
#
#Test Statistic Parameters:      n = 20
#                                m = 10
#
#P-value:                        0.001669561
#
#Alternative Hypothesis:         The cdf of 'dat1' does not equal
#                                the cdf of 'dat2'.

#----------
# Clean up

rm(dat1, dat2)
```

`gofTestCensored` *Goodness-of-Fit Test for Normal or Lognormal Distribution Based on Censored Data*

Description

Perform a goodness-of-fit test to determine whether a data set appears to come from a normal distribution, lognormal distribution, or lognormal distribution (alternative parameterization) based on a sample of data that has been subjected to Type I or Type II censoring.

Usage

```
gofTestCensored(x, censored, censoring.side = "left", test = "sf",
  distribution = "norm", est.arg.list = NULL,
  prob.method = "hirsch-stedinger", plot.pos.con = 0.375,
  keep.data = TRUE, data.name = NULL, censoring.name = NULL)
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`censored`	numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is `"logical"`, `TRUE` values correspond to elements of `x` that are censored, and `FALSE` values correspond to elements of `x` that are not censored. If the mode of `censored` is `"numeric"`, it must contain only `1`'s and `0`'s; `1` corresponds to `TRUE` and `0` corresponds to `FALSE`. Missing (`NA`) values are allowed but will be removed.
`censoring.side`	character string indicating on which side the censoring occurs. The possible values are `"left"` (the default) and `"right"`.

`test` character string defining which goodness-of-fit test to perform. Possible values are: `"sw"` Shapiro-Wilk. `"sf"` Shapiro-Francia; the default. `"ppcc"` Probability Plot Correlation Coefficient.

The Shapiro-Wilk test is only available for singly censored data.

See the DETAILS section for more information.

`distribution` a character string denoting the abbreviation of the assumed distribution. Only continous distributions are allowed. See the help file for `Distribution.df` for a list of distributions and their abbreviations. Examples of possible values are:
`distribution="norm"` (Normal distribution; the default),
`distribution="lnorm"` (Lognormal distribution),
`distribution="lnormAlt"` (Lognormal distribution, alternative parameterization,
`distribution="gamma"` (Gamma distribution),
`distribution="gammaAlt"` (Gamma distribution, alternative parameterization).

The results for the goodness-of-fit test are identical for `distribution="lnorm"` and `distribution="lnormAlt"`, the only difference in ouput is that when `distribution="lnorm"` the returned estimated parameters are the mean and standard deviation based on the log-scale of the data, whereas when `distribution="lnormAlt"` the returned estimated parameters are the mean and coefficient of variation based on the original scale of the data.

Also, the results for the goodness-of-fit test are identical for `distribution="gamma"` and `distribution="gammaAlt"`, the only difference in ouput is that when `distribution="gamma"` the returned estimated parameters are the shape and scale, whereas when `distribution="lnormAlt"` the returned estimated parameters are the mean and coefficient of variation.

`est.arg.list` a list of arguments to be passed to the function estimating the distribution parameters. For example, if `distribution="lnormAlt"` setting `est.arg.list=list(method="bcmle")` indicates using the bias-corrected maximum likelihood estimators (see the help file for `elnormAltCensored`). The default value is `est.arg.list=NULL` so that all default values for the estimating function are used. The estimated parameters are provided in the output merely for information, and the choice of the method of estimation has no effect on the goodness-of-fit test statistic or p-value.

`prob.method` character string indicating what method to use to compute the plotting positions (empirical probabilities) when `test="sf"` or `test="ppcc"`. Possible values are:
`"modified kaplan-meier"` (modification of product-limit method of Kaplan and Meier (1958)),
`"nelson"` (hazard plotting method of Nelson (1972)),
`"michael-schucany"` (generalization of the product-limit method due to Michael and Schucany (1986)), and
`"hirsch-stedinger"` (generalization of the product-limit method due to Hirsch and Stedinger (1987)).
The default value is `prob.method="hirsch-stedinger"`.

The `"nelson"` method is only available for `censoring.side="right"`, and the `"modified kaplan-meier"` method is only available for `censoring.side="left"`. See the DETAILS section and the help file for `ppointsCensored` for more information.

plot.pos.con	numeric scalar between 0 and 1 containing the value of the plotting position constant to use when test="sf" or test="ppcc". The default value is plot.pos.con=0.375. See the DETAILS section and the help file for ppointsCensored for more information.
keep.data	logical scalar indicating whether to return the original data. The default value is keep.data=TRUE.
data.name	optional character string indicating the name for the data used for argument x.
censoring.name	optional character string indicating the name for the data used for argument censored.

Details

Let $\underline{x} = c(x_1, x_2, \ldots, x_N)$ denote a vector of N observations from from some distribution with cdf F. Suppose we want to test the null hypothesis that F is the cdf of a normal (Gaussian) distribution with some arbitrary mean μ and standard deviation σ against the alternative hypothesis that F is the cdf of some other distribution. The table below shows the random variable for which F is the assumed cdf, given the value of the argument distribution.

Value of distribution	**Distribution Name**	**Random Variable for which F is the cdf**
"norm"	Normal	X
"lnorm"	Lognormal (Log-space)	$log(X)$
"lnormAlt"	Lognormal (Untransformed)	$log(X)$

Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_k;\ k \geq 1 \qquad (1)$$

For the case when $k \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $k = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j = 1, 2, \ldots, k$, so that

$$\sum_{i=1}^{k} c_j = c \qquad (2)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Note that for singly left-censored data:

$$x_{(1)} = x_{(2)} = \cdots = x_{(c)} = T \qquad (3)$$

and for singly right-censored data:

$$x_{(n+1)} = x_{(n+2)} = \cdots = x_{(N)} = T \qquad (4)$$

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations.

Shapiro-Wilk Goodness-of-Fit Test for Singly Censored Data (`test="sw"`)
Equation (8) in the help file for `gofTest` shows that for the case of **complete** ordered data $\underline{x}$, the Shapiro-Wilk W-statistic is the same as the square of the sample product-moment correlation between the vectors $\underline{a}$ and $\underline{x}$:

$$W = r(\underline{a}, \underline{x})^2 \qquad (5)$$

where

$$r(\underline{x}, \underline{y}) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{[\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2]^{1/2}} \qquad (6)$$

and $\underline{a}$ is defined by:

$$\underline{a} = \frac{\underline{m}^T V^{-1}}{[\underline{m}^T V^{-1} V^{-1} \underline{m}]^{1/2}} \qquad (7)$$

where ^{T} denotes the transpose operator, and $\underline{m}$ is the vector of expected values and V is the variance-covariance matrix of the order statistics of a random sample of size N from a standard normal distribution. That is, the values of $\underline{a}$ are the expected values of the standard normal order statistics weighted by their variance-covariance matrix, and normalized so that

$$\underline{a}^T \underline{a} = 1 \qquad (8)$$

Computing Shapiro-Wilk W-Statistic for Singly Censored Data
For the case of singly censored data, following Smith and Bain (1976) and Verrill and Johnson (1988), Royston (1993) generalizes the Shapiro-Wilk W-statistic to:

$$W = r(\underline{a}_\Delta, \underline{x}_\Delta)^2 \qquad (9)$$

where for left singly-censored data:

$$\underline{a}_\Delta = (a_{c+1}, a_{c+2}, \ldots, a_N) \qquad (10)$$

$$\underline{x}_\Delta = (x_{(c+1)}, x_{(c+2)}, \ldots, x_{(N)}) \qquad (11)$$

and for right singly-censored data:

$$\underline{a}_\Delta = (a_1, a_2, \ldots, a_n) \qquad (12)$$

$$\underline{x}_\Delta = (x_{(1)}, x_{(2)}, \ldots, x_{(n)}) \qquad (13)$$

Just like the function `gofTest`, when `test="sw"`, the function `gofTestCensored` uses Royston's (1992a) approximation for the coefficients $\underline{a}$ (see the help file for `gofTest`).

Computing P-Values for the Shapiro-Wilk Test
Verrill and Johnson (1988) show that the asymptotic distribution of the statistic in Equation (9) above is normal, but the rate of convergence is "surprisingly slow" even for complete samples. They provide a table of empirical percentiles of the distribution for the W-statistic shown in Equation (9) above for several sample sizes and percentages of censoring.

Based on the tables given in Verrill and Johnson (1988), Royston (1993) approximated the 90'th, 95'th, and 99'th percentiles of the distribution of the z-statistic computed from the W-statistic. (The distribution of this z-statistic is assumed to be normal, but not necessarily a standard normal.) Denote these percentiles by $Z_{0.90}$, $Z_{0.95}$, and $Z_{0.99}$. The true mean and standard deviation of the z-statistic are estimated by the intercept and slope, respectively, from the linear regression of Z_α on $\Phi^{-1}(\alpha)$ for α = 0.9, 0.95, and 0.99, where Φ denotes the cumulative distribution function of the standard normal distribution. The p-value associated with this test is then computed as:

$$p = 1 - \Phi(\frac{z - \mu_z}{\sigma_z}) \qquad (14)$$

Note: Verrill and Johnson (1988) produced their tables based on Type II censoring. Royston's (1993) approximation to the p-value of these tests, however, should be fairly accurate for Type I censored data as well.

Testing Goodness-of-Fit for Any Continuous Distribution
The function `gofTestCensored` extends the Shapiro-Wilk test that accounts for censoring to test for goodness-of-fit for any continuous distribution by using the idea of Chen and Balakrishnan (1995), who proposed a general purpose approximate goodness-of-fit test based on the Cramer-von Mises or Anderson-Darling goodness-of-fit tests for normality. The function `gofTestCensored` modifies the approach of Chen and Balakrishnan (1995) by using the same first 2 steps, and then applying the Shapiro-Wilk test that accounts for censoring:

1. Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote the vector of n ***ordered*** observations, ignoring censoring status. Compute cumulative probabilities for each x_i based on the cumulative distribution function for the hypothesized distribution. That is, compute $p_i = F(x_i, \hat{\theta})$, where $F(x, \theta)$ denotes the hypothesized cumulative distribution function with parameter(s) θ, and $\hat{\theta}$ denotes the estimated parameter(s) using an estimation method that accounts for censoring (e.g., assuming a Gamma distribution with alternative parameterization, call the function `link{egammaAltCensored}`).
2. Compute standard normal deviates based on the computed cumulative probabilities: $y_i = \Phi^{-1}(p_i)$
3. Perform the Shapiro-Wilk goodness-of-fit test (that accounts for censoring) on the y_i's.

Shapiro-Francia Goodness-of-Fit Test (`test="sf"`)
Equation (15) in the help file for `gofTest` shows that for the complete ordered data $\underline{x}$, the Shapiro-Francia W'-statistic is the same as the squared Pearson correlation coefficient associated with a normal probability plot.

Computing Shapiro-Francia W'-Statistic for Censored Data
For the case of singly censored data, following Smith and Bain (1976) and Verrill and Johnson (1988), Royston (1993) extends the computation of the Weisberg-Bingham Approximation to the W'-statistic to the case of singly censored data:

$$\tilde{W}' = r(\underline{c}_\Delta, \underline{x}_\Delta)^2 \qquad (14)$$

where for left singly-censored data:

$$\underline{c}_{\Delta} = (c_{c+1}, c_{c+2}, \ldots, c_N) \quad (15)$$

$$\underline{x}_{\Delta} = (x_{(c+1)}, x_{(c+2)}, \ldots, x_{(N)}) \quad (16)$$

and for right singly-censored data:

$$\underline{a}_{\Delta} = (a_1, a_2, \ldots, a_n) \quad (17)$$

$$\underline{x}_{\Delta} = (x_{(1)}, x_{(2)}, \ldots, x_{(n)}) \quad (18)$$

and $\underline{c}$ is defined as:

$$\underline{c} = \frac{\underline{\tilde{m}}}{[\underline{\tilde{m}}' \underline{\tilde{m}}]^{1/2}} \quad (19)$$

where

$$\tilde{m}_i = \Phi^{-1}\left(\frac{i - (3/8)}{n + (1/4)}\right) \quad (20)$$

and Φ denotes the standard normal cdf. **Note:** Do not confuse the elements of the vector $\underline{c}$ with the scalar c which denotes the number of censored observations. We use $\underline{c}$ here to be consistent with the notation in the help file for `gofTest`.

Just like the function `gofTest`, when `test="sf"`, the function `gofTestCensored` uses Royston's (1992a) approximation for the coefficients $\underline{c}$ (see the help file for `gofTest`).

In general, the Shapiro-Francia test statistic can be extended to multiply censored data using Equation (14) with $\underline{c}_{\Delta}$ defined as the orderd values of c_i associated with uncensored observations, and $\underline{x}_{\Delta}$ defined as the ordered values of x_i associated with uncensored observations:

$$\underline{c}_{\Delta} = \cup_{i \in \Omega} \; c_{(i)} \quad (21)$$

$$\underline{x}_{\Delta} = \cup_{i \in \Omega} \; x_{(i)} \quad (22)$$

and where the plotting positions in Equation (20) are replaced with any of the plotting positions available in `ppointsCensored` (see the description for the argument `prob.method`).

Computing P-Values for the Shapiro-Francia Test

Verrill and Johnson (1988) show that the asymptotic distribution of the statistic in Equation (14) above is normal, but the rate of convergence is "surprisingly slow" even for complete samples. They provide a table of empirical percentiles of the distribution for the $\tilde{W}'$-statistic shown in Equation (14) above for several sample sizes and percentages of censoring.

As for the Shapiro-Wilk test, based on the tables given in Verrill and Johnson (1988), Royston (1993) approximated the 90'th, 95'th, and 99'th percentiles of the distribution of the z-statistic computed from the $\tilde{W}'$-statistic. (The distribution of this z-statistic is assumed to be normal, but not necessarily a standard normal.) Denote these percentiles by $Z_{0.90}$, $Z_{0.95}$, and $Z_{0.99}$. The true mean and standard deviation of the z-statistic are estimated by the intercept and slope, respectively, from the linear regression of Z_{α} on $\Phi^{-1}(\alpha)$ for α = 0.9, 0.95, and 0.99, where Φ denotes the cumulative distribution function of the standard normal distribution. The p-value associated with this test is then computed as:

$$p = 1 - \Phi\left(\frac{z - \mu_z}{\sigma_z}\right) \quad (23)$$

Note: Verrill and Johnson (1988) produced their tables based on Type II censoring. Royston's (1993) approximation to the p-value of these tests, however, should be fairly accurate for Type I censored data as well, although this is an area that requires further investigation.

Testing Goodness-of-Fit for Any Continuous Distribution
The function `gofTestCensored` extends the Shapiro-Francia test that accounts for censoring to test for goodness-of-fit for any continuous distribution by using the idea of Chen and Balakrishnan (1995), who proposed a general purpose approximate goodness-of-fit test based on the Cramer-von Mises or Anderson-Darling goodness-of-fit tests for normality. The function `gofTestCensored` modifies the approach of Chen and Balakrishnan (1995) by using the same first 2 steps, and then applying the Shapiro-Francia test that accounts for censoring:

1. Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote the vector of n ***ordered*** observations, ignoring censoring status. Compute cumulative probabilities for each x_i based on the cumulative distribution function for the hypothesized distribution. That is, compute $p_i = F(x_i, \hat{\theta})$ where $F(x, \theta)$ denotes the hypothesized cumulative distribution function with parameter(s) θ, and $\hat{\theta}$ denotes the estimated parameter(s) using an estimation method that accounts for censoring (e.g., assuming a Gamma distribution with alternative parameterization, call the function `link{egammaAltCensored}`).
2. Compute standard normal deviates based on the computed cumulative probabilities: $y_i = \Phi^{-1}(p_i)$
3. Perform the Shapiro-Francia goodness-of-fit test (that accounts for censoring) on the y_i's.

Probability Plot Correlation Coefficient (PPCC) Goodness-of-Fit Test (`test="ppcc"`)
The function `gofTestCensored` computes the PPCC test statistic using Blom plotting positions. It can be shown that the square of this statistic is equivalent to the Weisberg-Bingham Approximation to the Shapiro-Francia W'-test (Weisberg and Bingham, 1975; Royston, 1993). Thus the PPCC goodness-of-fit test is equivalent to the Shapiro-Francia goodness-of-fit test.

Value

a list of class `"gofCensored"` containing the results of the goodness-of-fit test. See the help files for `gofCensored.object` for details.

Note

The Shapiro-Wilk test (Shapiro and Wilk, 1965) and the Shapiro-Francia test (Shapiro and Francia, 1972) are probably the two most commonly used hypothesis tests to test departures from normality. The Shapiro-Wilk test is most powerful at detecting short-tailed (platykurtic) and skewed distributions, and least powerful against symmetric, moderately long-tailed (leptokurtic) distributions. Conversely, the Shapiro-Francia test is more powerful against symmetric long-tailed distributions and less powerful against short-tailed distributions (Royston, 1992b; 1993).

In practice, almost any goodness-of-fit test will *not* reject the null hypothesis if the number of observations is relatively small. Conversely, almost any goodness-of-fit test *will* reject the null hypothesis if the number of observations is very large, since "real" data are never distributed according to any theoretical distribution (Conover, 1980, p.367). For most cases, however, the distribution of "real" data is close enough to some theoretical distribution that fairly accurate results may be provided by assuming that particular theoretical distribution. One way to asses the goodness of the fit is to use goodness-of-fit tests. Another way is to look at quantile-quantile (Q-Q) plots (see `qqPlotCensored`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Birnbaum, Z.W., and F.H. Tingey. (1951). One-Sided Confidence Contours for Probability Distribution Functions. *Annals of Mathematical Statistics* **22**, 592-596.

Blom, G. (1958). *Statistical Estimates and Transformed Beta Variables*. John Wiley and Sons, New York.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Dallal, G.E., and L. Wilkinson. (1986). An Analytic Approximation to the Distribution of Lilliefor's Test for Normality. *The American Statistician* **40**, 294-296.

D'Agostino, R.B. (1970). Transformation to Normality of the Null Distribution of $g1$. *Biometrika* **57**, 679-681.

D'Agostino, R.B. (1971). An Omnibus Test of Normality for Moderate and Large Size Samples. *Biometrika* **58**, 341-348.

D'Agostino, R.B. (1986b). Tests for the Normal Distribution. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York.

D'Agostino, R.B., and E.S. Pearson (1973). Tests for Departures from Normality. Empirical Results for the Distributions of $b2$ and $\sqrt{b1}$. *Biometrika* **60**(3), 613-622.

D'Agostino, R.B., and G.L. Tietjen (1973). Approaches to the Null Distribution of $\sqrt{b1}$. *Biometrika* **60**(1), 169-173.

Fisher, R.A. (1950). *Statistical Methods for Research Workers*. 11'th Edition. Hafner Publishing Company, New York, pp.99-100.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Kendall, M.G., and A. Stuart. (1991). *The Advanced Theory of Statistics, Volume 2: Inference and Relationship*. Fifth Edition. Oxford University Press, New York.

Royston, J.P. (1992a). Approximating the Shapiro-Wilk W-Test for Non-Normality. *Statistics and Computing* **2**, 117-119.

Royston, J.P. (1992b). Estimation, Reference Ranges and Goodness of Fit for the Three-Parameter Log-Normal Distribution. *Statistics in Medicine* **11**, 897-912.

Royston, J.P. (1992c). A Pocket-Calculator Algorithm for the Shapiro-Francia Test of Non-Normality: An Application to Medicine. *Statistics in Medicine* **12**, 181-184.

Royston, P. (1993). A Toolkit for Testing for Non-Normality in Complete and Censored Samples. *The Statistician* **42**, 37-43.

Ryan, T., and B. Joiner. (1973). *Normal Probability Plots and Tests for Normality*. Technical Report, Pennsylvannia State University, Department of Statistics.

Shapiro, S.S., and R.S. Francia. (1972). An Approximate Analysis of Variance Test for Normality. *Journal of the American Statistical Association* **67**(337), 215-219.

Shapiro, S.S., and M.B. Wilk. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika* **52**, 591-611.

Verrill, S., and R.A. Johnson. (1987). The Asymptotic Equivalence of Some Modified Shapiro-Wilk Statistics – Complete and Censored Sample Cases. *The Annals of Statistics* **15**(1), 413-419.

Verrill, S., and R.A. Johnson. (1988). Tables and Large-Sample Distribution Theory for Censored-Data Correlation Statistics for Testing Normality. *Journal of the American Statistical Association* **83**, 1192-1197.

Weisberg, S., and C. Bingham. (1975). An Approximate Analysis of Variance Test for Non-Normality Suitable for Machine Calculation. *Technometrics* **17**, 133-134.

See Also

gofTest, gofCensored.object, print.gofCensored, plot.gofCensored, shapiro.test, Normal, Lognormal, enormCensored, elnormCensored, elnormAltCensored, qqPlotCensored.

Examples

```
# Generate 30 observations from a gamma distribution with
# parameters mean=10 and cv=1 and then censor observations less than 5.
# Then test the hypothesis that these data came from a gamma
# distribution using the Shapiro-Wilk test.
#
# The p-value for the complete data is p = 0.86, while
# the p-value for the censored data is p = 0.52.
# (Note:  the call to set.seed lets you reproduce this example.)

set.seed(598)

dat <- sort(rgammaAlt(30, mean = 10, cv = 1))
dat
# [1]  0.5313509  1.4741833  1.9936208  2.7980636  3.4509840
# [6]  3.7987348  4.5542952  5.5207531  5.5253596  5.7177872
#[11]  5.7513827  9.1086375  9.8444090 10.6247123 10.9304922
#[16] 11.7925398 13.3432689 13.9562777 14.6029065 15.0563342
#[21] 15.8730642 16.0039936 16.6910715 17.0288922 17.8507891
#[26] 19.1105522 20.2657141 26.3815970 30.2912797 42.8726101

dat.censored <- dat
censored <- dat.censored < 5
dat.censored[censored] <- 5

# Results for complete data:
#---------------------------
gofTest(dat, test = "sw", dist = "gammaAlt")

#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method:                     Shapiro-Wilk GOF Based on
#                                 Chen & Balakrisnan (1995)
#
#Hypothesized Distribution:       Gamma
#
```

```
#Estimated Parameter(s):          mean = 12.4248552
#                                 cv   =  0.7901752
#
#Estimation Method:               MLE
#
#Data:                            dat
#
#Sample Size:                     30
#
#Test Statistic:                  W = 0.981471
#
#Test Statistic Parameter:        n = 30
#
#P-value:                         0.8631802
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Gamma Distribution.


# Results for censored data:
#---------------------------
gof.list <- gofTestCensored(dat.censored, censored, test = "sw",
  distribution = "gammaAlt")
gof.list

#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#                                 (Singly Censored Data)
#                                 Based on Chen & Balakrisnan (1995)
#
#Hypothesized Distribution:       Gamma
#
#Censoring Side:                  left
#
#Censoring Level(s):              5
#
#Estimated Parameter(s):          mean = 12.4911448
#                                 cv   =  0.7617343
#
#Estimation Method:               MLE
#
#Data:                            dat.censored
#
#Censoring Variable:              censored
#
#Sample Size:                     30
#
#Percent Censored:                23.3%
#
#Test Statistic:                  W = 0.9613711
```

```
#
#Test Statistic Parameters:       N     = 30.0000000
#                                DELTA =  0.2333333
#
#P-value:                        0.522329
#
#Alternative Hypothesis:          True cdf does not equal the
#                                Gamma Distribution.

# Plot the results for the censored data
#---------------------------------------
dev.new()
plot(gof.list)

#==========

# Continue the above example, but now test the hypothesis that
# these data came from a lognormal distribution
# (alternative parameterization) using the Shapiro-Wilk test.
#
# The p-value for the complete data is p = 0.056, while
# the p-value for the censored data is p = 0.11.

# Results for complete data:
#---------------------------
gofTest(dat, test = "sw", dist = "lnormAlt")

#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#
#Hypothesized Distribution:       Lognormal
#
#Estimated Parameter(s):          mean = 13.757239
#                                 cv   =  1.148872
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     30
#
#Test Statistic:                  W = 0.9322226
#
#Test Statistic Parameter:        n = 30
#
#P-value:                         0.05626823
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.
```

```
# Results for censored data:
#---------------------------
gof.list <- gofTestCensored(dat.censored, censored, test = "sw",
  distribution = "lnormAlt")
gof.list

#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#                                 (Singly Censored Data)
#
#Hypothesized Distribution:       Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              5
#
#Estimated Parameter(s):          mean = 13.0382221
#                                 cv   =  0.9129512
#
#Estimation Method:               MLE
#
#Data:                            dat.censored
#
#Censoring Variable:              censored
#
#Sample Size:                     30
#
#Percent Censored:                23.3%
#
#Test Statistic:                  W = 0.9292406
#
#Test Statistic Parameters:       N     = 30.0000000
#                                 DELTA =  0.2333333
#
#P-value:                         0.114511
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.

# Plot the results for the censored data
#---------------------------------------
dev.new()
plot(gof.list)

#----------

# Redo the above example, but specify the quasi-minimum variance
# unbiased estimator of the mean.  Note that the method of
# estimating the parameters has no effect on the goodness-of-fit
# test (see the DETAILS section above).
```

```
gofTestCensored(dat.censored, censored, test = "sw",
  distribution = "lnormAlt", est.arg.list = list(method = "qmvue"))

#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#                                 (Singly Censored Data)
#
#Hypothesized Distribution:       Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              5
#
#Estimated Parameter(s):          mean = 12.8722749
#                                 cv   =  0.8712549
#
#Estimation Method:               Quasi-MVUE
#
#Data:                            dat.censored
#
#Censoring Variable:              censored
#
#Sample Size:                     30
#
#Percent Censored:                23.3%
#
#Test Statistic:                  W = 0.9292406
#
#Test Statistic Parameters:       N     = 30.0000000
#                                 DELTA =  0.2333333
#
#P-value:                         0.114511
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.

#----------
# Clean up

rm(dat, dat.censored, censored, gof.list)
graphics.off()

#==========

# Check the assumption that the silver data stored in Helsel.Cohn.88.silver.df
# follows a lognormal distribution and plot the goodness-of-fit test results.
# Note that the small p-value and the shape of the Q-Q plot
# (an inverted S-shape) suggests that the log transformation is not quite strong
# enough to "bring in" the tails (i.e., the log-transformed silver data has tails
```

```
# that are slightly too long relative to a normal distribution).
# Helsel and Cohn (1988, p.2002) note that the gross outlier of 560 mg/L tends to
# make the shape of the data resemble a gamma distribution.

dum.list <- with(Helsel.Cohn.88.silver.df,
  gofTestCensored(Ag, Censored, test = "sf", dist = "lnorm"))

dum.list
#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#------------------------------
#
#Test Method:                     Shapiro-Francia GOF
#                                 (Multiply Censored Data)
#
#Hypothesized Distribution:       Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):               0.1  0.2  0.3  0.5  1.0  2.0  2.5  5.0
#                                  6.0 10.0 20.0 25.0
#
#Estimated Parameter(s):          meanlog = -1.040572
#                                 sdlog   =  2.354847
#
#Estimation Method:               MLE
#
#Data:                            Ag
#
#Censoring Variable:              Censored
#
#Sample Size:                     56
#
#Percent Censored:                60.7%
#
#Test Statistic:                  W = 0.8957198
#
#Test Statistic Parameters:       N     = 56.0000000
#                                 DELTA =  0.6071429
#
#P-value:                         0.03490314
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.

dev.new()
plot(dum.list)

#----------

# Clean up
#---------
```

```
rm(dum.list)
graphics.off()

#==========

# Chapter 15 of USEPA (2009) gives several examples of looking
# at normal Q-Q plots and estimating the mean and standard deviation
# for manganese concentrations (ppb) in groundwater at five background wells.
# In EnvStats these data are stored in the data frame
# EPA.09.Ex.15.1.manganese.df.

# Here we will test whether the data appear to come from a normal
# distribution, then we will test to see whether they appear to come
# from a lognormal distribution.
#---------------------------------------------------------------------


# First look at the data:
#------------------------

EPA.09.Ex.15.1.manganese.df

#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#...
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

longToWide(EPA.09.Ex.15.1.manganese.df,
  "Manganese.Orig.ppb", "Sample", "Well",
  paste.row.name = TRUE)

#         Well.1 Well.2 Well.3 Well.4 Well.5
#Sample.1     <5     <5     <5    6.3   17.9
#Sample.2   12.1    7.7    5.3   11.9   22.7
#Sample.3   16.9   53.6   12.6     10    3.3
#Sample.4   21.6    9.5  106.3     <2    8.4
#Sample.5     <2   45.9   34.5   77.2     <2


# Now test whether the data appear to come from
# a normal distribution.  Note that these data
# are multiply censored, so we'll use the
# Shapiro-Francia test.
#----------------------------------------------

gof.normal <- with(EPA.09.Ex.15.1.manganese.df,
  gofTestCensored(Manganese.ppb, Censored, test = "sf"))

gof.normal
```

```
#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#------------------------------
#
#Test Method:                     Shapiro-Francia GOF
#                                 (Multiply Censored Data)
#
#Hypothesized Distribution:       Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          mean = 15.23508
#                                 sd   = 30.62812
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Test Statistic:                  W = 0.8368016
#
#Test Statistic Parameters:       N     = 25.00
#                                 DELTA =  0.24
#
#P-value:                         0.004662658
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Normal Distribution.

# Plot the results:
#------------------

dev.new()
plot(gof.normal)

#----------

# Now test to see whether the data appear to come from
# a lognormal distribuiton.
#-----------------------------------------------------

gof.lognormal <- with(EPA.09.Ex.15.1.manganese.df,
  gofTestCensored(Manganese.ppb, Censored, test = "sf",
  distribution = "lnorm"))
```

```
gof.lognormal

#Results of Goodness-of-Fit Test
#Based on Type I Censored Data
#------------------------------
#
#Test Method:                     Shapiro-Francia GOF
#                                 (Multiply Censored Data)
#
#Hypothesized Distribution:       Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          meanlog = 2.215905
#                                 sdlog   = 1.356291
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              Censored
#
#Sample Size:                     25
#
#Percent Censored:                24%
#
#Test Statistic:                  W = 0.9864426
#
#Test Statistic Parameters:       N     = 25.00
#                                 DELTA =  0.24
#
#P-value:                         0.9767731
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.

# Plot the results:
#------------------

dev.new()
plot(gof.lognormal)

#----------

# Clean up
#---------

rm(gof.normal, gof.lognormal)
graphics.off()
```

gofTwoSample.object *S3 Class "gofTwoSample"*

Description

Objects of S3 class "gofTwoSample" are returned by the **EnvStats** function gofTest when both the x and y arguments are supplied.

Details

Objects of S3 class "gofTwoSample" are lists that contain information about the assumed distribution, the estimated or user-supplied distribution parameters, and the test statistic and p-value.

Value

Required Components

The following components must be included in a legitimate list of class "gofTwoSample".

distribution	a character string with the value "Equal".
statistic	a numeric scalar with a names attribute containing the name and value of the goodness-of-fit statistic.
sample.size	a numeric scalar containing the number of non-missing observations in the sample used for the goodness-of-fit test.
parameters	numeric vector with a names attribute containing the name(s) and value(s) of the parameter(s) associated with the test statistic given in the statistic component.
p.value	numeric scalar containing the p-value associated with the goodness-of-fit statistic.
alternative	character string indicating the alternative hypothesis.
method	character string indicating the name of the goodness-of-fit test.
data	a list of length 2 containing the numeric vectors actually used for the goodness-of-fit test (i.e., the original data but with any missing or infinite values removed).
data.name	a character vector of length 2 indicating the name of the data object used for the x argument and the name of the data object used for the y argument in the goodness-of-fit test.

Optional Component

The following component is included when the arguments x and/or y contain missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values.

bad.obs	numeric vector of length 2 indicating the number of missing (NA), undefined (NaN) and/or infinite (Inf, -Inf) values that were removed from the data in the x and y arguments prior to performing the goodness-of-fit test.

Methods

Generic functions that have methods for objects of class "gofTwoSample" include:
print, plot.

Note

Since objects of class "gofTwoSample" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

print.gofTwoSample, plot.gofTwoSample, Goodness-of-Fit Tests.

Examples

```
# Create an object of class "gofTwoSample", then print it out.

# Generate 20 observations from a normal distribution with mean=3 and sd=2, and
# generate 10 observaions from a normal distribution with mean=2 and sd=2 then
# test whether these sets of observations come from the same distribution.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(300)
dat1 <- rnorm(20, mean = 3, sd = 2)
dat2 <- rnorm(10, mean = 1, sd = 2)
gofTest(x = dat1, y = dat2, test = "ks")

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     2-Sample K-S GOF
#
#Hypothesized Distribution:       Equal
#
#Data:                            x = dat1
#                                 y = dat2
#
#Sample Sizes:                    n.x = 20
#                                 n.y = 10
#
#Test Statistic:                  ks = 0.7
#
#Test Statistic Parameters:       n = 20
#                                 m = 10
#
#P-value:                         0.001669561
#
#Alternative Hypothesis:          The cdf of 'dat1' does not equal
```

```
#                                  the cdf of 'dat2'.

#----------
# Clean up

rm(dat1, dat2)
```

---

gpqCiNormCensored *Generalized Pivotal Quantity for Confidence Interval for the Mean of a Normal Distribution Based on Censored Data*

Description

Generate a generalized pivotal quantity (GPQ) for a confidence interval for the mean of a Normal distribution based on singly or multiply censored data.

Usage

```
gpqCiNormSinglyCensored(n, n.cen, probs, nmc, method = "mle",
  censoring.side = "left", seed = NULL, names = TRUE)

gpqCiNormMultiplyCensored(n, cen.index, probs, nmc, method = "mle",
  censoring.side = "left", seed = NULL, names = TRUE)
```

Arguments

n	positive integer ≥ 3 indicating the sample size.
n.cen	for the case of singly censored data, a positive integer indicating the number of censored observations. The value of n.cen must be between 1 and n-2, inclusive.
cen.index	for the case of multiply censored data, a sorted vector of unique integers indicating the indices of the censored observations when the observations are "ordered". The length of cen.index must be between 1 and n-2, inclusive, and the values of cen.index must be between 1 and n.
probs	numeric vector of values between 0 and 1 indicating the confidence level(s) associated with the GPQ(s).
nmc	positive integer ≥ 10 indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).
method	character string indicating the method to use for parameter estimation. For singly censored data, possible values are "mle" (the default), "bcmle", "qq.reg", "qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", "iterative.impute.w.qq.reg", "m.est", and "half.cen.level". See the help file for enormCensored for details.

For multiply censored data, possible values are "mle" (the default), "qq.reg", "impute.w.qq.reg", and "half.cen.level". See the help file for enormCensored for details.

censoring.side character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".

seed positive integer to pass to the function set.seed. This argument is ignored if seed=NULL (the default). Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.

names a logical scalar passed to quantile indicating whether to add a names attribute to the resulting GPQ(s). The default value is names=TRUE.

Details

The functions gpqCiNormSinglyCensored and gpqCiNormMultiplyCensored are called by enormCensored when ci.method="gpq". They are used to construct generalized pivotal quantities to create confidence intervals for the mean μ of an assumed normal distribution.

This idea was introduced by Schmee et al. (1985) in the context of Type II singly censored data. The function gpqCiNormSinglyCensored generates GPQs using a modification of Algorithm 12.1 of Krishnamoorthy and Mathew (2009, p. 329). Algorithm 12.1 is used to generate GPQs for a tolerance interval. The modified algorithm for generating GPQs for confidence intervals for the mean μ is as follows:

1. Generate a random sample of n observations from a standard normal (i.e., N(0,1)) distribution and let $z_{(1)}, z_{(2)}, \ldots, z_{(n)}$ denote the ordered (sorted) observations.
2. Set the smallest n.cen observations as censored.
3. Compute the estimates of μ and σ by calling enormCensored using the method specified by the method argument, and denote these estimates as $\hat{\mu}^*,\ \hat{\sigma}^*$.
4. Compute the t-like pivotal quantity $\hat{t} = \hat{\mu}^*/\hat{\sigma}^*$.
5. Repeat steps 1-4 nmc times to produce an empirical distribution of the t-like pivotal quantity.

A two-sided $(1-\alpha)100\%$ confidence interval for μ is then computed as:

$$[\hat{\mu} - \hat{t}_{1-(\alpha/2)}\hat{\sigma},\ \hat{\mu} - \hat{t}_{\alpha/2}\hat{\sigma}]$$

where $\hat{t}_p$ denotes the p'th empirical quantile of the nmc generated $\hat{t}$ values.

Schmee at al. (1985) derived this method in the context of Type II singly censored data (for which these limits are exact within Monte Carlo error), but state that according to Regal (1982) this method produces confidence intervals that are close apporximations to the correct limits for Type I censored data.

The function gpqCiNormMultiplyCensored is an extension of this idea to multiply censored data. The algorithm is the same as for singly censored data, except Step 2 changes to:

2. Set observations as censored for elements of the argument cen.index that have the value TRUE.

The functions gpqCiNormSinglyCensored and gpqCiNormMultiplyCensored are computationally intensive and provided to the user to allow you to create your own tables.

Value

a numeric vector containing the GPQ(s).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Regal, R. (1982). Applying Order Statistic Censored Normal Confidence Intervals to Time Censored Data. Unpublished manuscript, University of Minnesota, Duluth, Department of Mathematical Sciences.

Schmee, J., D.Gladstein, and W. Nelson. (1985). Confidence Limits for Parameters of a Normal Distribution from Singly Censored Samples, Using Maximum Likelihood. *Technometrics* **27**(2) 119–128.

See Also

enormCensored, estimateCensored.object.

Examples

```
# Reproduce the entries for n=10 observations with n.cen=6 in Table 4
# of Schmee et al. (1985, p.122).
#
# Notes:
# 1. This table applies to right-censored data, and the
#    quantity "r" in this table refers to the number of
#    uncensored observations.
#
# 2. Passing a value for the argument "seed" simply allows
#    you to reproduce this example.

# NOTE:  Here to save computing time for the sake of example, we will specify
#        just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009)
#        suggest *10,000* Monte Carlos.

# Here are the values given in Schmee et al. (1985):
Schmee.values <- c(-3.59, -2.60, -1.73, -0.24, 0.43, 0.58, 0.73)
probs <- c(0.025, 0.05, 0.1, 0.5, 0.9, 0.95, 0.975)
names(Schmee.values) <- paste(probs * 100, "%", sep = "")

Schmee.values
# 2.5%    5%   10%   50%   90%   95% 97.5%
#-3.59 -2.60 -1.73 -0.24  0.43  0.58  0.73

gpqs <- gpqCiNormSinglyCensored(n = 10, n.cen = 6, probs = probs,
  nmc = 100, censoring.side = "right", seed = 529)
```

```
round(gpqs, 2)
# 2.5%    5%   10%   50%   90%   95% 97.5%
#-2.46 -2.03 -1.38 -0.14  0.54  0.65  0.84

# This is what you get if you specify nmc = 1000 with the
# same value for seed:
#-----------------------------------------------
# 2.5%    5%   10%   50%   90%   95% 97.5%
#-3.50 -2.49 -1.67 -0.25  0.41  0.57  0.71


# Clean up
#---------
rm(Schmee.values, probs, gpqs)

#==========

# Example of using gpqCiNormMultiplyCensored
#-------------------------------------------

# Consider the following set of multiply left-censored data:
dat <- 12:16
censored <- c(TRUE, FALSE, TRUE, FALSE, FALSE)

# Since the data are "ordered" we can identify the indices of the
# censored observations in the ordered data as follow:

cen.index <- (1:length(dat))[censored]
cen.index
#[1] 1 3

# Now we can generate a GPQ using gpqCiNormMultiplyCensored.
# Here we'll generate a GPQs to use to create a
# 95% confidence interval for left-censored data.

# NOTE:  Here to save computing time for the sake of example, we will specify
#        just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009)
#        suggest *10,000* Monte Carlos.

gpqCiNormMultiplyCensored(n = 5, cen.index = cen.index,
  probs = c(0.025, 0.975), nmc = 100, seed = 237)
#     2.5%     97.5%
#-1.315592  1.848513

#----------

# Clean up
#---------
rm(dat, censored, cen.index)
```

gpqTolIntNormCensored *Generalized Pivotal Quantity for Tolerance Interval for a Normal Distribution Based on Censored Data*

Description

Generate a generalized pivotal quantity (GPQ) for a tolerance interval for a Normal distribution based on singly or multiply censored data.

Usage

```
gpqTolIntNormSinglyCensored(n, n.cen, p, probs, nmc, method = "mle",
  censoring.side = "left", seed = NULL, names = TRUE)

gpqTolIntNormMultiplyCensored(n, cen.index, p, probs, nmc, method = "mle",
  censoring.side = "left", seed = NULL, names = TRUE)
```

Arguments

n	positive integer ≥ 3 indicating the sample size.
n.cen	for the case of singly censored data, a positive integer indicating the number of censored observations. The value of n.cen must be between 1 and n-2, inclusive.
cen.index	for the case of multiply censored data, a sorted vector of unique integers indicating the indices of the censored observations when the observations are "ordered". The length of cen.index must be between 1 and n-2, inclusive, and the values of cen.index must be between 1 and n.
p	numeric scalar strictly greater than 0 and strictly less than 1 indicating the quantile for which to generate the GPQ(s) (i.e., the coverage associated with a one-sided tolerance interval).
probs	numeric vector of values between 0 and 1 indicating the confidence level(s) associated with the GPQ(s).
nmc	positive integer ≥ 10 indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).
method	character string indicating the method to use for parameter estimation. For singly censored data, possible values are "mle" (the default), "bcmle", "qq.reg", "qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", "iterative.impute.w.qq.reg", "m.est", and "half.cen.level". See the help file for enormCensored for details. For multiply censored data, possible values are "mle" (the default), "qq.reg", "impute.w.qq.reg", and "half.cen.level". See the help file for enormCensored for details.

`censoring.side` character string indicating on which side the censoring occurs. The possible values are `"left"` (the default) and `"right"`.

`seed` positive integer to pass to the function `set.seed`. This argument is ignored if `seed=NULL` (the default). Using the `seed` argument lets you reproduce the exact same result if all other arguments stay the same.

`names` a logical scalar passed to `quantile` indicating whether to add a names attribute to the resulting GPQ(s). The default value is `names=TRUE`.

Details

The function `gpqTolIntNormSinglyCensored` generates GPQs as described in Algorithm 12.1 of Krishnamoorthy and Mathew (2009, p. 329). The function `gpqTolIntNormMultiplyCensored` is an extension of this idea to multiply censored data. These functions are called by `tolIntNormCensored` when `ti.method="gpq"`, and also by `eqnormCensored` when `ci=TRUE` and `ci.method="gpq"`. See the help files for these functions for an explanation of GPQs.

Note that technically these are only GPQs if the data are Type II censored. However, Krishnamoorthy and Mathew (2009, p. 328) state that in the case of Type I censored data these quantities should approximate the true GPQs and the results appear to be satisfactory, even for small sample sizes.

The functions `gpqTolIntNormSinglyCensored` and `gpqTolIntNormMultiplyCensored` are computationally intensive and provided to the user to allow you to create your own tables.

Value

a numeric vector containing the GPQ(s).

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

See Also

`tolIntNormCensored`, `eqnormCensored`, `enormCensored`, `estimateCensored.object`.

Examples

```
# Reproduce the entries for n=10 observations with n.cen=1 in Table 12.2
# of Krishnamoorthy and Mathew (2009, p.331).
#
# (Note: passing a value for the argument "seed" simply allows you to
# reproduce this example.)
#
# NOTE:  Here to save computing time for the sake of example, we will specify
#        just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009)
#        suggest *10,000* Monte Carlos.

gpqTolIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.05, probs = 0.05,
  nmc = 100, seed = 529)
#       5%
#-3.483403


gpqTolIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.1, probs = 0.05,
  nmc = 100, seed = 497)
#      5%
#-2.66705


gpqTolIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.9, probs = 0.95,
  nmc = 100, seed = 623)
#     95%
#2.478654

gpqTolIntNormSinglyCensored(n = 10, n.cen = 1, p = 0.95, probs = 0.95,
  nmc = 100, seed = 623)
#     95%
#3.108452

#==========

# Example of using gpqTolIntNormMultiplyCensored
#-----------------------------------------------

# Consider the following set of multiply left-censored data:
dat <- 12:16
censored <- c(TRUE, FALSE, TRUE, FALSE, FALSE)

# Since the data are "ordered" we can identify the indices of the
# censored observations in the ordered data as follow:

cen.index <- (1:length(dat))[censored]
cen.index
#[1] 1 3

# Now we can generate a GPQ using gpqTolIntNormMultiplyCensored.
# Here we'll generate a GPQ corresponding to an upper tolerance
# interval with coverage 90% with 95% confidence for
```

```
# left-censored data.
# NOTE:  Here to save computing time for the sake of example, we will specify
#        just 100 Monte Carlos, whereas Krishnamoorthy and Mathew (2009)
#        suggest *10,000* Monte Carlos.

gpqTolIntNormMultiplyCensored(n = 5, cen.index = cen.index, p = 0.9,
  probs = 0.95, nmc = 100, seed = 237)
#     95%
#3.952052

#==========

# Clean up
#---------
rm(dat, censored, cen.index)
```

`Graham.et.al.75.etu.df`

Ethylene Thiourea Dose-Response Data

Description

These data are the results of an experiment in which different groups of rats were exposed to different concentration levels of ethylene thiourea (ETU), which is a decomposition product of a certain class of fungicides that can be found in treated foods (Graham et al., 1975; Rodricks, 1992, p.133). In this experiment, the outcome of concern was the number of rats that developed thyroid tumors.

Usage

```
Graham.et.al.75.etu.df
```

Format

A data frame with 6 observations on the following 4 variables.

`dose` a numeric vector of dose (ppm/day) of ETU.

`tumors` a numeric vector indicating number of rats that developed thyroid tumors.

`n` a numeric vector indicating the number of rats in the dose group.

`proportion` a numeric vector indicating proportion of rats that developed thyroid tumors.

Source

Graham, S.L., K.J. Davis, W.H. Hansen, and C.H. Graham. (1975). Effects of Prolonged Ethylene Thiourea Ingestion on the Thyroid of the Rat. *Food and Cosmetics Toxicology*, **13**(5), 493–499.

References

Rodricks, J.V. (1992). *Calculated Risks: The Toxicity and Human Health Risks of Chemicals in Our Environment*. Cambridge University Press, New York, p.133.

`Grice.Bain.80.mat` *Adjusted Alpha Levels to Compute Confidence Intervals for the Mean of a Gamma Distribution*

Description

Adjusted alpha levels to compute confidence intervals for the mean of a gamma distribution, as presented in Table 2 of Grice and Bain (1980).

Usage

```
data("Grice.Bain.80.mat")
```

Format

A matrix of dimensions 5 by 7, with the first dimension indicating the sample size (between 5 and Inf), and the second dimension indicating the assumed significance level associated with the confidence interval (between 0.005 and 0.25). The assumed confidence level is 1 - assumed significance level.

Details

See Grice and Bain (1980) and the help file for `egamma` for more information. The data in this matrix are used when the function `egamma` is called with `ci.method="chisq.adj"`.

Source

Grice, J.V., and L.J. Bain. (1980). Inferences Concerning the Mean of the Gamma Distribution. *Journal of the American Statistical Association* **75**, 929-933.

References

Grice, J.V., and L.J. Bain. (1980). Inferences Concerning the Mean of the Gamma Distribution. *Journal of the American Statistical Association* **75**, 929-933.

USEPA. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C.

Examples

```
# Look at Grice.Bain.80.mat

Grice.Bain.80.mat
#         alpha.eq.005 alpha.eq.01 alpha.eq.025 alpha.eq.05 alpha.eq.075
```

```
#n.eq.5         0.0000      0.0000      0.0010      0.0086      0.0234
#n.eq.10        0.0003      0.0015      0.0086      0.0267      0.0486
#n.eq.20        0.0017      0.0046      0.0159      0.0380      0.0619
#n.eq.40        0.0030      0.0070      0.0203      0.0440      0.0685
#n.eq.Inf       0.0050      0.0100      0.0250      0.0500      0.0750

#         alpha.eq.10 alpha.eq.25
#n.eq.5        0.0432      0.2038
#n.eq.10       0.0724      0.2294
#n.eq.20       0.0866      0.2403
#n.eq.40       0.0934      0.2453
#n.eq.Inf      0.1000      0.2500
```

`Helsel.Cohn.88.app.b.df`

Example of Multiply Left-censored Data from Literature

Description

Made up multiply left-censored data. There are 9 observations out of a total of 18 that are reported as $<DL$, where DL denotes a detection limit. There are 2 distinct detection limits.

Usage

```
Helsel.Cohn.88.app.b.df
```

Format

A data frame with 18 observations on the following 3 variables.

`Conc.orig` a character vector of original observations

`Conc` a numeric vector of observations with censored values coded to censoring levels

`Censored` a logical vector indicating which values are censored

Source

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997–2004, Appendix B.

Helsel.Cohn.88.silver.df

Silver Concentrations From An Interlab Comparison

Description

Silver concentrations (mg/L) from an interlab comparison. There are 34 observations out of a total of 56 that are reported as <DL, where DL denotes a detection limit. There are 12 distinct detection limits.

Usage

```
Helsel.Cohn.88.silver.df
```

Format

A data frame with 56 observations on the following 4 variables.

`Ag.orig` a character vector of original silver concentrations (mg/L)

`Ag` a numeric vector with nondetects coded to the detection limit

`Censored` a logical vector indicating which observations are censored

`log.Ag` the natural logarithm of `Ag`

Source

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997–2004.

References

Janzer, V.J. (1986). *Report of the U.S. Geological Survey's Analytical Evaluation Program–Standard Reference Water Samples M6, M94, T95, N16, P8, and SED3*. Technical Report, Branch of Quality Assurance, U.S. Geological Survey, Arvada, CO.

Helsel.Hirsch.02.Mayfly.df

Paired Counts of Mayfly Nymphs Above and Below Industrial Outfalls

Description

Counts of mayfly nymphs at low flow in 12 small streams. In each stream, counts were recorded above and below industrial outfalls.

Usage

```
data(Helsel.Hirsch.02.Mayfly.df)
```

Format

A data frame with 24 observations on the following 3 variables.

`Mayfly.Count` Number of mayfly nymphs counted

`Stream` a factor indicating the stream number

`Location` a factor indicating the location of the count (above vs. below)

Source

Helsel, D.R., and R.M. Hirsch. (2002). *Statistical Methods in Water Resources Research*. Techniques of Water Resources Investigations, Book 4, Chapter A3. U.S. Geological Survey, 139–140. https://pubs.usgs.gov/tm/04/a03/tm4a3.pdf.

`HoskingEtAl1985` *Abstract: Hosking et al. (1985)*

Description

Detailed abstract of the manuscript:

Hosking, J.R.M., J.R. Wallis, and E.F. Wood. (1985). Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments. *Technometrics* **27**(3), 251–261.

Details

Abstract

Hosking et al. (1985) use the method of probability-weighted moments, introduced by Greenwood et al. (1979), to estimate the parameters of the generalized extreme value distribution (GEVD) with parameters `location`=η, `scale`=θ, and `shape`=κ. Hosking et al. (1985) derive the asymptotic distributions of the probability-weighted moment estimators (PWME), and compare the asymptotic and small-sample statistical properties (via computer simulation) of the PWME with maximum likelihood estimators (MLE) and Jenkinson's (1969) method of sextiles estimators (JSE). They also compare the statistical properties of quantile estimators (which are based on the distribution parameter estimators). Finally, they derive a test of the null hypothesis that the shape parameter is zero, and assess its performance via computer simulation.

Hosking et al. (1985) note that when $\kappa \leq -1$, the moments and probability-weighted moments of the GEVD do not exist. They also note that in practice the shape parameter usually lies between -1/2 and 1/2.

Hosking et al. (1985) found that the asymptotic efficiency of the PWME (the limit as the sample size approaches infinity of the ratio of the variance of the MLE divided by the variance of the PWME) tends to 0 as the shape parameter approaches 1/2 or -1/2. For values of κ within the range $[-0.2, 0.2]$, however, the efficiency of the estimator of location is close to 100 are greater than 70 Hosking et al. (1985) found that the asymptotic efficiency of the PWME is poor for κ outside the range $[-0.2, 0.2]$.

For the small sample results, Hosking et al. (1985) considered several possible forms of the PWME (see equations (8)-(10) below). The best overall results were given by the plotting-position PWME defined by equations (9) and (10) with $a = 0.35$ and $b = 0$.

Small sample results for estimating the parameters show that for $n \geq 50$ all three methods give almost identical results. For $n < 50$ the results for the different estimators are a bit different, but not dramatically so. The MLE tends to be slightly less biased than the other two methods. For estimating the shape parameter, the MLE has a slightly larger standard deviation, and the PWME has consistently the smallest standard deviation.

Small sample results for estimating large quantiles show that for $n \geq 100$ all three methods are comparable. For $n < 100$ the PWME and JSE are comparable and in general have much smaller standard deviations than the MLE. All three methods are very inaccurate for estimating large quantiles in small samples, especially when $\kappa < 0$.

Hosking et al. (1985) derive a test of the null hypothesis $H_0 : \kappa = 0$ based on the PWME of κ. The test is performed by computing the statistic:

$$z = \frac{\hat{\kappa_{pwme}}}{\sqrt{0.5663/n}} \quad (1)$$

and comparing z to a standard normal distribution (see `zTestGevdShape`). Based on computer simulations using the plotting-position PWME, they found that a sample size of $n \geq 25$ ensures an adequate normal approximation. They also found this test has power comparable to the modified likelihood-ratio test, which was found by Hosking (1984) to be the best overall test of $H_0 : \kappa = 0$ of the thirteen tests he considered.

More Details

Probability-Weighted Moments and Parameters of the GEVD

The definition of a probability-weighted moment, introduced by Greenwood et al. (1979), is as follows. Let X denote a random variable with cdf F, and let $x(p)$ denote the p'th quantile of the distribution. Then the ijk'th probability-weighted moment is given by:

$$M(i,j,k) = E[X^i F^j (1-F)^k] = \int_0^1 [x(F)]^i F^j (1-F)^k \, dF \quad (2)$$

where i, j, and k are real numbers.

Hosking et al. (1985) set

$$\beta_j = M(i,j,0) \quad (3)$$

and Greenwood et al. (1979) show that

$$\beta_j = \frac{1}{j+1} E[X_{j+1:j+1}] \quad (4)$$

where

$$E[X_{j+1:j+1}]$$

denotes the expected value of the $j+1$'th order statistic (i.e., the maximum) in a sample of size $j+1$. Hosking et al. (1985) show that if X has a GEVD with parameters `location`=η, `scale`=θ, and `shape`=κ, where $\kappa \neq 0$, then

$$\beta_j = \frac{1}{j+1}\{\eta + \frac{\theta[1-(j+1)^{-\kappa}\Gamma(1+\kappa)]}{\kappa}\} \quad (5)$$

for $\kappa > -1$, where $\Gamma()$ denotes the gamma function. Thus,

$$\beta_0 = \eta + \frac{\theta[1 - \Gamma(1+\kappa)]}{\kappa} \quad (6)$$

$$2\beta_1 - \beta_0 = \frac{\theta[\Gamma(1+\kappa)](1 - 2^{-\kappa})}{\kappa} \quad (7)$$

$$\frac{3\beta_2 - \beta_0}{2\beta_1 - \beta_0} = \frac{1 - 3^{-\kappa}}{1 - 2^{-kappa}} \quad (8)$$

Estimating Distribution Parameters

Using the results of Landwehr et al. (1979), Hosking et al. (1985) show that given a random sample of n values from some arbitrary distribution, an unbiased, distribution-free, and parameter-free estimator of the probability-weighted moment $\beta_j = M(i, j, 0)$ defined above is given by:

$$b_j = \frac{1}{n} \sum_{i=j+1}^{n} x_{i,n} \frac{\binom{i-1}{j}}{\binom{n-1}{j}} \quad (9)$$

where the quantity $x_{i,n}$ denotes the i'th order statistic in the random sample of size n. Hosking et al. (1985) note that this estimator is closely related to U-statistics (Hoeffding, 1948; Lehmann, 1975, pp. 362-371).

An alternative "plotting position" estimator is given by:

$$\hat{\beta}_j[p_{i,n}] = \frac{1}{n} \sum_{i=1}^{n} p_{i,n}^{j} x_{i,n} \quad (10)$$

where

$$p_{i,n} = \hat{F}(x_{i,n}) \quad (11)$$

denotes the plotting position of the i'th order statistic in the random sample of size n, that is, a distribution-free estimate of the cdf of X evaluated at the i'th order statistic. Typically, plotting positions have the form:

$$p_{i,n} = \frac{i - a}{n + b} \quad (12)$$

where $b > -a > -1$. For this form of plotting position, the plotting-position estimators in (10) are asymptotically equivalent to the U-statistic estimators in (9).

Although the unbiased and plotting position estimators are asymptotically equivalent (Hosking, 1990), Hosking and Wallis (1995) recommend using the unbiased estimator for almost all applications because of its superior performance in small and moderate samples.

Using equations (6)-(8) above, i.e., the three equations involving β_0, β_1, and β_2, Hosking et al. (1985) define the probability-weighted moment estimators of η, θ, and κ as the solutions to these three simultaneous equations, with the values of the probability-weighted moments replaced by their estimated values (using either the unbiased or plotting posistion estiamtors in (9) and (10) above). Hosking et al. (1985) note that the third equation (equation (8)) must be solved iteratively for the PWME of κ. Using the unbiased estimators of the PWMEs to solve for κ, the PWMEs of η and θ are given by:

$$\hat{\eta}_{pwme} = b_0 + \frac{\hat{\theta}_{pwme}[\Gamma(1 + \hat{\kappa}_{pwme}) - 1]}{\hat{\kappa}_{pwme}} \quad (13)$$

$$\hat{\theta}_{pwme} = \frac{(2b_1 - b_0)\hat{\kappa}_{pwme}}{\Gamma(1+\hat{\kappa}_{pwme})(1-2^{-\hat{\kappa}_{pwme}})} \quad (14)$$

Hosking et al. (1985) show that when the unbiased estimates of the PWMEs are used to estimate the probability-weighted moments, the estimates of θ and κ satisfy the feasibility criteria

$$\hat{\theta}_{pwme} > 0;\ \hat{\kappa}_{pwme} > -1$$

almost surely.

Hosking et al. (1985) show that the asymptotic distribution of the PWME is multivariate normal with mean equal to (η, θ, κ), and they derive the formula for the asymptotic variance-covariance matrix as:

$$V_{\hat{\eta},\hat{\theta},\hat{\kappa}} = \frac{1}{n} G V_{\hat{\beta}_0,\hat{\beta}_1,\hat{\beta}_2} G^T \quad (15)$$

where

$$V_{\hat{\beta}_0,\hat{\beta}_1,\hat{\beta}_2}$$

denotes the variance-covariance matrix of the estimators of the probability-weighted moments defined in either equation (9) or (10) above (recall that these two estimators are asymptotically equivalent), and the matrix G is defined by:

$$G_{i1} = \frac{\partial \eta}{\partial \beta_{i-1}},\ G_{i2} = \frac{\partial \theta}{\partial \beta_{i-1}},\ G_{i3} = \frac{\partial \kappa}{\partial \beta_{i-1}} \quad (16)$$

for $i = 1, 2, 3$. Hosking et al. (1985) provide formulas for the matrix

$$V_{\hat{\beta}_0,\hat{\beta}_1,\hat{\beta}_2}$$

in Appendix C of their manuscript. Note that there is a typographical error in equation (C.11) (Jon Hosking, personal communication, 1996). In the second line of this equation, the quantity $-(r+s)^{-k}$ should be replaced with $-(r+s)^{-2k}$.

The matrix G in equation (16) is not easily computed. Its inverse, however, is easy to compute and then can be inverted numerically (Jon Hosking, 1996, personal communication). The inverse of G is given by:

$$G_{i1}^{-1} = \frac{\partial \beta_{i-1}}{,}\frac{\partial \eta}{} \ G_{i2}^{-1} = \frac{\partial \beta_{i-1}}{,}\frac{\partial \theta}{} \ G_{i3}^{-1} = \frac{\partial \beta_{i-1} \partial \kappa}{} \quad (17)$$

and by equation (5) above it can be shown that:

$$\frac{\partial \beta_j}{\partial \eta} = \frac{1}{j+1} \quad (18)$$

$$\frac{\partial \beta_j}{\partial \theta} = \frac{1-(j+1)^{-\kappa}\Gamma(1+\kappa)}{(j+1)\kappa} \quad (19)$$

$$\frac{\partial \beta_j}{\partial \kappa} = \frac{\theta}{j+1}\{\frac{(j+1)^{-\kappa}[log(j+1)\Gamma(1+\kappa) - \Gamma'(1+\kappa)]}{\kappa} - \frac{1-(j+1)^{-\kappa}\Gamma(1+\kappa)}{\kappa^2}\} \quad (20)$$

for $i = 1, 2, 3$.

Estimating Distribution Quantiles

If X has a GEVD with parameters `location`=η, `scale`=θ, and `shape`=κ, where $\kappa \neq 0$, then the p'th quantile of the distribution is given by:

$$x(p) = \eta + \frac{\theta\{1-[-log(p)]^\kappa\}}{\kappa} \quad (21)$$

$(0 \leq p \leq 1)$. Given estimated values of the location, scale, and shape parameters, the p'th quantile of the distribution is estimated as:

$$\hat{x}(p) = \hat{\eta} + \frac{\hat{\theta}\{1 - [-log(p)]^{\hat{\kappa}}\}}{\hat{\kappa}} \quad (22)$$

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Greenwood, J.A., J.M. Landwehr, N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments: Definition and Relation to Parameters of Several Distributions Expressible in Inverse Form. *Water Resources Research* **15**(5), 1049–1054.

Hoeffding, W. (1948). A Class of Statistics with Asymptotically Normal Distribution. *Annals of Mathematical Statistics* **19**, 293–325.

Hosking, J.R.M. (1985). Algorithm AS 215: Maximum-Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Applied Statistics* **34**(3), 301–310.

Hosking, J.R.M. (1990). *L*-Moments: Analysis and Estimation of Distributions Using Linear Combinations of Order Statistics. *Journal of the Royal Statistical Society, Series B* **52**(1), 105–124.

Hosking, J.R.M., and J.R. Wallis (1995). A Comparison of Unbiased and Plotting-Position Estimators of *L* Moments. *Water Resources Research* **31**(8), 2019–2025.

Jenkinson, A.F. (1969). Statistics of Extremes. *Technical Note 98*, World Meteorological Office, Geneva.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, pp.4-8.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

Lehmann, E.L. (1975). *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, Oakland, CA, 457pp.

See Also

Generalized Extreme Value Distribution, egevd.

htest.htestEnvStats.object

S3 Classes "htest" and "htestEnvStats"

Description

These classes of objects are returned by functions that perform hypothesis tests (e.g., the R function `t.test` returns an object of class `"htest"`, while the **EnvStats** functions `quantileTest` and `kendallSeasonalTrendTest` return objects of class `"htestEnvStats"`). Objects of class `"htest"` and `"htestEnvStats"` are lists that contain information about the null and alternative hypotheses, the estimated distribution parameters, the test statistic, the p-value, and (optionally) confidence intervals for distribution parameters.

Details

Objects of S3 class `"htestEnvStats"` returned by **EnvStats** functions that perform hypothesis tests include:
`chenTTest`
`kendallTrendTest`
`kendallSeasonalTrendTest`
`quantileTest`
`serialCorrelationTest`
`signTest`
`twoSampleLinearRankTest`
`varTest`
`varGroupTest`
`zTestGevdShape`

These functions are listed in the help file Hypothesis Tests, along with other functions that perform hypothesis tests that return objects of different classes (e.g., the function `link{oneSamplePermutationTest}` returns an object of class `"permutationTest"`).

Note that functions that perform goodness-of-fit tests return objects of class `"gof"` or `"gofTwoSample"`.

Objects of class `"htestEnvStats"` generated by **EnvStats** functions may contain additional components called `estimation.method` (method used to estimate the population parameter(s)), `sample.size`, and `bad.obs` (number of missing (`NA`), undefined (`NaN`), or infinite (`Inf`, `-Inf`) values removed prior to performing the hypothesis test), and `interval` (a list with information about a confidence, prediction, or tolerance interval).

Value

Required Components

The following components must be included in a legitimate list of class `"htest"` or `"htestEnvStats"`.

`null.value`	numeric vector containing the value(s) of the population parameter(s) specified by the null hypothesis. This vector has a `names` attribute describing its elements.
`alternative`	character string indicating the alternative hypothesis (the value of the input argument `alternative`). Possible values are `"greater"`, `"less"`, or `"two-sided"`.

`method`	character string giving the name of the test used.
`estimate`	numeric vector containing the value(s) of the estimated population parameter(s) involved in the null hypothesis. This vector has a `names` attribute describing its element(s).
`data.name`	character string containing the actual name(s) of the input data.
`statistic`	numeric scalar containing the value of the test statistic, with a `names` attribute indicating the null distribution.
`parameters`	numeric vector containing the parameter(s) associated with the null distribution of the test statistic. This vector has a `names` attribute describing its element(s).
`p.value`	numeric scalar containing the p-value for the test under the null hypothesis.

Optional Components

The following component may optionally be included in an object of of class `"htest"` generated by R functions that test hypotheses:

`conf.int`	numeric vector of length 2 containing lower and upper confidence limits for the estimated population parameter. This vector has an attribute called `"conf.level"` that is a numeric scalar indicating the confidence level associated with the confidence interval.

The following components may be included in objects of class `"htestEnvStats"` generated by **EnvStats** functions:

`sample.size`	numeric scalar containing the number of non-missing observations in the sample used for the hypothesis test.
`estimation.method`	character string containing the method used to compute the estimated distribution parameter(s). The value of this component will depend on the available estimation methods (see `Distribution.df`).
`bad.obs`	the number of missing (`NA`), undefined (`NaN`) and/or infinite (`Inf`, `-Inf`) values that were removed from the data object prior to performing the hypothesis test.
`interval`	a list containing information about a confidence, prediction, or tolerance interval.

Methods

Generic functions that have methods for objects of class `"htest"` and `"htestEnvStats"` include: `print`.

Note

Since objects of class `"htest"` and `"htestEnvStats"` are lists, you may extract their components with the `$` and `[[` operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

print.htest, print.htestEnvStats, Hypothesis Tests.

Examples

```
# Create an object of class "htestEnvStats", then print it out.
#-----------------------------------------------------------

htestEnvStats.obj <- chenTTest(EPA.02d.Ex.9.mg.per.L.vec, mu = 30)

mode(htestEnvStats.obj)
#[1] "list"

class(htestEnvStats.obj)
#[1] "htestEnvStats"

names(htestEnvStats.obj)
# [1] "statistic"   "parameters"  "p.value"     "estimate"
# [5] "null.value"  "alternative" "method"      "sample.size"
# [9] "data.name"   "bad.obs"     "interval"

htestEnvStats.obj

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 mean = 30
#
#Alternative Hypothesis:          True mean is greater than 30
#
#Test Name:                       One-sample t-Test
#                                 Modified for
#                                 Positively-Skewed Distributions
#                                 (Chen, 1995)
#
#Estimated Parameter(s):          mean = 34.566667
#                                 sd   = 27.330598
#                                 skew =  2.365778
#
#Data:                            EPA.02d.Ex.9.mg.per.L.vec
#
#Sample Size:                     60
#
#Test Statistic:                  t = 1.574075
#
#Test Statistic Parameter:        df = 59
#
#P-values:                        z               = 0.05773508
#                                 t               = 0.06040889
#                                 Avg. of z and t = 0.05907199
#
#Confidence Interval for:         mean
```

```
#
#Confidence Interval Method:      Based on z
#
#Confidence Interval Type:        Lower
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 29.82
#                                 UCL =   Inf

#==========

# Extract the test statistic
#---------------------------

htestEnvStats.obj$statistic
#       t
#1.574075

#================================================================================

# Now create an object of class "htest" and note the difference in how it is
# printed out depending on whether or not you explicitly use the print() command.
#--------------------------------------------------------------------------------

htest.obj <- t.test(EPA.02d.Ex.9.mg.per.L.vec, mu = 30, alternative = "greater")

class(htest.obj)
#[1] "htest"

names(htest.obj)
# [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
# [6] "null.value"  "stderr"      "alternative" "method"      "data.name"

htest.obj

#        One Sample t-test
#
#data:  EPA.02d.Ex.9.mg.per.L.vec
#t = 1.2943, df = 59, p-value = 0.1003
#alternative hypothesis: true mean is greater than 30
#95 percent confidence interval:
# 28.67044      Inf
#sample estimates:
#mean of x
# 34.56667


print(htest.obj)

#Results of Hypothesis Test
#--------------------------
#
```

```
#Null Hypothesis:                 mean = 30
#
#Alternative Hypothesis:          True mean is greater than 30
#
#Test Name:                       One Sample t-test
#
#Estimated Parameter(s):          mean of x = 34.56667
#
#Data:                            EPA.02d.Ex.9.mg.per.L.vec
#
#Test Statistic:                  t = 1.294273
#
#Test Statistic Parameter:        df = 59
#
#P-value:                         0.1003072
#
#95% Confidence Interval:         LCL = 28.67044
#                                 UCL =      Inf

#==========

# Clean up
#---------
rm(htestEnvStats.obj, htest.obj)
```

`htestCensored.object` *S3 Class "htestCensored"*

Description

This class of objects is returned by **EnvStats** functions that perform hypothesis tests based on ***censored*** data. Objects of class `"htestCensored"` are lists that contain information about the null and alternative hypotheses, the censoring side, the censoring levels, the percentage of observations that are censored, the estimated distribution parameters (if applicable), the test statistic, the p-value, and (optionally, if applicable) confidence intervals for distribution parameters.

Details

Objects of S3 class `"htestCensored"` are returned by the functions listed in the section **Hypothesis Tests** in the help file EnvStats Functions for Censored Data. Currently, the only function listed is `twoSampleLinearRankTestCensored`.

Value

Required Components

The following components must be included in a legitimate list of class `"htestCensored"`.

`statistic` numeric scalar containing the value of the test statistic, with a `names` attribute indicating the null distribution.

`parameters` numeric vector containing the parameter(s) associated with the null distribution of the test statistic. This vector has a `names` attribute describing its element(s).

`p.value` numeric scalar containing the p-value for the test under the null hypothesis.

`null.value` numeric vector containing the value(s) of the population parameter(s) specified by the null hypothesis. This vector has a `names` attribute describing its elements.

`alternative` character string indicating the alternative hypothesis (the value of the input argument `alternative`). Possible values are `"greater"`, `"less"`, or `"two-sided"`.

`method` character string giving the name of the test used.

`sample.size` numeric scalar containing the number of non-missing observations in the sample used for the hypothesis test.

`data.name` character string containing the actual name(s) of the input data.

`bad.obs` the number of missing (`NA`), undefined (`NaN`) and/or infinite (`Inf`, `-Inf`) values that were removed from the data object prior to performing the hypothesis test.

`censoring.side` character string indicating whether the data are left- or right-censored.

`censoring.name` character string indicating the name of the data object used to identify which values are censored.

`censoring.levels` numeric scalar or vector indicating the censoring level(s).

`percent.censored` numeric scalar indicating the percent of non-missing observations that are censored.

Optional Components

The following component may optionally be included in an object of of class `"htestCensored"`:

`estimate` numeric vector containing the value(s) of the estimated population parameter(s) involved in the null hypothesis. This vector has a `names` attribute describing its element(s).

`estimation.method` character string containing the method used to compute the estimated distribution parameter(s). The value of this component will depend on the available estimation methods (see `Distribution.df`).

`interval` a list containing information about a confidence, prediction, or tolerance interval.

Methods

Generic functions that have methods for objects of class `"htestCensored"` include:
`print`.

Note

Since objects of class `"htestCensored"` are lists, you may extract their components with the `$` and `[[` operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

print.htestCensored, Censored Data.

Examples

```
# Create an object of class "htestCensored", then print it out.
#-------------------------------------------------------------

htestCensored.obj <-   with(EPA.09.Ex.16.5.PCE.df,
  twoSampleLinearRankTestCensored(
    x = PCE.ppb[Well.type == "Compliance"],
    x.censored = Censored[Well.type == "Compliance"],
    y = PCE.ppb[Well.type == "Background"],
    y.censored = Censored[Well.type == "Background"],
    test = "tarone-ware", alternative = "greater"))

mode(htestCensored.obj)
#[1] "list"

class(htestCensored.obj)
#[1] "htest"

names(htestCensored.obj)
# [1] "statistic"         "parameters"        "p.value"
# [4] "estimate"          "null.value"        "alternative"
# [7] "method"            "estimation.method" "sample.size"
#[10] "data.name"         "bad.obs"           "censoring.side"
#[13] "censoring.name"    "censoring.levels"  "percent.censored"

htestCensored.obj

#Results of Hypothesis Test
#Based on Censored Data
#--------------------------
#
#Null Hypothesis:                 Fy(t) = Fx(t)
#
#Alternative Hypothesis:          Fy(t) > Fx(t) for at least one t
#
#Test Name:                       Two-Sample Linear Rank Test:
#                                 Tarone-Ware Test
#                                 with Hypergeometric Variance
#
#Censoring Side:                  left
#
#Data:                            x = PCE.ppb[Well.type == "Compliance"]
#                                 y = PCE.ppb[Well.type == "Background"]
#
```

```
#Censoring Variable:              x = Censored[Well.type == "Compliance"]
#                                 y = Censored[Well.type == "Background"]
#
#Sample Sizes:                    nx = 8
#                                 ny = 6
#
#Percent Censored:                x = 12.5%
#                                 y = 50.0%
#
#Test Statistics:                 nu     =  8.458912
#                                 var.nu = 20.912407
#                                 z      =  1.849748
#
#P-value:                         0.03217495

#==========

# Extract the test statistics
#----------------------------

htestCensored.obj$statistic
#       nu    var.nu         z
# 8.458912 20.912407  1.849748

#==========

# Clean up
#---------
rm(htestCensored.obj)
```

`inversePredictCalibrate` *Predict Concentration Using Calibration*

Description

Predict concentration using a calibration line (or curve) and inverse regression.

Usage

```
inversePredictCalibrate(object, obs.y = NULL,
  n.points = ifelse(is.null(obs.y), 100, length(obs.y)),
  intervals = FALSE, coverage = 0.99, simultaneous = FALSE,
  individual = FALSE, trace = FALSE)
```

Arguments

`object`	an object of class `"calibrate"` that is the result of calling the function `calibrate`.

obs.y	optional numeric vector of observed values for the machine signal. The default value is obs.y=NULL, in which case obs.y is set equal to a vector of values (of length n.points) ranging from the minimum to the maximum of the fitted values from the calibrate object.
n.points	optional integer indicating the number of points at which to predict concentrations (i.e., perform inverse regression). The default value is n.points=100. This argument is ignored when obs.y is supplied.
intervals	optional logical scalar indicating whether to compute confidence intervals for the predicted concentrations. The default value is intervals=FALSE.
coverage	optional numeric scalar between 0 and 1 indicating the confidence level associated with the confidence intervals for the predicted concentrations. The default value is coverage=0.99.
simultaneous	optional logical scalar indicating whether to base the confidence intervals for the predicted values on simultaneous or non-simultaneous prediction limits. The default value is simultaneous=FALSE.
individual	optional logical scalar indicating whether to base the confidence intervals for the predicted values on prediction limits for the mean (individual=FALSE) or prediction limits for an individual observation (individual=TRUE). The default value is individual=FALSE.
trace	optional logical scalar indicating whether to print out (trace) the progress of the inverse prediction for each of the specified values of obs.y. The default value is trace=FALSE.

Details

A simple and frequently used calibration model is a straight line where the response variable S denotes the signal of the machine and the predictor variable C denotes the true concentration in the physical sample. The error term is assumed to follow a normal distribution with mean 0. Note that the average value of the signal for a blank ($C = 0$) is the intercept. Other possible calibration models include higher order polynomial models such as a quadratic or cubic model.

In a typical setup, a small number of samples (e.g., $n = 6$) with known concentrations are measured and the signal is recorded. A sample with no chemical in it, called a blank, is also measured. (You have to be careful to define exactly what you mean by a "blank." A blank could mean a container from the lab that has nothing in it but is prepared in a similar fashion to containers with actual samples in them. Or it could mean a field blank: the container was taken out to the field and subjected to the same process that all other containers were subjected to, except a physical sample of soil or water was not placed in the container.) Usually, replicate measures at the same known concentrations are taken. (The term "replicate" must be well defined to distinguish between for example the same physical samples that are measured more than once vs. two different physical samples of the same known concentration.)

The function calibrate initially fits a linear calibration line or curve. Once the calibration line is fit, samples with unknown concentrations are measured and their signals are recorded. In order to produce estimated concentrations, you have to use inverse regression to map the signals to the estimated concentrations. We can quantify the uncertainty in the estimated concentration by combining inverse regression with prediction limits for the signal S.

Value

A numeric matrix containing the results of the inverse calibration. The first two columns are labeled `obs.y` and `pred.x` containing the values of the argument `obs.y` and the predicted values of `x` (the concentration), respectively. If `intervals=TRUE`, then the matrix also contains the columns `lpl.x` and `upl.x` corresponding to the lower and upper prediction limits for `x`. Also, if `intervals=TRUE`, then the matrix has the attributes `coverage` (the value of the argument `coverage`) and `simultaneous` (the value of the argument `simultaneous`).

Note

Almost always the process of determining the concentration of a chemical in a soil, water, or air sample involves using some kind of machine that produces a signal, and this signal is related to the concentration of the chemical in the physical sample. The process of relating the machine signal to the concentration of the chemical is called **calibration** (see `calibrate`). Once calibration has been performed, estimated concentrations in physical samples with unknown concentrations are computed using inverse regression. The uncertainty in the process used to estimate the concentration may be quantified with decision, detection, and quantitation limits.

In practice, only the point estimate of concentration is reported (along with a possible qualifier), without confidence bounds for the true concentration C. This is most unfortunate because it gives the impression that there is no error associated with the reported concentration. Indeed, both the International Organization for Standardization (ISO) and the International Union of Pure and Applied Chemistry (IUPAC) recommend always reporting both the estimated concentration and the uncertainty associated with this estimate (Currie, 1997).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Currie, L.A. (1997). Detection: International Update, and Some Emerging Di-Lemmas Involving Calibration, the Blank, and Multiple Detection Decisions. *Chemometrics and Intelligent Laboratory Systems* **37**, 151–181.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, Chapter 3 and p.335.

Hubaux, A., and G. Vos. (1970). Decision and Detection Limits for Linear Calibration Curves. *Annals of Chemistry* **42**, 849–855.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.562–575.

See Also

`pointwise`, `calibrate`, `detectionLimitCalibrate`, `lm`

Examples

```
# The data frame EPA.97.cadmium.111.df contains calibration data
# for cadmium at mass 111 (ng/L) that appeared in
```

```
# Gibbons et al. (1997b) and were provided to them by the U.S. EPA.
# Here we
# 1. Display a plot of these data along with the fitted calibration
#    line and 99% non-simultaneous prediction limits.
# 2. Then based on an observed signal of 60 from a sample with
#    unknown concentration, we use the calibration line to estimate
#    the true concentration and use the prediction limits to compute
#    confidence bounds for the true concentration.
# An observed signal of 60 results in an estimated value of cadmium
# of 59.97 ng/L and a confidence interval of [53.83, 66.15].
# See Millard and Neerchal (2001, pp.566-569) for more details on
# this example.

Cadmium <- EPA.97.cadmium.111.df$Cadmium

Spike <- EPA.97.cadmium.111.df$Spike

calibrate.list <- calibrate(Cadmium ~ Spike,
  data = EPA.97.cadmium.111.df)

newdata <- data.frame(Spike = seq(min(Spike), max(Spike),
  length.out = 100))

pred.list <- predict(calibrate.list, newdata = newdata, se.fit = TRUE)

pointwise.list <- pointwise(pred.list, coverage = 0.99,
  individual = TRUE)

plot(Spike, Cadmium, ylim = c(min(pointwise.list$lower),
  max(pointwise.list$upper)), xlab = "True Concentration (ng/L)",
  ylab = "Observed Concentration (ng/L)")

abline(calibrate.list, lwd=2)

lines(newdata$Spike, pointwise.list$lower, lty=8, lwd=2)

lines(newdata$Spike, pointwise.list$upper, lty=8, lwd=2)

title(paste("Calibration Line and 99% Prediction Limits",
  "for US EPA Cadmium 111 Data", sep = "\n"))


# Now estimate the true concentration based on
# an observed signal of 60 ng/L.

inversePredictCalibrate(calibrate.list, obs.y = 60,
  intervals = TRUE, coverage = 0.99, individual = TRUE)

#     obs.y   pred.x   lpl.x    upl.x
#[1,]    60 59.97301 53.8301 66.15422
#attr(, "coverage"):
#[1] 0.99
#attr(, "simultaneous"):
```

```
#[1] FALSE

rm(Cadmium, Spike, calibrate.list, newdata, pred.list, pointwise.list)
```

---

iqr *Interquartile Range*

Description

Compute the interquartile range for a set of data.

Usage

```
iqr(x, na.rm = FALSE)
```

Arguments

x	numeric vector of observations.
na.rm	logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.

Details

Let $\underline{x}$ denote a random sample of n observations from some distribution associated with a random variable X. The sample interquartile range is defined as:

$$IQR = \hat{X}_{0.75} - \hat{X}_{0.25} \qquad (1)$$

where X_p denotes the p'th quantile of the distribution and $\hat{X}_p$ denotes the estimate of this quantile (i.e., the sample p'th quantile).

See the R help file for quantile for information on how sample quantiles are computed.

Value

A numeric scalar – the interquartile range.

Note

The interquartile range is a robust estimate of the spread of the distribution. It is the distance between the two ends of a boxplot (see the R help file for boxplot). For a normal distribution with standard deviation σ it can be shown that:

$$IQR = 1.34898\sigma \qquad (2)$$

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY.

Hirsch, R.M., D.R. Helsel, T.A. Cohn, and E.J. Gilroy. (1993). Statistical Analysis of Hydrologic Data. In: Maidment, D.R., ed. *Handbook of Hydrology*. McGraw-Hill, New York, Chapter 17, pp.5–7.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

Summary Statistics, `summaryFull`, `var`, `sd`.

Examples

```
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=2, and compute the standard deviation and
# interquartile range.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rnorm(20, mean=10, sd=2)

sd(dat)
#[1] 1.180226

iqr(dat)
#[1] 1.489932

#----------

# Repeat the last example, but add a couple of large "outliers" to the
# data.  Note that the estimated standard deviation is greatly affected
# by the outliers, while the interquartile range is not.

summaryStats(dat, quartiles = TRUE)
#     N   Mean     SD Median    Min     Max 1st Qu. 3rd Qu.
#dat 20 9.8612 1.1802 9.6978 7.6042 11.8756  9.1618 10.6517

new.dat <- c(dat, 20, 50)

sd(dat)
#[1] 1.180226

sd(new.dat)
#[1] 8.79796

iqr(dat)
```

```
#[1] 1.489932

iqr(new.dat)
#[1] 1.851472

#----------
# Clean up
rm(dat, new.dat)
```

`kendallSeasonalTrendTest` *Nonparametric Test for Monotonic Trend Within Each Season Based on Kendall's Tau Statistic*

Description

Perform a nonparametric test for a monotonic trend within each season based on Kendall's tau statistic, and optionally compute a confidence interval for the slope across all seasons.

Usage

```
kendallSeasonalTrendTest(y, ...)

## S3 method for class 'formula'
kendallSeasonalTrendTest(y, data = NULL, subset,
  na.action = na.pass, ...)

## Default S3 method:
kendallSeasonalTrendTest(y, season, year,
  alternative = "two.sided", correct = TRUE, ci.slope = TRUE, conf.level = 0.95,
  independent.obs = TRUE, data.name = NULL, season.name = NULL, year.name = NULL,
  parent.of.data = NULL, subset.expression = NULL, ...)

## S3 method for class 'data.frame'
kendallSeasonalTrendTest(y, ...)

## S3 method for class 'matrix'
kendallSeasonalTrendTest(y, ...)
```

Arguments

`y`	an object containing data for the trend test. In the default method, the argument `y` must be numeric vector of observations. When `y` is a data frame, all columns must be numeric. When `y` is a matrix, it must be a numeric matrix. In the formula method, `y` must be a formula of the form `y ~ season + year`, where `y`, `season`, and `year` specify what variables to use for the these arguments in the call to `kendallSeasonalTrendTest.default`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.

data	specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which kendallTrendTest is called.
subset	specifies an optional vector specifying a subset of observations to be used.
na.action	specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.
season	numeric or character vector or a factor indicating the seasons in which the observations in y were taken. The length of season must equal the length of y.
year	numeric vector indicating the years in which the observations in y were taken. The length of year must equal the length of y.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (tau not equal to 0; the default), "less" (tau less than 0), and "greater" (tau greater than 0).
correct	logical scalar indicating whether to use the correction for continuity in computing the z-statistic that is based on the test statistic S'. The default value is TRUE.
ci.slope	logical scalar indicating whether to compute a confidence interval for the slope. The default value is TRUE.
conf.level	numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the slope. The default value is 0.95.
independent.obs	logical scalar indicating whether to assume the observations in y are seially independent. The default value is TRUE.
data.name	character string indicating the name of the data used for the trend test. The default value is deparse(substitute(y)).
season.name	character string indicating the name of the data used for the season. The default value is deparse(substitute(season)).
year.name	character string indicating the name of the data used for the year. The default value is deparse(substitute(year)).
parent.of.data	character string indicating the source of the data used for the trend test.
subset.expression	character string indicating the expression used to subset the data.
...	additional arguments affecting the test for trend.

Details

Hirsch et al. (1982) introduced a modification of Kendall's test for trend (see kendallTrendTest) that allows for seasonality in observations collected over time. They call this test the *seasonal Kendall test*. Their test is appropriate for testing for trend in each season when the trend is always in the same direction across all seasons. van Belle and Hughes (1984) introduced a heterogeneity test for trend which is appropriate for testing for trend in any direction in any season. Hirsch and Slack (1984) proposed an extension to the seasonal Kendall test that allows for serial dependence in the observations. The function kendallSeasonalTrendTest includes all of these tests, as well as an extension of the van Belle-Hughes heterogeneity test to the case of serial dependence.

Testing for Trend Assuming Serial Independence

The Model

Assume observations are taken over two or more years, and assume a single year can be divided into two or more seasons. Let p denote the number of seasons. Let X and Y denote two continuous random variables with some joint (bivariate) distribution (which may differ from season to season). Let N_j denote the number of bivariate observations taken in the j'th season (over two or more years) ($j = 1, 2, \ldots, p$), so that

$$(X_{1j}, Y_{1j}), (X_{2j}, Y_{2j}), \ldots, (X_{N_j j}, Y_{N_j j})$$

denote the N_j bivariate observations from this distribution for season j, assume these bivariate observations are mutually independent, and let

$$\tau_j = \{2Pr[(X_{2j} - X_{1j})(Y_{2j} - Y_{1j}) > 0]\} - 1 \qquad (1)$$

denote the value of Kendall's tau for that season (see `kendallTrendTest`). Also, assume all of the Y observations are independent.

The function `kendallSeasonalTrendTest` assumes that the X values always denote the year in which the observation was taken. Note that within any season, the X values need not be unique. That is, there may be more than one observation within the same year within the same season. In this case, the argument `y` must be a numeric vector, and you must supply the additional arguments `season` and `year`.

If there is only one observation per season per year (missing values allowed), it is usually easiest to supply the argument `y` as an $n \times p$ matrix or data frame, where n denotes the number of years. In this case

$$N_1 = N_2 = \cdots = N_p = n \qquad (2)$$

and

$$X_{ij} = i \qquad (3)$$

for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$, so if Y denotes the $n \times p$ matrix of observed Y's and X denotes the $n \times p$ matrix of the X's, then

$$\underline{Y} = \begin{matrix} Y_{11} & Y_{12} & \cdots & Y_{1p} \\ Y_{21} & Y_{22} & \cdots & Y_{2p} \\ . & & & \\ . & & & \\ . & & & \\ Y_{n1} & Y_{n2} & \cdots & Y_{np} \end{matrix} \qquad (4)$$

$$\underline{X} = \begin{matrix} 1 & 1 & \cdots & 1 \\ 2 & 2 & \cdots & 2 \\ . & & & \\ . & & & \\ . & & & \\ n & n & \cdots & n \end{matrix} \qquad (5)$$

The null hypothesis is that within each season the X and Y random variables are independent; that is, within each season there is no trend in the Y observations over time. This null hypothesis can be expressed as:

$$H_0 : \tau_1 = \tau_2 = \cdots = \tau_p = 0 \qquad (6)$$

The Seasonal Kendall Test for Trend

Hirsch et al.'s (1982) extension of Kendall's tau statistic to test the null hypothesis (6) is based on simply summing together the Kendall S-statistics for each season and computing the following statistic:

$$z = \frac{S'}{\sqrt{Var(S')}} \qquad (7)$$

or, using the correction for continuity,

$$z = \frac{S' - sign(S')}{\sqrt{Var(S')}} \qquad (8)$$

where

$$S' = \sum_{j=1}^{p} S_j \qquad (9)$$

$$S_j = \sum_{i=1}^{N_j - 1} \sum_{k=i+1}^{N_j} sign[(X_{kj} - X_{ij})(Y_{kj} - Y_{ij})] \qquad (10)$$

and $sign()$ denotes the `sign` function:

$$sign(x) = \begin{array}{cc} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{array} \qquad (11)$$

Note that the quantity in Equation (10) is simply the Kendall S-statistic for season j ($j = 1, 2, \ldots, p$) (see Equation (3) in the help file for `kendallTrendTest`).

For each season, if the predictor variables (the X's) are strictly increasing (e.g., Equation (3) above), then Equation (10) simplifies to

$$S_j = \sum_{i=1}^{N_j - 1} \sum_{k=i+1}^{N_j} sign[(Y_{kj} - Y_{ij})] \qquad (12)$$

Under the null hypothesis (6), the quantity z defined in Equation (7) or (8) is approximately distributed as a standard normal random variable.

Note that there may be missing values in the observations, so let n_j denote the number of (X, Y) pairs without missing values for season j.

The statistic S' in Equation (9) has mean and variance given by:

$$E(S') = \sum_{j=1}^{p} E(S_j) \qquad (13)$$

$$Var(S') = \sum_{j=1}^{p} Var(S_j) + \sum_{g=1}^{p-1} \sum_{h=g+1}^{p} 2Cov(S_g, S_h) \qquad (14)$$

Since all the observations are assumed to be mutually independent,

$$\sigma_{gh} = Cov(S_g, S_h) = 0, \quad g \neq h, \quad g, h = 1, 2, \ldots, p \qquad (15)$$

Furthermore, under the null hypothesis (6),

$$E(S_j) = 0, \quad j = 1, 2, \ldots, p \qquad (16)$$

and, in the case of no tied observations,

$$Var(S_j) = \frac{n_j(n_j - 1)(2n_j + 5)}{18} \qquad (17)$$

for $j = 1, 2, \ldots, p$ (see equation (7) in the help file for `kendallTrendTest`).

In the case of tied observations,

$$\begin{aligned} Var(S_j) = \quad & \tfrac{n_j(n_j-1)(2n_j+5)}{18} - \\ & \tfrac{\sum_{i=1}^{g} t_i(t_i-1)(2t_i+5)}{18} - \\ & \tfrac{\sum_{k=1}^{h} u_k(u_k-1)(2u_k+5)}{18} + \\ & \tfrac{[\sum_{i=1}^{g} t_i(t_i-1)(t_i-2)][\sum_{k=1}^{h} u_k(u_k-1)(u_k-2)]}{9n_k(n_k-1)(n_k-2)} + \\ & \tfrac{[\sum_{i=1}^{g} t_i(t_i-1)][\sum_{k=1}^{h} u_k(u_k-1)]}{2n_k(n_k-1)} \end{aligned} \qquad (18)$$

where g is the number of tied groups in the X observations for season j, t_i is the size of the i'th tied group in the X observations for season j, h is the number of tied groups in the Y observations for season j, and u_k is the size of the k'th tied group in the Y observations for season j (see Equation (9) in the help file for `kendallTrendTest`).

Estimating τ, Slope, and Intercept

The function `kendall.SeasonalTrendTest` returns estimated values of Kendall's τ, the slope, and the intercept for each season, as well as a single estimate for each of these three quantities combined over all seasons. The overall estimate of τ is the weighted average of the p seasonal τ's:

$$\hat{\tau} = \frac{\sum_{j=1}^{p} n_j \hat{\tau}_j}{\sum_{j=1}^{p} n_j} \qquad (19)$$

where

$$\hat{\tau}_j = \frac{2S_j}{n_j(n_j - 1)} \qquad (20)$$

(see Equation (2) in the help file for `kendallTrendTest`).

We can compute the estimated slope for season j as:

$$\hat{\beta}_{1_j} = Median(\frac{Y_{kj} - Y_{ij}}{X_{kj} - X_{ij}}); \quad i < k; \quad X_{kj} \neq X_{ik} \qquad (21)$$

for $j = 1, 2, \ldots, p$. The overall estimate of slope, however, is **not** the median of these p estimates of slope; instead, following Hirsch et al. (1982, p.117), the overall estimate of slope is the median of all two-point slopes computed within each season:

$$\hat{\beta}_1 = Median(\frac{Y_{kj} - Y_{ij}}{X_{kj} - X_{ij}}); \;\; i < k; \;\; X_{kj} \neq X_{ik}; \;\; j = 1, 2, \ldots, p \qquad (22)$$

(see Equation (15) in the help file for `kendallTrendTest`).

The overall estimate of intercept is the median of the p seasonal estimates of intercept:

$$\hat{\beta}_0 = Median(\hat{\beta}_{0_1}, \hat{\beta}_{0_2}, \ldots, \hat{\beta}_{0_p}) \qquad (23)$$

where

$$\hat{\beta}_{0_j} = Y_{0.5_j} - \hat{\beta}_{1_j} X_{0.5_j} \qquad (24)$$

and $X_{0.5_j}$ and $Y_{0.5_j}$ denote the sample medians of the X's and Y's, respectively, for season j (see Equation (16) in the help file for `kendallTrendTest`).

Confidence Interval for the Slope

Gilbert (1987, p.227-228) extends his method of computing a confidence interval for the slope to the case of seasonal observations. Let N' denote the number of defined two-point estimated slopes that are used in Equation (22) above and let

$$\hat{\beta}_{1_{(1)}}, \hat{\beta}_{1_{(2)}}, \ldots, \hat{\beta}_{1_{(N')}}$$

denote the N' ordered slopes. For Gilbert's (1987) method, a $100(1-\alpha)\%$ two-sided confidence interval for the true over-all slope across all seasons is given by:

$$[\hat{\beta}_{1_{(M1)}}, \hat{\beta}_{1_{(M2+1)}}] \qquad (25)$$

where

$$M1 = \frac{N' - C_\alpha}{2} \qquad (26)$$

$$M2 = \frac{N' + C_\alpha}{2} \qquad (27)$$

$$C_\alpha = z_{1-\alpha/2}\sqrt{Var(S')} \qquad (28)$$

$Var(S')$ is defined in Equation (14), and z_p denotes the p'th quantile of the standard normal distribution. One-sided confidence intervals may computed in a similar fashion.

Usually the quantities $M1$ and $M2$ will not be integers. Gilbert (1987, p.219) suggests interpolating between adjacent values in this case, which is what the function `kendallSeasonalTrendTest` does.

The Van Belle-Hughes Heterogeneity Test for Trend

The seasonal Kendall test described above is appropriate for testing the null hypothesis (6) against the alternative hypothesis of a trend in at least one season. All of the trends in each season should be in the same direction.

The seasonal Kendall test is not appropriate for testing for trend when there are trends in a positive direction in one or more seasons and also negative trends in one or more seasons. For example, for the following set of observations, the seasonal Kendall statistic S' is 0 with an associated two-sided

p-value of 1, even though there is clearly a positive trend in season 1 and a negative trend in season 2.

Year	Season 1	Season 2
1	5	8
2	6	7
3	7	6
4	8	5

Van Belle and Hughes (1984) suggest using the following statistic to test for heterogeneity in trend prior to applying the seasonal Kendall test:

$$\chi^2_{het} = \sum_{j=1}^{p} Z_j^2 - p\bar{Z}^2 \qquad (29)$$

where

$$Z_j = \frac{S_j}{Var(S_j)} \qquad (30)$$

$$\bar{Z} = \frac{1}{p}\sum_{j=1}^{p} Z_j \qquad (31)$$

Under the null hypothesis (6), the statistic defined in Equation (29) is approximately distributed as a chi-square random variable with $p - 1$ degrees of freedom. Note that the continuity correction is not used to compute the Z_j's defined in Equation (30) since using it results in an unacceptably conservative test (van Belle and Hughes, 1984, p.132). Van Belle and Hughes (1984) actually call the statistic in (29) a homogeneous chi-square statistic. Here it is called a heterogeneous chi-square statistic after the alternative hypothesis it is meant to test.

Van Belle and Hughes (1984) imply that the heterogeneity statistic defined in Equation (29) may be used to test the null hypothesis:

$$H_0 : \tau_1 = \tau_2 = \cdots = \tau_p = \tau \qquad (32)$$

where τ is some arbitrary number between -1 and 1. For this case, however, the distribution of the test statistic in Equation (29) is unknown since it depends on the unknown value of τ (Equations (16)-(18) above assume $\tau = 0$ and are not correct if $\tau \neq 0$). The heterogeneity chi-square statistic of Equation (29) may be assumed to be approximately distributed as chi-square with $p - 1$ degrees of freedom under the null hypothesis (32), but further study is needed to determine how well this approximation works.

Testing for Trend Assuming Serial Dependence

The Model

Assume the same model as for the case of serial independence, except now the observed Y's are not assumed to be independent of one another, but are allowed to be serially correlated over time. Furthermore, assume one observation per season per year (Equations (2)-(5) above).

The Seasonal Kendall Test for Trend Modified for Serial Dependence

Hirsch and Slack (1984) introduced a modification of the seasonal Kendall test that is robust against

serial dependence (in terms of Type I error) except when the observations have a very strong long-term persistence (very large autocorrelation) or when the sample sizes are small (e.g., 5 years of monthly data). This modification is based on a multivariate test introduced by Dietz and Killeen (1981).

In the case of serial dependence, Equation (15) is no longer true, so an estimate of the correct value of σ_{gh} must be used to compute Var(S') in Equation (14). Let R denote the $n \times p$ matrix of ranks for the Y observations (Equation (4) above), where the Y's are ranked within season:

$$\underline{R} = \begin{matrix} R_{11} & R_{12} & \cdots & R_{1p} \\ R_{21} & R_{22} & \cdots & R_{2p} \\ . & & & \\ . & & & \\ . & & & \\ R_{n1} & R_{n2} & \cdots & R_{np} \end{matrix} \qquad (33)$$

where

$$R_{ij} = \frac{1}{2}[n_j + 1 \sum_{k=1}^{n_j} sign(Y_{ij} - Y_{kj})] \qquad (34)$$

the $sign$ function is defined in Equation (11) above, and as before n_j denotes the number of (X, Y) pairs without missing values for season j. Note that by this definition, missing values are assigned the mid-rank of the non-missing values.

Hirsch and Slack (1984) suggest using the following formula, given by Dietz and Killeen (1981), in the case where there are no missing values:

$$\hat{\sigma}_{gh} = \frac{1}{3}[K_{gh} + 4 \sum_{i=1}^{n} R_{ig} R_{ih} - n(n+1)^2] \qquad (35)$$

where

$$K_{gh} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sign[(Y_{jg} - Y_{ig})(Y_{jh} - Y_{ih})] \qquad (36)$$

Note that the quantity defined in Equation (36) is Kendall's tau for season g vs. season h.

For the case of missing values, Hirsch and Slack (1984) derive the following modification of Equation (35):

$$\hat{\sigma}_{gh} = \frac{1}{3}[K_{gh} + 4 \sum_{i=1}^{n} R_{ig} R_{ih} - n(n_g + 1)(n_h + 1)] \qquad (37)$$

Technically, the estimates in Equations (35) and (37) are not correct estimators of covariance, and Equations (17) and (18) are not correct estimators of variance, because the model Dietz and Killeen (1981) use assumes that observations within the rows of Y (Equation (4) above) may be correlated, but observations between rows are independent. Serial dependence induces correlation between all of the Y's. In most cases, however, the serial dependence shows an exponential decay in correlation across time and so these estimates work fairly well (see more discussion in the BACKGROUND section below).

Estimates and Confidence Intervals

The seasonal and over-all estimates of τ, slope, and intercept are computed using the same methods

as in the case of serial independence. Also, the method for computing the confidence interval for the slope is the same as in the case of serial independence. Note that the serial dependence is accounted for in the term $Var(S')$ in Equation (28).

The Van Belle-Hughes Heterogeneity Test for Trend Modified for Serial Dependence
Like its counterpart in the case of serial independence, the seasonal Kendall test modified for serial dependence described above is appropriate for testing the null hypothesis (6) against the alternative hypothesis of a trend in at least one season. All of the trends in each season should be in the same direction.

The modified seasonal Kendall test is not appropriate for testing for trend when there are trends in a positive direction in one or more seasons and also negative trends in one or more seasons. This section describes a modification of the van Belle-Hughes heterogeneity test for trend in the presence of serial dependence.

Let $\underline{S}$ denote the $p \times 1$ vector of Kendall S-statistics for each season:

$$\underline{S} = \begin{matrix} S_1 \\ S_2 \\ . \\ . \\ . \\ S_p \end{matrix} \tag{38}$$

The distribution of $\underline{S}$ is approximately multivariate normal with

$$E(\underline{S}) = \underline{\mu} = \begin{matrix} \mu_1 \\ \mu_2 \\ . \\ . \\ . \\ \mu_p \end{matrix} \tag{39}$$

$$Var(\underline{S}) = \Sigma = \begin{matrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2p} \\ . & & & \\ . & & & \\ . & & & \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{matrix} \tag{40}$$

where

$$\mu_j = \frac{n_j(n_j-1)}{2}\tau_j, \quad j = 1, 2, \ldots, p \tag{41}$$

Define the $p \times p$ matrix $\underline{m}$ as

$$\underline{m} = \begin{matrix} \frac{2}{n_1(n_1-1)} & 0 & \cdots & 0 \\ 0 & \frac{2}{n_2(n_2-1)} & \cdots & 0 \\ . & & & \end{matrix} \tag{42}$$

$$\begin{matrix} . \\ . \\ 0 & 0 & \cdots & \frac{2}{n_p(n_p-1)} \end{matrix}$$

Then the vector of the seasonal estimates of τ can be written as:

$$\underline{\hat{\tau}} = \begin{matrix} \hat{\tau}_1 \\ \hat{\tau}_2 \\ . \\ . \\ . \\ \hat{\tau}_p \end{matrix} = \begin{matrix} 2S_1/[n_1(n_1-1)] \\ 2S_2/[n_2(n_2-1)] \\ . \\ . \\ . \\ 2S_p/[n_p(n_p-1)] \end{matrix} = \underline{m}\ \underline{S} \qquad (43)$$

so the distribution of the vector in Equation (43) is approximately multivariate normal with

$$E(\underline{\hat{\tau}}) = \quad E(\underline{mS}) = \quad \underline{m}\underline{\mu} = \quad \underline{\tau} = \begin{matrix} \tau_1 \\ \tau_2 \\ . \\ . \\ . \\ \tau_p \end{matrix} \qquad (44)$$

$$Var(\underline{\hat{\tau}}) = Var(\underline{m}\ \underline{S}) = \underline{m}\Sigma\underline{m}^T = \Sigma_{\hat{\tau}} \qquad (45)$$

where ^{T} denotes the transpose operator. Let $\underline{C}$ denote the $(p-1) \times p$ contrast matrix

$$\underline{C} = [\ \underline{1}\ |\ -I_p] \qquad (46)$$

where I_p denotes the $p \times p$ identity matrix. That is,

$$\underline{C} = \begin{matrix} 1 & -1 & 0 & \cdots & 0 \\ 1 & 0 & -1 & \cdots & 0 \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ 1 & 0 & 0 & \cdots & -1 \end{matrix}$$

Then the null hypothesis (32) is equivalent to the null hypothesis:

$$H_0 : \underline{C\tau} = 0 \qquad (47)$$

Based on theory for samples from a multivariate normal distribution (Johnson and Wichern, 2007), under the null hypothesis (47) the quantity

$$\chi^2_{het} = (\underline{C}\ \underline{\hat{\tau}})^T(\underline{C}\hat{\Sigma}_{\hat{\tau}}\underline{C}^T)^{-1}(\underline{C}\ \underline{\hat{\tau}}) \qquad (48)$$

has approximately a chi-square distribution with $p-1$ degrees of freedom for "large" values of seasonal sample sizes, where

$$\hat{\Sigma}_{\hat{\tau}} = \underline{m}\hat{\Sigma}\underline{m}^T \qquad (49)$$

The estimate of Σ in Equation (49) can be computed using the same formulas that are used for the modified seasonal Kendall test (i.e., Equation (35) or (37) for the off-diagonal elements and Equation (17) or (18) for the diagonal elements). As previously noted, the formulas for the variances are actually only valid if $t = 0$ and there is no correlation between the rows of Y. The same is true of the formulas for the covariances. More work is needed to determine the goodness of the chi-square approximation for the test statistic in (48). The pseudo-heterogeneity test statistic of Equation (48), however, should provide some guidance as to whether the null hypothesis (32) (or equivalently (47)) appears to be true.

Value

A list of class `"htestEnvStats"` containing the results of the hypothesis test. See the help file for `htestEnvStats.object` for details. In addition, the following components are part of the list returned by `kendallSeasonalTrendTest`:

`seasonal.S` numeric vector. The value of the Kendall S-statistic for each season.

`var.seasonal.S` numeric vector. The variance of the Kendall S-statistic for each season. This component only appears when `independent.obs=TRUE`.

`var.cov.seasonal.S` numeric matrix. The estimated variance-covariance matrix of the Kendall S-statistics for each season. This component only appears when `independent.obs=FALSE`.

`seasonal.estimates` numeric matrix. The estimated Kendall's tau, slope, and intercept for each season.

Note

Kendall's test for independence or trend is a nonparametric test. No assumptions are made about the distribution of the X and Y variables. Hirsch et al. (1982) introduced the seasonal Kendall test to test for trend within each season. They note that Kendall's test for trend is easy to compute, even in the presence of missing values, and can also be used with censored values.

van Belle and Hughes (1984) note that the seasonal Kendall test introduced by Hirsch et al. (1982) is similar to a multivariate extension of the sign test proposed by Jonckheere (1954). Jonckheeere's test statistic is based on the unweighted sum of the seasonal tau statistics, while Hirsch et al.'s test is based on the weighted sum (weighted by number of observations within a season) of the seasonal tau statistics.

van Belle and Hughes (1984) also note that Kendall's test for trend is slightly less powerful than the test based on Spearman's rho, but it converges to normality faster. Also, Bradley (1968, p.288) shows that for the case of a linear model with normal (Gaussian) errors, the asymptotic relative efficiency of Kendall's test for trend versus the parametric test for a zero slope is 0.98.

Based on the work of Dietz and Killeen (1981), Hirsch and Slack (1984) describe a modified version of the seasonal Kendall test that allows for serial dependence in the observations. They performed a Monte Carlo study to determine the empirical significance level and power of this modified test vs. the test that assumes independent observations and found a trade-off between power and the correct significance level. For $p = 12$ seasons, they found the modified test gave correct significance levels for $n \geq 10$ as long as the lag-one autocorrelation was 0.6 or less, while the original test that assumes

independent observations yielded highly inflated significance levels. On the other hand, if in fact the observations are serially independent, the original test is more powerful than the modified test.

Hirsch and Slack (1984) also looked at the performance of the test for trend introduced by Dietz and Killeen (1981), which is a weighted sums of squares of the seasonal Kendall S-statistics, where the matrix of weights is the inverse of the covariance matrix. The Dietz-Killeen test statistic, unlike the one proposed by Hirsh and Slack (1984), tests for trend in either direction in any season, and is asymptotically distributed as a chi-square random variable with p (number of seasons) degrees of freedom. Hirsch and Slack (1984), however, found that the test based on this statistic is quite conservative (i.e., the significance level is much smaller than the assumed significance level) and has poor power even for moderate sample sizes. The chi-square approximation becomes reasonably close only when $n > 40$ if $p = 12$, $n > 30$ if $p = 4$, and $n > 20$ if $p = 2$.

Lettenmaier (1988) notes the poor power of the test proposed by Dietz and Killeen (1981) and states the poor power apparently results from an upward bias in the estimated variance of the statistic, which can be traced to the inversion of the estimated covariance matrix. He suggests an alternative test statistic (to test trend in either direction in any season) that is the sum of the squares of the scaled seasonal Kendall S-statistics (scaled by their standard deviations). Note that this test statistic ignores information about the covariance between the seasonal Kendall S-statistics, although its distribution depends on these covariances. In the case of no serial dependence, Lettenmaier's test statistic is exactly the same as the Dietz-Killeen test statistic. In the case of serial dependence, Lettenmaier (1988) notes his test statistic is a quadratic form of a multivariate normal random variable and therefore all the moments of this random variable are easily computed. Lettenmaier (1988) approximates the distribution of his test statistic as a scaled non-central chi-square distribution (with fractional degrees of freedom). Based on extensive Monte Carlo studies, Lettenmaier (1988) shows that for the case when the trend is the same in all seasons, the seasonal Kendall's test of Hirsch and Slack (1984) is superior to his test and far superior to the Dietz-Killeen test. The power of Lettenmaier's test approached that of the seasonal Kendall test for large trend magnitudes.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bradley, J.V. (1968). *Distribution-Free Statistical Tests*. Prentice-Hall, Englewood Cliffs, NJ.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York, pp.256-272.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, Chapter 16.

Helsel, D.R. and R.M. Hirsch. (1988). Discussion of Applicability of the t-test for Detecting Trends in Water Quality Variables. *Water Resources Bulletin* **24**(1), 201-204.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, NY.

Helsel, D.R., and R. M. Hirsch. (2002). *Statistical Methods in Water Resources*. Techniques of Water Resources Investigations, Book 4, chapter A3. U.S. Geological Survey. Available on-line at https://pubs.usgs.gov/tm/04/a03/tm4a3.pdf.

Hirsch, R.M., J.R. Slack, and R.A. Smith. (1982). Techniques of Trend Analysis for Monthly Water Quality Data. *Water Resources Research* **18**(1), 107-121.

Hirsch, R.M. and J.R. Slack. (1984). A Nonparametric Trend Test for Seasonal Data with Serial Dependence. *Water Resources Research* **20**(6), 727-732.

Hirsch, R.M., R.B. Alexander, and R.A. Smith. (1991). Selection of Methods for the Detection and Estimation of Trends in Water Quality. *Water Resources Research* **27**(5), 803-813.

Hollander, M., and D.A. Wolfe. (1999). *Nonparametric Statistical Methods, Second Edition*. John Wiley and Sons, New York.

Johnson, R.A., and D.W. Wichern. (2007). *Applied Multivariate Statistical Analysis*, Sixth Edition. Pearson Prentice Hall, Upper Saddle River, NJ.

Kendall, M.G. (1938). A New Measure of Rank Correlation. *Biometrika* **30**, 81-93.

Kendall, M.G. (1975). *Rank Correlation Methods*. Charles Griffin, London.

Mann, H.B. (1945). Nonparametric Tests Against Trend. *Econometrica* **13**, 245-259.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Sen, P.K. (1968). Estimates of the Regression Coefficient Based on Kendall's Tau. *Journal of the American Statistical Association* **63**, 1379-1389.

Theil, H. (1950). A Rank-Invariant Method of Linear and Polynomial Regression Analysis, I-III. *Proc. Kon. Ned. Akad. v. Wetensch. A.* **53**, 386-392, 521-525, 1397-1412.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

van Belle, G., and J.P. Hughes. (1984). Nonparametric Tests for Trend in Water Quality. *Water Resources Research* **20**(1), 127-136.

See Also

kendallTrendTest, htestEnvStats.object, cor.test.

Examples

```
# Reproduce Example 14-10 on page 14-38 of USEPA (2009).  This example
# tests for trend in analyte concentrations (ppm) collected monthly
# between 1983 and 1985.

head(EPA.09.Ex.14.8.df)
#     Month Year Unadj.Conc Adj.Conc
#1  January 1983       1.99     2.11
#2 February 1983       2.10     2.14
#3    March 1983       2.12     2.10
#4    April 1983       2.12     2.13
#5      May 1983       2.11     2.12
```

```
  #6     June 1983       2.15     2.12

  tail(EPA.09.Ex.14.8.df)
  #       Month Year Unadj.Conc Adj.Conc
  #31      July 1985       2.31     2.23
  #32    August 1985       2.32     2.24
  #33 September 1985       2.28     2.23
  #34   October 1985       2.22     2.24
  #35  November 1985       2.19     2.25
  #36  December 1985       2.22     2.23


  # Plot the data
  #--------------
  Unadj.Conc <- EPA.09.Ex.14.8.df$Unadj.Conc
  Adj.Conc   <- EPA.09.Ex.14.8.df$Adj.Conc
  Month      <- EPA.09.Ex.14.8.df$Month
  Year       <- EPA.09.Ex.14.8.df$Year
  Time       <- paste(substring(Month, 1, 3), Year - 1900, sep = "-")
  n          <- length(Unadj.Conc)
  Three.Yr.Mean <- mean(Unadj.Conc)

  dev.new()
  par(mar = c(7, 4, 3, 1) + 0.1, cex.lab = 1.25)
  plot(1:n, Unadj.Conc, type = "n", xaxt = "n",
xlab = "Time (Month)",
ylab = "ANALYTE CONCENTRATION (mg/L)",
main = "Figure 14-15. Seasonal Time Series Over a Three Year Period",
cex.main = 1.1)
  axis(1, at = 1:n, labels = rep("", n))
  at <- rep(c(1, 5, 9), 3) + rep(c(0, 12, 24), each = 3)
  axis(1, at = at, labels = Time[at])
  points(1:n, Unadj.Conc, pch = 0, type = "o", lwd = 2)
  points(1:n, Adj.Conc, pch = 3, type = "o", col = 8, lwd = 2)
  abline(h = Three.Yr.Mean, lwd = 2)
  legend("topleft", c("Unadjusted", "Adjusted", "3-Year Mean"), bty = "n",
    pch = c(0, 3, -1), lty = c(1, 1, 1), lwd = 2, col = c(1, 8, 1),
    inset = c(0.05, 0.01))


  # Perform the seasonal Kendall trend test
  #----------------------------------------

  kendallSeasonalTrendTest(Unadj.Conc ~ Month + Year,
    data = EPA.09.Ex.14.8.df)

  #Results of Hypothesis Test
  #--------------------------
  #
  #Null Hypothesis:                 All 12 values of tau = 0
  #
  #Alternative Hypothesis:          The seasonal taus are not all equal
  #                                 (Chi-Square Heterogeneity Test)
```

```
#                                 At least one seasonal tau != 0
#                                 and all non-zero tau's have the
#                                 same sign (z Trend Test)
#
#Test Name:                       Seasonal Kendall Test for Trend
#                                 (with continuity correction)
#
#Estimated Parameter(s):          tau       =    0.9722222
#                                 slope     =    0.0600000
#                                 intercept = -131.7350000
#
#Estimation Method:               tau:        Weighted Average of
#                                             Seasonal Estimates
#                                 slope:      Hirsch et al.'s
#                                             Modification of
#                                             Thiel/Sen Estimator
#                                 intercept:  Median of
#                                             Seasonal Estimates
#
#Data:                            y      = Unadj.Conc
#                                 season = Month
#                                 year   = Year
#
#Data Source:                     EPA.09.Ex.14.8.df
#
#Sample Sizes:                    January   =  3
#                                 February  =  3
#                                 March     =  3
#                                 April     =  3
#                                 May       =  3
#                                 June      =  3
#                                 July      =  3
#                                 August    =  3
#                                 September =  3
#                                 October   =  3
#                                 November  =  3
#                                 December  =  3
#                                 Total     = 36
#
#Test Statistics:                 Chi-Square (Het) = 0.1071882
#                                 z (Trend)        = 5.1849514
#
#Test Statistic Parameter:        df = 11
#
#P-values:                        Chi-Square (Het) = 1.000000e+00
#                                 z (Trend)        = 2.160712e-07
#
#Confidence Interval for:         slope
#
#Confidence Interval Method:      Gilbert's Modification of
#                                 Theil/Sen Method
#
#Confidence Interval Type:        two-sided
```

```
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 0.05786914
#                                 UCL = 0.07213086

#==========

# Clean up
#---------

rm(Unadj.Conc, Adj.Conc, Month, Year, Time, n, Three.Yr.Mean, at)
graphics.off()
```

---

kendallTrendTest *Kendall's Nonparametric Test for Montonic Trend*

Description

Perform a nonparametric test for a monotonic trend based on Kendall's tau statistic, and optionally compute a confidence interval for the slope.

Usage

```
kendallTrendTest(y, ...)

## S3 method for class 'formula'
kendallTrendTest(y, data = NULL, subset,
  na.action = na.pass, ...)

## Default S3 method:
kendallTrendTest(y, x = seq(along = y),
  alternative = "two.sided", correct = TRUE, ci.slope = TRUE,
  conf.level = 0.95, warn = TRUE, data.name = NULL, data.name.x = NULL,
  parent.of.data = NULL, subset.expression = NULL, ...)
```

Arguments

y	an object containing data for the trend test. In the default method, the argument y must be numeric vector of observations. In the formula method, y must be a formula of the form y ~ 1 or y ~ x. The form y ~ 1 indicates use the observations in the vector y for the test for trend and use the default value of the argument x in the call to kendallTrendTest.default. The form y ~ x indicates use the observations in the vector y for the test for trend and use the specified value of the argument x in the call to kendallTrendTest.default. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

data	specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which kendallTrendTest is called.
subset	specifies an optional vector specifying a subset of observations to be used.
na.action	specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.
x	numeric vector of "predictor" values. The length of x must equal the length of y. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. The default value of x is the vector of numbers $1, 2, \ldots, n$ where n is the number of elements in y.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (tau not equal to 0; the default), "less" (tau less than 0), and "greater" (tau greater than 0).
correct	logical scalar indicating whether to use the correction for continuity in computing the z-statistic that is based on the test statistic S. The default value is TRUE.
ci.slope	logical scalar indicating whether to compute a confidence interval for the slope. The default value is TRUE.
conf.level	numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the slope. The default value is 0.95.
warn	logical scalar indicating whether to print a warning message when y does not contain at least two non-missing values, or when x does not contain at least two unique non-missing values. The default value is TRUE.
data.name	character string indicating the name of the data used for the trend test. The default value is deparse(substitute(y)).
data.name.x	character string indicating the name of the data used for the predictor variable x. If x is not supplied this argument is ignored. When x is supplied, the default value is deparse(substitute(x)).
parent.of.data	character string indicating the source of the data used for the trend test.
subset.expression	character string indicating the expression used to subset the data.
...	additional arguments affecting the test for trend.

Details

kendallTrendTest performs Kendall's nonparametric test for a monotonic trend, which is a special case of the test for independence based on Kendall's tau statistic (see cor.test). The slope is estimated using the method of Theil (1950) and Sen (1968). When ci.slope=TRUE, the confidence interval for the slope is computed using Gilbert's (1987) Modification of the Theil/Sen Method.

Kendall's test for a monotonic trend is a special case of the test for independence based on Kendall's tau statistic. The first section below explains the general case of testing for independence. The second section explains the special case of testing for monotonic trend. The last section explains how a simple linear regression model is a special case of a monotonic trend and how the slope may be

estimated.

The General Case of Testing for Independence

Definition of Kendall's Tau
Let X and Y denote two continuous random variables with some joint (bivariate) distribution. Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ denote a set of n bivariate observations from this distribution, and assume these bivariate observations are mutually independent. Kendall (1938, 1975) proposed a test for the hypothesis that the X and Y random variables are independent based on estimating the following quantity:

$$\tau = \{2Pr[(X_2 - X_1)(Y_2 - Y_1) > 0]\} - 1 \qquad (1)$$

The quantity in Equation (1) is called Kendall's tau, although this term is more often applied to the estimate of τ (see Equation (2) below). If X and Y are independent, then $\tau = 0$. Furthermore, for most distributions of interest, if $\tau = 0$ then the random variables X and Y are independent. (It can be shown that there exist some distributions for which $\tau = 0$ and the random variables X and Y are not independent; see Hollander and Wolfe (1999, p.364)).

Note that Kendall's tau is similar to a correlation coefficient in that $-1 \leq \tau \leq 1$. If X and Y always vary in the same direction, that is if $X_1 < X_2$ always implies $Y_1 < Y_2$, then $\tau = 1$. If X and Y always vary in the opposite direction, that is if $X_1 < X_2$ always implies $Y_1 > Y_2$, then $\tau = -1$. If $\tau > 0$, this indicates X and Y are positively associated. If $\tau < 0$, this indicates X and Y are negatively associated.

Estimating Kendall's Tau
The quantity in Equation (1) can be estimated by:

$$\hat{\tau} = \frac{2S}{n(n-1)} \qquad (2)$$

where

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sign[(X_j - X_i)(Y_j - Y_i)] \qquad (3)$$

and $sign()$ denotes the `sign` function:

$$sign(x) = \begin{array}{rl} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{array} \qquad (4)$$

(Hollander and Wolfe, 1999, Chapter 8; Conover, 1980, pp.256–260; Gilbert, 1987, Chapter 16; Helsel and Hirsch, 1992, pp.212–216; Gibbons et al., 2009, Chapter 11). The quantity defined in Equation (2) is called Kendall's rank correlation coefficient or more often Kendall's tau.

Note that the quantity S defined in Equation (3) is equal to the number of concordant pairs minus the number of discordant pairs. Hollander and Wolfe (1999, p.364) use the notation K instead of S, and Conover (1980, p.257) uses the notation T.

Testing the Null Hypothesis of Independence

The null hypothesis $H_0 : \tau = 0$, can be tested using the statistic S defined in Equation (3) above. Tables of the distribution of S for small samples are given in Hollander and Wolfe (1999), Conover (1980, pp.458–459), Gilbert (1987, p.272), Helsel and Hirsch (1992, p.469), and Gibbons (2009, p.210). The function `kendallTrendTest` uses the large sample approximation to the distribution of S under the null hypothesis, which is given by:

$$z = \frac{S - E(S)}{\sqrt{Var(S)}} \qquad (5)$$

where

$$E(S) = 0 \qquad (6)$$

$$Var(S) = \frac{n(n-1)(2n+5)}{18} \qquad (7)$$

Under the null hypothesis, the quantity z defined in Equation (5) is approximately distributed as a standard normal random variable.

Both Kendall (1975) and Mann (1945) show that the normal approximation is excellent even for samples as small as $n = 10$, provided that the following continuity correction is used:

$$z = \frac{S - sign(S)}{\sqrt{Var(S)}} \qquad (8)$$

The function `kendallTrendTest` performs the usual one-sample z-test using the statistic computed in Equation (8) or Equation (5). The argument `correct` determines which equation is used to compute the z-statistic. By default, `correct=TRUE` so Equation (8) is used.

In the case of tied observations in either the observed X's and/or observed Y's, the formula for the variance of S given in Equation (7) must be modified as follows:

$$\begin{aligned} Var(S) = \quad & \tfrac{n(n-1)(2n+5)}{18} - \\ & \tfrac{\sum_{i=1}^{g} t_i(t_i-1)(2t_i+5)}{18} - \\ & \tfrac{\sum_{j=1}^{h} u_j(u_j-1)(2u_j+5)}{18} + \\ & \tfrac{[\sum_{i=1}^{g} t_i(t_i-1)(t_i-2)][\sum_{j=1}^{h} u_j(u_j-1)(u_j-2)]}{9n(n-1)(n-2)} + \\ & \tfrac{[\sum_{i=1}^{g} t_i(t_i-1)][\sum_{j=1}^{h} u_j(u_j-1)]}{2n(n-1)} \qquad (9) \end{aligned}$$

where g is the number of tied groups in the X observations, t_i is the size of the i'th tied group in the X observations, h is the number of tied groups in the Y observations, and u_j is the size of the j'th tied group in the Y observations. In the case of no ties in either the X or Y observations, Equation (9) reduces to Equation (7).

The Special Case of Testing for Monotonic Trend

Often in environmental sampling, observations are taken periodically over time (Hirsch et al., 1982; van Belle and Hughes, 1984; Hirsch and Slack, 1984). In this case, the random variables

$Y_1, Y_2, \ldots, Y_n$ can be thought of as representing the observations, and the variables $X_1, X_2, \ldots, X_n$ are no longer random but represent the time at which the i'th observation was taken. If the observations are equally spaced over time, then it is useful to make the simplification $X_i = i$ for $i = 1, 2, \ldots, n$. This is in fact the default value of the argument `x` for the function `kendallTrendTest`.

In the case where the X's represent time and are all distinct, the test for independence between X and Y is equivalent to testing for a monotonic trend in Y, and the test statistic S simplifies to:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sign(Y_j - Y_i) \qquad (10)$$

Also, the formula for the variance of S in the presence of ties (under the null hypothesis $H_0 : \tau = 0$) simplifies to:

$$Var(S) = \frac{n(n-1)(2n+5)}{18} - \frac{\sum_{j=1}^{h} u_j(u_j-1)(2u_j+5)}{18} \qquad (11)$$

A form of the test statistic S in Equation (10) was introduced by Mann (1945).

The Special Case of a Simple Linear Model: Estimating the Slope

Consider the simple linear regression model

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \qquad (12)$$

where β_0 denotes the intercept, β_1 denotes the slope, $i = 1, 2, \ldots, n$, and the ϵ's are assumed to be independent and identically distributed random variables from the same distribution. This is a special case of dependence between the X's and the Y's, and the null hypothesis of a zero slope can be tested using Kendall's test statistic S (Equation (3) or (10) above) and the associated z-statistic (Equation (5) or (8) above) (Hollander and Wolfe, 1999, pp.415–420).

Theil (1950) proposed the following nonparametric estimator of the slope:

$$\hat{\beta}_1 = Median(\frac{Y_j - Y_i}{X_j - X_i}); \;\; i < j \qquad (13)$$

Note that the computation of the estimated slope involves computing

$$N = \binom{n}{2} = \frac{n(n-1)}{2} \qquad (14)$$

"two-point" estimated slopes (assuming no tied X values), and taking the median of these N values.

Sen (1968) generalized this estimator to the case where there are possibly tied observations in the X's. In this case, Sen simply ignores the two-point estimated slopes where the X's are tied and computes the median based on the remaining N' two-point estimated slopes. That is, Sen's estimator is given by:

$$\hat{\beta}_1 = Median(\frac{Y_j - Y_i}{X_j - X_i}); \;\; i < j, X_i \neq X_j \qquad (15)$$

(Hollander and Wolfe, 1999, pp.421–422).

Conover (1980, p. 267) suggests the following estimator for the intercept:

$$\hat{\beta}_0 = Y_{0.5} - \hat{\beta}_1 X_{0.5} \qquad (16)$$

where $X_{0.5}$ and $Y_{0.5}$ denote the sample medians of the X's and Y's, respectively. With these estimators of slope and intercept, the estimated regression line passes through the point $(X_{0.5}, Y_{0.5})$.

NOTE: The function `kendallTrendTest` always returns estimates of slope and intercept assuming a linear model (Equation (12)), while the p-value is based on Kendall's tau, which is testing for the broader alternative of any kind of dependence between the X's and Y's.

Confidence Interval for the Slope

Theil (1950) and Sen (1968) proposed methods to compute a confidence interval for the true slope, assuming the linear model of Equation (12) (see Hollander and Wolfe, 1999, pp.421-422). Gilbert (1987, p.218) illustrates a simpler method than the one given by Sen (1968) that is based on a normal approximation. Gilbert's (1987) method is an extension of the one given in Hollander and Wolfe (1999, p.424) that allows for ties and/or multiple observations per time period. This method is valid for a sample size as small as $n = 10$ unless there are several tied observations.

Let N' denote the number of defined two-point estimated slopes that are used in Equation (15) above (if there are no tied X values then $N' = N$), and let $\hat{\beta}_{1_{(1)}}, \hat{\beta}_{1_{(2)}}, \ldots, \hat{\beta}_{1_{(N')}}$ denote the N' ordered slopes. For Gilbert's (1987) method, a $100(1-\alpha)\%$ two-sided confidence interval for the true slope is given by:

$$[\hat{\beta}_{1_{(M1)}}, \hat{\beta}_{1_{(M2+1)}}] \quad (17)$$

where

$$M1 = \frac{N' - C_\alpha}{2} \quad (18)$$

$$M2 = \frac{N' + C_\alpha}{2} \quad (19)$$

$$C_\alpha = z_{1-\alpha/2}\sqrt{Var(S)} \quad (20)$$

$Var(S)$ is defined in Equations (7), (9), or (11), and z_p denotes the p'th quantile of the standard normal distribution. One-sided confidence intervals may computed in a similar fashion.

Usually the quantities $M1$ and $M2$ will not be integers. Gilbert (1987, p.219) suggests interpolating between adjacent values in this case, which is what the function `kendallTrendTest` does.

Value

A list of class `"htestEnvStats"` containing the results of the hypothesis test. See the help file for `htestEnvStats.object` for details. In addition, the following components are part of the list returned by `kendallTrendTest`:

`S`	The value of the Kendall S-statistic.
`var.S`	The variance of the Kendall S-statistic.
`slopes`	A numeric vector of all possible two-point slope estimates. This component is used by the function `kendallSeasonalTrendTest`.

Note

Kendall's test for independence or trend is a nonparametric test. No assumptions are made about the distribution of the X and Y variables. Hirsch et al. (1982) introduced the "seasonal Kendall test" to test for trend within each season. They note that Kendall's test for trend is easy to compute, even in the presence of missing values, and can also be used with censored values.

van Belle and Hughes (1984) note that Kendall's test for trend is slightly less powerful than the test based on Spearman's rho, but it converges to normality faster. Also, Bradley (1968, p.288) shows that for the case of a linear model with normal (Gaussian) errors, the asymptotic relative efficiency of Kendall's test for trend versus the parametric test for a zero slope is 0.98.

The results of the function `kendallTrendTest` are similar to the results of the built-in R function `cor.test` with the argument `method="kendall"` except that `cor.test` 1) computes exact p-values when the number of pairs is less than 50 and there are no ties, and 2) does not return a confidence interval for the slope.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bradley, J.V. (1968). *Distribution-Free Statistical Tests*. Prentice-Hall, Englewood Cliffs, NJ.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York, pp.256-272.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, Chapter 16.

Helsel, D.R. and R.M. Hirsch. (1988). Discussion of Applicability of the t-test for Detecting Trends in Water Quality Variables. *Water Resources Bulletin* **24**(1), 201-204.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, NY.

Helsel, D.R., and R. M. Hirsch. (2002). *Statistical Methods in Water Resources*. Techniques of Water Resources Investigations, Book 4, chapter A3. U.S. Geological Survey. Available on-line at https://pubs.usgs.gov/tm/04/a03/tm4a3.pdf.

Hirsch, R.M., J.R. Slack, and R.A. Smith. (1982). Techniques of Trend Analysis for Monthly Water Quality Data. *Water Resources Research* **18**(1), 107-121.

Hirsch, R.M. and J.R. Slack. (1984). A Nonparametric Trend Test for Seasonal Data with Serial Dependence. *Water Resources Research* **20**(6), 727-732.

Hirsch, R.M., R.B. Alexander, and R.A. Smith. (1991). Selection of Methods for the Detection and Estimation of Trends in Water Quality. *Water Resources Research* **27**(5), 803-813.

Hollander, M., and D.A. Wolfe. (1999). *Nonparametric Statistical Methods, Second Edition*. John Wiley and Sons, New York.

Kendall, M.G. (1938). A New Measure of Rank Correlation. *Biometrika* **30**, 81-93.

Kendall, M.G. (1975). *Rank Correlation Methods*. Charles Griffin, London.

Mann, H.B. (1945). Nonparametric Tests Against Trend. *Econometrica* **13**, 245-259.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Sen, P.K. (1968). Estimates of the Regression Coefficient Based on Kendall's Tau. *Journal of the American Statistical Association* **63**, 1379-1389.

Theil, H. (1950). A Rank-Invariant Method of Linear and Polynomial Regression Analysis, I-III. *Proc. Kon. Ned. Akad. v. Wetensch. A.* **53**, 386-392, 521-525, 1397-1412.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

van Belle, G., and J.P. Hughes. (1984). Nonparametric Tests for Trend in Water Quality. *Water Resources Research* **20**(1), 127-136.

See Also

cor.test, kendallSeasonalTrendTest, htestEnvStats.object.

Examples

```
# Reproduce Example 17-6 on page 17-33 of USEPA (2009).  This example
# tests for trend in sulfate concentrations (ppm) collected at various
# months between 1989 and 1996.

head(EPA.09.Ex.17.6.sulfate.df)
#  Sample.No Year Month Sampling.Date       Date Sulfate.ppm
#1         1   89     6          89.6 1989-06-01         480
#2         2   89     8          89.8 1989-08-01         450
#3         3   90     1          90.1 1990-01-01         490
#4         4   90     3          90.3 1990-03-01         520
#5         5   90     6          90.6 1990-06-01         485
#6         6   90     8          90.8 1990-08-01         510


# Plot the data
#--------------
dev.new()
with(EPA.09.Ex.17.6.sulfate.df,
  plot(Sampling.Date, Sulfate.ppm, pch = 15, ylim = c(400, 900),
  xlab = "Sampling Date", ylab = "Sulfate Conc (ppm)",
  main = "Figure 17-6. Time Series Plot of \nSulfate Concentrations (ppm)")
)
Sulfate.fit <- lm(Sulfate.ppm ~ Sampling.Date,
  data = EPA.09.Ex.17.6.sulfate.df)
abline(Sulfate.fit, lty = 2)


# Perform the Kendall test for trend
#-----------------------------------
kendallTrendTest(Sulfate.ppm ~ Sampling.Date,
  data = EPA.09.Ex.17.6.sulfate.df)

#Results of Hypothesis Test
```

```
#--------------------------
#
#Null Hypothesis:                 tau = 0
#
#Alternative Hypothesis:          True tau is not equal to 0
#
#Test Name:                       Kendall's Test for Trend
#                                 (with continuity correction)
#
#Estimated Parameter(s):          tau       =     0.7667984
#                                 slope     =    26.6666667
#                                 intercept = -1909.3333333
#
#Estimation Method:               slope:      Theil/Sen Estimator
#                                 intercept:  Conover's Estimator
#
#Data:                            y = Sulfate.ppm
#                                 x = Sampling.Date
#
#Data Source:                     EPA.09.Ex.17.6.sulfate.df
#
#Sample Size:                     23
#
#Test Statistic:                  z = 5.107322
#
#P-value:                         3.267574e-07
#
#Confidence Interval for:         slope
#
#Confidence Interval Method:      Gilbert's Modification
#                                 of Theil/Sen Method
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 20.00000
#                                 UCL = 35.71182


# Clean up
#---------
rm(Sulfate.fit)
graphics.off()
```

---

kurtosis *Coefficient of (Excess) Kurtosis*

Description

Compute the sample coefficient of kurtosis or excess kurtosis.

Usage

```
kurtosis(x, na.rm = FALSE, method = "fisher", l.moment.method = "unbiased",
  plot.pos.cons = c(a = 0.35, b = 0), excess = TRUE)
```

Arguments

x	numeric vector of observations.
na.rm	logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the coefficient of variation.
method	character string specifying what method to use to compute the sample coefficient of kurtosis. The possible values are "fisher" (ratio of unbiased moment estimators; the default), "moments" (ratio of product moment estimators), or "l.moments" (ratio of L-moment estimators).
l.moment.method	character string specifying what method to use to compute the L-moments when method="l.moments". The possible values are "ubiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula).
plot.pos.cons	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="l.moments" and l.moment.method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b".
excess	logical scalar indicating whether to compute the kurtosis (excess=FALSE) or excess kurtosis (excess=TRUE; the default).

Details

Let $\underline{x}$ denote a random sample of n observations from some distribution with mean μ and standard deviation σ.

Product Moment Coefficient of Kurtosis
(method="moment" or method="fisher")
The ***coefficient of kurtosis*** of a distribution is the fourth standardized moment about the mean:

$$\eta_4 = \beta_2 = \frac{\mu_4}{\sigma^4} \qquad (1)$$

where

$$\eta_r = E[(\frac{X-\mu}{\sigma})^r] = \frac{1}{\sigma^r}E[(X-\mu)^r] = \frac{\mu_r}{\sigma^r} \qquad (2)$$

and

$$\mu_r = E[(X-\mu)^r] \qquad (3)$$

denotes the r'th moment about the mean (central moment).

The ***coefficient of excess kurtosis*** is defined as:

$$\beta_2 - 3 \qquad (4)$$

For a normal distribution, the coefficient of kurtosis is 3 and the coefficient of excess kurtosis is 0. Distributions with kurtosis less than 3 (excess kurtosis less than 0) are called ***platykurtic***: they have shorter tails than a normal distribution. Distributions with kurtosis greater than 3 (excess kurtosis greater than 0) are called ***leptokurtic***: they have heavier tails than a normal distribution.

When `method="moment"`, the coefficient of kurtosis is estimated using the method of moments estimator for the fourth central moment and and the method of moments estimator for the variance:

$$\hat{\eta}_4 = \frac{\hat{\mu}_4}{\sigma^4} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2]^2} \qquad (5)$$

where

$$\hat{\sigma}_m^2 = s_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (6)$$

This form of estimation should be used when resampling (bootstrap or jackknife).

When `method="fisher"`, the coefficient of kurtosis is estimated using the unbiased estimator for the fourth central moment (Serfling, 1980, p.73) and the unbiased estimator for the variance.

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (7)$$

L-Moment Coefficient of Kurtosis (`method="l.moments"`)
Hosking (1990) defines the L-moment analog of the coefficient of kurtosis as:

$$\tau_4 = \frac{\lambda_4}{\lambda_2} \qquad (8)$$

that is, the fourth L-moment divided by the second L-moment. He shows that this quantity lies in the interval (-1, 1).

When `l.moment.method="unbiased"`, the L-kurtosis is estimated by:

$$t_4 = \frac{l_4}{l_2} \qquad (9)$$

that is, the unbiased estimator of the fourth L-moment divided by the unbiased estimator of the second L-moment.

When `l.moment.method="plotting.position"`, the L-kurtosis is estimated by:

$$\tilde{\tau}_4 = \frac{\tilde{\lambda}_4}{\tilde{\lambda}_2} \qquad (10)$$

that is, the plotting-position estimator of the fourth L-moment divided by the plotting-position estimator of the second L-moment.

See the help file for `lMoment` for more information on estimating L-moments.

Value

A numeric scalar – the sample coefficient of kurtosis or excess kurtosis.

Note

Traditionally, the coefficient of kurtosis has been estimated using product moment estimators. Sometimes an estimate of kurtosis is used in a goodness-of-fit test for normality (D'Agostino and Stephens, 1986). Hosking (1990) introduced the idea of L-moments and L-kurtosis.

Vogel and Fennessey (1993) argue that L-moment ratios should replace product moment ratios because of their superior performance (they are nearly unbiased and better for discriminating between distributions). They compare product moment diagrams with L-moment diagrams.

Hosking and Wallis (1995) recommend using unbiased estimators of L-moments (vs. plotting-position estimators) for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Taylor, J.K. (1990). *Statistical Techniques for Data Analysis*. Lewis Publishers, Boca Raton, FL.

Vogel, R.M., and N.M. Fennessey. (1993). L Moment Diagrams Should Replace Product Moment Diagrams. *Water Resources Research* **29**(6), 1745–1752.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

var, sd, cv, skewness, summaryFull, Summary Statistics.

Examples

```
# Generate 20 observations from a lognormal distribution with parameters
# mean=10 and cv=1, and estimate the coefficient of kurtosis and
# coefficient of excess kurtosis.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

dat <- rlnormAlt(20, mean = 10, cv = 1)

# Compute standard kurtosis first
#--------------------------------
kurtosis(dat, excess = FALSE)
#[1] 2.964612
```

```
kurtosis(dat, method = "moment", excess = FALSE)
#[1] 2.687146

kurtosis(dat, method = "l.moment", excess = FALSE)
#[1] 0.1444779


# Now compute excess kurtosis
#----------------------------
kurtosis(dat)
#[1] -0.0353876

kurtosis(dat, method = "moment")
#[1] -0.3128536

kurtosis(dat, method = "l.moment")
#[1] -2.855522

#----------
# Clean up
rm(dat)
```

`Lin.Evans.80.df` *Fecal Coliform Data from the Illinois River*

Description

Lin and Evans (1980) reported fecal coliform measures (organisms per 100 ml) from the Illinois River taken between 1971 and 1976. The object `Lin.Evans.80.df` is a small subset of these data that were reported by Helsel and Hirsch (1992, p.162).

Usage

```
Lin.Evans.80.df
```

Format

A data frame with 24 observations on the following 2 variables.

`Fecal.Coliform` a numeric vector of fecal coliform measure (organisms per 100 ml).

`Season` an ordered factor indicating the season of collection

Source

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, p.162.

References

Lin, S.D., and R.L. Evans. (1980). *Coliforms and fecal streptococcus in the Illinois River at Peoria, 1971-1976*. Illinois State Water Survey Report of Investigations No. 93. Urbana, IL, 28pp.

`linearTrendTestN` *Sample Size for a t-Test for Linear Trend*

Description

Compute the sample size necessary to achieve a specified power for a t-test for linear trend, given the scaled slope and significance level.

Usage

```
linearTrendTestN(slope.over.sigma, alpha = 0.05, power = 0.95,
  alternative = "two.sided", approx = FALSE, round.up = TRUE,
  n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

`slope.over.sigma`
: numeric vector specifying the ratio of the true slope to the standard deviation of the error terms (σ). This is also called the "scaled slope". The default value is `slope.over.sigma=0`.

`alpha`
: numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.

`power`
: numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.

`alternative`
: character string indicating the kind of alternative hypothesis. The possible values are `"two.sided"` (the default), `"greater"`, and `"less"`.

`approx`
: logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `approx=FALSE`.

`round.up`
: logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is `TRUE`.

`n.max`
: positive integer greater than 2 indicating the maximum sample size. The default value is `n.max=5000`.

`tol`
: numeric scalar indicating the toloerance to use in the `uniroot` search algorithm. The default value is `tol=1e-7`.

`maxiter`
: positive integer indicating the maximum number of iterations argument to pass to the `uniroot` function. The default value is `maxiter=1000`.

Details

If the arguments `slope.over.sigma`, `alpha`, and `power` are not all the same length, they are replicated to be the same length as the length of the longest argument.

Formulas for the power of the t-test of linear trend for specified values of the sample size, scaled slope, and Type I error level are given in the help file for `linearTrendTestPower`. The function `linearTrendTestN` uses the `uniroot` search algorithm to determine the required sample size(s) for specified values of the power, scaled slope, and Type I error level.

Value

a numeric vector of sample sizes.

Note

See the help file for `linearTrendTestPower`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `linearTrendTestPower`.

See Also

`linearTrendTestPower`, `linearTrendTestScaledMds`, `plotLinearTrendTestDesign`, `lm`, `summary.lm`, `kendallTrendTest`, Power and Sample Size, Normal, `t.test`.

Examples

```
# Look at how the required sample size for the t-test for zero slope
# increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

linearTrendTestN(slope.over.sigma = 0.1, power = seq(0.5, 0.9, by = 0.1))
#[1] 18 19 21 22 25

#----------

# Repeat the last example, but compute the sample size based on the approximate
# power instead of the exact:

linearTrendTestN(slope.over.sigma = 0.1, power = seq(0.5, 0.9, by = 0.1),
  approx = TRUE)
#[1] 18 19 21 22 25

#==========
```

```
# Look at how the required sample size for the t-test for zero slope decreases
# with increasing scaled slope:

seq(0.05, 0.2, by = 0.05)
#[1] 0.05 0.10 0.15 0.20

linearTrendTestN(slope.over.sigma = seq(0.05, 0.2, by = 0.05))
#[1] 41 26 20 17

#==========

# Look at how the required sample size for the t-test for zero slope decreases
# with increasing values of Type I error:

linearTrendTestN(slope.over.sigma = 0.1, alpha = c(0.001, 0.01, 0.05, 0.1))
#[1] 33 29 26 25
```

---

linearTrendTestPower *Power of a t-Test for Linear Trend*

Description

Compute the power of a parametric test for linear trend, given the sample size or predictor variable values, scaled slope, and significance level.

Usage

```
linearTrendTestPower(n, x = lapply(n, seq), slope.over.sigma = 0, alpha = 0.05,
  alternative = "two.sided", approx = FALSE)
```

Arguments

n	numeric vector of sample sizes. All values of n must be positive integers larger than 2. This argument is ignored when x is supplied. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
x	numeric vector of predictor variable values, or a list in which each component is a numeric vector of predictor variable values. Usually, the predictor variable is time (e.g., days, months, quarters, etc.). The default value is x=lapply(n,seq), which yields a list in which the i'th component is the seqence of integers from 1 to the i'th value of the vector n. If x is a numeric vector, it must contain at least three elements, two of which must be unique. If x is a list of numeric vectors, each component of x must contain at least three elements, two of which must be unique. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
slope.over.sigma	numeric vector specifying the ratio of the true slope to the standard deviation of the error terms (σ). This is also called the "scaled slope". The default value is slope.over.sigma=0.

alpha	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
approx	logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.

Details

If the argument x is a vector, it is converted into a list with one component. If the arguments n, x, slope.over.sigma, and alpha are not all the same length, they are replicated to be the same length as the length of the longest argument.

Basic Model

Consider the simple linear regression model

$$Y = \beta_0 + \beta_1 X + \epsilon \qquad (1)$$

where X denotes the predictor variable (observed without error), β_0 denotes the intercept, β_1 denotes the slope, and the error term ϵ is assumed to be a random variable from a normal distribution with mean 0 and standard deviation σ. Let

$$(\underline{x}, \underline{y}) = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \qquad (2)$$

denote n independent observed (X, Y) pairs from the model (1).

Often in environmental data analysis, we are interested in determining whether there is a trend in some indicator variable over time. In this case, the predictor variable X is time (e.g., day, month, quarter, year, etc.), and the n values of the response variable Y represent measurements taken over time. The slope then represents the change in the average of the response variable per one unit of time.

When the argument x is a numeric vector, it represents the n values of the predictor variable. When the argument x is a list, each component of x is a numeric vector that represents a set values of the predictor variable (and the number of elements may vary by component). By default, the argument x is a list for which the i'th component is simply the integers from 1 to the value of the i'th element of the argument n, representing, for example, Day 1, Day2, ..., Day n[i].

In the discussion that follows, be sure not to confuse the intercept and slope coefficients β_0 and β_1 with the Type II error of the hypothesis test, which is denoted by β.

Estimation of Coefficients and Confidence Interval for Slope

The standard least-squares estimators of the slope and intercept are given by:

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} \qquad (3)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \qquad (4)$$

where

$$S_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \qquad (5)$$

$$S_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (6)$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (7)$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad (8)$$

(Draper and Smith, 1998, p.25; Zar, 2010, p.332-334; Berthoux and Brown, 2002, p.297; Helsel and Hirsch, p.226). The estimator of slope in Equation (3) has a normal distribution with mean equal to the true slope, and variance given by:

$$Var(\hat{\beta}_1) = \sigma^2_{\hat{\beta}_1} = \frac{\sigma^2}{S_{xx}} \qquad (9)$$

(Draper and Smith, 1998, p.35; Zar, 2010, p.341; Berthoux and Brown, 2002, p.299; Helsel and Hirsch, 1992, p.227). Thus, a $(1-\alpha)100\%$ two-sided confidence interval for the slope is given by:

$$[\hat{\beta}_1 - t_{n-2}(1-\alpha/2)\hat{\sigma}_{\hat{\beta}_1},\ \ \hat{\beta}_1 + t_{n-2}(1-\alpha/2)\hat{\sigma}_{\hat{\beta}_1}] \qquad (10)$$

where

$$\hat{\sigma}_{\hat{\beta}_1} = \frac{\hat{\sigma}}{\sqrt{S_{xx}}} \qquad (11)$$

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (12)$$

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \qquad (13)$$

and $t_\nu(p)$ denotes the p'th quantile of Student's t-distribution with ν degrees of freedom (Draper and Smith, 1998, p.36; Zar, 2010, p.343; Berthoux and Brown, 2002, p.300; Helsel and Hirsch, 1992, p.240).

Testing for a Non-Zero Slope

Consider the null hypothesis of a zero slope coefficient:

$$H_0 : \beta_1 = 0 \qquad (14)$$

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`):

$$H_a : \beta_1 > 0 \qquad (15)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : \beta_1 < 0 \qquad (16)$$

and the two-sided alternative (`alternative="two.sided"`)

$$H_a : \beta_1 \neq 0 \qquad (17)$$

The test of the null hypothesis (14) versus any of the three alternatives (15)-(17) is based on the Student t-statistic:

$$t = \frac{\hat{\beta}_1}{\hat{\sigma}_{\hat{\beta}_1}} = \frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \qquad (18)$$

Under the null hypothesis (14), the t-statistic in (18) follows a Student's t-distribution with $n-2$ degrees of freedom (Draper and Smith, 1998, p.36; Zar, 2010, p.341; Helsel and Hirsch, 1992, pp.238-239).

The formula for the power of the test of a zero slope depends on which alternative is being tested. The two subsections below describe exact and approximate formulas for the power of the test. Note that none of the equations for the power of the t-test requires knowledge of the values β_1 or σ (the population standard deviation of the error terms), only the ratio β_1/σ. The argument `slope.over.sigma` is this ratio, and it is referred to as the "scaled slope".

Exact Power Calculations (`approx=FALSE`)
This subsection describes the exact formulas for the power of the t-test for a zero slope.

Upper one-sided alternative (`alternative="greater"`)
The standard Student's t-test rejects the null hypothesis (1) in favor of the upper alternative hypothesis (2) at level-α if

$$t \geq t_\nu(1-\alpha) \qquad (19)$$

where

$$\nu = n - 2 \qquad (20)$$

and, as noted previously, $t_\nu(p)$ denotes the p'th quantile of Student's t-distribution with ν degrees of freedom. The power of this test, denoted by $1-\beta$, where β denotes the probability of a Type II error, is given by:

$$1 - \beta = Pr[t_{\nu,\Delta} \geq t_\nu(1-\alpha)] = 1 - G[t_\nu(1-\alpha), \nu, \Delta] \qquad (21)$$

where

$$\Delta = \sqrt{S_{xx}}\frac{\beta_1}{\sigma} \qquad (22)$$

and $t_{\nu,\Delta}$ denotes a non-central Student's t-random variable with ν degrees of freedom and non-centrality parameter Δ, and $G(x, \nu, \Delta)$ denotes the cumulative distribution function of this random variable evaluated at x (Johnson et al., 1995, pp.508-510). Note that when the predictor variable X represents equally-spaced measures of time (e.g., days, months, quarters, etc.) and

$$x_i = i, \quad i = 1, 2, \ldots, n \qquad (23)$$

then the non-centrality parameter in Equation (22) becomes:

$$\Delta = \sqrt{\frac{(n-1)n(n+1)}{12}}\frac{\beta_1}{\sigma} \qquad (24)$$

Lower one-sided alternative (`alternative="less"`)
The standard Student's t-test rejects the null hypothesis (1) in favor of the lower alternative hypothesis (3) at level-α if

$$t \leq t_\nu(\alpha) \qquad (25)$$

and the power of this test is given by:

$$1 - \beta = Pr[t_{\nu,\Delta} \leq t_\nu(\alpha)] = G[t_\nu(\alpha), \nu, \Delta] \qquad (26)$$

Two-sided alternative (`alternative="two.sided"`)
The standard Student's t-test rejects the null hypothesis (14) in favor of the two-sided alternative hypothesis (17) at level-α if

$$|t| \geq t_\nu(1 - \alpha/2) \qquad (27)$$

and the power of this test is given by:

$$1 - \beta = Pr[t_{\nu,\Delta} \leq t_\nu(\alpha/2)] + Pr[t_{\nu,\Delta} \geq t_\nu(1 - \alpha/2)]$$

$$= G[t_\nu(\alpha/2), \nu, \Delta] + 1 - G[t_\nu(1 - \alpha/2), \nu, \Delta] \qquad (28)$$

The power of the t-test given in Equation (28) can also be expressed in terms of the cumulative distribution function of the non-central F-distribution as follows. Let $F_{\nu_1,\nu_2,\Delta}$ denote a non-central F random variable with ν_1 and ν_2 degrees of freedom and non-centrality parameter Δ, and let $H(x, \nu_1, \nu_2, \Delta)$ denote the cumulative distribution function of this random variable evaluated at x. Also, let $F_{\nu_1,\nu_2}(p)$ denote the p'th quantile of the central F-distribution with ν_1 and ν_2 degrees of freedom. It can be shown that

$$(t_{\nu,\Delta})^2 \cong F_{1,\nu,\Delta^2} \qquad (29)$$

where $\cong$ denotes "equal in distribution". Thus, it follows that

$$[t_\nu(1 - \alpha/2)]^2 = F_{1,\nu}(1 - \alpha) \qquad (30)$$

so the formula for the power of the t-test given in Equation (28) can also be written as:

$$1 - \beta = Pr\{(t_{\nu,\Delta})^2 \geq [t_\nu(1 - \alpha/2)]^2\}$$

$$= Pr[F_{1,\nu,\Delta^2} \geq F_{1,\nu}(1 - \alpha)] = 1 - H[F_{1,\nu}(1 - \alpha), 1, \nu, \Delta^2] \qquad (31)$$

Approximate Power Calculations (`approx=TRUE`)
Zar (2010, pp.115–118) presents an approximation to the power for the t-test given in Equations (21), (26), and (28) above. His approximation to the power can be derived by using the approximation

$$\sqrt{S_{xx}}\frac{\beta_1}{s} \approx \sqrt{SS_{xx}}\frac{\beta_1}{\sigma} = \Delta \qquad (32)$$

where $\approx$ denotes "approximately equal to". Zar's approximation can be summarized in terms of the cumulative distribution function of the non-central t-distribution as follows:

$$G(x, \nu, \Delta) \approx G(x - \Delta, \nu, 0) = G(x - \Delta, \nu) \qquad (33)$$

where $G(x, \nu)$ denotes the cumulative distribution function of the central Student's t-distribution with ν degrees of freedom evaluated at x.

The following three subsections explicitly derive the approximation to the power of the t-test for each of the three alternative hypotheses.

Upper one-sided alternative (`alternative="greater"`)
The power for the upper one-sided alternative (15) given in Equation (21) can be approximated as:

$$1-\beta = Pr[t \geq t_\nu(1-\alpha)]$$

$$= Pr[\frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \geq t_\nu(1-\alpha) - \sqrt{S_{xx}}\frac{\beta_1}{s}]$$

$$\approx Pr[t_\nu \geq t_\nu(1-\alpha) - \Delta]$$

$$= 1 - Pr[t_\nu \leq t_\nu(1-\alpha) - \Delta]$$

$$= 1 - G[t_\nu(1-\alpha) - \Delta, \nu] \qquad (34)$$

where t_ν denotes a central Student's t-random variable with ν degrees of freedom.

Lower one-sided alternative (`alternative="less"`)
The power for the lower one-sided alternative (16) given in Equation (26) can be approximated as:

$$1-\beta = Pr[t \leq t_\nu(\alpha)]$$

$$= Pr[\frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \leq t_\nu(\alpha) - \sqrt{S_{xx}}\frac{\beta_1}{s}]$$

$$\approx Pr[t_\nu \leq t_\nu(\alpha) - \Delta]$$

$$= G[t_\nu(\alpha) - \Delta, \nu] \qquad (35)$$

Two-sided alternative (`alternative="two.sided"`)
The power for the two-sided alternative (17) given in Equation (28) can be approximated as:

$$1-\beta = Pr[t \leq t_\nu(\alpha/2)] + Pr[t \geq t_\nu(1-\alpha/2)]$$

$$= Pr[\frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \leq t_\nu(\alpha/2) - \sqrt{SS_{xx}}\frac{\beta_1}{s}] + Pr[\frac{\hat{\beta}_1}{s/\sqrt{S_{xx}}} \geq t_\nu(1-\alpha) - \sqrt{SS_{xx}}\frac{\beta_1}{s}]$$

$$\approx Pr[t_\nu \leq t_\nu(\alpha/2) - \Delta] + Pr[t_\nu \geq t_\nu(1-\alpha/2) - \Delta]$$

$$= G[t_\nu(\alpha/2) - \Delta, \nu] + 1 - G[t_\nu(1-\alpha/2) - \Delta, \nu] \qquad (36)$$

Value

a numeric vector powers.

Note

Often in environmental data analysis, we are interested in determining whether there is a trend in some indicator variable over time. In this case, the predictor variable X is time (e.g., day, month, quarter, year, etc.), and the n values of the response variable represent measurements taken over time. The slope then represents the change in the average of the response variable per one unit of time.

You can use the parametric model (1) to model your data, then use the R function `lm` to fit the regression coefficients and the `summary.lm` function to perform a test for the significance of the slope coefficient. The function `linearTrendTestPower` computes the power of this t-test, given a fixed value of the sample size, scaled slope, and significance level.

You can also use Kendall's nonparametric test for trend if you don't want to assume the error terms are normally distributed. When the errors are truly normally distributed, the asymptotic relative efficiency of Kendall's test for trend versus the parametric t-test for a zero slope is 0.98, and Kendall's test can be more powerful than the parametric t-test when the errors are not normally distributed. Thus the function `linearTrendTestPower` can also be used to estimate the power of Kendall's test for trend.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled slope if one of the objectives of the sampling program is to determine whether a trend is occurring. The functions `linearTrendTestPower`, `linearTrendTestN`, `linearTrendTestScaledMds`, and `plotLinearTrendTestDesign` can be used to investigate these relationships.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, Chapter 1.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 9.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York, Chapters 28, 31

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`linearTrendTestN`, `linearTrendTestScaledMds`, `plotLinearTrendTestDesign`, `lm`, `summary.lm`, `kendallTrendTest`, Power and Sample Size, Normal, `t.test`.

Examples

```
# Look at how the power of the t-test for zero slope increases with increasing
# sample size:

seq(5, 30, by = 5)
#[1] 5 10 15 20 25 30

power <- linearTrendTestPower(n = seq(5, 30, by = 5), slope.over.sigma = 0.1)

round(power, 2)
#[1] 0.06 0.13 0.34 0.68 0.93 1.00

#----------

# Repeat the last example, but compute the approximate power instead of the
# exact:

power <- linearTrendTestPower(n = seq(5, 30, by = 5), slope.over.sigma = 0.1,
  approx = TRUE)

round(power, 2)
#[1] 0.05 0.11 0.32 0.68 0.93 0.99

#----------

# Look at how the power of the t-test for zero slope increases with increasing
# scaled slope:

seq(0.05, 0.2, by = 0.05)
#[1] 0.05 0.10 0.15 0.20

power <- linearTrendTestPower(15, slope.over.sigma = seq(0.05, 0.2, by = 0.05))

round(power, 2)
#[1] 0.12 0.34 0.64 0.87

#----------

# Look at how the power of the t-test for zero slope increases with increasing
# values of Type I error:

power <- linearTrendTestPower(20, slope.over.sigma = 0.1,
  alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
#[1] 0.14 0.41 0.68 0.80

#----------

# Show that for a simple regression model, you get a greater power of detecting
# a non-zero slope if you take all the observations at two endpoints, rather than
# spreading the observations evenly between two endpoints.
```

```
# (Note: This design usually cannot work with environmental monitoring data taken
# over time since usually observations taken close together in time are not
# independent.)

linearTrendTestPower(x = 1:10, slope.over.sigma = 0.1)
#[1] 0.1265976


linearTrendTestPower(x = c(rep(1, 5), rep(10, 5)), slope.over.sigma = 0.1)
#[1] 0.2413823

#==========

# Clean up
#---------
rm(power)
```

`linearTrendTestScaledMds` *Scaled Minimal Detectable Slope for a t-Test for Linear Trend*

Description

Compute the scaled minimal detectable slope associated with a t-test for liner trend, given the sample size or predictor variable values, power, and significance level.

Usage

```
linearTrendTestScaledMds(n, x = lapply(n, seq), alpha = 0.05, power = 0.95,
  alternative = "two.sided", two.sided.direction = "greater", approx = FALSE,
  tol = 1e-07, maxiter = 1000)
```

Arguments

`n`	numeric vector of sample sizes. All values of `n` must be positive integers larger than 2. This argument is ignored when `x` is supplied. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`x`	numeric vector of predictor variable values, or a list in which each component is a numeric vector of predictor variable values. Usually, the predictor variable is time (e.g., days, months, quarters, etc.). The default value is `x=lapply(n,seq)`, which yields a list in which the i'th component is the seqence of integers from 1 to the i'th value of the vector `n`. If `x` is a numeric vector, it must contain at least three elements, two of which must be unique. If `x` is a list of numeric vectors, each component of `x` must contain at least three elements, two of which must be unique. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`alpha`	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.

power numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

two.sided.direction character string indicating the direction (positive or negative) for the scaled minimal detectable slope when alternative="two.sided". When two.sided.direction="greater" (the default), the scaled minimal detectable slope is positive. When two.sided.direction="less", the scaled minimal detectable slope is negative. This argument is ignored if alternative="less" or alternative="greater".

approx logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is approx=FALSE.

tol numeric scalar indicating the toloerance to use in the uniroot search algorithm. The default value is tol=1e-7.

maxiter positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

Details

If the argument x is a vector, it is converted into a list with one component. If the arguments n, x, alpha, and power are not all the same length, they are replicated to be the same length as the length of the longest argument.

Formulas for the power of the t-test of linear trend for specified values of the sample size, scaled slope, and Type I error level are given in the help file for linearTrendTestPower. The function linearTrendTestScaledMds uses the uniroot search algorithm to determine the minimal detectable scaled slope for specified values of the power, sample size, and Type I error level.

Value

numeric vector of computed scaled minimal detectable slopes. When alternative="less", or alternative="two.sided" and two.sided.direction="less", the computed slopes are negative. Otherwise, the slopes are positive.

Note

See the help file for linearTrendTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for linearTrendTestPower.

See Also

linearTrendTestPower, linearTrendTestN, plotLinearTrendTestDesign, lm, summary.lm, kendallTrendTest, Power and Sample Size, Normal, t.test.

Examples

```
# Look at how the scaled minimal detectable slope for the t-test for linear
# trend increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

scaled.mds <- linearTrendTestScaledMds(n = 10, power = seq(0.5, 0.9, by = 0.1))

round(scaled.mds, 2)
#[1] 0.25 0.28 0.31 0.35 0.41

#----------

# Repeat the last example, but compute the scaled minimal detectable slopes
# based on the approximate power instead of the exact:

scaled.mds <- linearTrendTestScaledMds(n = 10, power = seq(0.5, 0.9, by = 0.1),
  approx = TRUE)

round(scaled.mds, 2)
#[1] 0.25 0.28 0.31 0.35 0.41

#==========

# Look at how the scaled minimal detectable slope for the t-test for linear trend
# decreases with increasing sample size:

seq(10, 50, by = 10)
#[1] 10 20 30 40 50

scaled.mds <- linearTrendTestScaledMds(seq(10, 50, by = 10), alternative = "greater")

round(scaled.mds, 2)
#[1] 0.40 0.13 0.07 0.05 0.03

#==========

# Look at how the scaled minimal detectable slope for the t-test for linear trend
# decreases with increasing values of Type I error:

scaled.mds <- linearTrendTestScaledMds(10, alpha = c(0.001, 0.01, 0.05, 0.1),
  alternative="greater")

round(scaled.mds, 2)
#[1] 0.76 0.53 0.40 0.34
```

```
#----------

# Repeat the last example, but compute the scaled minimal detectable slopes
# based on the approximate power instead of the exact:

scaled.mds <- linearTrendTestScaledMds(10, alpha = c(0.001, 0.01, 0.05, 0.1),
  alternative="greater", approx = TRUE)

round(scaled.mds, 2)
#[1] 0.70 0.52 0.41 0.36

#==========

# Clean up
#---------
rm(scaled.mds)
```

---

lMoment *Estimate L-Moments*

Description

Estimate the r'th L-moment from a random sample.

Usage

```
lMoment(x, r = 1, method = "unbiased",
  plot.pos.cons = c(a = 0.35, b = 0), na.rm = FALSE)
```

Arguments

x	numeric vector of observations.
r	positive integer specifying the order of the moment.
method	character string specifying what method to use to compute the L-moment. The possible values are "unbiased" (method based on the U-statistic; the default), or "plotting.position" (method based on the plotting position formula). See the DETAILS section for more information.
plot.pos.cons	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when method="plotting.position". The default value is plot.pos.cons=c(a=0.35, b=0). If this vector has a names attribute with the value c("a","b") or c("b","a"), then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name "a" and the second element to the name "b". See the DETAILS section for more information. This argument is ignored if method="ubiased".
na.rm	logical scalar indicating whether to remove missing values from x. If na.rm=FALSE (the default) and x contains missing values, then a missing value (NA) is returned. If na.rm=TRUE, missing values are removed from x prior to computing the L-moment.

Details

Definitions: L-Moments and L-Moment Ratios

The definition of an L-moment given by Hosking (1990) is as follows. Let X denote a random variable with cdf F, and let $x(p)$ denote the p'th quantile of the distribution. Furthermore, let

$$x_{1:n} \leq x_{2:n} \leq \ldots \leq x_{n:n}$$

denote the order statistics of a random sample of size n drawn from the distribution of X. Then the r'th L-moment is given by:

$$\lambda_r = \frac{1}{r} \sum_{k=0}^{r-1} (-1)^k \binom{r-1}{k} E[X_{r-k:r}]$$

for $r = 1, 2, \ldots$.

Hosking (1990) shows that the above equation can be rewritten as:

$$\lambda_r = \int_0^1 x(u) P^*_{r-1}(u) du$$

where

$$P^*_r(u) = \sum_{k=0}^{r} p^*_{r,k} u^k$$

$$p^*_{r,k} = (-1)^{r-k} \binom{r}{k} \binom{r+k}{k} = \frac{(-1)^{r-k}(r+k)!}{(k!)^2 (r-k)!}$$

The first four L-moments are given by:

$$\lambda_1 = E[X]$$

$$\lambda_2 = \frac{1}{2} E[X_{2:2} - X_{1:2}]$$

$$\lambda_3 = \frac{1}{3} E[X_{3:3} - 2X_{2:3} + X_{1:3}]$$

$$\lambda_4 = \frac{1}{4} E[X_{4:4} - 3X_{3:4} + 3X_{2:4} - X_{1:4}]$$

Thus, the first L-moment is a measure of location, and the second L-moment is a measure of scale. Hosking (1990) defines the L-moment ratios of X to be:

$$\tau_r = \frac{\lambda_r}{\lambda_2}$$

for $r = 2, 3, \ldots$. He shows that for a non-degenerate random variable with a finite mean, these quantities lie in the interval $(-1, 1)$. The quantity

$$\tau_3 = \frac{\lambda_3}{\lambda_2}$$

is the L-moment analog of the coefficient of skewness, and the quantity

$$\tau_4 = \frac{\lambda_4}{\lambda_2}$$

is the L-moment analog of the coefficient of kurtosis. Hosking (1990) also defines an L-moment analog of the coefficient of variation (denoted the L-CV) as:

$$\lambda = \frac{\lambda_2}{\lambda_1}$$

He shows that for a positive-valued random variable, the L-CV lies in the interval $(0, 1)$.

Relationship Between L-Moments and Probability-Weighted Moments
Hosking (1990) and Hosking and Wallis (1995) show that L-moments can be written as linear combinations of probability-weighted moments:

$$\lambda_r = (-1)^{r-1} \sum_{k=0}^{r-1} p^*_{r-1,k} \alpha_k = \sum_{j=0}^{r-1} p^*_{r-1,j} \beta_j$$

where

$$\alpha_k = M(1, 0, k) = \frac{1}{k+1} E[X_{1:k+1}]$$

$$\beta_j = M(1, j, 0) = \frac{1}{j+1} E[X_{j+1:j+1}]$$

See the help file for `pwMoment` for more information on probability-weighted moments.

Estimating L-Moments
The two commonly used methods for estimating L-moments are the "unbiased" method based on U-statistics (Hoeffding, 1948; Lehmann, 1975, pp. 362-371), and the "plotting-position" method. Hosking and Wallis (1995) recommend using the unbiased method for almost all applications.

Unbiased Estimators (`method="unbiased"`)
Using the relationship between L-moments and probability-weighted moments explained above, the unbiased estimator of the r'th L-moment is based on unbiased estimators of probability-weighted moments and is given by:

$$l_r = (-1)^{r-1} \sum_{k=0}^{r-1} p^*_{r-1,k} a_k = \sum_{j=0}^{r-1} p^*_{r-1,j} b_j$$

where

$$a_k = \frac{1}{n} \sum_{i=1}^{n-k} x_{i:n} \frac{\binom{n-i}{k}}{\binom{n-1}{k}}$$

$$b_j = \frac{1}{n} \sum_{i=j+1}^{n} x_{i:n} \frac{\binom{i-1}{j}}{\binom{n-1}{j}}$$

Plotting-Position Estimators (`method="plotting.position"`)
Using the relationship between L-moments and probability-weighted moments explained above, the plotting-position estimator of the r'th L-moment is based on the plotting-position estimators of probability-weighted moments and is given by:

$$\tilde{\lambda}_r = (-1)^{r-1} \sum_{k=0}^{r-1} p^*_{r-1,k} \tilde{\alpha}_k = \sum_{j=0}^{r-1} p^*_{r-1,j} \tilde{\beta}_j$$

where

$$\tilde{\alpha}_k = \frac{1}{n}\sum_{i=1}^{n}(1-p_{i:n})^k x_{i:n}$$

$$\tilde{\beta}_j = \frac{1}{n}\sum_{i=1}^{n}p_{i:n}^j x_{i:n}$$

and

$$p_{i:n} = \hat{F}(x_{i:n})$$

denotes the plotting position of the i'th order statistic in the random sample of size n, that is, a distribution-free estimate of the cdf of X evaluated at the i'th order statistic. Typically, plotting positions have the form:

$$p_{i:n} = \frac{i-a}{n+b}$$

where $b > -a > -1$. For this form of plotting position, the plotting-position estimators are asymptotically equivalent to their unbiased estimator counterparts.

Estimating L-Moment Ratios

L-moment ratios are estimated by simply replacing the population L-moments with the estimated L-moments. The estimated ratios based on the unbiased estimators are given by:

$$t_r = \frac{l_r}{l_2}$$

and the estimated ratios based on the plotting-position estimators are given by:

$$\tilde{\tau}_r = \frac{\tilde{\lambda}_r}{\tilde{\lambda}_2}$$

In particular, the L-moment skew is estimated by:

$$t_3 = \frac{l_3}{l_2}$$

or

$$\tilde{\tau}_3 = \frac{\tilde{\lambda}_3}{\tilde{\lambda}_2}$$

and the L-moment kurtosis is estimated by:

$$t_4 = \frac{l_4}{l_2}$$

or

$$\tilde{\tau}_4 = \frac{\tilde{\lambda}_4}{\tilde{\lambda}_2}$$

Similarly, the L-moment coefficient of variation can be estimated using the unbiased L-moment estimators:

$$l = \frac{l_2}{l_1}$$

or using the plotting-position L-moment estimators:

$$\tilde{\lambda} = \frac{\tilde{\lambda}_2}{\tilde{\lambda}_1}$$

Value

A numeric scalar–the value of the r'th L-moment as defined by Hosking (1990).

Note

Hosking (1990) introduced the idea of L-moments, which are expectations of certain linear combinations of order statistics, as the basis of a general theory of describing theoretical probability distributions, computing summary statistics from observed data, estimating distribution parameters and quantiles, and performing hypothesis tests. The theory of L-moments parallels the theory of conventional moments. L-moments have several advantages over conventional moments, including:

- L-moments can characterize a wider range of distributions because they always exist as long as the distribution has a finite mean.
- L-moments are estimated by linear combinations of order statistics, so estimators based on L-moments are more robust to the presence of outliers than estimators based on conventional moments.
- Based on the author's and others' experience, L-moment estimators are less biased and approximate their asymptotic distribution more closely in finite samples than estimators based on conventional moments.
- L-moment estimators are sometimes more efficient (smaller RMSE) than even maximum likelihood estimators for small samples.

Hosking (1990) presents a table with formulas for the L-moments of common probability distributions. Articles that illustrate the use of L-moments include Fill and Stedinger (1995), Hosking and Wallis (1995), and Vogel and Fennessey (1993).

Hosking (1990) and Hosking and Wallis (1995) show the relationship between probabiity-weighted moments and L-moments.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Fill, H.D., and J.R. Stedinger. (1995). L Moment and Probability Plot Correlation Coefficient Goodness-of-Fit Tests for the Gumbel Distribution and Impact of Autocorrelation. *Water Resources Research* **31**(1), 225–229.

Hosking, J.R.M. (1990). L-Moments: Analysis and Estimation of Distributions Using Linear Combinations of Order Statistics. *Journal of the Royal Statistical Society, Series B* **52**(1), 105–124.

Hosking, J.R.M., and J.R. Wallis (1995). A Comparison of Unbiased and Plotting-Position Estimators of L Moments. *Water Resources Research* **31**(8), 2019–2025.

Vogel, R.M., and N.M. Fennessey. (1993). L Moment Diagrams Should Replace Product Moment Diagrams. *Water Resources Research* **29**(6), 1745–1752.

See Also

cv, skewness, kurtosis, pwMoment.

Examples

```
# Generate 20 observations from a generalized extreme value distribution
# with parameters location=10, scale=2, and shape=.25, then compute the
# first four L-moments.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rgevd(20, location = 10, scale = 2, shape = 0.25)

lMoment(dat)
#[1] 10.59556

lMoment(dat, 2)
#[1] 1.0014

lMoment(dat, 3)
#[1] 0.1681165

lMoment(dat, 4)
#[1] 0.08732692

#----------

# Now compute some L-moments based on the plotting-position estimators:

lMoment(dat, method = "plotting.position")
#[1] 10.59556

lMoment(dat, 2, method = "plotting.position")
#[1] 1.110264

lMoment(dat, 3, method="plotting.position", plot.pos.cons = c(.325,1))
#[1] -0.4430792

#----------

# Clean up
#---------
rm(dat)
```

Lognormal3 *The Three-Parameter Lognormal Distribution*

Description

Density, distribution function, quantile function, and random generation for the three-parameter lognormal distribution with parameters `meanlog`, `sdlog`, and `threshold`.

Usage

```
dlnorm3(x, meanlog = 0, sdlog = 1, threshold = 0)
plnorm3(q, meanlog = 0, sdlog = 1, threshold = 0)
qlnorm3(p, meanlog = 0, sdlog = 1, threshold = 0)
rlnorm3(n, meanlog = 0, sdlog = 1, threshold = 0)
```

Arguments

`x`	vector of quantiles.
`q`	vector of quantiles.
`p`	vector of probabilities between 0 and 1.
`n`	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
`meanlog`	vector of means of the distribution of the random variable on the log scale. The default is `meanlog=0`.
`sdlog`	vector of (positive) standard deviations of the random variable on the log scale. The default is `sdlog=1`.
`threshold`	vector of thresholds of the random variable on the log scale. The default is `threshold=0`.

Details

The three-parameter lognormal distribution is simply the usual two-parameter lognormal distribution with a location shift.

Let X be a random variable with a three-parameter lognormal distribution with parameters `meanlog`=μ, `sdlog`=σ, and `threshold`=γ. Then the random variable $Y = X - \gamma$ has a lognormal distribution with parameters `meanlog`=μ and `sdlog`=σ. Thus,

- `dlnorm3` calls `dlnorm` using the arguments `x = x - threshold`, `meanlog = meanlog`, `sdlog = sdlog`
- `plnorm3` calls `plnorm` using the arguments `q = q - threshold`, `meanlog = meanlog`, `sdlog = sdlog`
- `qlnorm3` calls `qlnorm` using the arguments `q = q`, `meanlog = meanlog`, `sdlog = sdlog` and then adds the argument `threshold` to the result.
- `rlnorm3` calls `rlnorm` using the arguments `n = n`, `meanlog = meanlog`, `sdlog = sdlog` and then adds the argument `threshold` to the result.

The threshold parameter γ affects only the location of the three-parameter lognormal distribution; it has no effect on the variance or the shape of the distribution.

Denote the mean, variance, and coefficient of variation of $Y = X - \gamma$ by:

$$E(Y) = \theta$$

$$Var(Y) = \eta^2$$

$$CV(Y) = \tau = \eta/\theta$$

Then the mean, variance, and coefficient of variation of X are given by:

$$E(X) = \theta + \eta$$

$$Var(X) = \eta^2$$

$$CV(X) = \frac{\eta}{\theta + \gamma} = \frac{\tau\theta}{\theta + \gamma}$$

The relationships between the parameters μ, σ, θ, η, and τ are as follows:

$$\theta = \beta\sqrt{\omega}$$

$$\eta = \beta\sqrt{\omega(\omega - 1)}$$

$$\tau = \sqrt{\omega - 1}$$

$$\mu = log(\frac{\theta}{\sqrt{\tau^2 + 1}})$$

$$\sigma = \sqrt{log(\tau^2 + 1)}$$

where

$$\beta = e^{\mu}, \omega = exp(\sigma^2)$$

Since quantiles of a distribution are preserved under monotonic transformations, the median of X is:

$$Median(X) = \gamma + \beta$$

Value

`dlnorm3` gives the density, `plnorm3` gives the distribution function, `qlnorm3` gives the quantile function, and `rlnorm3` generates random deviates.

Note

The two-parameter lognormal distribution is the distribution of a random variable whose logarithm is normally distributed. The two major characteristics of the two-parameter lognormal distribution are that it is bounded below at 0, and it is skewed to the right. The three-parameter lognormal distribution is a generalization of the two-parameter lognormal distribution in which the distribution is shifted so that the threshold parameter is some arbitrary number, not necessarily 0.

The three-parameter lognormal distribution was introduced by Wicksell (1917) in a study of the distribution of ages at first marriage. Both the two- and three-parameter lognormal distributions have been used in a variety of fields, including economics and business, industry, biology, ecology, atmospheric science, and geology (Crow and Shimizu, 1988). Royston (1992) has discussed the application of the three-parameter lognormal distribution in the field of medicine.

The two-parameter lognormal distribution is often used to characterize chemical concentrations in the environment. Ott (1990) has shown mathematically how a series of successive random dilutions gives rise to a distribution that can be approximated by a two-parameter lognormal distribution.

The three-pararameter lognormal distribution starts to resemble a normal distribution as the parameter σ (the standard deviation of $log(X - \gamma)$ tends to 0.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special references to its uses in economics)*. Cambridge University Press, London, 176pp.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, 387pp.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Ott, W.R. (1990). A Physical Explanation of the Lognormality of Pollutant Concentrations. *Journal of the Air and Waste Management Association* **40**, 1378–1383.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL, Chapter 9.

Royston, J.P. (1992b). Estimation, Reference Ranges and Goodness of Fit for the Three-Parameter Log-Normal Distribution. *Statistics in Medicine* **11**, 897–912.

Wicksell, S.D. (1917). On Logarithmic Correlation with an Application to the Distribution of Ages at First Marriage. *Medd. Lunds. Astr. Obs.* **84**, 1–21.

See Also

Lognormal, `elnorm3`, Probability Distributions and Random Numbers.

Examples

```
# Density of the three-parameter lognormal distribution with
# parameters meanlog=1, sdlog=2, and threshold=10, evaluated at 10.5:

dlnorm3(10.5, 1, 2, 10)
#[1] 0.278794

#----------

# The cdf of the three-parameter lognormal distribution with
# parameters meanlog=2, sdlog=3, and threshold=5, evaluated at 9:

plnorm3(9, 2, 3, 5)
#[1] 0.4189546

#----------

# The median of the three-parameter lognormal distribution with
# parameters meanlog=2, sdlog=3, and threshold=20:

qlnorm3(0.5, 2, 3, 20)
```

```
#[1] 27.38906

#----------

# Random sample of 3 observations from the three-parameter lognormal
# distribution with parameters meanlog=2, sdlog=1, and threshold=-5.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rlnorm3(3, 2, 1, -5)
#[1] 18.6339749 -0.8873173 39.0561521
```

LognormalAlt *The Lognormal Distribution (Alternative Parameterization)*

Description

Density, distribution function, quantile function, and random generation for the lognormal distribution with parameters mean and cv.

Usage

```
dlnormAlt(x, mean = exp(1/2), cv = sqrt(exp(1) - 1), log = FALSE)
plnormAlt(q, mean = exp(1/2), cv = sqrt(exp(1) - 1),
    lower.tail = TRUE, log.p = FALSE)
qlnormAlt(p, mean = exp(1/2), cv = sqrt(exp(1) - 1),
    lower.tail = TRUE, log.p = FALSE)
rlnormAlt(n, mean = exp(1/2), cv = sqrt(exp(1) - 1))
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
mean	vector of (positive) means of the distribution of the random variable.
cv	vector of (positive) coefficients of variation of the random variable.
log, log.p	logical; if TRUE, probabilities/densities p are returned as $log(p)$.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Details

Let X be a random variable with a lognormal distribution with parameters `meanlog`=μ and `sdlog`=σ. That is, μ and σ denote the mean and standard deviation of the random variable on the log scale. The relationship between these parameters and the mean (`mean`=θ) and coefficient of variation (`cv`=τ) of the distribution on the original scale is given by:

$$\mu = log(\frac{\theta}{\sqrt{\tau^2 + 1}}) \quad (1)$$

$$\sigma = [log(\tau^2 + 1)]^{1/2} \quad (2)$$

$$\theta = exp[\mu + (\sigma^2/2)] \quad (3)$$

$$\tau = [exp(\sigma^2) - 1]^{1/2} \quad (4)$$

Thus, the functions `dlnormAlt`, `plnormAlt`, `qlnormAlt`, and `rlnormAlt` call the R functions `dlnorm`, `plnorm`, `qlnorm`, and `rlnorm`, respectively using the following values for the `meanlog` and `sdlog` parameters:

```
sdlog <- sqrt(log(1 + cv^2)),
meanlog <- log(mean) - (sdlog^2)/2
```

Value

`dlnormAlt` gives the density, `plnormAlt` gives the distribution function, `qlnormAlt` gives the quantile function, and `rlnormAlt` generates random deviates.

Note

The two-parameter lognormal distribution is the distribution of a random variable whose logarithm is normally distributed. The two major characteristics of the lognormal distribution are that it is bounded below at 0, and it is skewed to the right.

Because the empirical distribution of many variables is inherently positive and skewed to the right (e.g., size of organisms, amount of rainfall, size of income, etc.), the lognormal distribution has been widely applied in several fields, including economics, business, industry, biology, ecology, atmospheric science, and geology (Aitchison and Brown, 1957; Crow and Shimizu, 1988).

Gibrat (1930) derived the lognormal distribution from theoretical assumptions, calling it the "law of proportionate effect", but Kapteyn (1903) had described a machine that was the mechanical equivalent. The basic idea is that the Central Limit Theorem states that the distribution of the sum of several independent random variables tends to look like a normal distribution, no matter what the underlying distribution(s) of the original random variables, hence the product of several independent random variables tends to look like a lognormal distribution.

The lognormal distribution is often used to characterize chemical concentrations in the environment. Ott (1990) has shown mathematically how a series of successive random dilutions gives rise to a distribution that can be approximated by a lognormal distribution.

A lognormal distribution starts to resemble a normal distribution as the parameter σ (the standard deviation of the log of the distribution) tends to 0.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) discourage using the assumption of a lognormal distribution for some types of environmental data and recommend instead assessing whether the data appear to fit a gamma distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Limpert, E., W.A. Stahel, and M. Abbt. (2001). Log-Normal Distributions Across the Sciences: Keys and Clues. *BioScience* **51**, 341–352.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

See Also

Lognormal, `elnormAlt`, Probability Distributions and Random Numbers.

Examples

```
# Density of the lognormal distribution with parameters
# mean=10 and cv=1, evaluated at 5:

dlnormAlt(5, mean = 10, cv = 1)
#[1] 0.08788173

#----------

# The cdf of the lognormal distribution with parameters mean=2 and cv=3,
# evaluated at 4:

plnormAlt(4, 2, 3)
#[1] 0.8879132

#----------

# The median of the lognormal distribution with parameters
# mean=10 and cv=1:

qlnormAlt(0.5, mean = 10, cv = 1)
#[1] 7.071068

#----------
```

```
# Random sample of 3 observations from a lognormal distribution with
# parameters mean=10 and cv=1.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rlnormAlt(3, mean = 10, cv = 1)
#[1] 18.615797  4.341402 31.265293
```

LognormalMix *Mixture of Two Lognormal Distributions*

Description

Density, distribution function, quantile function, and random generation for a mixture of two lognormal distribution with parameters meanlog1, sdlog1, meanlog2, sdlog2, and p.mix.

Usage

```
dlnormMix(x, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)
plnormMix(q, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)
qlnormMix(p, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)
rlnormMix(n, meanlog1 = 0, sdlog1 = 1, meanlog2 = 0, sdlog2 = 1, p.mix = 0.5)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
meanlog1	vector of means of the first lognormal random variable on the log scale. The default is meanlog1=0.
sdlog1	vector of standard deviations of the first lognormal random variable on the log scale. The default is sdlog1=1.
meanlog2	vector of means of the second lognormal random variable on the log scale. The default is meanlog2=0.
sdlog2	vector of standard deviations of the second lognormal random variable on the log scale. The default is sdlog2=1.
p.mix	vector of probabilities between 0 and 1 indicating the mixing proportion. For rlnormMix this must be a single, non-missing number.

Details

Let $f(x; \mu, \sigma)$ denote the density of a lognormal random variable with parameters meanlog=μ and sdlog=σ. The density, g, of a lognormal mixture random variable with parameters meanlog1=μ_1, sdlog1=σ_1, meanlog2=μ_2, sdlog2=σ_2, and p.mix=p is given by:

$$g(x; \mu_1, \sigma_1, \mu_2, \sigma_2, p) = (1 - p) f(x; \mu_1, \sigma_1) + p f(x; \mu_2, \sigma_2)$$

Value

`dlnormMix` gives the density, `plnormMix` gives the distribution function, `qlnormMix` gives the quantile function, and `rlnormMix` generates random deviates.

Note

A lognormal mixture distribution is often used to model positive-valued data that appear to be "contaminated"; that is, most of the values appear to come from a single lognormal distribution, but a few "outliers" are apparent. In this case, the value of `meanlog2` would be larger than the value of `meanlog1`, and the mixing proportion `p.mix` would be fairly close to 0 (e.g., `p.mix=0.1`). The value of the second standard deviation (`sdlog2`) may or may not be the same as the value for the first (`sdlog1`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135-146.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, pp.53-54, and Chapter 8.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

Lognormal, NormalMix, Probability Distributions and Random Numbers.

Examples

```
# Density of a lognormal mixture with parameters meanlog1=0, sdlog1=1,
# meanlog2=2, sdlog2=3, p.mix=0.5, evaluated at 1.5:

dlnormMix(1.5, meanlog1 = 0, sdlog1 = 1, meanlog2 = 2, sdlog2 = 3, p.mix = 0.5)
#[1] 0.1609746

#----------

# The cdf of a lognormal mixture with parameters meanlog1=0, sdlog1=1,
# meanlog2=2, sdlog2=3, p.mix=0.2, evaluated at 4:

plnormMix(4, 0, 1, 2, 3, 0.2)
#[1] 0.8175281

#----------

# The median of a lognormal mixture with parameters meanlog1=0, sdlog1=1,
# meanlog2=2, sdlog2=3, p.mix=0.2:
```

```
qlnormMix(0.5, 0, 1, 2, 3, 0.2)
#[1] 1.156891

#----------

# Random sample of 3 observations from a lognormal mixture with
# parameters meanlog1=0, sdlog1=1, meanlog2=3, sdlog2=4, p.mix=0.2.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rlnormMix(3, 0, 1, 2, 3, 0.2)
#[1] 0.08975283 1.07591103 7.85482514
```

`LognormalMixAlt` *Mixture of Two Lognormal Distributions (Alternative Parameterization)*

Description

Density, distribution function, quantile function, and random generation for a mixture of two lognormal distribution with parameters `mean1`, `cv1`, `mean2`, `cv2`, and `p.mix`.

Usage

```
dlnormMixAlt(x, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
    mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
plnormMixAlt(q, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
    mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
qlnormMixAlt(p, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
    mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
rlnormMixAlt(n, mean1 = exp(1/2), cv1 = sqrt(exp(1) - 1),
    mean2 = exp(1/2), cv2 = sqrt(exp(1) - 1), p.mix = 0.5)
```

Arguments

`x`	vector of quantiles.
`q`	vector of quantiles.
`p`	vector of probabilities between 0 and 1.
`n`	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
`mean1`	vector of means of the first lognormal random variable. The default is `meanlog1=sqrt(exp(1) - 1)`.
`cv1`	vector of coefficient of variations of the first lognormal random variable. The default is `sdlog1=sqrt(exp(1) - 1)`.
`mean2`	vector of means of the second lognormal random variable. The default is `mean2=sqrt(exp(1) - 1)`.

cv2	vector of coefficient of variations of the second lognormal random variable. The default is sdlog2=sqrt(exp(1) - 1).
p.mix	vector of probabilities between 0 and 1 indicating the mixing proportion. For rlnormMixAlt this must be a single, non-missing number.

Details

Let $f(x; \eta, \theta)$ denote the density of a lognormal random variable with parameters mean=η and cv=θ. The density, g, of a lognormal mixture random variable with parameters mean1=η_1, cv1=θ_1, mean2=η_2, cv2=θ_2, and p.mix=p is given by:

$$g(x; \eta_1, \theta_1, \eta_2, \theta_2, p) = (1 - p) f(x; \eta_1, \theta_1) + p f(x; \eta_2, \theta_2)$$

The default values for mean1 and cv1 correspond to a lognormal distribution with parameters meanlog=0 and sdlog=1. Similarly for the default values of mean2 and cv2.

Value

dlnormMixAlt gives the density, plnormMixAlt gives the distribution function, qlnormMixAlt gives the quantile function, and rlnormMixAlt generates random deviates.

Note

A lognormal mixture distribution is often used to model positive-valued data that appear to be "contaminated"; that is, most of the values appear to come from a single lognormal distribution, but a few "outliers" are apparent. In this case, the value of mean2 would be larger than the value of mean1, and the mixing proportion p.mix would be fairly close to 0 (e.g., p.mix=0.1).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135-146.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, pp.53-54, and Chapter 8.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

LognormalAlt, LognormalMix, Lognormal, NormalMix, Probability Distributions and Random Numbers.

Examples

```
# Density of a lognormal mixture with parameters mean=2, cv1=3,
# mean2=4, cv2=5, p.mix=0.5, evaluated at 1.5:

dlnormMixAlt(1.5, mean1 = 2, cv1 = 3, mean2 = 4, cv2 = 5, p.mix = 0.5)
#[1] 0.1436045

#----------

# The cdf of a lognormal mixture with parameters mean=2, cv1=3,
# mean2=4, cv2=5, p.mix=0.5, evaluated at 1.5:

plnormMixAlt(1.5, mean1 = 2, cv1 = 3, mean2 = 4, cv2 = 5, p.mix = 0.5)
#[1] 0.6778064

#----------

# The median of a lognormal mixture with parameters mean=2, cv1=3,
# mean2=4, cv2=5, p.mix=0.5:

qlnormMixAlt(0.5, 2, 3, 4, 5, 0.5)
#[1] 0.6978355

#----------

# Random sample of 3 observations from a lognormal mixture with
# parameters mean1=2, cv1=3, mean2=4, cv2=5, p.mix=0.5.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rlnormMixAlt(3, 2, 3, 4, 5, 0.5)
#[1]  0.70672151 14.43226313  0.05521329
```

LognormalTrunc *The Truncated Lognormal Distribution*

Description

Density, distribution function, quantile function, and random generation for the truncated lognormal distribution with parameters `meanlog`, `sdlog`, `min`, and `max`.

Usage

```
dlnormTrunc(x, meanlog = 0, sdlog = 1, min = 0, max = Inf)
plnormTrunc(q, meanlog = 0, sdlog = 1, min = 0, max = Inf)
qlnormTrunc(p, meanlog = 0, sdlog = 1, min = 0, max = Inf)
rlnormTrunc(n, meanlog = 0, sdlog = 1, min = 0, max = Inf)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
meanlog	vector of means of the distribution of the non-truncated random variable on the log scale. The default is meanlog=0.
sdlog	vector of (positive) standard deviations of the non-truncated random variable on the log scale. The default is sdlog=1.
min	vector of minimum values for truncation on the left. The default value is min=0.
max	vector of maximum values for truncation on the right. The default value is max=Inf.

Details

See the help file for the lognormal distribution for information about the density and cdf of a log-normal distribution.

Probability Density and Cumulative Distribution Function
Let X denote a random variable with density function $f(x)$ and cumulative distribution function $F(x)$, and let Y denote the truncated version of X where Y is truncated below at min=A and above atmax=B. Then the density function of Y, denoted $g(y)$, is given by:

$$g(y) = fracf(y)F(B) - F(A), A \le y \le B$$

and the cdf of Y, denoted $G(y)$, is given by:

$$G(y) = \begin{array}{ll} 0 & \text{for } y < A \\ \frac{F(y)-F(A)}{F(B)-F(A)} & \text{for } A \le y \le B \\ 1 & \text{for } y > B \end{array}$$

Quantiles
The p^{th} quantile y_p of Y is given by:

$$y_p = \begin{array}{ll} A & \text{for } p = 0 \\ F^{-1}\{p[F(B) - F(A)] + F(A)\} & \text{for } 0 < p < 1 \\ B & \text{for } p = 1 \end{array}$$

Random Numbers
Random numbers are generated using the inverse transformation method:

$$y = G^{-1}(u)$$

where u is a random deviate from a uniform $[0, 1]$ distribution.

Value

`dlnormTrunc` gives the density, `plnormTrunc` gives the distribution function, `qlnormTrunc` gives the quantile function, and `rlnormTrunc` generates random deviates.

Note

A truncated lognormal distribution is sometimes used as an input distribution for probabilistic risk assessment.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Schneider, H. (1986). *Truncated and Censored Samples from Normal Populations*. Marcel Dekker, New York, Chapter 2.

See Also

Lognormal, Probability Distributions and Random Numbers.

Examples

```
# Density of a truncated lognormal distribution with parameters
# meanlog=1, sdlog=0.75, min=0, max=10, evaluated at 2 and 4:

dlnormTrunc(c(2, 4), 1, 0.75, 0, 10)
#[1] 0.2551219 0.1214676

#----------

# The cdf of a truncated lognormal distribution with parameters
# meanlog=1, sdlog=0.75, min=0, max=10, evaluated at 2 and 4:

plnormTrunc(c(2, 4), 1, 0.75, 0, 10)
#[1] 0.3558867 0.7266934

#----------

# The median of a truncated lognormal distribution with parameters
# meanlog=1, sdlog=0.75, min=0, max=10:

qlnormTrunc(.5, 1, 0.75, 0, 10)
#[1] 2.614945
```

```
#----------

# A random sample of 3 observations from a truncated lognormal distribution
# with parameters meanlog=1, sdlog=0.75, min=0, max=10.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rlnormTrunc(3, 1, 0.75, 0, 10)
#[1] 5.754805 4.372218 1.706815
```

LognormalTruncAlt *The Truncated Lognormal Distribution (Alternative Parameterization)*

Description

Density, distribution function, quantile function, and random generation for the truncated lognormal distribution with parameters mean, cv, min, and max.

Usage

```
dlnormTruncAlt(x, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)
plnormTruncAlt(q, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)
qlnormTruncAlt(p, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)
rlnormTruncAlt(n, mean = exp(1/2), cv = sqrt(exp(1) - 1), min = 0, max = Inf)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
mean	vector of means of the distribution of the non-truncated random variable. The default is mean=exp(1/2).
cv	vector of (positive) coefficient of variations of the non-truncated random variable. The default is cv=sqrt(exp(1)-1).
min	vector of minimum values for truncation on the left. The default value is min=0.
max	vector of maximum values for truncation on the right. The default value is max=Inf.

Details

See the help file for LognormalAlt for information about the density and cdf of a lognormal distribution with this alternative parameterization.

Let X denote a random variable with density function $f(x)$ and cumulative distribution function $F(x)$, and let Y denote the truncated version of X where Y is truncated below at `min`=A and above at`max`=B. Then the density function of Y, denoted $g(y)$, is given by:

$$g(y) = fracf(y)F(B) - F(A), A \le y \le B$$

and the cdf of Y, denoted $G(y)$, is given by:

$$G(y) = \begin{cases} 0 & \text{for } y < A \\ \frac{F(y)-F(A)}{F(B)-F(A)} & \text{for } A \le y \le B \\ 1 & \text{for } y > B \end{cases}$$

The p^{th} quantile y_p of Y is given by:

$$y_p = \begin{cases} A & \text{for } p = 0 \\ F^{-1}\{p[F(B) - F(A)] + F(A)\} & \text{for } 0 < p < 1 \\ B & \text{for } p = 1 \end{cases}$$

Random numbers are generated using the inverse transformation method:

$$y = G^{-1}(u)$$

where u is a random deviate from a uniform $[0, 1]$ distribution.

Value

`dlnormTruncAlt` gives the density, `plnormTruncAlt` gives the distribution function, `qlnormTruncAlt` gives the quantile function, and `rlnormTruncAlt` generates random deviates.

Note

A truncated lognormal distribution is sometimes used as an input distribution for probabilistic risk assessment.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Schneider, H. (1986). *Truncated and Censored Samples from Normal Populations*. Marcel Dekker, New York, Chapter 2.

See Also

LognormalAlt, Probability Distributions and Random Numbers.

Examples

```
# Density of a truncated lognormal distribution with parameters
# mean=10, cv=1, min=0, max=20, evaluated at 2 and 12:

dlnormTruncAlt(c(2, 12), 10, 1, 0, 20)
#[1] 0.08480874 0.03649884

#----------

# The cdf of a truncated lognormal distribution with parameters
# mean=10, cv=1, min=0, max=20, evaluated at 2 and 12:

plnormTruncAlt(c(2, 4), 10, 1, 0, 20)
#[1] 0.07230627 0.82467603

#----------

# The median of a truncated lognormal distribution with parameters
# mean=10, cv=1, min=0, max=20:

qlnormTruncAlt(.5, 10, 1, 0, 20)
#[1] 6.329505

#----------

# A random sample of 3 observations from a truncated lognormal distribution
# with parameters mean=10, cv=1, min=0, max=20.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rlnormTruncAlt(3, 10, 1, 0, 20)
#[1]  6.685391 17.445387 18.543553
```

---

longToWide *Convert a Long Format Data Set into a Wide Format*

Description

Given a data frame or matrix in long format, convert it to wide format based on the levels of two variables in the data frame. This is a simplified version of the R function `reshape` with the argument `direction="wide"`.

Usage

```
longToWide(x, data.var, row.var, col.var,
  row.labels = levels(factor(x[, row.var])),
  col.labels = levels(factor(x[, col.var])),
  paste.row.name = FALSE, paste.col.name = FALSE, sep = ".",
  check.names = FALSE, ...)
```

Arguments

x	data frame or matrix to convert to wide format. Must have at least 3 columns corresponding to the data variable, row variable, and column variable, respectively.
data.var	character string or numeric scalar indicating column variable name in x for data values.
row.var	character string or numeric scalar indicating column variable name in x for defining rows of output. The indicated column in x cannot have missing values.
col.var	character string or numeric scalar indicating column variable name in x for defining columns of output. The indicated column in x cannot have missing values.
row.labels	optional character vector indicating labels to use for rows. The default value is the levels of the variable indicated by row.var when coerced to a factor.
col.labels	optional character vector indicating labels to use for columns. The default value is the levels of the variable indicated by col.var when coerced to a factor.
paste.row.name	logical scalar indicating whether to paste the name of the variable used to define the row names (i.e., the value of row.var) in front of the values defining the row names. The default value is paste.row.name=FALSE.
paste.col.name	logical scalar indicating whether to paste the name of the variable used to define the column names (i.e., the value of col.var) in front of the values defining the column names. The default value is paste.col.name=FALSE.
sep	character string separator used when paste.row.name=TRUE and/or paste.col.name=TRUE. The default value is sep=".".
check.names	argument to data.frame. Used to convert the return value to a data frame when the argument x is a data frame. This argument is ignored if x is a matrix.
...	other arguments to data.frame. This argument is ignored if x is a matrix.

Details

The combination of values in x[, row.var] and x[, col.var] must yield n unique values, where n is the number of rows in x.

Value

longToWide returns a matrix when x is a matrix and a data frame when x is a data frame. The number of rows is equal to the number of unique values in x[, row.var] and the number of columns is equal to the number of unique values in x[, col.var].

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>), based on a template from Phil Dixon.

See Also

reshape, data.frame, matrix.

Examples

```
EPA.09.Ex.10.1.nickel.df
#   Month   Well Nickel.ppb
#1      1 Well.1       58.8
#2      3 Well.1        1.0
#3      6 Well.1      262.0
#4      8 Well.1       56.0
#5     10 Well.1        8.7
#6      1 Well.2       19.0
#7      3 Well.2       81.5
#8      6 Well.2      331.0
#9      8 Well.2       14.0
#10    10 Well.2       64.4
#11     1 Well.3       39.0
#12     3 Well.3      151.0
#13     6 Well.3       27.0
#14     8 Well.3       21.4
#15    10 Well.3      578.0
#16     1 Well.4        3.1
#17     3 Well.4      942.0
#18     6 Well.4       85.6
#19     8 Well.4       10.0
#20    10 Well.4      637.0

longToWide(EPA.09.Ex.10.1.nickel.df,
  "Nickel.ppb", "Month", "Well", paste.row.name = TRUE)
#         Well.1 Well.2 Well.3 Well.4
#Month.1    58.8   19.0   39.0    3.1
#Month.3     1.0   81.5  151.0  942.0
#Month.6   262.0  331.0   27.0   85.6
#Month.8    56.0   14.0   21.4   10.0
#Month.10    8.7   64.4  578.0  637.0
```

Millard.Deverel.88.df *Copper and Zinc Concentrations in Shallow Ground Water*

Description

Copper and zinc concentrations (mg/L) in shallow ground water from two geological zones (Alluvial Fan and Basin-Trough) in the San Joaquin Valley, CA. There are 68 samples from the Alluvial Fan zone and 50 from the Basin-Trough zone. Some observations are reported as $<DL$, where DL

denotes a detection limit. There are multiple detection limits for both the copper and zinc data in each of the geological zones.

Usage

```
Millard.Deverel.88.df
```

Format

A data frame with 118 observations on the following 8 variables.

`Cu.orig` a character vector of original copper concentrations (mg/L)

`Cu` a numeric vector of copper concentrations with nondetects coded to their detection limit

`Cu.censored` a logical vector indicating which copper concentrations are censored

`Zn.orig` a character vector of original zinc concentrations (mg/L)

`Zn` a numeric vector of zinc concentrations with nondetects coded to their detection limit

`Zn.censored` a logical vector indicating which zinc concentrations are censored

`Zone` a factor indicating the zone (alluvial fan vs. basin trough)

`Location` a numeric vector indicating the sampling location

Source

Millard, S.P., and S.J. Deverel. (1988). Nonparametric Statistical Methods for Comparing Two Sites Based on Data With Multiple Nondetect Limits. *Water Resources Research*, **24**(12), 2087-2098.

References

Deverel, S.J., R.J. Gilliom, R. Fujii, J.A. Izbicki, and J.C. Fields. (1984). *Areal Distribution of Selenium and Other Inorganic Constituents in Shallow Ground Water of the San Luis Drain Service Area, San Joaquin, California: A Preliminary Study*. U.S. Geological Survey Water Resources Investigative Report 84-4319.

`Modified.TcCB.df` *Modified 1,2,3,4-Tetrachlorobenzene Data with Censored Values*

Description

Artificial 1,2,3,4-Tetrachlorobenzene (TcCB) concentrations with censored values; based on the reference area data stored in `EPA.94b.tccb.df`. The data frame `EPA.94b.tccb.df` contains TcCB concentrations (ppb) in soil samples at a reference area and a cleanup area. The data frame `Modified.TcCB.df` contains a modified version of the data from the reference area. For this data set, the concentrations of TcCB less than 0.5 ppb have been recoded as `<0.5`.

Usage

```
Modified.TcCB.df
```

Format

A data frame with 47 observations on the following 3 variables.

`TcCB.orig` a character vector of original TcCB concentrations (ppb)

`TcCB` a numeric vector with censored observations set to their detection level

`Censored` a logical vector indicating which observations are censored

Source

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, p.595.

References

USEPA. (1994b). *Statistical Methods for Evaluating the Attainment of Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid Media*. EPA/230-R-94-004. Office of Policy, Planning, and Evaluation, U.S. Environmental Protection Agency, Washington, D.C.

See Also

`EPA.94b.tccb.df`.

`newsEnvStats` *Show the EnvStats NEWS File*

Description

Show the NEWS file of the **EnvStats** package.

Usage

```
newsEnvStats()
```

Details

The function `newsEnvStats` displays the contents of the **EnvStats** NEWS file in a separate text window. You can also access the NEWS file with the command `news(package="EnvStats")`, which returns the contents of the file to the R command window.

Value

None.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

`news`.

NIOSH.89.air.lead.vec *NIOSH Air Lead Levels Data*

Description

Air lead levels collected by the National Institute for Occupational Safety and Health (NIOSH) at 15 different areas within the Alma American Labs, Fairply, CO, for health hazard evaluation (HETA 89-052) on Februay 23, 1989.

Usage

```
NIOSH.89.air.lead.vec
```

Format

A numeric vector with 15 elements containing air lead concentrations ($\mu g/m^3$).

Source

Krishnamoorthy, K., T. Matthew, and G. Ramachandran. (2006). Generalized P-Values and Confidence Intervals: A Novel Approach for Analyzing Lognormally Distributed Exposure Data. *Journal of Occupational and Environmental Hygiene*, **3**, 642–650.

References

Zou, G.Y., C.Y. Huo, and J. Taleban. (2009). Simple Confidence Intervals for Lognormal Means and their Differences with Environmental Applications. *Environmetrics*, **20**, 172–180.

NormalMix *Mixture of Two Normal Distributions*

Description

Density, distribution function, quantile function, and random generation for a mixture of two normal distribution with parameters `mean1`, `sd1`, `mean2`, `sd2`, and `p.mix`.

Usage

```
dnormMix(x, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
pnormMix(q, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
qnormMix(p, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
rnormMix(n, mean1 = 0, sd1 = 1, mean2 = 0, sd2 = 1, p.mix = 0.5)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
mean1	vector of means of the first normal random variable. The default is mean1=0.
sd1	vector of standard deviations of the first normal random variable. The default is sd1=1.
mean2	vector of means of the second normal random variable. The default is mean2=0.
sd2	vector of standard deviations of the second normal random variable. The default is sd2=1.
p.mix	vector of probabilities between 0 and 1 indicating the mixing proportion. For rnormMix this must be a single, non-missing number.

Details

Let $f(x; \mu, \sigma)$ denote the density of a normal random variable with parameters mean=μ and sd=σ. The density, g, of a normal mixture random variable with parameters mean1=μ_1, sd1=σ_1, mean2=μ_2, sd2=σ_2, and p.mix=p is given by:

$$g(x; \mu_1, \sigma_1, \mu_2, \sigma_2, p) = (1 - p) f(x; \mu_1, \sigma_1) + p f(x; \mu_2, \sigma_2)$$

Value

dnormMix gives the density, pnormMix gives the distribution function, qnormMix gives the quantile function, and rnormMix generates random deviates.

Note

A normal mixture distribution is sometimes used to model data that appear to be "contaminated"; that is, most of the values appear to come from a single normal distribution, but a few "outliers" are apparent. In this case, the value of mean2 would be larger than the value of mean1, and the mixing proportion p.mix would be fairly close to 0 (e.g., p.mix=0.1). The value of the second standard deviation (sd2) may or may not be the same as the value for the first (sd1).

Another application of the normal mixture distribution is to bi-modal data; that is, data exhibiting two modes.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, pp.53-54, and Chapter 8.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

Normal, LognormalMix, Probability Distributions and Random Numbers.

Examples

```
# Density of a normal mixture with parameters mean1=0, sd1=1,
#  mean2=4, sd2=2, p.mix=0.5, evaluated at 1.5:

dnormMix(1.5, mean2=4, sd2=2)
#[1] 0.1104211

#----------

# The cdf of a normal mixture with parameters mean1=10, sd1=2,
# mean2=20, sd2=2, p.mix=0.1, evaluated at 15:

pnormMix(15, 10, 2, 20, 2, 0.1)
#[1] 0.8950323

#----------

# The median of a normal mixture with parameters mean1=10, sd1=2,
# mean2=20, sd2=2, p.mix=0.1:

qnormMix(0.5, 10, 2, 20, 2, 0.1)
#[1] 10.27942

#----------

# Random sample of 3 observations from a normal mixture with
# parameters mean1=0, sd1=1, mean2=4, sd2=2, p.mix=0.5.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rnormMix(3, mean2=4, sd2=2)
#[1] 0.07316778 2.06112801 1.05953620
```

NormalTrunc *The Truncated Normal Distribution*

Description

Density, distribution function, quantile function, and random generation for the truncated normal distribution with parameters `mean`, `sd`, `min`, and `max`.

Usage

```
dnormTrunc(x, mean = 0, sd = 1, min = -Inf, max = Inf)
pnormTrunc(q, mean = 0, sd = 1, min = -Inf, max = Inf)
qnormTrunc(p, mean = 0, sd = 1, min = -Inf, max = Inf)
rnormTrunc(n, mean = 0, sd = 1, min = -Inf, max = Inf)
```

Arguments

x	vector of quantiles.
q	vector of quantiles.
p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
mean	vector of means of the distribution of the non-truncated random variable. The default is mean=0.
sd	vector of (positive) standard deviations of the non-truncated random variable. The default is sd=1.
min	vector of minimum values for truncation on the left. The default value is min=-Inf.
max	vector of maximum values for truncation on the right. The default value is max=Inf.

Details

See the help file for the normal distribution for information about the density and cdf of a normal distribution.

Probability Density and Cumulative Distribution Function
Let X denote a random variable with density function $f(x)$ and cumulative distribution function $F(x)$, and let Y denote the truncated version of X where Y is truncated below at min=A and above atmax=B. Then the density function of Y, denoted $g(y)$, is given by:

$$g(y) = fracf(y)F(B) - F(A), A \leq y \leq B$$

and the cdf of Y, denoted $G(y)$, is given by:

$$G(y) = \begin{array}{ll} 0 & \text{for } y < A \\ \frac{F(y)-F(A)}{F(B)-F(A)} & \text{for } A \leq y \leq B \\ 1 & \text{for } y > B \end{array}$$

Quantiles
The p^{th} quantile y_p of Y is given by:

$$y_p = \begin{array}{ll} A & \text{for } p = 0 \\ F^{-1}\{p[F(B) - F(A)] + F(A)\} & \text{for } 0 < p < 1 \\ B & \text{for } p = 1 \end{array}$$

Random Numbers
Random numbers are generated using the inverse transformation method:

$$y = G^{-1}(u)$$

where u is a random deviate from a uniform $[0, 1]$ distribution.

Mean and Variance

The expected value of a truncated normal random variable with parameters `mean`=μ, `sd`=σ, `min`=A, and `max`=B is given by:

$$E(Y) = \mu + \sigma^2 \frac{f(A) - f(B)}{F(B) - F(A)}$$

(Johnson et al., 1994, p.156; Schneider, 1986, p.17).

The variance of this random variable is given by:

$$\sigma^2 + \sigma^3\{z_A f(A) - z_B f(B) - \sigma[f(A) - f(B)]^2\}$$

where

$$z_A = \frac{A - \mu}{\sigma};\ z_B = \frac{B - \mu}{\sigma}$$

(Johnson et al., 1994, p.158; Schneider, 1986, p.17).

Value

`dnormTrunc` gives the density, `pnormTrunc` gives the distribution function, `qnormTrunc` gives the quantile function, and `rnormTrunc` generates random deviates.

Note

A truncated normal distribution is sometimes used as an input distribution for probabilistic risk assessment.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Schneider, H. (1986). *Truncated and Censored Samples from Normal Populations*. Marcel Dekker, New York, Chapter 2.

See Also

Normal, Probability Distributions and Random Numbers.

Examples

```
# Density of a truncated normal distribution with parameters
# mean=10, sd=2, min=8, max=13, evaluated at 10 and 11.5:

dnormTrunc(c(10, 11.5), 10, 2, 8, 13)
#[1] 0.2575358 0.1943982
```

```
#----------

# The cdf of a truncated normal distribution with parameters
# mean=10, sd=2, min=8, max=13, evaluated at 10 and 11.5:

pnormTrunc(c(10, 11.5), 10, 2, 8, 13)
#[1] 0.4407078 0.7936573

#----------

# The median of a truncated normal distribution with parameters
# mean=10, sd=2, min=8, max=13:

qnormTrunc(.5, 10, 2, 8, 13)
#[1] 10.23074

#----------

# A random sample of 3 observations from a truncated normal distribution
# with parameters mean=10, sd=2, min=8, max=13.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rnormTrunc(3, 10, 2, 8, 13)
#[1] 11.975223 11.373711  9.361258
```

`Olympic.NH4.df` *Ammonium Concentration in Precipitation Measured at Olympic National Park Hoh Ranger Station*

Description

Ammonium (NH_4) concentration (mg/L) in precipitation measured at Olympic National Park, Hoh Ranger Station (WA14), weekly or every other week from January 6, 2009 through December 20, 2011.

Usage

```
Olympic.NH4.df
```

Format

A data frame with 102 observations on the following 6 variables.

`Date.On` Start of collection period. Date on which the sample bucket was installed on the collector.

`Date.Off` End of collection period. Date on which the sample bucket was removed from the collector.

`Week` a numeric vector indicating the cumulative week number starting from January 1, 2009.

`NH4.Orig.mg.per.L` a character vector of the original NH_4 concentrations reported either as the observed value or less than some detection limit. For values reported as less than a detection limit, the value reported is the actual limit of detection or, in the case of a diluted sample, the product of the detection limit value and the dilution factor.

`NH4.mg.per.L` a numeric vector of NH_4 concentrations with non-detects coded to their detection limit.

`Censored` a logical vector indicating which observations are censored.

Details

Station Olympic National Park-Hoh Ranger Station (WA14)

Location Jefferson County, Washington

Latitude 47.8597

Longitude -123.9325

Elevation 182 meters

USGS 1:24000 Map Name Owl Mountain

Operating Agency Olympic National Park

Sponsoring Agency NPS-Air Resources Division

Source

National Atmospheric Deposition Program, National Trends Network (NADP/NTN).
https://nadp.slh.wisc.edu/sites/ntn-WA14/

---

oneSamplePermutationTest

Fisher's One-Sample Randomization (Permutation) Test for Location

---

Description

Perform Fisher's one-sample randomization (permutation) test for location.

Usage

```
oneSamplePermutationTest(x, alternative = "two.sided", mu = 0, exact = FALSE,
  n.permutations = 5000, seed = NULL, ...)
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`alternative`	character string indicating the kind of alternative hypothesis. The possible values are `"two.sided"` (the default), `"less"`, and `"greater"`.

`mu`	numeric scalar indicating the hypothesized value of the mean. The default value is `mu=0`.
`exact`	logical scalar indicating whether to perform the exact permutation test (i.e., enumerate all possible permutations) or simply sample from the permutation distribution. The default value is `exact=FALSE`.
`n.permutations`	integer indicating how many times to sample from the permutation distribution when `exact=FALSE`. The default value is `n.permutations=5000`. This argument is ignored when `exact=TRUE`.
`seed`	positive integer to pass to the R function `set.seed`. The default is `seed=NULL`, in which case the current value of `.Random.seed` is used. Using the `seed` argument lets you reproduce the exact same result if all other arguments stay the same.
`...`	arguments that can be supplied to the `format` function. This argument is used when creating the `names` attribute for the `statistic` component of the returned list (see `permutationTest.object`).

Details

Randomization Tests

In 1935, R.A. Fisher introduced the idea of a ***randomization test*** (Manly, 2007, p. 107; Efron and Tibshirani, 1993, Chapter 15), which is based on trying to answer the question: "Did the observed pattern happen by chance, or does the pattern indicate the null hypothesis is not true?" A randomization test works by simply enumerating all of the possible outcomes under the null hypothesis, then seeing where the observed outcome fits in. A randomization test is also called a ***permutation test***, because it involves permuting the observations during the enumeration procedure (Manly, 2007, p. 3).

In the past, randomization tests have not been used as extensively as they are now because of the "large" computing resources needed to enumerate all of the possible outcomes, especially for large sample sizes. The advent of more powerful personal computers and software has allowed randomization tests to become much easier to perform. Depending on the sample size, however, it may still be too time consuming to enumerate all possible outcomes. In this case, the randomization test can still be performed by sampling from the randomization distribution, and comparing the observed outcome to this sampled permutation distribution.

Fisher's One-Sample Randomization Test for Location

Let $\underline{x} = x_1, x_2, \ldots, x_n$ be a vector of n independent and identically distributed (i.i.d.) observations from some symmetric distribution with mean μ. Consider the test of the null hypothesis that the mean μ is equal to some specified value μ_0:

$$H_0 : \mu = \mu_0 \qquad (1)$$

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`)

$$H_a : \mu > \mu_0 \qquad (2)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : \mu < \mu_0 \qquad (3)$$

and the two-sided alternative

$$H_a : \mu \neq \mu_0 \qquad (4)$$

To perform the test of the null hypothesis (1) versus any of the three alternatives (2)-(4), Fisher proposed using the test statistic

$$T = \sum_{i=1}^{n} y_i \qquad (5)$$

where

$$y_i = x_i - \mu_0 \qquad (6)$$

(Manly, 2007, p. 112). The test assumes all of the observations come from the same distribution that is symmetric about the true population mean (hence the mean is the same as the median for this distribution). Under the null hypothesis, the y_i's are equally likely to be positive or negative. Therefore, the permutation distribution of the test statistic T consists of enumerating all possible ways of permuting the signs of the y_i's and computing the resulting sums. For n observations, there are 2^n possible permutations of the signs, because each observation can either be positive or negative.

For a one-sided upper alternative hypothesis (Equation (2)), the p-value is computed as the proportion of sums in the permutation distribution that are greater than or equal to the observed sum T. For a one-sided lower alternative hypothesis (Equation (3)), the p-value is computed as the proportion of sums in the permutation distribution that are less than or equal to the observed sum T. For a two-sided alternative hypothesis (Equation (4)), the p-value is computed by using the permutation distribution of the absolute value of T (i.e., $|T|$) and computing the proportion of values in this permutation distribution that are greater than or equal to the observed value of $|T|$.

Confidence Intervals Based on Permutation Tests

Based on the relationship between hypothesis tests and confidence intervals, it is possible to construct a two-sided or one-sided $(1 - \alpha)100\%$ confidence interval for the mean μ based on the one-sample permutation test by finding the values of μ_0 that correspond to obtaining a p-value of α (Manly, 2007, pp. 18–20, 113). A confidence interval based on the bootstrap however, will yield a similar type of confidence interval (Efron and Tibshirani, 1993, p. 214); see the help file for `boot` in the R package **boot**.

Value

A list of class `"permutationTest"` containing the results of the hypothesis test. See the help file for `permutationTest.object` for details.

Note

A frequent question in environmental statistics is "Is the concentration of chemical X greater than Y units?". For example, in groundwater assessment (compliance) monitoring at hazardous and solid waste sites, the concentration of a chemical in the groundwater at a downgradient well must be compared to a groundwater protection standard (GWPS). If the concentration is "above" the GWPS, then the site enters corrective action monitoring. As another example, soil screening at a Superfund site involves comparing the concentration of a chemical in the soil with a pre-determined soil screening level (SSL). If the concentration is "above" the SSL, then further investigation and possible remedial action is required. Determining what it means for the chemical concentration to

be "above" a GWPS or an SSL is a policy decision: the average of the distribution of the chemical concentration must be above the GWPS or SSL, or the median must be above the GWPS or SSL, or the 95'th percentile must be above the GWPS or SSL, or something else. Often, the first interpretation is used.

Hypothesis tests you can use to perform tests of location include: Student's t-test, Fisher's randomization test, the Wilcoxon signed rank test, Chen's modified t-test, the sign test, and a test based on a bootstrap confidence interval. For a discussion comparing the performance of these tests, see Millard and Neerchal (2001, pp.408-409).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, pp.224–227.

Manly, B.F.J. (2007). *Randomization, Bootstrap and Monte Carlo Methods in Biology*. Third Edition. Chapman & Hall, New York, pp.112-113.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.404–406.

See Also

permutationTest.object, Hypothesis Tests, boot.

Examples

```
# Generate 10 observations from a logistic distribution with parameters
# location=7 and scale=2, and test the null hypothesis that the true mean
# is equal to 5 against the alternative that the true mean is greater than 5.
# Use the exact permutation distribution.
# (Note: the call to set.seed() allows you to reproduce this example).

set.seed(23)

dat <- rlogis(10, location = 7, scale = 2)

test.list <- oneSamplePermutationTest(dat, mu = 5,
  alternative = "greater", exact = TRUE)

# Print the results of the test
#------------------------------
test.list

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 Mean (Median) = 5
#
```

```
#Alternative Hypothesis:          True Mean (Median) is greater than 5
#
#Test Name:                       One-Sample Permutation Test
#                                 (Exact)
#
#Estimated Parameter(s):          Mean = 9.977294
#
#Data:                            dat
#
#Sample Size:                     10
#
#Test Statistic:                  Sum(x - 5) = 49.77294
#
#P-value:                         0.001953125


# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

#==========

# The guidance document "Supplemental Guidance to RAGS: Calculating the
# Concentration Term" (USEPA, 1992d) contains an example of 15 observations
# of chromium concentrations (mg/kg) which are assumed to come from a
# lognormal distribution.  These data are stored in the vector
# EPA.92d.chromium.vec.  Here, we will use the permutation test to test
# the null hypothesis that the mean (median) of the log-transformed chromium
# concentrations is less than or equal to log(100 mg/kg) vs. the alternative
# that it is greater than log(100 mg/kg).  Note that we *cannot* use the
# permutation test to test a hypothesis about the mean on the original scale
# because the data are not assumed to be symmetric about some mean, they are
# assumed to come from a lognormal distribution.
#
# We will sample from the permutation distribution.
# (Note: setting the argument seed=542 allows you to reproduce this example).

test.list <- oneSamplePermutationTest(log(EPA.92d.chromium.vec),
  mu = log(100), alternative = "greater", seed = 542)

test.list

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 Mean (Median) = 4.60517
#
#Alternative Hypothesis:          True Mean (Median) is greater than 4.60517
#
#Test Name:                       One-Sample Permutation Test
#                                 (Based on Sampling
#                                 Permutation Distribution
```

```
#                                5000 Times)
#
#Estimated Parameter(s):         Mean = 4.378636
#
#Data:                           log(EPA.92d.chromium.vec)
#
#Sample Size:                    15
#
#Test Statistic:                 Sum(x - 4.60517) = -3.398017
#
#P-value:                        0.7598


# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

#----------

# Clean up
#---------
rm(test.list)
graphics.off()
```

`Ozone.NE.df` *Ozone Concentrations in the Northeast U.S.*

Description

Ozone concentrations in 41 U.S. cities based on daily maxima collected between June and August 1974.

Usage

```
Ozone.NE.df
```

Format

A data frame with 41 observations on the following 5 variables.

`Median` median of daily maxima ozone concentration (ppb).

`Quartile` Upper quartile (i.e., 75th percentile) of daily maxima ozone concentration (ppb).

`City` a factor indicating the city

`Longitude` negative longitude of the city

`Latitude` latitude of the city

Source

Cleveland, W.S., Kleiner, B., McRae, J.E., Warner, J.L., and Pasceri, P.E. (1975). *The Analysis of Ground-Level Ozone Data from New Jersey, New York, Connecticut, and Massachusetts: Data Quality Assessment and Temporal and Geographical Properties*. Bell Laboratories Memorandum.

The original data were collected by the New Jersey Department of Environmental Protection, the New York State Department of Environmental Protection, the Boyce Thompson Institute (Yonkers, for New York data), the Connecticut Department of Environmental Protection, and the Massachusetts Department of Public Health.

Examples

```
summary(Ozone.NE.df)
#     Median          Quartile              City      Longitude
# Min.   : 34.00   Min.   : 48.00   Asbury Park: 1   Min.   :-74.71
# 1st Qu.: 58.00   1st Qu.: 79.75   Babylon    : 1   1st Qu.:-73.74
# Median : 65.00   Median : 90.00   Bayonne    : 1   Median :-73.17
# Mean   : 68.15   Mean   : 95.10   Boston     : 1   Mean   :-72.94
# 3rd Qu.: 80.00   3rd Qu.:112.25   Bridgeport : 1   3rd Qu.:-72.08
# Max.   :100.00   Max.   :145.00   Cambridge  : 1   Max.   :-71.05
#                  NA's   :  1.00   (Other)    :35
#    Latitude
# Min.   :40.22
# 1st Qu.:40.97
# Median :41.56
# Mean   :41.60
# 3rd Qu.:42.25
# Max.   :43.32
```

Pareto *The Pareto Distribution*

Description

Density, distribution function, quantile function, and random generation for the Pareto distribution with parameters `location` and `shape`.

Usage

```
dpareto(x, location, shape = 1)
ppareto(q, location, shape = 1)
qpareto(p, location, shape = 1)
rpareto(n, location, shape = 1)
```

Arguments

`x`	vector of quantiles.
`q`	vector of quantiles.

p	vector of probabilities between 0 and 1.
n	sample size. If length(n) is larger than 1, then length(n) random values are returned.
location	vector of (positive) location parameters.
shape	vector of (positive) shape parameters. The default is shape=1.

Details

Let X be a Pareto random variable with parameters location=η and shape=θ. The density function of X is given by:

$$f(x;\eta,\theta) = \frac{\theta\eta^{\theta}}{x^{\theta+1}},\ \eta > 0,\ \theta > 0,\ x \geq \eta$$

The cumulative distribution function of X is given by:

$$F(x;\eta,\theta) = 1 - (\frac{\eta}{x})^{\theta}$$

and the p'th quantile of X is given by:

$$x_p = \eta(1-p)^{-1/\theta},\ 0 \leq p \leq 1$$

The mode, mean, median, variance, and coefficient of variation of X are given by:

$$Mode(X) = \eta$$

$$E(X) = \frac{\theta\eta}{\theta - 1},\ \theta > 1$$

$$Median(X) = x_{0.5} = 2^{1/\theta}\eta$$

$$Var(X) = \frac{\theta\eta^2}{(\theta-1)^2(\theta-1)},\ \theta > 2$$

$$CV(X) = [\theta(\theta-2)]^{-1/2},\ \theta > 2$$

Value

dpareto gives the density, ppareto gives the distribution function, qpareto gives the quantile function, and rpareto generates random deviates.

Note

The Pareto distribution is named after Vilfredo Pareto (1848-1923), a professor of economics. It is derived from Pareto's law, which states that the number of persons N having income $\geq x$ is given by:

$$N = Ax^{-\theta}$$

where θ denotes Pareto's constant and is the shape parameter for the probability distribution.

The Pareto distribution takes values on the positive real line. All values must be larger than the "location" parameter η, which is really a threshold parameter. There are three kinds of Pareto distributions. The one described here is the Pareto distribution of the first kind. Stable Pareto distributions have $0 < \theta < 2$. Note that the r'th moment only exists if $r < \theta$.

The Pareto distribution is related to the exponential distribution and logistic distribution as follows. Let X denote a Pareto random variable with `location`=η and `shape`=θ. Then $log(X/\eta)$ has an exponential distribution with parameter `rate`=θ, and $-log\{[(X/\eta)^{\theta}] - 1\}$ has a logistic distribution with parameters `location`=0 and `scale`=1.

The Pareto distribution has a very long right-hand tail. It is often applied in the study of socioeconomic data, including the distribution of income, firm size, population, and stock price fluctuations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

See Also

`epareto`, `eqpareto`, Exponential, Probability Distributions and Random Numbers.

Examples

```
# Density of a Pareto distribution with parameters location=1 and shape=1,
# evaluated at 2, 3 and 4:

dpareto(2:4, 1, 1)
#[1] 0.2500000 0.1111111 0.0625000

#----------

# The cdf of a Pareto distribution with parameters location=2 and shape=1,
# evaluated at 3, 4, and 5:

ppareto(3:5, 2, 1)
#[1] 0.3333333 0.5000000 0.6000000

#----------

# The 25'th percentile of a Pareto distribution with parameters
# location=1 and shape=1:

qpareto(0.25, 1, 1)
#[1] 1.333333

#----------

# A random sample of 4 numbers from a Pareto distribution with parameters
# location=3 and shape=2.
# (Note: the call to set.seed simply allows you to reproduce this example.)
```

```
set.seed(10)
rpareto(4, 3, 2)
#[1] 4.274728 3.603148 3.962862 5.415322
```

---

pdfPlot *Plot Probability Density Function*

Description

Produce a probability density function (pdf) plot for a user-specified distribution.

Usage

```
pdfPlot(distribution = "norm", param.list = list(mean = 0, sd = 1),
  left.tail.cutoff = ifelse(is.finite(supp.min), 0, 0.001),
  right.tail.cutoff = ifelse(is.finite(supp.max), 0, 0.001),
  plot.it = TRUE, add = FALSE, n.points = 1000, pdf.col = "black",
  pdf.lwd = 3 * par("cex"), pdf.lty = 1, curve.fill = !add,
  curve.fill.col = "cyan", x.ticks.at.all.x.max = 15,
  hist.col = ifelse(add, "black", "cyan"), density = 5,
  digits = .Options$digits, ..., type = "l", main = NULL, xlab = NULL,
  ylab = NULL, xlim = NULL, ylim = NULL)
```

Arguments

`distribution` a character string denoting the distribution abbreviation. The default value is `distribution="norm"`. See the help file for `Distribution.df` for a list of possible distribution abbreviations.

`param.list` a list with values for the parameters of the distribution. The default value is `param.list=list(mean=0, sd=1)`. See the help file for `Distribution.df` for the names and possible values of the parameters associated with each distribution.

`left.tail.cutoff`
a numeric scalar indicating what proportion of the left-tail of the probability distribution to omit from the plot. For densities with a finite support minimum (e.g., Lognormal) the default value is `0`; for all other densities the default value is `0.001`.

`right.tail.cutoff`
a scalar indicating what proportion of the right-tail of the probability distribution to omit from the plot. For densities with a finite support maximum (e.g., Binomial) the default value is `0`; for all other densities the default value is `0.001`.

`plot.it` a logical scalar indicating whether to create a plot or add to the existing plot (see `add`) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x, y) values is returned (see the section VALUE below). The default value is `plot.it=TRUE`.

add — a logical scalar indicating whether to add the probability density curve to the existing plot (add=TRUE), or to create a new plot (add=FALSE; the default). This argument is ignored if plot.it=FALSE.

n.points — a numeric scalar specifying at how many evenly-spaced points the probability density function will be evaluated. The default value is n.points=1000.

pdf.col — for continuous distributions, a numeric scalar or character string determining the color of the pdf line in the plot. The default value is pdf.col="black". See the entry for col in the help file for par for more information.

pdf.lwd — for continuous distributions, a numeric scalar determining the width of the pdf line in the plot. The default value is pdf.lwd=3*par("cex"). See the entry for lwd in the help file for par for more information.

pdf.lty — for continuous distributions, a numeric scalar determining the line type of the pdf line in the plot. The default value is pdf.lty=1. See the entry for lty in the help file for par for more information.

curve.fill — for continuous distributions, a logical value indicating whether to fill in the area below the probability density curve with the color specified by curve.fill.col. The default value is TRUE unless add=TRUE.

curve.fill.col — for continuous distributions, when curve.fill=TRUE, a numeric scalar or character string indicating what color to use to fill in the area below the probability density curve. The default value is curve.fill.col="cyan". See the entry for col in the help file for par for more information.

x.ticks.at.all.x.max — a numeric scalar indicating the maximum number of ticks marks on the x-axis. The default value is x.ticks.at.all.x.max=15.

hist.col — for discrete distributions, a numeric scalar or character string indicating what color to use to fill in the histogram if add=FALSE, or the color of the shading lines if add=TRUE. The default is "cyan" if add=FALSE and "black" if add=TRUE. See the entry for col in the help file for par for more information.

density — for discrete distributions, a scalar indicting the density of line shading for the histogram when add=TRUE. This argument is ignored if add=FALSE.

digits — a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

type, main, xlab, ylab, xlim, ylim, ... — additional graphical parameters. See plot.default and par).

Details

The ***probability density function (pdf)*** of a random variable X, usually denoted f, is defined as:

$$f(x) = \frac{dF(x)}{dx} \qquad (1)$$

where F is the cumulative distribution function (cdf) of X. That is, $f(x)$ is the derivative of the cdf F with respect to x (where this derivative exists).

For discrete distributions, the probability density function is simply:

$$f(x) = Pr(X = x) \qquad (2)$$

In this case, f is sometimes called the ***probability function*** or ***probability mass function***.

The probability that the random variable X takes on a value in the interval $[a, b]$ is simply the (Lebesgue) integral of the pdf evaluated between a and b. That is,

$$Pr(a \leq X \leq b) = \int_a^b f(x)dx \qquad (3)$$

For discrete distributions, Equation (3) translates to summing up the probabilities of all values in this interval:

$$Pr(a \leq X \leq b) = \sum_{x \in [a,b]} f(x) = \sum_{x \in [a,b]} Pr(X = x) \qquad (4)$$

A ***probability density function (pdf) plot*** plots the values of the pdf against quantiles of the specified distribution. Theoretical pdf plots are sometimes plotted along with empirical pdf plots (density plots), histograms or bar graphs to visually assess whether data have a particular distribution.

Value

`pdfPlot` invisibly returns a list giving coordinates of the points that have been or would have been plotted:

`Quantiles` The quantiles used for the plot.

`Probability.Densities`
The values of the pdf associated with the quantiles.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions, Second Edition*. John Wiley and Sons, New York.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

`Distribution.df`, `epdfPlot`, `cdfPlot`.

Examples

```
# Plot the pdf of the standard normal distribution
#-------------------------------------------------
dev.new()
pdfPlot()

#==========

# Plot the pdf of the standard normal distribution
# and a N(2, 2) distribution on the sample plot.
#-------------------------------------------------
dev.new()
pdfPlot(param.list = list(mean=2, sd=2),
  curve.fill = FALSE, ylim = c(0, dnorm(0)), main = "")

pdfPlot(add = TRUE, pdf.col = "red")

legend("topright", legend = c("N(2,2)", "N(0,1)"),
  col = c("black", "red"), lwd = 3 * par("cex"))

title("PDF Plots for Two Normal Distributions")

#==========

# Clean up
#---------
graphics.off()
```

---

permutationTest.object

S3 Class "permutationTest"

Description

This class of objects is returned by functions that perform permutation tests. Objects of class `"permutationTest"` are lists that contain information about the null and alternative hypotheses, the estimated distribution parameters, the test statistic and the p-value. They also contain the permutation distribution of the statistic (or a sample of the permutation distribution).

Details

Objects of S3 class `"permutationTest"` are returned by any of the **EnvStats** functions that perform permutation tests. Currently, these are: `oneSamplePermutationTest`, `twoSamplePermutationTestLocation`, and `twoSamplePermutationTestProportion`.

Value

A legitimate list of class `"permutationTest"` includes the components listed in the help file for `htest.object`. In addition, the following components must be included in a legitimate list of class `"permutationTest"`:

Required Components

The following components must be included in a legitimate list of class "permutationTest".

stat.dist	numeric vector containing values of the statistic for the permutation distribution. When exact=FALSE, the vector is comprised of values sampled from the full permutation distribution.
exact	logical scalar indicating whether the exact permutation distribution was used for the test (exact=TRUE), or if instead the permutation distribution was sampled (exact=FALSE).

Optional Components

The following component may optionally be included in an object of of class "permutationTest":

seed	integer or vector of integers indicating the seed that was used for sampling the permutation distribution. This component is present only if exact=FALSE.
prob.stat.dist	numeric vector containing the probabilities associated with each element of the component stat.dist. This component is only returned by the function twoSamplePermutationTestProportion.

Methods

Generic functions that have methods for objects of class "permutationTest" include: print, plot.

Note

Since objects of class "permutationTest" are lists, you may extract their components with the $ and [[operators.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

print.permutationTest, plot.permutationTest, oneSamplePermutationTest, twoSamplePermutationTestLocation, twoSamplePermutationTestProportion, Hypothesis Tests.

Examples

```
# Create an object of class "permutationTest", then print it and plot it.
#-----------------------------------------------------------------------

set.seed(23)

dat <- rlogis(10, location = 7, scale = 2)

permutationTest.obj <- oneSamplePermutationTest(dat, mu = 5,
  alternative = "greater", exact = TRUE)
```

```
mode(permutationTest.obj)
#[1] "list"

class(permutationTest.obj)
#[1] "permutationTest"

names(permutationTest.obj)
# [1] "statistic"         "parameters"        "p.value"
# [4] "estimate"          "null.value"        "alternative"
# [7] "method"            "estimation.method" "sample.size"
#[10] "data.name"         "bad.obs"           "stat.dist"
#[13] "exact"

#==========

# Print the results of the test
#------------------------------
permutationTest.obj

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 Mean (Median) = 5
#
#Alternative Hypothesis:          True Mean (Median) is greater than 5
#
#Test Name:                       One-Sample Permutation Test
#                                 (Exact)
#
#Estimated Parameter(s):          Mean = 9.977294
#
#Data:                            dat
#
#Sample Size:                     10
#
#Test Statistic:                  Sum(x - 5) = 49.77294
#
#P-value:                         0.001953125

#==========

# Plot the results of the test
#-----------------------------
dev.new()
plot(permutationTest.obj)

#==========

# Extract the test statistic
#---------------------------

permutationTest.obj$statistic
#Sum(x - 5)
```

```
#  49.77294

#==========

# Clean up
#---------
rm(permutationTest.obj)
graphics.off()
```

---

plot.boxcox *Plot Results of Box-Cox Transformations*

Description

Plot the results of calling the function boxcox, which returns an object of class "boxcox". Three different kinds of plots are available.

The function plot.boxcox is automatically called by plot when given an object of class "boxcox". The names of other functions associated with Box-Cox transformations are listed under Data Transformations.

Usage

```
## S3 method for class 'boxcox'
plot(x, plot.type = "Objective vs. lambda", same.window = TRUE,
    ask = same.window & plot.type != "Ojective vs. lambda",
    plot.pos.con = 0.375, estimate.params = FALSE,
    equal.axes = qq.line.type == "0-1" || estimate.params, add.line = TRUE,
    qq.line.type = "least squares", duplicate.points.method = "standard",
    points.col = 1, line.col = 1, line.lwd = par("cex"), line.lty = 1,
    digits = .Options$digits, cex.main = 1.4 * par("cex"), cex.sub = par("cex"),
    main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL,
    ylim = NULL, ...)
```

Arguments

x	an object of class "boxcox". See boxcox.object for details.
plot.type	character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Objective vs. lambda" (the default), "Q-Q Plots", "Tukey M-D Q-Q Plots", and "All".
same.window	logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE; the default), or to create a new graphics window for each separate plot (same.window=FALSE). The argument is relevant only when plot.type produces more than one plot (i.e., when plot.type is not equal to "Objective vs. lambda").

`ask` logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. This argument is ignored when `plot.type="Objective vs. lambda"` (since only one plot is produced) or when `same.window=FALSE`, otherwise the default value is `ask=TRUE`.

`points.col` numeric scalar determining the color of the points in the plot. The default value is `points.col=1`. See the entry for `col` in the R help file for par for more information.

The following arguments can be supplied when `plot.type="Q-Q Plots"`, `plot.type="Tukey M-D Q-Q Plots"`, or `plot.type="All"` (supplied to qqPlot):

`plot.pos.con` numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the Q-Q plots and/or Tukey Mean-Difference Q-Q plots. The default value is `plot.pos.con=0.375`. See the help files for qqPlot for more information and the motivation for this choice.

`estimate.params` logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (`estimate.params=TRUE`) or using the distribution parameters for a standard normal distribution (i.e, `mean=0`, `sd=1`). The default value is `estimate.params=FALSE` because a standard normal Q-Q plot will yield roughly a straight line if the observations are from *any* normal distribution. If you specify `plot.type="Tukey M-D Q-Q Plots"`, then you need to set `estiamte.params=TRUE` unless you want to assume the transformed data come from a standard normal distribution.

`equal.axes` logical scalar indicating whether to use the same range on the x- and y-axes when `plot.type="Q-Q Plots"`. The default value is `TRUE` if `qq.line.type="0-1"` or `estimate.params=TRUE`, otherwise it is `FALSE`.

`add.line` logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q Plots"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey M-D Q-Q Plots"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=TRUE`.

`qq.line.type` character string determining what kind of line to add to the plot when `plot.type="Q-Q Plots"`. Possible values are:
`"least squares"` (a least squares line; the default),
`"0-1"` (a line with intercept 0 and slope 1), and
`"robust"` (a line is fit through the first and third quartiles of the x and y data).
This argument is ignored if `add.line=FALSE`.

`duplicate.points.method` a character string denoting how to plot points with duplicate (x, y) values. Possible values are `"standard"` (a single plotting symbol is plotted; the default), `"jitter"` (a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y), and `"number"` (a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate).

`line.col` numeric scalar determining the color of the line in the plot. The default value is `line.col=1`. See the entry for `col` in the R help file for par for more information. This argument is ignored if `add.line=FALSE`.

`line.lwd` numeric scalar determining the width of the line in the plot. The default value is `line.lwd=par("cex")`. See the entry for `lwd` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`line.lty` numeric scalar determining the line type (style) of the line in the plot. The default value is `line.lty=1`. See the entry for `lty` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`digits` scalar indicating how many significant digits to print for the distribution parameters and the value of the objective in the sub-title. The default value is the current setting of `options("digits")`.

Graphics parameters:

`cex.main`, `cex.sub`, `main`, `sub`, `xlab`, `ylab`, `xlim`, `ylim`, `...`
graphics parameters; see `par` for more information. The default value of `cex.main` is `cex.main=1.4 * par("cex")`. The default value of `cex.sub` is `cex.sub=par("cex")`.

Details

The function `plot.boxcox` is a method for the generic function `plot` for the class `"boxcox"` (see `boxcox.object`). It can be invoked by calling `plot` and giving it an object of class `"boxcox"` as the first argument, or by calling `plot.boxcox` directly, regardless of the class of the object given as the first argument to `plot.boxcox`.

Plots associated with Box-Cox transformations are produced on the current graphics device. These can be one or all of the following:

- Objective vs. λ.
- Observed Quantiles vs. Normal Quantiles (Q-Q Plot) for the transformed observations for each of the values of λ.
- Tukey Mean-Difference Q-Q Plots for the transformed observations for each of the values of λ.

See the help files for `boxcox` and `qqPlot` for more information.

Value

`plot.boxcox` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`qqPlot`, `boxcox`, `boxcox.object`, `print.boxcox`, Data Transformations, `plot`.

Examples

```
# Generate 30 observations from a lognormal distribution with
# mean=10 and cv=2, call the function boxcox, and then plot
# the results.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
x <- rlnormAlt(30, mean = 10, cv = 2)

# Plot the results based on the PPCC objective
#---------------------------------------------
boxcox.list <- boxcox(x)
dev.new()
plot(boxcox.list)

# Look at Q-Q Plots for the candidate values of lambda
#-----------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)


# Look at Tukey Mean-Difference Q-Q Plots
# for the candidate values of lambda
#---------------------------------------
plot(boxcox.list, plot.type = "Tukey M-D Q-Q Plots", same.window = FALSE)

#==========

# Clean up
#---------
rm(x, boxcox.list)
graphics.off()
```

`plot.boxcoxCensored` *Plot Results of Box-Cox Transformations Based on Type I Censored Data*

Description

Plot the results of calling the function `boxcoxCensored`, which returns an object of class `"boxcoxCensored"`. Three different kinds of plots are available.

The function `plot.boxcoxCensored` is automatically called by `plot` when given an object of class `"boxcoxCensored"`.

Usage

```
## S3 method for class 'boxcoxCensored'
plot(x, plot.type = "Objective vs. lambda", same.window = TRUE,
    ask = same.window & plot.type != "Ojective vs. lambda",
    prob.method = "michael-schucany", plot.pos.con = 0.375,
```

```
estimate.params = FALSE, equal.axes = qq.line.type == "0-1" || estimate.params,
 add.line = TRUE, qq.line.type = "least squares",
 duplicate.points.method = "standard", points.col = 1, line.col = 1,
 line.lwd = par("cex"), line.lty = 1, digits = .Options$digits,
 cex.main = 1.4 * par("cex"), cex.sub = par("cex"),
 main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL,
 ylim = NULL, ...)
```

Arguments

x	an object of class "boxcoxCensored". See boxcoxCensored.object for details.
plot.type	character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Objective vs. lambda" (the default), "Q-Q Plots", "Tukey M-D Q-Q Plots", and "All".
same.window	logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE; the default), or to create a new graphics window for each separate plot (same.window=FALSE). The argument is relevant only when plot.type produces more than one plot (i.e., when plot.type is not equal to "Objective vs. lambda").
ask	logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. This argument is ignored when plot.type="Objective vs. lambda" (since only one plot is produced) or when same.window=FALSE, otherwise the default value is ask=TRUE.
points.col	numeric scalar determining the color of the points in the plot. The default value is points.col=1. See the entry for col in the R help file for par for more information.

The following arguments can be supplied when plot.type="Q-Q Plots"**,** plot.type="Tukey M-D Q-Q Plots"**, or** plot.type="All" **(supplied to** qqPlot**):**

prob.method	character string indicating what method to use to compute the plotting positions for Q-Q plots or Tukey Mean-Difference Q-Q plots. Possible values are "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany". The "nelson" method is only available for objects that are the result of calling boxcoxCensored with the argument censoring.side="right". See the help file for qqPlotCensored for more information. This argument is ignored if plot.type="Objective vs. lambda".

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the Q-Q plots and/or Tukey Mean-Difference Q-Q plots. The default value is plot.pos.con=0.375. See the help file for qqPlotCensored for more information.

estimate.params
logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (estimate.params=TRUE) or using the distribution parameters for a standard normal distribution (i.e, mean=0, sd=1). The default value is estimate.params=FALSE because a standard normal Q-Q plot will yield roughly a straight line if the observations are from *any* normal distribution. If you specify plot.type="Tukey M-D Q-Q Plots", then you need to set estiamte.params=TRUE unless you want to assume the transformed data come from a standard normal distribution.

equal.axes logical scalar indicating whether to use the same range on the x- and y-axes when plot.type="Q-Q Plots". The default value is TRUE if qq.line.type="0-1" or estimate.params=TRUE, otherwise it is FALSE.

add.line logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q Plots", a line determined by the value of qq.line.type is added to the plot. If add.line=TRUE and plot.type="Tukey M-D Q-Q Plots", a horizontal line at $y = 0$ is added to the plot. The default value is add.line=TRUE.

qq.line.type character string determining what kind of line to add to the plot when plot.type="Q-Q Plots". Possible values are "least squares" (a least squares line; the default), "0-1" (a line with intercept 0 and slope 1), and "robust" (a line is fit through the first and third quartiles of the x and y data). This argument is ignored if add.line=FALSE.

duplicate.points.method
a character string denoting how to plot points with duplicate (x, y) values. Possible values are "standard" (a single plotting symbol is plotted; the default), "jitter" (a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y), and "number" (a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate).

line.col numeric scalar determining the color of the line in the plot. The default value is line.col=1. See the entry for col in the R help file for par for more information. This argument is ignored if add.line=FALSE.

line.lwd numeric scalar determining the width of the line in the plot. The default value is line.lwd=par("cex"). See the entry for lwd in the R help file for par for more information. This argument is ignored if add.line=FALSE.

line.lty numeric scalar determining the line type (style) of the line in the plot. The default value is line.lty=1. See the entry for lty in the R help file for par for more information. This argument is ignored if add.line=FALSE.

digits scalar indicating how many significant digits to print for the distribution parameters and the value of the objective in the sub-title. The default value is the current setting of options("digits").

Graphics parameters:

`cex.main`, `cex.sub`, `main`, `sub`, `xlab`, `ylab`, `xlim`, `ylim`, `...`
: graphics parameters; see `par` for more information. The default value of `cex.main` is `cex.main=1.4 * par("cex")`. The default value of `cex.sub` is `cex.sub=par("cex")`.

Details

The function `plot.boxcoxCensored` is a method for the generic function `plot` for the class `"boxcoxCensored"` (see `boxcoxCensored.object`). It can be invoked by calling `plot` and giving it an object of class `"boxcoxCensored"` as the first argument, or by calling `plot.boxcoxCensored` directly, regardless of the class of the object given as the first argument to `plot.boxcoxCensored`.

Plots associated with Box-Cox transformations are produced on the current graphics device. These can be one or all of the following:

- Objective vs. λ.
- Observed Quantiles vs. Normal Quantiles (Q-Q Plot) for the transformed observations for each of the values of λ.
- Tukey Mean-Difference Q-Q Plots for the transformed observations for each of the values of λ.

See the help files for `boxcoxCensored` and `qqPlotCensored` for more information.

Value

`plot.boxcoxCensored` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`qqPlotCensored`, `boxcoxCensored`, `boxcoxCensored.object`, `print.boxcoxCensored`, Data Transformations, `plot`.

Examples

```
# Generate 15 observations from a lognormal distribution with
# mean=10 and cv=2 and censor the observations less than 2.
# Then generate 15 more observations from this distribution and
# censor the observations less than 4.
# Then call the function boxcoxCensored, and then plot the results.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

x.1 <- rlnormAlt(15, mean = 10, cv = 2)
```

```
censored.1 <- x.1 < 2
x.1[censored.1] <- 2

x.2 <- rlnormAlt(15, mean = 10, cv = 2)
censored.2 <- x.2 < 4
x.2[censored.2] <- 4

x <- c(x.1, x.2)
censored <- c(censored.1, censored.2)

# Plot the results based on the PPCC objective
#---------------------------------------------
boxcox.list <- boxcoxCensored(x, censored)
dev.new()
plot(boxcox.list)

# Look at Q-Q Plots for the candidate values of lambda
#-----------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)


# Look at Tukey Mean-Difference Q-Q Plots
# for the candidate values of lambda
#---------------------------------------
plot(boxcox.list, plot.type = "Tukey M-D Q-Q Plots", same.window = FALSE)

#==========

# Clean up
#---------
rm(x.1, censored.1, x.2, censored.2, x, censored, boxcox.list)
graphics.off()
```

`plot.boxcoxLm` *Plot Results of Box-Cox Transformations for a Linear Model*

Description

Plot the results of calling the function `boxcox` when the argument `x` supplied to `boxcox` is an object of class `"lm"`. Three different kinds of plots are available.

The function `plot.boxcoxLm` is automatically called by `plot` when given an object of class `"boxcoxLm"`. The names of other functions associated with Box-Cox transformations are listed under Data Transformations.

Usage

```
## S3 method for class 'boxcoxLm'
plot(x, plot.type = "Objective vs. lambda", same.window = TRUE,
    ask = same.window & plot.type != "Ojective vs. lambda",
```

```
plot.pos.con = 0.375, estimate.params = FALSE,
equal.axes = qq.line.type == "0-1" || estimate.params, add.line = TRUE,
qq.line.type = "least squares", duplicate.points.method = "standard",
points.col = 1, line.col = 1, line.lwd = par("cex"), line.lty = 1,
digits = .Options$digits, cex.main = 1.4 * par("cex"), cex.sub = par("cex"),
main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL,
ylim = NULL, ...)
```

Arguments

x	an object of class "boxcoxLm". See boxcoxLm.object for details.
plot.type	character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Objective vs. lambda" (the default), "Q-Q Plots", "Tukey M-D Q-Q Plots", and "All".
same.window	logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE; the default), or to create a new graphics window for each separate plot (same.window=FALSE). The argument is relevant only when plot.type produces more than one plot (i.e., when plot.type is not equal to "Objective vs. lambda").
ask	logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. This argument is ignored when plot.type="Objective vs. lambda" (since only one plot is produced) or when same.window=FALSE, otherwise the default value is ask=TRUE.
points.col	numeric scalar determining the color of the points in the plot. The default value is points.col=1. See the entry for col in the R help file for par for more information.

The following arguments can be supplied when plot.type="Q-Q Plots"**,** plot.type="Tukey M-D Q-Q Plots"**, or** plot.type="All" **(supplied to** qqPlot**):**

plot.pos.con	numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the Q-Q plots and/or Tukey Mean-Difference Q-Q plots. The default value is plot.pos.con=0.375. See the help files for qqPlot for more information and the motivation for this choice.
estimate.params	logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (estimate.params=TRUE) or using the distribution parameters for a standard normal distribution (i.e, mean=0, sd=1). The default value is estimate.params=FALSE because a standard normal Q-Q plot will yield roughly a straight line if the observations are from *any* normal distribution. If you specify plot.type="Tukey M-D Q-Q Plots", then you need to set estiamte.params=TRUE unless you want to assume the transformed data come from a standard normal distribution.

`equal.axes` logical scalar indicating whether to use the same range on the x- and y-axes when `plot.type="Q-Q Plots"`. The default value is `TRUE` if `qq.line.type="0-1"` or `estimate.params=TRUE`, otherwise it is `FALSE`.

`add.line` logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q Plots"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey M-D Q-Q Plots"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=TRUE`.

`qq.line.type` character string determining what kind of line to add to the plot when `plot.type="Q-Q Plots"`. Possible values are `"least squares"` (a least squares line; the default), `"0-1"` (a line with intercept 0 and slope 1), and `"robust"` (a line is fit through the first and third quartiles of the x and y data). This argument is ignored if `add.line=FALSE`.

`duplicate.points.method` a character string denoting how to plot points with duplicate (x, y) values. Possible values are `"standard"` (a single plotting symbol is plotted; the default), `"jitter"` (a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y), and `"number"` (a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate).

`line.col` numeric scalar determining the color of the line in the plot. The default value is `line.col=1`. See the entry for `col` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`line.lwd` numeric scalar determining the width of the line in the plot. The default value is `line.lwd=par("cex")`. See the entry for `lwd` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`line.lty` numeric scalar determining the line type (style) of the line in the plot. The default value is `line.lty=1`. See the entry for `lty` in the R help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`digits` scalar indicating how many significant digits to print for the distribution parameters and the value of the objective in the sub-title. The default value is the current setting of `options("digits")`.

Graphics parameters:

`cex.main`, `cex.sub`, `main`, `sub`, `xlab`, `ylab`, `xlim`, `ylim`, `...` graphics parameters; see `par` for more information. The default value of `cex.main` is `cex.main=1.4 * par("cex")`. The default value of `cex.sub` is `cex.sub=par("cex")`.

Details

The function `plot.boxcoxLm` is a method for the generic function `plot` for the class `"boxcoxLm"` (see `boxcoxLm.object`). It can be invoked by calling `plot` and giving it an object of class `"boxcoxLm"` as the first argument, or by calling `plot.boxcoxLm` directly, regardless of the class of the object given as the first argument to `plot.boxcoxLm`.

Plots associated with Box-Cox transformations are produced on the current graphics device. These can be one or all of the following:

- Objective vs. λ.
- Observed Quantiles vs. Normal Quantiles (Q-Q Plot) for the residuals of the linear model based on transformed values of the response variable for each of the values of λ.
- Tukey Mean-Difference Q-Q Plots for the residuals of the linear model based on transformed values of the response variable for each of the values of λ.

See the help files for `boxcox` and `qqPlot` for more information.

Value

`plot.boxcoxLm` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`qqPlot`, `boxcox`, `boxcoxLm.object`, `print.boxcoxLm`, Data Transformations, `plot`.

Examples

```
# Create an object of class "boxcoxLm", then plot the results.

# The data frame Environmental.df contains daily measurements of
# ozone concentration, wind speed, temperature, and solar radiation
# in New York City for 153 consecutive days between May 1 and
# September 30, 1973.  In this example, we'll model ozone as a
# function of temperature.

# Fit the model with the raw Ozone data
#--------------------------------------
ozone.fit <- lm(ozone ~ temperature, data = Environmental.df)

boxcox.list <- boxcox(ozone.fit)

# Plot PPCC vs. lambda based on Q-Q plots of residuals
#----------------------------------------------------
dev.new()
plot(boxcox.list)

# Look at Q-Q plots of residuals for the various transformation
#--------------------------------------------------------------
plot(boxcox.list, plot.type = "Q-Q Plots", same.window = FALSE)
```

```
# Look at Tukey Mean-Difference Q-Q plots of residuals
# for the various transformation
#-----------------------------------------------------
plot(boxcox.list, plot.type = "Tukey M-D Q-Q Plots", same.window = FALSE)

#==========

# Clean up
#---------
rm(ozone.fit, boxcox.list)
graphics.off()
```

---

plot.gof *Plot Results of Goodness-of-Fit Test*

Description

Plot the results of calling the function gofTest, which returns an object of class "gof" when testing the goodness-of-fit of a set of data to a distribution (i.e., when supplied with the y argument but not the x argument). Five different kinds of plots are available.

The function plot.gof is automatically called by plot when given an object of class "gof". The names of other functions associated with goodness-of-fit test are listed under Goodness-of-Fit Tests.

Usage

```
## S3 method for class 'gof'
plot(x, plot.type = "Summary",
  captions = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL, Results = NULL),
  x.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
  y.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
  same.window = FALSE, ask = same.window & plot.type == "All", hist.col = "cyan",
  fitted.pdf.col = "black", fitted.pdf.lwd = 3 * par("cex"), fitted.pdf.lty = 1,
  plot.pos.con = switch(dist.abb, norm = , lnorm = , lnormAlt = , lnorm3 = 0.375,
    evd = 0.44, 0.4), ecdf.col = "cyan", fitted.cdf.col = "black",
  ecdf.lwd = 3 * par("cex"), fitted.cdf.lwd = 3 * par("cex"), ecdf.lty = 1,
  fitted.cdf.lty = 2, add.line = TRUE,
  digits = ifelse(plot.type == "Summary", 2, .Options$digits),
  test.result.font = 1,
  test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
  test.result.mar = c(0, 0, 3, 0) + 0.1,
  cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
  cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
  cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
  main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
  add.om.title = TRUE,
 oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 2.5, 0) else c(0, 0, 0, 0),
 om.title = NULL, om.font = 2, om.cex.main = 1.75 * par("cex"), om.line = 0.5, ...)
```

Arguments

x	an object of class "gof". See gof.object for details.
plot.type	character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Summary" (the default), "PDFs: Observed and Fitted", "CDFs: Observed and Fitted", "Q-Q Plot", "Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.
captions	a list with 1 to 5 components with the names "PDFs", "CDFs", "QQ", "MDQQ", and/or "Results". Each component either has the value NULL or else it is a character string containing the title for that particular kind of plot. When the component has the value NULL (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gof (i.e., when plot.type="Summary" or plot.type="All") and you want to specify titles different from the default ones. If you are creating only one kind of plot, then you can just use the main argument to specify a title different from the default one.
x.labels	a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the x-axis for that particular kind of plot. When the component has the value NULL (the default), a default x-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gof (i.e., when plot.type="Summary" or plot.type="All") and you want to specify x-axis labels different from the default ones. If you are creating only one plot, then you can just use the xlab argument to specify an x-axis label different from the default one.
y.labels	a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the y-axis for that particular kind of plot. When the component has the value NULL (the default), a default y-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gof (i.e., when plot.type="Summary" or plot.type="All") and you want to specify y-axis labels different from the default ones. If you are creating only one plot, then you can just use the ylab argument to specify a y-axis label different from the default one.
same.window	logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE), or to create a new graphics window for each separate plot (same.window=FALSE; the default). The argument is relevant only when plot.type="All".
ask	logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is FALSE unless same.window=TRUE and plot.type == "All".
digits	scalar indicating how many significant digits to print for the distribution parameters. If plot.type == "Summary", the default value is digits=2, otherwise it is .Options$digits (i.e., the current setting of options("digits")). This argument is ignored when plot.type="PDFs: Observed and Fitted".

Arguments associated with `plot.type="PDFs: Observed and Fitted"`**:**

`hist.col` a character string or numeric scalar determining the color of the histogram used to display the distribution of the observed values. The default value is `hist.col="cyan"`. See the entry for `col` in the R help file for `par` for more information.

`fitted.pdf.col` a character string or numeric scalar determining the color of the fitted PDF (which is displayed as a line for continuous distributions and a histogram for discrete distributions). The default value is `fitted.pdf.col="black"`. See the entry for `col` in the R help file for `par` for more information.

`fitted.pdf.lwd` numeric scalar determining the width of the line used to display the fitted PDF. The default value is `fitted.pdf.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

`fitted.pdf.lty` numeric scalar determining the line type used to display the fitted PDF. The default value is `fitted.pdf.lty=1`. See the entry for `lty` in the R help file for `par` for more information.

Arguments associated with `plot.type="CDFs: Observed and Fitted"`**:**

`plot.pos.con` numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the observed (empirical) CDF. The default value of `plot.pos.con` depends on the value of `gof.obj$distribution` (i.e., the distribution assumed for the goodness-of-fit test). For the normal, lognormal, and three-parameter lognormal distributions, the default value is `plot.pos.con=0.375`. For the Type I extreme value (Gumbel) distribution, the default value is `plot.pos.con=0.44`. For all other distributions, the default value is `plot.pos.con=0.4`. See the help files for `ecdfPlot` and `qqPlot` for more information and the motivation for these choices of values.

NOTE: This argument is also used to determine the value of the plotting position constant for the Q-Q plot (`plot.type="Q-Q Plot"`), or the Tukey Mean-Difference Q-Q plot (`plot.type="Tukey M-D Q-Q Plot"`).

`ecdf.col` a character string or numeric scalar determining the color of the line used to display the empirical CDF. The default value is `ecdf.col="cyan"`. See the entry for `col` in the R help file for `par` for more information.

`fitted.cdf.col` a character string or numeric scalar determining the color of the line used to display the fitted CDF. The default value is `fitted.cdf.col="black"`. See the entry for `col` in the R help file for `par` for more information.

`ecdf.lwd` numeric scalar determining the width of the line used to display the empirical CDF. The default value is `ecdf.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

`fitted.cdf.lwd` numeric scalar determining the width of the line used to display the fitted CDF. The default value is `fitted.cdf.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

`ecdf.lty` numeric scalar determining the line type used to display the empirical CDF. The default value is `ecdf.lty=1`. See the entry for `lty` in the R help file for `par` for more information.

`fitted.cdf.lty` numeric scalar determining the line type used to display the fitted CDF. The default value is `fitted.cdf.lty=2`. See the entry for `lty` in the R help file for `par` for more information.

Arguments associated with `plot.type="Q-Q Plot"` or `plot.type="Tukey M-D Q-Q Plot"`:

As explained above, `plot.pos.con` is used for these plot types. Also:

`add.line` logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q Plot"`, a 0-1 line is added to the plot. If `add.line=TRUE` and `plot.type="Tukey M-D Q-Q Plot"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=TRUE`.

Arguments associated with `plot.type="Test Results"`

`test.result.font`
numeric scalar indicating which font to use to print out the test results. The default value is `test.result.font=1`. See the description of the `font` argument in the help file for `par` for more information. You may get better results if you use a font number that corresponds to a fixed font (e.g., courier).

`test.result.cex`
numeric scalar indicating the value of `cex` to use to print out the test results. The default value is `0.9*par("cex")` when `plot.type="Summary"`, otherwise it is `par("cex")`. See the description of the `cex` argument in the help file for `par` for more information.

`test.result.mar`
numeric vector indicating the value of `mar` to use to print out the test results. The default value is `test.result.mar=c(0, 0, 3, 0)+0.1`. See the description of the `mar` argument in the help file for `par` for more information.

Arguments associated with `plot.type="Summary"`

`add.om.title` logical scalar indicating whether to add a title in the outer margin when `plot.type="Summary"`. The default value is `add.om.title=TRUE`.

`om.title` character string containing the outer margin title. The default value is `om.title=NULL`, which will result in a default title.

`om.font` numeric scalar indicating the font to use for the outer margin. The default value is `om.font=2`.

`om.cex.main` numeric scalar indicating the value of `cex` for the outer margin title. The default value is `1.75 * par("cex")`.

`om.line` numeric scalar indicating the line to place the outer margin title on. The default value is `om.line=0.5`.

Graphics parameters:

cex.main, cex.axis, cex.lab, main, xlab, ylab, xlim, ylim, oma, ...
: additional graphics parameters. See the help file for par.

Details

The function plot.gof is a method for the generic function plot for objects that inherit from class "gof"
(see gof.object). It can be invoked by calling plot and giving it an object of class "gof" as the first argument, or by calling plot.gof directly, regardless of the class of the object given as the first argument to plot.gof.

Plots associated with the goodness-of-fit test are produced on the current graphics device. These can be one or all of the following:

- Observed distribution overlaid with fitted distribution (plot.type="PDFs: Observed and Fitted"). See the help files for hist and pdfPlot.
- Observed empirical distribution overlaid with fitted cumulative distribution (plot.type="CDFs: Observed and Fitted"). See the help file for cdfCompare.
- Observed quantiles vs. fitted quantiles (Q-Q Plot) (plot.type="Q-Q Plot"). See the help file for qqPlot.
- Tukey mean-difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot"). See the help file for qqPlot.
- Results of the goodness-of-fit test (plot.type="Test Results"). See the help file for print.gof.

See the help file for gofTest for more information.

Value

plot.gof invisibly returns the first argument, x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

gofTest, gof.object, print.gof, Goodness-of-Fit Tests, plot.

Examples

```
# Create an object of class "gof" then plot the results.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)
dat <- rnorm(20, mean = 3, sd = 2)
gof.obj <- gofTest(dat)
```

```
# Summary plot (the default)
#---------------------------
dev.new()
plot(gof.obj)


# Make your own titles for the summary plot
#------------------------------------------
dev.new()
plot(gof.obj, captions = list(PDFs = "Compare PDFs",
  CDFs = "Compare CDFs", QQ = "Q-Q Plot", Results = "Results"),
  om.title = "Summary")


# Just the Q-Q Plot
#------------------
dev.new()
plot(gof.obj, plot.type="Q-Q")


# Make your own title for the Q-Q Plot
#-------------------------------------
dev.new()
plot(gof.obj, plot.type="Q-Q", main = "Q-Q Plot")

#==========

# Clean up
#---------
rm(dat, gof.obj)
graphics.off()
```

`plot.gofCensored` *Plot Results of Goodness-of-Fit Test Based on Censored Data*

Description

Plot the results of calling the function `gofTestCensored`, which returns an object of class `"gofCensored"` when testing the goodness-of-fit of a set of data to a distribution. Five different kinds of plots are available.

The function `plot.gofCensored` is automatically called by `plot` when given an object of class `"gofCensored"`.

Usage

```
## S3 method for class 'gofCensored'
plot(x, plot.type = "Summary",
  captions = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL, Results = NULL),
```

```
  x.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
  y.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
  same.window = FALSE, ask = same.window & plot.type == "All", hist.col = "cyan",
  fitted.pdf.col = "black", fitted.pdf.lwd = 3 * par("cex"), fitted.pdf.lty = 1,
  prob.method = "michael-schucany", plot.pos.con = 0.375, ecdf.col = "cyan",
  fitted.cdf.col = "black", ecdf.lwd = 3 * par("cex"),
 fitted.cdf.lwd = 3 * par("cex"), ecdf.lty = 1, fitted.cdf.lty = 2, add.line = TRUE,
 digits = ifelse(plot.type == "Summary", 2, .Options$digits), test.result.font = 1,
  test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
  test.result.mar = c(0, 0, 3, 0) + 0.1,
  cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
  cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
  cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
 main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, add.om.title = TRUE,
 oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 4, 0) else c(0, 0, 0, 0),
  om.title = NULL, om.font = 2, om.cex.main = 1.5 * par("cex"), om.line = 0, ...)
```

Arguments

x	an object of class "gofCensored". See gofCensored.object for details.
plot.type	character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Summary" (the default), "PDFs: Observed and Fitted", "CDFs: Observed and Fitted", "Q-Q Plot", "Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.
captions	a list with 1 to 5 components with the names "PDFs", "CDFs", "QQ", "MDQQ", and/or "Results". Each component either has the value NULL or else it is a character string containing the title for that particular kind of plot. When the component has the value NULL (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gofCensored (i.e., when plot.type="Summary" or plot.type="All") and you want to specify titles different from the default ones. If you are creating only one kind of plot, then you can just use the main argument to specify a title different from the default one.
x.labels	a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the x-axis for that particular kind of plot. When the component has the value NULL (the default), a default x-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gofCensored (i.e., when plot.type="Summary" or plot.type="All") and you want to specify x-axis labels different from the default ones. If you are creating only one plot, then you can just use the xlab argument to specify an x-axis label different from the default one.
y.labels	a list of 1 to 4 components with the names "PDFs", "CDFs", "QQ", and/or "MDQQ". Each component either has the value NULL or else it is a character string containing the label for the y-axis for that particular kind of plot. When the component has the value NULL (the default), a default y-axis label

is used. This argument is useful when you are creating more than one kind of plot with a single call to `plot.gofCensored` (i.e., when `plot.type="Summary"` or `plot.type="All"`) and you want to specify y-axis labels different from the default ones. If you are creating only one plot, then you can just use the `ylab` argument to specify a y-axis label different from the default one.

`same.window` logical scalar indicating whether to produce all plots in the same graphics window (`same.window=TRUE`), or to create a new graphics window for each separate plot (`same.window=FALSE`; the default). The argument is relevant only when `plot.type="All"`.

`ask` logical scalar supplied to the function `devAskNewPage`, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is `FALSE` unless `same.window=TRUE` and `plot.type == "All"`.

`digits` scalar indicating how many significant digits to print for the distribution parameters. If `plot.type == "Summary"`, the default value is `digits=2`, otherwise it is `.Options$digits` (i.e., the current setting of `options("digits")`). This argument is ignored when `plot.type="PDFs: Observed and Fitted"`.

Arguments associated with `plot.type="PDFs: Observed and Fitted"`**:**

`hist.col` a character string or numeric scalar determining the color of the histogram used to display the distribution of the observed values. The default value is `hist.col="cyan"`. See the entry for `col` in the R help file for `par` for more information.

`fitted.pdf.col` a character string or numeric scalar determining the color of the fitted PDF (which is displayed as a line for continuous distributions and a histogram for discrete distributions). The default value is `fitted.pdf.col="black"`. See the entry for `col` in the R help file for `par` for more information.

`fitted.pdf.lwd` numeric scalar determining the width of the line used to display the fitted PDF. The default value is `fitted.pdf.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

`fitted.pdf.lty` numeric scalar determining the line type used to display the fitted PDF. The default value is `fitted.pdf.lty=1`. See the entry for `lty` in the R help file for `par` for more information.

Arguments associated with `plot.type="CDFs: Observed and Fitted"`**:**

`prob.method` character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are:
`"kaplan-meier"` (product-limit method of Kaplan and Meier (1958)),
`"modified kaplan-meier"` (modification of Kaplan-Meier method),
`"nelson"` (hazard plotting method of Nelson (1972)),
`"michael-schucany"` (generalization of the product-limit method due to Michael and Schucany (1986)), and
`"hirsch-stedinger"` (generalization of the product-limit method due to Hirsch and Stedinger (1987)).
The default value is `prob.method="michael-schucany"`.

The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" method is only available for censoring.side="left". See the help file for ppointsCensored for more explanation.

NOTE: This argument is also used to determine the plotting position method for the Q-Q plot (plot.type="Q-Q Plot"), or the Tukey Mean-Difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot").

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the observed (empirical) CDF. The default value is plot.pos.con=0.375. See the help files for ecdfPlot and qqPlot for more information and the motivation for this choice of value.

This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

NOTE: This argument is also used to determine the value of the plotting position constant for the Q-Q plot (plot.type="Q-Q Plot"), or the Tukey Mean-Difference Q-Q plot (plot.type="Tukey M-D Q-Q Plot").

ecdf.col a character string or numeric scalar determining the color of the line used to display the empirical CDF. The default value is ecdf.col="cyan". See the entry for col in the R help file for par for more information.

fitted.cdf.col a character string or numeric scalar determining the color of the line used to display the fitted CDF. The default value is fitted.cdf.col="black". See the entry for col in the R help file for par for more information.

ecdf.lwd numeric scalar determining the width of the line used to display the empirical CDF. The default value is ecdf.lwd=3*par("cex"). See the entry for lwd in the R help file for par for more information.

fitted.cdf.lwd numeric scalar determining the width of the line used to display the fitted CDF. The default value is fitted.cdf.lwd=3*par("cex"). See the entry for lwd in the R help file for par for more information.

ecdf.lty numeric scalar determining the line type used to display the empirical CDF. The default value is ecdf.lty=1. See the entry for lty in the R help file for par for more information.

fitted.cdf.lty numeric scalar determining the line type used to display the fitted CDF. The default value is fitted.cdf.lty=2. See the entry for lty in the R help file for par for more information.

Arguments associated with plot.type="Q-Q Plot" **or** plot.type="Tukey M-D Q-Q Plot"**:**

As explained above, prob.method and plot.pos.con are used for these plot types. Also:

add.line logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q Plot", a 0-1 line is added to the plot. If add.line=TRUE and plot.type="Tukey M-D Q-Q Plot", a horizontal line at $y = 0$ is added to the plot. The default value is add.line=TRUE.

Arguments associated with plot.type="Test Results"

test.result.font
: numeric scalar indicating which font to use to print out the test results. The default value is test.result.font=1. See the description of the font argument in the help file for par for more information. You may get better results if you use a font number that corresponds to a fixed font (e.g., courier).

test.result.cex
: numeric scalar indicating the value of cex to use to print out the test results. The default value is 0.9*par("cex") when plot.type="Summary", otherwise it is par("cex"). See the description of the cex argument in the help file for par for more information.

test.result.mar
: numeric vector indicating the value of mar to use to print out the test results. The default value is test.result.mar=c(0, 0, 3, 0) + 0.1. See the description of the mar argument in the help file for par for more information.

Arguments associated with plot.type="Summary"

add.om.title
: logical scalar indicating whether to add a title in the outer margin when plot.type="Summary". The default value is add.om.title=TRUE.

om.title
: character string containing the outer margin title. The default value is om.title=NULL, which will result in a default title.

om.font
: numeric scalar indicating the font to use for the outer margin. The default value is om.font=2.

om.cex.main
: numeric scalar indicating the value of cex for the outer margin title. The default value is 1.75 * par("cex").

om.line
: numeric scalar indicating the line to place the outer margin title on. The default value is om.line=0.5.

Graphics parameters:

cex.main, cex.axis, cex.lab, main, xlab, ylab, xlim, ylim, oma, ...
: additional graphics parameters. See the help file for par.

Details

The function plot.gofCensored is a method for the generic function plot for objects that inherit from the class "gofCensored" (see gofCensored.object). It can be invoked by calling plot and giving it an object of class "gofCensored" as the first argument, or by calling plot.gofCensored directly, regardless of the class of the object given as the first argument to plot.gofCensored.

Plots associated with the goodness-of-fit test are produced on the current graphics device. These can be one or all of the following:

- Observed distribution overlaid with fitted distribution (plot.type="PDFs: Observed and Fitted"). See the help files for hist and pdfPlot. **Note:** This kind of plot is only available for singly-censored data.

- Observed empirical distribution overlaid with fitted cumulative distribution (`plot.type="CDFs: Observed and Fitted"`). See the help file for `cdfCompareCensored`.
- Observed quantiles vs. fitted quantiles (Q-Q Plot) (`plot.type="Q-Q Plot"`). See the help file for `qqPlotCensored`.
- Tukey mean-difference Q-Q plot (`plot.type="Tukey M-D Q-Q Plot"`). See the help file for `qqPlotCensored`.
- Results of the goodness-of-fit test (`plot.type="Test Results"`). See the help file for `print.gofCensored`.

See the help file for `gofTestCensored` for more information.

Value

`plot.gofCensored` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`gofTestCensored`, `gofCensored.object`, `print.gofCensored`, Censored Data, `plot`.

Examples

```
# Create an object of class "gofCensored", then plot the results.
#----------------------------------------------------------------

gofCensored.obj <- with(EPA.09.Ex.15.1.manganese.df,
  gofTestCensored(Manganese.ppb, Censored, test = "sf"))

mode(gofCensored.obj)
#[1] "list"

class(gofCensored.obj)
#[1] "gofCensored"

# Summary plot (the default)
#---------------------------
dev.new()
plot(gofCensored.obj)


# Make your own titles for the summary plot
#------------------------------------------
dev.new()
plot(gofCensored.obj, captions = list(CDFs = "Compare CDFs",
  QQ = "Q-Q Plot", Results = "Results"), om.title = "Summary")
```

```
# Just the Q-Q Plot
#------------------
dev.new()
plot(gofCensored.obj, plot.type="Q-Q")


# Make your own title for the Q-Q Plot
#-------------------------------------
dev.new()
plot(gofCensored.obj, plot.type="Q-Q", main = "Q-Q Plot")

#==========

# Clean up
#---------
rm(gofCensored.obj)
graphics.off()
```

---

plot.gofGroup *Plot Results of Group Goodness-of-Fit Test*

Description

Plot the results of calling the function gofGroupTest, which returns an object of class "gofGroup" when performing a goodness-of-fit test to determine whether data in a set of groups appear to all come from the same probability distribution (with possibly different parameters for each group). Five different kinds of plots are available.

The function plot.gofGroup is automatically called by plot when given an object of class "gofGroup". The names of other functions associated with goodness-of-fit test are listed under Goodness-of-Fit Tests.

Usage

```
## S3 method for class 'gofGroup'
plot(x, plot.type = "Summary",
    captions = list(QQ = NULL, MDQQ = NULL, ScoresQQ = NULL, ScoresMDQQ = NULL,
      Results = NULL),
    x.labels = list(QQ = NULL, MDQQ = NULL, ScoresQQ = NULL, ScoresMDQQ = NULL),
    y.labels = list(QQ = NULL, MDQQ = NULL, ScoresQQ = NULL, ScoresMDQQ = NULL),
    same.window = FALSE, ask = same.window & plot.type == "All", add.line = TRUE,
   digits = ifelse(plot.type == "Summary", 2, .Options$digits), test.result.font = 1,
    test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
    test.result.mar = c(0, 0, 3, 0) + 0.1, individual.p.values = FALSE,
    cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
    cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
    cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
```

```
main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, add.om.title = TRUE,
oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 5, 0) else c(0, 0, 0, 0),
 om.title = NULL, om.font = 2, om.cex.main = 1.5 * par("cex"), om.line = 1, ...)
```

Arguments

x	an object of class "gofGroup". See gofGroup.object for details.
plot.type	character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Summary" (the default), "Q-Q Plot" "Tukey M-D Q-Q Plot", "Scores Q-Q Plot", "Scores Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.
captions	a list with 1 to 5 components with the names "QQ", "MDQQ", "ScoresQQ", "ScoresMDQQ", and/or "Results". Each component either has the value NULL or else it is a character string containing the title for that particular kind of plot. When the component has the value NULL (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gofGroup (i.e., when plot.type="Summary" or plot.type="All") and you want to specify titles different from the default ones. If you are creating only one kind of plot, then you can just use the main argument to specify a title different from the default one.
x.labels	a list of 1 to 4 components with the names "QQ", "MDQQ", "ScoresQQ", and/or "ScoresMDQQ". Each component either has the value NULL or else it is a character string containing the label for the x-axis for that particular kind of plot. When the component has the value NULL (the default), a default x-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gofGroup (i.e., when plot.type="Summary" or plot.type="All") and you want to specify x-axis labels different from the default ones. If you are creating only one plot, then you can just use the xlab argument to specify an x-axis label different from the default one.
y.labels	a list of 1 to 4 components with the names "QQ", "MDQQ", "ScoresQQ", and/or "ScoresMDQQ". Each component either has the value NULL or else it is a character string containing the label for the y-axis for that particular kind of plot. When the component has the value NULL (the default), a default y-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gofGroup (i.e., when plot.type="Summary" or plot.type="All") and you want to specify y-axis labels different from the default ones. If you are creating only one plot, then you can just use the ylab argument to specify a y-axis label different from the default one.
same.window	logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE), or to create a new graphics window for each separate plot (same.window=FALSE; the default). The argument is relevant only when plot.type="All".
ask	logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is FALSE unless same.window=TRUE and plot.type == "All".

add.line logical scalar indicating whether to add a line to the plot. If add.line=TRUE and plot.type="Q-Q Plot" or plot.type="Scores Q-Q Plot", a 0-1 line is added to the plot. If add.line=TRUE and plot.type="Tukey M-D Q-Q Plot" or plot.type="Scores Tukey M-D Q-Q Plot", a horizontal line at $y = 0$ is added to the plot. The default value is add.line=TRUE. This argument is ignored if plot.type="Test Results".

Arguments associated with plot.type="Test Results"

digits scalar indicating how many significant digits to print for the test results when plot.type="Summary" or plot.type="Test Results". If plot.type == "Summary", the default value is digits=2, otherwise it is .Options$digits (i.e., the current setting of options("digits")).

individual.p.values
logical scalar indicating whether to display the p-values associated with each individual group. The default value is individual.p.values=FALSE.

test.result.font
numeric scalar indicating which font to use to print out the test results. The default value is test.result.font=1. See the description of the font argument in the help file for par for more information. You may get better results if you use a font number that corresponds to a fixed font (e.g., courier).

test.result.cex
numeric scalar indicating the value of cex to use to print out the test results. The default value is 0.9*par("cex") when plot.type="Summary", otherwise it is par("cex"). See the description of the cex argument in the help file for par for more information.

test.result.mar
numeric vector indicating the value of mar to use to print out the test results. The default value is test.result.mar=c(0, 0, 3, 0)+0.1. See the description of the mar argument in the help file for par for more information.

Arguments associated with plot.type="Summary"

add.om.title logical scalar indicating whether to add a title in the outer margin when plot.type="Summary". The default value is add.om.title=TRUE.

om.title character string containing the outer margin title. The default value is om.title=NULL, which will result in a default title.

om.font numeric scalar indicating the font to use for the outer margin. The default value is om.font=2.

om.cex.main numeric scalar indicating the value of cex for the outer margin title. The default value is 1.5 * par("cex").

om.line numeric scalar indicating the line to place the outer margin title on. The default value is om.line=1.

Graphics parameters:

cex.main, cex.axis, cex.lab, main, xlab, ylab, xlim, ylim, oma, ...
additional graphics parameters. See the help file for par.

Details

The function plot.gofGroup is a method for the generic function plot for the class "gofGroup" (see gofGroup.object). It can be invoked by calling plot and giving it an object of class "gofGroup" as the first argument, or by calling plot.gofGroup directly, regardless of the class of the object given as the first argument to plot.gofGroup.

Plots associated with the goodness-of-fit test are produced on the current graphics device. These can be one or all of the following:

- plot.type="Q-Q Plot". Q-Q Plot of observed p-values vs. quantiles from a Uniform [0,1] distribution. See the help file for qqPlot.
- plot.type="Tukey M-D Q-Q Plot". Tukey mean-difference Q-Q plot for observed p-values and quantiles from a Uniform [0,1] distribution. See the help file for qqPlot.
- plot.type="Scores Q-Q Plot". Q-Q Plot of Normal scores vs. quantiles from a Normal(0,1) distribution or Q-Q Plot of Chisquare scores vs. quantiles from a Chisquare distribution with 2 degrees of freedom. See the help file for qqPlot.
- plot.type="Scores Tukey M-D Q-Q Plot". Tukey mean-difference Q-Q plot based on Normal scores or Chisquare scores. See the help file for qqPlot.
- Results of the goodness-of-fit test (plot.type="Test Results"). See the help file for print.gofGroup.

See the help file for gofGroupTest for more information.

Value

plot.gofGroup invisibly returns the first argument, x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

gofGroupTest, gofGroup.object, print.gofGroup, Goodness-of-Fit Tests, plot.

Examples

```
# Create an object of class "gofGroup" then plot it.

# Example 10-4 of USEPA (2009, page 10-20) gives an example of
# simultaneously testing the assumption of normality for nickel
# concentrations (ppb) in groundwater collected at 4 monitoring
# wells over 5 months.  The data for this example are stored in
# EPA.09.Ex.10.1.nickel.df.

EPA.09.Ex.10.1.nickel.df
#   Month   Well Nickel.ppb
```

```
#1      1 Well.1       58.8
#2      3 Well.1        1.0
#3      6 Well.1      262.0
#...
#18     6 Well.4       85.6
#19     8 Well.4       10.0
#20    10 Well.4      637.0


# Test for a normal distribution at each well:
#---------------------------------------------

gofGroup.obj <- gofGroupTest(Nickel.ppb ~ Well,
  data = EPA.09.Ex.10.1.nickel.df)

dev.new()
plot(gofGroup.obj)

# Make your own titles for the summary plot
#------------------------------------------
dev.new()
plot(gofGroup.obj, captions = list(QQ = "Q-Q Plot",
  ScoresQQ = "Scores Q-Q Plot", Results = "Results"),
  om.title = "Summary Plot")

# Just the Q-Q Plot
#------------------
dev.new()
plot(gofGroup.obj, plot.type="Q-Q")


# Make your own title for the Q-Q Plot
#-------------------------------------
dev.new()
plot(gofGroup.obj, plot.type="Q-Q", main = "Q-Q Plot")

#==========

# Clean up
#---------
rm(gofGroup.obj)
graphics.off()
```

`plot.gofTwoSample` *Plot Results of Goodness-of-Fit Test to Compare Two Samples*

Description

Plot the results of calling the function `gofTest` to compare two samples. `gofTest` returns an object of class `"gofTwoSample"` when supplied with both the arguments `y` and `x`. `plot.gofTwoSample` provides five different kinds of plots.

The function plot.gofTwoSample is automatically called by plot when given an object of class "gofTwoSample". The names of other functions associated with goodness-of-fit test are listed under Goodness-of-Fit Tests.

Usage

```
## S3 method for class 'gofTwoSample'
plot(x, plot.type = "Summary",
   captions = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL, Results = NULL),
    x.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
    y.labels = list(PDFs = NULL, CDFs = NULL, QQ = NULL, MDQQ = NULL),
   same.window = FALSE, ask = same.window & plot.type == "All", x.points.col = "blue",
   y.points.col = "black", points.pch = 1, jitter.points = TRUE, discrete = FALSE,
    plot.pos.con = 0.375, x.ecdf.col = "blue", y.ecdf.col = "black",
    x.ecdf.lwd = 3 * par("cex"), y.ecdf.lwd = 3 * par("cex"), x.ecdf.lty = 1,
    y.ecdf.lty = 4, add.line = TRUE,
   digits = ifelse(plot.type == "Summary", 2, .Options$digits), test.result.font = 1,
    test.result.cex = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
    test.result.mar = c(0, 0, 3, 0) + 0.1,
    cex.main = ifelse(plot.type == "Summary", 1.2, 1.5) * par("cex"),
    cex.axis = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
    cex.lab = ifelse(plot.type == "Summary", 0.9, 1) * par("cex"),
    main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
    add.om.title = TRUE,
   oma = if (plot.type == "Summary" & add.om.title) c(0, 0, 4, 0) else c(0, 0, 0, 0),
   om.title = NULL, om.font = 2, om.cex.main = 1.5 * par("cex"), om.line = 0, ...)
```

Arguments

x	an object of class "gof". See gof.object for details.
plot.type	character string indicating what kind of plot to create. Only one particular plot type will be created, unless plot.type="All", in which case all plots will be created sequentially. The possible values of plot.type are: "Summary" (the default), "PDFs: Observed", "CDFs: Observed", "Q-Q Plot", "Tukey M-D Q-Q Plot", "Test Results", and "All". See the DETAILS section for more information.
captions	a list with 1 to 5 components with the names "PDFs", "CDFs", "QQ", "MDQQ", and/or "Results". Each component either has the value NULL or else it is a character string containing the title for that particular kind of plot. When the component has the value NULL (the default), a default title is used. This argument is useful when you are creating more than one kind of plot with a single call to plot.gofTwoSample (i.e., when plot.type="Summary" or plot.type="All") and you want to specify titles different from the default ones. If you are creating

only one kind of plot, then you can just use the `main` argument to specify a title different from the default one.

`x.labels` a list of 1 to 4 components with the names `"PDFs"`, `"CDFs"`, `"QQ"`, and/or `"MDQQ"`. Each component either has the value `NULL` or else it is a character string containing the label for the x-axis for that particular kind of plot. When the component has the value `NULL` (the default), a default x-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to `plot.gofTwoSample` (i.e., when `plot.type="Summary"` or `plot.type="All"`) and you want to specify x-axis labels different from the default ones. If you are creating only one plot, then you can just use the `xlab` argument to specify an x-axis label different from the default one.

`y.labels` a list of 1 to 4 components with the names `"PDFs"`, `"CDFs"`, `"QQ"`, and/or `"MDQQ"`. Each component either has the value `NULL` or else it is a character string containing the label for the y-axis for that particular kind of plot. When the component has the value `NULL` (the default), a default y-axis label is used. This argument is useful when you are creating more than one kind of plot with a single call to `plot.gofTwoSample` (i.e., when `plot.type="Summary"` or `plot.type="All"`) and you want to specify y-axis labels different from the default ones. If you are creating only one plot, then you can just use the `ylab` argument to specify a y-axis label different from the default one.

`same.window` logical scalar indicating whether to produce all plots in the same graphics window (`same.window=TRUE`), or to create a new graphics window for each separate plot (`same.window=FALSE`; the default). The argument is relevant only when `plot.type="All"`.

`ask` logical scalar supplied to the function `devAskNewPage`, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is `FALSE` unless `same.window=TRUE` and `plot.type == "All"`.

Arguments associated with `plot.type="PDFs: Observed"`:

`x.points.col` a character string or numeric scalar determining the color of the plotting symbol used to display the distribution of the observed `x` values that were supplied to `gofTest`. The default value is `x.points.col="blue"`. See the entry for `col` in the R help file for `par` for more information.

`y.points.col` a character string or numeric scalar determining the color of the plotting symbol used to display the distribution of the observed `y` values that were supplied to `gofTest`. The default value is `y.points.col="black"`. See the entry for `col` in the R help file for `par` for more information.

`points.pch` a character string or numeric scalar determining the plotting symbol used to display the distribution of the observed `x` and `y` values that were supplied to `gofTest`. The default value is `points.pch=1`. See the entry for `pch` in the R help file for `par` for more information.

`jitter.points` logical scalar indicating whether to jitter the points in the strip chart. The default value is `jitter.points=TRUE`.

Arguments associated with `plot.type="CDFs: Observed"`:

`discrete` logical scalar indicating whether the two distributions are considered to be discrete (`discrete=TRUE`) or not(`discrete=FALSE`; the default). When `discrete=TRUE`, the empirical CDFs are plotted as step functions.

`plot.pos.con` numeric scalar between 0 and 1 containing the value of the plotting position constant used to construct the observed (empirical) CDFs. The default value is `plot.pos.con=0.375`. See the help files for `ecdfPlot` and `qqPlot` for more information and the motivation for this choice of values.

NOTE: This argument is also used to determine the value of the plotting position constant for the Q-Q plot (`plot.type="Q-Q Plot"`), or the Tukey Mean-Difference Q-Q plot (`plot.type="Tukey M-D Q-Q Plot"`).

`x.ecdf.col` a character string or numeric scalar determining the color of the line used to display the empirical CDF for the `x` values that were supplied to `gofTest`. The default value is `x.ecdf.col="blue"`. See the entry for `col` in the R help file for `par` for more information.

`y.ecdf.col` a character string or numeric scalar determining the color of the line used to display the empirical CDF for the `y` values that were supplied to `gofTest`. The default value is `y.ecdf.col="black"`. See the entry for `col` in the R help file for `par` for more information.

`x.ecdf.lwd` numeric scalar determining the width of the line used to display the empirical CDF for the `x` values that were supplied to `gofTest`. The default value is `x.ecdf.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

`y.ecdf.lwd` numeric scalar determining the width of the line used to display the empirical CDF for the `y` values that were supplied to `gofTest`. The default value is `y.ecdf.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

`x.ecdf.lty` numeric scalar determining the line type used to display the empirical CDF for the `x` values that were supplied to `gofTest`. The default value is `x.ecdf.lty=1`. See the entry for `lty` in the R help file for `par` for more information.

`y.ecdf.lty` numeric scalar determining the line type used to display the empirical CDF for the `y` values that were supplied to `gofTest`. The default value is `y.ecdf.lty=4`. See the entry for `lty` in the R help file for `par` for more information.

Arguments associated with `plot.type="Q-Q Plot"` or `plot.type="Tukey M-D Q-Q Plot"`:

As explained above, `plot.pos.con` is used for these plot types. Also:

`add.line` logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q Plot"`, a 0-1 line is added to the plot. If `add.line=TRUE` and `plot.type="Tukey M-D Q-Q Plot"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=TRUE`.

Arguments associated with `plot.type="Test Results"`

`digits` scalar indicating how many significant digits to print for the test results when `plot.type="Summary"` or `plot.type="Test Results"`. If `plot.type == "Summary"`, the default value is `digits=2`, otherwise it is `.Options$digits` (i.e., the current setting of `options("digits")`).

`test.result.font`
numeric scalar indicating which font to use to print out the test results. The default value is `test.result.font=1`. See the description of the `font` argument in the help file for `par` for more information. You may get better results if you use a font number that corresponds to a fixed font (e.g., courier).

`test.result.cex`
numeric scalar indicating the value of `cex` to use to print out the test results. The default value is `0.9*par("cex")` when `plot.type="Summary"`, otherwise it is `par("cex")`. See the description of the `cex` argument in the help file for `par` for more information.

`test.result.mar`
numeric vector indicating the value of `mar` to use to print out the test results. The default value is `test.result.mar=c(0, 0, 3, 0)+0.1`. See the description of the `mar` argument in the help file for `par` for more information.

Arguments associated with `plot.type="Summary"`

`add.om.title` logical scalar indicating whether to add a title in the outer margin when `plot.type="Summary"`. The default value is `add.om.title=TRUE`.

`om.title` character string containing the outer margin title. The default value is `om.title=NULL`, which will result in a default title.

`om.font` numeric scalar indicating the font to use for the outer margin. The default value is `om.font=2`.

`om.cex.main` numeric scalar indicating the value of `cex` for the outer margin title. The default value is `1.75 * par("cex")`.

`om.line` numeric scalar indicating the line to place the outer margin title on. The default value is `om.line=0.5`.

Graphics parameters:

`cex.main`, `cex.axis`, `cex.lab`, `main`, `xlab`, `ylab`, `xlim`, `ylim`, `oma`, `...`
additional graphics parameters. See the help file for `par`.

Details

The function `plot.gofTwoSample` is a method for the generic function `plot` for the class `"gofTwoSample"` (see `gofTwoSample.object`). It can be invoked by calling `plot` and giving it an object of class `"gofTwoSample"` as the first argument, or by calling `plot.gofTwoSample` directly, regardless of the class of the object given as the first argument to `plot.gofTwoSample`.

Plots associated with the goodness-of-fit test are produced on the current graphics device. These can be one or all of the following:

- Observed distributions (`plot.type="PDFs: Observed"`).
- Observed CDFs (`plot.type="CDFs: Observed"`). See the help file for `cdfCompare`.
- Q-Q Plot (`plot.type="Q-Q Plot"`). See the help file for `qqPlot`.
- Tukey mean-difference Q-Q plot (`plot.type="Tukey M-D Q-Q Plot"`). See the help file for `qqPlot`.
- Results of the goodness-of-fit test (`plot.type="Test Results"`). See the help file for `print.gofTwoSample`.

See the help file for `gofTest` for more information.

Value

`plot.gofTwoSample` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`gofTest`, `gofTwoSample.object`, `print.gofTwoSample`, Goodness-of-Fit Tests, `plot`.

Examples

```
# Create an object of class "gofTwoSample" then plot the results.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(300)
dat1 <- rnorm(20, mean = 3, sd = 2)
dat2 <- rnorm(10, mean = 1, sd = 2)
gof.obj <- gofTest(x = dat1, y = dat2)

# Summary plot (the default)
#---------------------------
dev.new()
plot(gof.obj)


# Make your own titles for the summary plot
#------------------------------------------
dev.new()
plot(gof.obj, captions = list(PDFs = "Compare PDFs",
  CDFs = "Compare CDFs", QQ = "Q-Q Plot", Results = "Results"),
  om.title = "Summary Plot")


# Just the Q-Q Plot
```

```
#------------------
dev.new()
plot(gof.obj, plot.type="Q-Q")


# Make your own title for the Q-Q Plot
#-------------------------------------
dev.new()
plot(gof.obj, plot.type="Q-Q", main = "Q-Q Plot")

#==========

# Clean up
#---------
rm(dat1, dat2, gof.obj)
graphics.off()
```

`plot.permutationTest` *Plot Results of Permutation Test*

Description

Plot the results of calling functions that return an object of class `"permutationTest"`. Currently, the **EnvStats** functions that perform permutation tests and produce objects of class `"permutationTest"` are: `oneSamplePermutationTest`, `twoSamplePermutationTestLocation`, and `twoSamplePermutationTestProportion`.

The function `plot.permutationTest` is automatically called by `plot` when given an object of class `"permutationTest"`.

Usage

```
## S3 method for class 'permutationTest'
plot(x, hist.col = "cyan", stat.col = "black",
  stat.lwd = 3 * par("cex"), stat.lty = 1, cex.main = par("cex"),
  digits = .Options$digits, main = NULL, xlab = NULL, ylab = NULL,
  xlim = NULL, ylim = NULL, ...)
```

Arguments

`x`	an object of class `"permutationTest"`. See `permutationTest.object` for details.
`hist.col`	a character string or numeric scalar determining the color of the histogram used to display the permutation distribution. The default value is `hist.col="cyan"`. See the entry for `col` in the R help file for `par` for more information.
`stat.col`	a character string or numeric scalar determining the color of the line indicating the value of the observed test statistic. The default value is `stat.col="black"`. See the entry for `col` in the R help file for `par` for more information.

`stat.lwd` numeric scalar determining the width of the line indicating the value of the observed test statistic. The default value is `stat.lwd=3*par("cex")`. See the entry for `lwd` in the R help file for `par` for more information.

`stat.lty` numeric scalar determining the line type used to display the value of the observed test statistic. The default value is `stat.lty=1`. See the entry for `lty` in the R help file for `par` for more information.

`digits` scalar indicating how many significant digits to print for the distribution parameters. The default value is `.Options$digits` (i.e., the current setting of `options("digits")`).

`cex.main`, `main`, `xlab`, `ylab`, `xlim`, `ylim`, `...`
graphics parameters. See the help file for `par`.

Details

Produces a plot displaying the permutation distribution (`exact=TRUE`) or a sample of the permutation distribution (`exact=FALSE`), and a line indicating the observed value of the test statistic. The title in the plot includes information on the data used, null hypothesis, and p-value.

The function `plot.permutationTest` is a method for the generic function `plot` for the class `"permutationTest"` (see `permutationTest.object`). It can be invoked by calling `plot` and giving it an object of class `"permutationTest"` as the first argument, or by calling `plot.permutationTest` directly, regardless of the class of the object given as the first argument to `plot.permutationTest`.

Value

`plot.permutationTest` invisibly returns the first argument, `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`permutationTest.object`, `print.permutationTest`, `oneSamplePermutationTest`, `twoSamplePermutationTestLocation`, `twoSamplePermutationTestProportion`, Hypothesis Tests, `plot`.

Examples

```
# Create an object of class "permutationTest", then print it and plot it.
# (Note:  the call to set.seed() allows you to reproduce this example.)
#-----------------------------------------------------------------------

set.seed(23)
```

```
dat <- rlogis(10, location = 7, scale = 2)

permutationTest.obj <- oneSamplePermutationTest(dat, mu = 5,
  alternative = "greater", exact = TRUE)

mode(permutationTest.obj)
#[1] "list"

class(permutationTest.obj)
#[1] "permutationTest"

names(permutationTest.obj)
# [1] "statistic"         "parameters"        "p.value"
# [4] "estimate"          "null.value"        "alternative"
# [7] "method"            "estimation.method" "sample.size"
#[10] "data.name"         "bad.obs"           "stat.dist"
#[13] "exact"

#==========

# Print the results of the test
#------------------------------
permutationTest.obj

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 Mean (Median) = 5
#
#Alternative Hypothesis:          True Mean (Median) is greater than 5
#
#Test Name:                       One-Sample Permutation Test
#                                 (Exact)
#
#Estimated Parameter(s):          Mean = 9.977294
#
#Data:                            dat
#
#Sample Size:                     10
#
#Test Statistic:                  Sum(x - 5) = 49.77294
#
#P-value:                         0.001953125

#==========

# Plot the results of the test
#-----------------------------
dev.new()
plot(permutationTest.obj)

#==========
```

```
# Extract the test statistic
#---------------------------

permutationTest.obj$statistic
#Sum(x - 5)
#  49.77294

#==========

# Clean up
#---------
rm(permutationTest.obj)
graphics.off()
```

---

plotAovDesign *Create Plots for a Sampling Design Based on a One-Way Fixed-Effects Analysis of Variance*

Description

Create plots involving sample size, power, scaled difference, and significance level for a one-way fixed-effects analysis of variance.

Usage

```
plotAovDesign(x.var = "n", y.var = "power", range.x.var = NULL,
  n.vec = c(25, 25), mu.vec = c(0, 1), sigma = 1, alpha = 0.05, power = 0.95,
  round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE,
  add = FALSE, n.points = 50, plot.col = 1, plot.lwd = 3 * par("cex"),
 plot.lty = 1, digits = .Options$digits, main = NULL, xlab = NULL, ylab = NULL,
  type = "l", ...)
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "power" (power of the test), and "alpha" (significance level of the test).
y.var	character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default) and "n" (sample size).
range.x.var	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="power" the default value is c(alpha+.Machine$double.eps, 0.95). When x.var="alpha", the default value is c(0.01, 0.2).
n.vec	numeric vector indicating the sample size for each group. The default value is n.vec=c(25, 25). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument must be the same length as mu.vec. This argument is ignored if either x.var="n" or y.var="n".

`mu.vec` numeric vector indicating the population mean for each group. The default value is `mu.vec=c(0, 1)`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed. This argument must be the same length as `n.vec`.

`sigma` numeric scalar indicating the population standard deviation for all groups. The default value is `sigma=1`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.

`alpha` numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`. This argument is ignored when `x.var="alpha"`.

`power` numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`. This argument is ignored when `x.var="power"` or `y.var="power"`.

`round.up` logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is FALSE. This argument is ignored unless `y.var="n"`.

`n.max` for the case when `y.var="n"`, a positive integer greater than 2 indicating the maximum sample size per group. The default value is `n.max=5000`.

`tol` for the case when `y.var="n"`, numeric scalar indicating the tolerance to use in the `uniroot` search for the sample size. The default value is `tol=1e-7`.

`maxiter` for the case when `y.var="n"`, positive integer greater then 1 indicating the maximum number of iterations to use in the `uniroot` search for the sample size. The default value is `maxiter=1000`.

`plot.it` a logical scalar indicating whether to create a plot or add to the existing plot (see `add`) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is `plot.it=TRUE`.

`add` a logical scalar indicating whether to add the design plot to the existing plot (`add=TRUE`), or to create a new plot (`add=FALSE`). The default value is `add=FALSE`. This argument is ignored if `plot.it=FALSE`.

`n.points` a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are `n.points` x-values evenly spaced between `range.x.var[1]` and `range.x.var[2]`. The default value is `n.points=50`.

`plot.col` a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col=1`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits` a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`main, xlab, ylab, type, ...`
additional graphical parameters (see `par`).

Details

See the help files for aovPower and aovN for information on how to compute the power and sample size for a one-way fixed-effects analysis of variance.

Value

plotAovDesign invisibly returns a list with components:

x.var	x-coordinates of the points that have been or would have been plotted
y.var	y-coordinates of the points that have been or would have been plotted

Note

The normal and lognormal distribution are probably the two most frequently used distributions to model environmental data. Sometimes it is necessary to compare several means to determine whether any are significantly different from each other (e.g., USEPA, 2009, p.6-38). In this case, assuming normally distributed data, you perform a one-way parametric analysis of variance.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, Type I error level, power, and differences in means if one of the objectives of the sampling program is to determine whether a particular mean differs from a group of means. The functions aovPower, aovN, and plotAovDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapter 17.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York, Chapters 27, 29, 30.

Scheffe, H. (1959). *The Analysis of Variance*. John Wiley and Sons, New York, 477pp.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 10.

See Also

aovPower, aovN, Normal, aov.

Examples

```
# Look at the relationship between power and sample size
# for a one-way ANOVA, assuming k=2 groups, group means of
# 0 and 1, a population standard deviation of 1, and a
# 5% significance level:

dev.new()
plotAovDesign()

#----------------------------------------------------------------------

# Plot power vs. sample size for various levels of significance:

dev.new()
plotAovDesign(mu.vec = c(0, 0.5, 1), ylim=c(0, 1), main="")

plotAovDesign(mu.vec = c(0, 0.5, 1), alpha=0.1, add=TRUE, plot.col=2)

plotAovDesign(mu.vec = c(0, 0.5, 1), alpha=0.2, add=TRUE, plot.col=3)

legend(35, 0.6, c("20%", "10%", "   5%"), lty=1, lwd = 3, col=3:1,
  bty = "n")

mtext("Power vs. Sample Size for One-Way ANOVA", line = 3, cex = 1.25)
mtext(expression(paste("with ", mu, "=(0, 0.5, 1), ", sigma,
  "=1, and Various Significance Levels", sep="")),
  line = 1.5, cex = 1.25)

#----------------------------------------------------------------------

# The example on pages 5-11 to 5-14 of USEPA (1989b) shows
# log-transformed concentrations of lead (mg/L) at two
# background wells and four compliance wells, where
# observations were taken once per month over four months
# (the data are stored in EPA.89b.loglead.df).
# Assume the true mean levels at each well are
# 3.9, 3.9, 4.5, 4.5, 4.5, and 5, respectively.  Plot the
# power vs. sample size of a one-way ANOVA to test for mean
# differences between wells.  Use alpha=0.05, and assume the
# true standard deviation is equal to the one estimated
# from the data in this example.

names(EPA.89b.loglead.df)
#[1] "LogLead"   "Month"     "Well"      "Well.type"

# Perform the ANOVA and get the estimated sd
aov.list <- aov(LogLead ~ Well, data=EPA.89b.loglead.df)

summary(aov.list)
#            Df Sum Sq Mean Sq F value  Pr(>F)
#Well         5 5.7447 1.14895  3.3469 0.02599 *
#Residuals   18 6.1791 0.34328
```

```
#---
#Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Now create the plot
dev.new()
plotAovDesign(range.x.var = c(2, 20),
  mu.vec = c(3.9,3.9,4.5,4.5,4.5,5),
  sigma=sqrt(0.34),
  ylim = c(0, 1), digits=2)

# Clean up
#---------
rm(aov.list)
graphics.off()
```

`plotCiBinomDesign` *Plots for Sampling Design Based on Confidence Interval for Binomial Proportion or Difference Between Two Proportions*

Description

Create plots for a sampling design based on a confidence interval for a binomial proportion or the difference between two proportions.

Usage

```
plotCiBinomDesign(x.var = "n", y.var = "half.width",
  range.x.var = NULL, n.or.n1 = 25, p.hat.or.p1.hat = 0.5,
  n2 = n.or.n1, p2.hat = 0.4, ratio = 1, half.width = 0.05,
  conf.level = 0.95, sample.type = "one.sample", ci.method = "score",
  correct = TRUE, warn = TRUE, n.or.n1.min = 2,
  n.or.n1.max = 10000, tol.half.width = 0.005, tol.p.hat = 0.005,
  maxiter = 10000, plot.it = TRUE, add = FALSE, n.points = 100,
  plot.col = 1, plot.lwd = 3 * par("cex"), plot.lty = 1,
  digits = .Options$digits,
  main = NULL, xlab = NULL, ylab = NULL, type = "l", ...)
```

Arguments

`x.var`	character string indicating what variable to use for the x-axis. Possible values are `"n"` (sample size; the default), `"half.width"` (the half-width of the confidence interval), `"p.hat"` (the estimated probability of "success"), and `"conf.level"` (the confidence level).
`y.var`	character string indicating what variable to use for the y-axis. Possible values are `"half.width"` (the half-width of the confidence interval; the default), and `"n"` (sample size).

`range.x.var` numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of `x.var`.
When `x.var="n"` the default value is `c(10,50)`.
When `x.var="half.width"`, the default value is `c(0.03, 0.1)`.
When `x.var="p.hat"`, the default value is `c(0.5, 0.9)`.
When `x.var="conf.level"`, the default value is `c(0.8, 0.99)`.

`n.or.n1` numeric scalar indicating the sample size. The default value is `n.or.n1=25`. When `sample.type="one.sample"`, `n.or.n1` denotes the number of observations in the single sample. When `sample.type="two.sample"`, `n.or.n1` denotes the number of observations from group 1. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed. This argument is ignored if either `x.var="n"` or `y.var="n"`.

`p.hat.or.p1.hat`
numeric scalar indicating an estimated proportion.
When `sample.type="one.sample"`, `p.hat.or.p1.hat` denotes the estimated value of p, the probability of "success".
When `sample.type="two.sample"`, `p.hat.or.p1.hat` denotes the estimated value of p_1, the probability of "success" in group 1.
Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed. This argument is ignored if `x.var="p.hat"`.

`n2` numeric scalar indicating the sample size for group 2. The default value is the value of `n.or.n1`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed. This argument is ignored when `sample.type="one.sample"`.

`p2.hat` numeric scalar indicating the estimated proportion for group 2. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed. This argument is ignored if `sample.type="one.sample"`.

`ratio` numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 (n_2/n_1). The default value is `ratio=1`. All values of `ratio` must be greater than or equal to 1. This argument is only used when `sample.type="two.sample"` and either `x.var="n"` or `y.var="n"`.

`half.width` positive numeric scalar indicating the half-width of the confidence interval. The default value is `half.width=0.05`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed. This argument is ignored if either `x.var="half.width"` or `y.var="half.width"`.

`conf.level` a numeric scalar between 0 and 1 indicating the confidence level associated with the confidence intervals. The default value is `conf.level=0.95`. This argument is ignored when `x.var="conf.level"`.

`sample.type` character string indicating whether this is a one-sample or two-sample confidence interval. When `sample.type="one.sample"`, the computations for the plot are based on a confidence interval for a single proportion. When `sample.type="two.sample"`, the computations for the plot are based on a confidence interval for the difference between two proportions. The default value is `sample.type="one.sample"` unless the arguments `n2`, `p2.hat`, and/or `ratio` are supplied.

`ci.method` character string indicating which method to use to construct the confidence interval. Possible values are `"score"` (the default), `"exact"`, `"adjusted Wald"`,

and "Wald" (the "Wald" method is **never** recommended but is included for historical purposes). The exact method is only available for the one-sample case, i.e., when sample.type="one.sample".

correct — logical scalar indicating whether to use the continuity correction when ci.method="score" or ci.method="Wald". The default value is correct=TRUE. This argument is ignored if ci.method="exact" or ci.method="adjusted Wald".

warn — logical scalar indicating whether to issue a warning when ci.method="Wald" for cases when the normal approximation to the binomial distribution probably is not accurate. The default value is warn=TRUE.

n.or.n1.min — for the case when y.var="n", integer indicating the minimum allowed value for n (sample.type="one.sample") or n_1 (sample.type="two.sample"). The default value is n.or.n1.min=2.

n.or.n1.max — for the case when y.var="n", integer indicating the maximum allowed value for n (sample.type="one.sample") or n_1 (sample.type="two.sample"). The default value is n.or.n1.max=10000.

tol.half.width — for the case when y.var="n", numeric scalar indicating the tolerance to use for the half width for the search algorithm. The sample sizes are computed so that the actual half width is less than or equal to half.width + tol.half.width. The default value is tol.half.width=0.005.

tol.p.hat — for the case when y.var="n", numeric scalar indicating the tolerance to use for the estimated proportion(s) for the search algorithm. For the one-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of p.hat.or.p1.hat and the actual estimated proportion is less than or equal to tol.p.hat. For the two-sample case, the sample sizes are computed so that the absolute value of the difference between the user supplied value of p.hat.or.p1.hat and the actual estimated proportion for group 1 is less than or equal to tol.p.hat, and the absolute value of the difference between the user supplied value of p2.hat and the actual estimated proportion for group 2 is less than or equal to tol.p.hat. The default value is tol.p.hat=0.005.

maxiter — for the case when y.var="n", integer indicating the maximum number of iterations to use for the search algorithm. The default value is maxiter=1000.

plot.it — a logical scalar indicating whether to create a plot or add to the existing plot (see description of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the VALUE section below). The default value is plot.it=TRUE.

add — a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points — a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and

range.x.var[2]. The default value is n.points=100. This argument is ignored when x.var="n", in which case the x-values are all the integers between range.x.var[1] and range.x.var[2].

plot.col — a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col=1. See the entry for col in the help file for par for more information.

plot.lwd — a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty — a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits — a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ... — additional graphical parameters (see par).

Details

See the help files for ciBinomHalfWidth and ciBinomN for information on how to compute a one-sample confidence interval for a single binomial proportion or a two-sample confidence interval for the difference between two proportions, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

plotCiBinomDesign invisibly returns a list with components:

x.var — x-coordinates of the points that have been or would have been plotted

y.var — y-coordinates of the points that have been or would have been plotted

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", p, is the same on each trial. A binomial discrete random variable X is the number of "successes" in n independent trials. A special case of the binomial distribution occurs when $n = 1$, in which case X is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7). (However, USEPA 2009, p.8-27 recommends using the Wilcoxon rank sum test (wilcox.test) instead of comparing proportions.)

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions ciBinomHalfWidth, ciBinomN, and plotCiBinomDesign can be used to investigate these relationships for the case of binomial proportions.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Agresti, A., and B.A. Coull. (1998). Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions. *The American Statistician*, **52**(2), 119–126.

Agresti, A., and B. Caffo. (2000). Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures. *The American Statistician*, **54**(4), 280–288.

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapters 2 and 15.

Cochran, W.G. (1977). *Sampling Techniques*. John Wiley and Sons, New York, Chapter 3.

Fisher, R.A., and F. Yates. (1963). *Statistical Tables for Biological, Agricultural, and Medical Research*. 6th edition. Hafner, New York, 146pp.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. Second Edition. John Wiley and Sons, New York, Chapters 1-2.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY, Chapter 11.

Newcombe, R.G. (1998a). Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods. *Statistics in Medicine*, **17**, 857–872.

Newcombe, R.G. (1998b). Interval Estimation for the Difference Between Independent Proportions: Comparison of Eleven Methods. *Statistics in Medicine*, **17**, 873–890.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL, Chapter 4.

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 24.

See Also

ciBinomHalfWidth, ciBinomN, ebinom, binom.test, prop.test, par.

Examples

```
# Look at the relationship between half-width and sample size
# for a one-sample confidence interval for a binomial proportion,
# assuming an estimated proportion of 0.5 and a confidence level of
# 95%.  The jigsaw appearance of the plot is the result of using the
```

```
# score method:

dev.new()
plotCiBinomDesign()

#----------

# Redo the example above, but use the traditional (and inaccurate)
# Wald method.

dev.new()
plotCiBinomDesign(ci.method = "Wald")

#--------------------------------------------------------------------

# Plot sample size vs. the estimated proportion for various half-widths,
# using a 95% confidence level and the adjusted Wald method:

# NOTE:  This example takes several seconds to run so it has been
#        commented out.  Simply remove the pound signs (#) from in front
#        of the R commands to run it.

#dev.new()
#plotCiBinomDesign(x.var = "p.hat", y.var = "n",
#    half.width = 0.04, ylim = c(0, 600), main = "",
#    xlab = expression(hat(p)))
#
#plotCiBinomDesign(x.var = "p.hat", y.var = "n",
#    half.width = 0.05, add = TRUE, plot.col = 2)
#
#plotCiBinomDesign(x.var = "p.hat", y.var = "n",
#    half.width = 0.06, add = TRUE, plot.col = 3)
#
#legend(0.5, 150, paste("Half-Width =", c(0.04, 0.05, 0.06)),
#    lty = rep(1, 3), lwd = rep(2, 3), col=1:3, bty = "n")
#
#mtext(expression(paste("Sample Size vs. ", hat(p),
#  " for Confidence Interval for p")), line = 2.5, cex = 1.25)
#mtext("with Confidence=95%  and Various Values of Half-Width",
#  line = 1.5, cex = 1.25)
#mtext(paste("CI Method = Score Normal Approximation",
#  "with Continuity Correction"), line = 0.5)

#--------------------------------------------------------------------

# Modifying the example on pages 8-5 to 8-7 of USEPA (1989b),
# look at the relationship between half-width and sample size
# for a 95% confidence interval for the difference between the
# proportion of detects at the background and compliance wells.
# Use the estimated proportion of detects from the original data.
# (The data are stored in EPA.89b.cadmium.df.)
# Assume equal sample sizes at each well.
```

```
EPA.89b.cadmium.df
#   Cadmium.orig Cadmium Censored  Well.type
#1           0.1   0.100    FALSE Background
#2          0.12   0.120    FALSE Background
#3           BDL   0.000     TRUE Background
# ....................................
#86          BDL   0.000     TRUE Compliance
#87          BDL   0.000     TRUE Compliance
#88          BDL   0.000     TRUE Compliance

p.hat.back <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Background"]))

p.hat.back
#[1] 0.3333333

p.hat.comp <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Compliance"]))

p.hat.comp
#[1] 0.375

dev.new()
plotCiBinomDesign(p.hat.or.p1.hat = p.hat.back,
    p2.hat = p.hat.comp, digits=3)

#==========

# Clean up
#---------
rm(p.hat.back, p.hat.comp)
graphics.off()
```

`plotCiNormDesign` *Plots for Sampling Design Based on Confidence Interval for Mean of a Normal Distribution or Difference Between Two Means*

Description

Create plots involving sample size, half-width, estimated standard deviation, and confidence level for a confidence interval for the mean of a normal distribution or the difference between two means.

Usage

```
plotCiNormDesign(x.var = "n", y.var = "half.width",
  range.x.var = NULL, n.or.n1 = 25, n2 = n.or.n1,
  half.width = sigma.hat/2, sigma.hat = 1, conf.level = 0.95,
  sample.type = ifelse(missing(n2), "one.sample", "two.sample"),
  round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000,
  plot.it = TRUE, add = FALSE, n.points = 100,
```

```
    plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
    digits = .Options$digits,
    main = NULL, xlab = NULL, ylab = NULL, type = "l", ...)
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "half.width" (the half-width of the confidence interval), "sigma.hat" (the estimated standard deviation), and "conf.level" (the confidence level).
y.var	character string indicating what variable to use for the y-axis. Possible values are "half.width" (the half-width of the confidence interval; the default), and "n" (sample size).
range.x.var	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="half.width" the default value is c(0.1/sigma.hat, 2/sigma.hat). When x.var="sigma.hat", the default value is c(0.1, 2). When x.var="conf.level", the default value is c(0.5, 0.99).
n.or.n1	numeric scalar indicating the sample size. The default value is n.or.n1=25. When sample.type="one.sample", this argument denotes the number of observations in the single sample. When sample.type="two.sample", this argument denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".
n2	numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".
half.width	positive numeric scalar indicating the half-width of the confidence interval. The default value is sigma.hat/2. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="half.width" or y.var="half.width".
sigma.hat	positive numeric scalar specifying the estimated standard deviation. The default value is sigma.hat=1. This argument is ignored if x.var="sigma.hat".
conf.level	a scalar between 0 and 1 indicating the confidence level associated with the confidence interval. The default value is conf.level=0.95. This argument is ignored if x.var="conf.level".
sample.type	character string indicating whether this is a one-sample or two-sample confidence interval. When sample.type="one.sample", the computations for the plot are based on a confidence interval for a single mean. When sample.type="two.sample", the computations for the plot are based on a confidence interval for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.

round.up — logical scalar indicating whether to round up the computed sample sizes to the next smallest integer. The default value is round.up=FALSE. This argument is ignored unless y.var="n".

n.max — for the case when y.var="n", positive integer greater than 1 specifying the maximum sample size for the single group when sample.type="one.sample" or for group 1 when sample.type="two.sample". The default value is n.max=5000.

tol — for the case when y.var="n", numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.

maxiter — for the case when y.var="n", positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.

plot.it — a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add — a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points — a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col — a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col=1. See the entry for col in the help file for par for more information.

plot.lwd — a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty — a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits — a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

main, xlab, ylab, type, ... — additional graphical parameters (see par).

Details

See the help files for ciNormHalfWidth and ciNormN for information on how to compute a one-sample confidence interval for the mean of a normal distribution or a two-sample confidence interval for the difference between two means, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

plotCiNormDesign invisibly returns a list with components:

x.var	x-coordinates of points that have been or would have been plotted.
y.var	y-coordinates of points that have been or would have been plotted.

Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. In order to make any kind of probability statement about a normally-distributed population (of chemical concentrations for example), you have to first estimate the mean and standard deviation (the population parameters) of the distribution. Once you estimate these parameters, it is often useful to characterize the uncertainty in the estimate of the mean. This is done with confidence intervals.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce confidence intervals. The functions ciNormHalfWidth, ciNormN, and plotCiNormDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Millard, S.P., and N. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. p.21-3.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapters 7 and 8.

See Also

ciNormHalfWidth, ciNormN, Normal, enorm, t.test,
Estimating Distribution Parameters.

Examples

```
# Look at the relationship between half-width and sample size
# for a one-sample confidence interval for the mean, assuming
# an estimated standard deviation of 1 and a confidence level of 95%.

dev.new()
plotCiNormDesign()

#----------------------------------------------------------------------

# Plot sample size vs. the estimated standard deviation for
# various levels of confidence, using a half-width of 0.5.

dev.new()
plotCiNormDesign(x.var = "sigma.hat", y.var = "n", main = "")

plotCiNormDesign(x.var = "sigma.hat", y.var = "n", conf.level = 0.9,
  add = TRUE, plot.col = 2)

plotCiNormDesign(x.var = "sigma.hat", y.var = "n", conf.level = 0.8,
  add = TRUE, plot.col = 3)

legend(0.25, 60, c("95%", "90%", "80%"), lty = 1, lwd = 3, col = 1:3)

mtext("Sample Size vs. Estimated SD for Confidence Interval for Mean",
  font = 2, cex = 1.25, line = 2.75)
mtext("with Half-Width=0.5 and Various Confidence Levels", font = 2,
  cex = 1.25, line = 1.25)

#----------------------------------------------------------------------

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
# look at the relationship between half-width and sample size for a
# 95% confidence interval for the mean level of Aldicarb at the
# first compliance well.  Use the estimated standard deviation from
# the first four months of data.
# (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)

EPA.09.Ex.21.1.aldicarb.df
#   Month   Well Aldicarb.ppb
#1      1 Well.1         19.9
#2      2 Well.1         29.6
#3      3 Well.1         18.7
#4      4 Well.1         24.2
#...

mu.hat <- with(EPA.09.Ex.21.1.aldicarb.df,
  mean(Aldicarb.ppb[Well=="Well.1"]))

mu.hat
#[1] 23.1
```

```
sigma.hat <- with(EPA.09.Ex.21.1.aldicarb.df,
  sd(Aldicarb.ppb[Well=="Well.1"]))

sigma.hat
#[1] 4.93491

dev.new()
plotCiNormDesign(sigma.hat = sigma.hat, digits = 2,
  range.x.var = c(2, 25))

#==========

# Clean up
#---------
rm(mu.hat, sigma.hat)
graphics.off()
```

`plotCiNparDesign` *Plots for Sampling Design Based on Nonparametric Confidence Interval for a Quantile*

Description

Create plots involving sample size, quantile, and confidence level for a nonparametric confidence interval for a quantile.

Usage

```
plotCiNparDesign(x.var = "n", y.var = "conf.level", range.x.var = NULL,
  n = 25, p = 0.5, conf.level = 0.95, ci.type = "two.sided",
  lcl.rank = ifelse(ci.type == "upper", 0, 1),
  n.plus.one.minus.ucl.rank = ifelse(ci.type == "lower", 0, 1),
  plot.it = TRUE, add = FALSE, n.points = 100, plot.col = "black",
  plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits,
  cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL,
  type = "l")
```

Arguments

`x.var`	character string indicating what variable to use for the x-axis. Possible values are `"n"` (sample size; the default), `"p"` (quantile), and `"conf.level"` (the confidence level).
`y.var`	character string indicating what variable to use for the y-axis. Possible values are `conf.level` (confidence level; the default), and `"n"` (sample size).
`range.x.var`	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of `x.var`. When `x.var="n"` the default value is `c(2,50)`. When `x.var="p"` the default value is `c(0.5, 0.99)`. When `x.var="conf.level"`, the default value is `c(0.5, 0.99)`.

n numeric scalar indicating the sample size. The default value is n=25. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

p numeric scalar specifying the quantile. The value of this argument must be between 0 and 1. The default value is p=0.5. The argument is ignored if x.var="p".

conf.level a scalar between 0 and 1 indicating the confidence level associated with the confidence interval. The default value is conf.level=0.95. This argument is ignored if x.var="conf.level" or y.var="conf.level".

ci.type character string indicating what kind of confidence interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".

lcl.rank, n.plus.one.minus.ucl.rank
numeric vectors of non-negative integers indicating the ranks of the order statistics that are used for the lower and upper bounds of the confidence interval for the specified quantile(s). When lcl.rank=1 that means use the smallest value as the lower bound, when lcl.rank=2 that means use the second to smallest value as the lower bound, etc. When n.plus.one.minus.ucl.rank=1 that means use the largest value as the upper bound, when n.plus.one.minus.ucl.rank=2 that means use the second to largest value as the upper bound, etc. A value of 0 for lcl.rank indicates no lower bound (i.e., -Inf) and a value of 0 for n.plus.one.minus.ucl.rank indicates no upper bound (i.e., Inf). When ci.type="upper" then lcl.rank is set to 0 by default, otherwise it is set to 1 by default. When ci.type="lower" then n.plus.one.minus.ucl.rank is set to 0 by default, otherwise it is set to 1 by default.

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ...
additional graphical parameters (see par).

Details

See the help files for eqnpar, ciNparConfLevel, and ciNparN for information on how to compute a nonparametric confidence interval for a quantile, how the confidence level is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

plotCiNparDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note

See the help file for eqnpar.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for eqnpar.

See Also

eqnpar, ciNparConfLevel, ciNparN.

Examples

```
# Look at the relationship between confidence level and sample size for
# a two-sided nonparametric confidence interval for the 90'th percentile.

dev.new()
plotCiNparDesign(p = 0.9)

#----------

# Plot sample size vs. quantile for various levels of confidence:

dev.new()
plotCiNparDesign(x.var = "p", y.var = "n", range.x.var = c(0.8, 0.95),
  ylim = c(0, 60), main = "")

plotCiNparDesign(x.var = "p", y.var = "n", conf.level = 0.9, add = TRUE,
  plot.col = 2, plot.lty = 2)

plotCiNparDesign(x.var = "p", y.var = "n", conf.level = 0.8, add = TRUE,
  plot.col = 3, plot.lty = 3)

legend("topleft", c("95%", "90%", "80%"), lty = 1:3, col = 1:3,
  lwd = 3 * par('cex'), bty = 'n')
```

```
title(main = paste("Sample Size vs. Quantile for ",
  "Nonparametric CI for \nQuantile, with ",
  "Various Confidence Levels", sep=""))

#==========

# Clean up
#---------
graphics.off()
```

---

plotLinearTrendTestDesign
Plots for a Sampling Design Based on a t-Test for Linear Trend

---

Description

Create plots involving sample size, power, scaled difference, and significance level for a t-test for linear trend.

Usage

```
plotLinearTrendTestDesign(x.var = "n", y.var = "power",
  range.x.var = NULL, n = 12,
  slope.over.sigma = switch(alternative, greater = 0.1, less = -0.1,
    two.sided = ifelse(two.sided.direction == "greater", 0.1, -0.1)),
  alpha = 0.05, power = 0.95, alternative = "two.sided",
  two.sided.direction = "greater", approx = FALSE, round.up = FALSE,
  n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE,
  n.points = ifelse(x.var == "n", diff(range.x.var) + 1, 50),
  plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
  digits = .Options$digits, ..., main = NULL, xlab = NULL, ylab = NULL,
  type = "l")
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "slope.over.sigma" (scaled minimal detectable slope), "power" (power of the test), and "alpha" (significance level of the test).
y.var	character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "slope.over.sigma" (scaled minimal detectable slope), and "n" (sample size).
range.x.var	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(3,25). When x.var="slope.over.sigma" and alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is c(0.1, 1). When x.var="slope.over.sigma"

and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is -c(1, 0.1). When x.var="power" the default value is c(alpha + .Machine$double.eps, 0.95). When x.var="alpha", the default value is c(0.01, 0.2).

n numeric scalar indicating the sample size. The default value is n=12. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

slope.over.sigma
numeric scalar specifying the ratio of the true slope (β_1) to the population standard deviation of the error terms (σ). This is also called the "scaled slope". When alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is slope.over.sigma=0.1. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is delta.over.sigma=-0.1. This argument is ignored when x.var="slope.over.sigma" or y.var="slope.over.sigma".

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05. This argument is ignored when x.var="alpha".

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95. This argument is ignored when x.var="power" or y.var="power".

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

two.sided.direction
character string indicating the direction (positive or negative) for the scaled minimal detectable slope when alternative="two.sided". When two.sided.direction="greater" (the default), the scaled minimal detectable slope is positive. When two.sided.direction="less", the scaled minimal detectable slope is negative. This argument is ignored if alternative="less" or alternative="greater".

approx logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is approx=FALSE.

round.up logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is FALSE.

n.max for the case when y.var="n", a positive integer greater than 2 indicating the maximum sample size. The default value is n.max=5000.

tol numeric scalar indicating the toloerance to use in the uniroot search algorithm. The default value is tol=1e-7.

maxiter positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

plot.it a logical scalar indicating whether to create a new plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add	a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.
n.points	a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=50 unless x.var="n", in which case n.points=diff(range.x.var)+1.
plot.col	a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.
plot.lwd	a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.
plot.lty	a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.
digits	a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").
main, xlab, ylab, type, ...	additional graphical parameters (see par).

Details

See the help files for linearTrendTestPower, linearTrendTestN, and linearTrendTestScaledMds for information on how to compute the power, sample size, or scaled minimal detectable slope for a t-test for linear trend.

Value

plotlinearTrendTestDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note

See the help files for linearTrendTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for linearTrendTestPower.

See Also

linearTrendTestPower, linearTrendTestN, linearTrendTestScaledMds.

Examples

```
# Look at the relationship between power and sample size for the t-test for
# liner trend, assuming a scaled slope of 0.1 and a 5% significance level:

dev.new()
plotLinearTrendTestDesign()

#==========

# Plot sample size vs. the scaled minimal detectable slope for various
# levels of power, using a 5% significance level:

dev.new()
plotLinearTrendTestDesign(x.var = "slope.over.sigma", y.var = "n",
  ylim = c(0, 30), main = "")

plotLinearTrendTestDesign(x.var = "slope.over.sigma", y.var = "n",
  power = 0.9, add = TRUE, plot.col = "red")

plotLinearTrendTestDesign(x.var = "slope.over.sigma", y.var = "n",
  power = 0.8, add = TRUE, plot.col = "blue")

legend("topright", c("95%", "90%", "80%"), lty = 1, bty = "n",
  lwd = 3 * par("cex"), col = c("black", "red", "blue"))

title(main = paste("Sample Size vs. Scaled Slope for t-Test for Linear Trend",
  "with Alpha=0.05 and Various Powers", sep="\n"))

#==========

# Clean up
#---------
graphics.off()
```

---

plotPredIntLnormAltSimultaneousTestPowerCurve

Power Curves for Sampling Design for Test Based on Simultaneous Prediction Interval for Lognormal Distribution

---

Description

Plot power vs. θ_1/θ_2 (ratio of means) for a sampling design for a test based on a simultaneous prediction interval for a lognormal distribution.

Usage

```
plotPredIntLnormAltSimultaneousTestPowerCurve(n = 8, df = n - 1, n.geomean = 1,
  k = 1, m = 2, r = 1, rule = "k.of.m", cv = 1, range.ratio.of.means = c(1, 5),
  pi.type = "upper", conf.level = 0.95, r.shifted = r,
```

```
K.tol = .Machine$double.eps^(1/2), integrate.args.list = NULL, plot.it = TRUE,
 add = FALSE, n.points = 20, plot.col = "black", plot.lwd = 3 * par("cex"),
 plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ...,
 main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

Arguments

n	positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is n=8.
df	positive integer indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.geomean	positive integer specifying the sample size associated with the future geometric mean(s). The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), positive integer specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	positive integer specifying the maximum number of future observations (or averages) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	positive integer specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule).
cv	positive value specifying the coefficient of variation for both the population that was sampled to construct the prediction interval **and** the population that will be sampled to produce the future observations. The default value is cv=1.
range.ratio.of.means	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is range.ratio.of.means=c(1,5).
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".
conf.level	numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
r.shifted	positive integer between 1 and r specifying the number of future sampling occasions for which the mean is shifted. The default value is r.shifted=r.
K.tol	numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is K.tol=.Machine$double.eps^(1/2). For many applications, the value of K needs to be known only to the second decimal place, in which case setting K.tol=1e-4 will speed up computation a bit.

integrate.args.list
: a list of arguments to supply to the integrate function. The default value is integrate.args.list=NULL which means that the default values of integrate are used.

plot.it
: a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add
: a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points
: a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col
: a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd
: a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty
: a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits
: a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ...
: additional graphical parameters (see par).

Details

See the help file for predIntLnormAltSimultaneousTestPower for information on how to compute the power of a hypothesis test for the difference between two means of lognormal distributions based on a simultaneous prediction interval for a lognormal distribution.

Value

plotPredIntLnormAltSimultaneousTestPowerCurve invisibly returns a list with components:

x.var
: x-coordinates of points that have been or would have been plotted.

y.var
: y-coordinates of points that have been or would have been plotted.

Note

See the help file for predIntNormSimultaneous.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the

objectives of the sampling program is to determine whether two distributions differ from each other. The functions predIntLnormAltSimultaneousTestPower and plotPredIntLnormAltSimultaneousTestPowerCurve can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNormSimultaneous.

See Also

predIntLnormAltSimultaneousTestPower, predIntLnormAltSimultaneous, predIntLnormAlt, predIntLnormAltTestPower, Prediction Intervals, LognormalAlt.

Examples

```
# USEPA (2009) contains an example on page 19-23 that involves monitoring
# nw=100 compliance wells at a large facility with minimal natural spatial
# variation every 6 months for nc=20 separate chemicals.
# There are n=25 background measurements for each chemical to use to create
# simultaneous prediction intervals.  We would like to determine which kind of
# resampling plan based on normal distribution simultaneous prediction intervals to
# use (1-of-m, 1-of-m based on means, or Modified California) in order to have
# adequate power of detecting an increase in chemical concentration at any of the
# 100 wells while at the same time maintaining a site-wide false positive rate
# (SWFPR) of 10% per year over all 4,000 comparisons
# (100 wells x 20 chemicals x semi-annual sampling).

# The function predIntNormSimultaneousTestPower includes the argument "r"
# that is the number of future sampling occasions (r=2 in this case because
# we are performing semi-annual sampling), so to compute the individual test
# Type I error level alpha.test (and thus the individual test confidence level),
# we only need to worry about the number of wells (100) and the number of
# constituents (20): alpha.test = 1-(1-alpha)^(1/(nw x nc)).  The individual
# confidence level is simply 1-alpha.test.  Plugging in 0.1 for alpha,
# 100 for nw, and 20 for nc yields an individual test confidence level of
# 1-alpha.test = 0.9999473.

nc <- 20
nw <- 100
conf.level <- (1 - 0.1)^(1 / (nc * nw))
conf.level
#[1] 0.9999473

# The help file for predIntNormSimultaneousTestPower shows how to
# create the results below for various sampling plans:

#         Rule k m N.Mean    K Power Total.Samples
```

```
#1      k.of.m 1 2      1 3.16  0.39            2
#2      k.of.m 1 3      1 2.33  0.65            3
#3      k.of.m 1 4      1 1.83  0.81            4
#4 Modified.CA 1 4      1 2.57  0.71            4
#5      k.of.m 1 1      2 3.62  0.41            2
#6      k.of.m 1 2      2 2.33  0.85            4
#7      k.of.m 1 1      3 2.99  0.71            3

# The above table shows the K-multipliers for each prediction interval, along with
# the power of detecting a change in concentration of three standard deviations at
# any of the 100 wells during the course of a year, for each of the sampling
# strategies considered.  The last three rows of the table correspond to sampling
# strategies that involve using the mean of two or three observations.

# Here we will create a variation of this example based on
# using a lognormal distribution and plotting power versus ratio of the
# means assuming cv=1.

# Here is the power curve for the 1-of-4 sampling strategy:

dev.new()
plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
  rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
  conf.level = conf.level, ylim = c(0, 1), main = "")

title(main = paste("Power Curves for 1-of-4 Sampling Strategy Based on 25 Background",
  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))
mtext("Assuming Lognormal Data with CV=1", line = 0)

#----------

# Here are the power curves for the first four sampling strategies.
# Because this takes several seconds to run, here we have commented out
# the R commands.  To run this example, just remove the pound signs (#)
# from in front of the R commands.

#dev.new()
#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
#  rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
#  conf.level = conf.level, ylim = c(0, 1), main = "")

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 3, r = 2,
#  rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
#  conf.level = conf.level, add = TRUE, plot.col = "red", plot.lty = 2)

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 2, r = 2,
#  rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
#  conf.level = conf.level, add = TRUE, plot.col = "blue", plot.lty = 3)

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, r = 2, rule="Modified.CA",
#  range.ratio.of.means = c(1, 10), pi.type = "upper", conf.level = conf.level,
#  add = TRUE, plot.col = "green3", plot.lty = 4)
```

```
#legend("topleft", c("1-of-4", "Modified CA", "1-of-3", "1-of-2"),
#  col = c("black", "green3", "red", "blue"), lty = c(1, 4, 2, 3),
#  lwd = 3 * par("cex"), bty = "n")

#title(main = paste("Power Curves for 4 Sampling Strategies Based on 25 Background",
#  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))
#mtext("Assuming Lognormal Data with CV=1", line = 0)

#----------

# Here are the power curves for the last 3 sampling strategies:
# Because this takes several seconds to run, here we have commented out
# the R commands.  To run this example, just remove the pound signs (#)
# from in front of the R commands.

#dev.new()
#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 2, n.geomean = 2,
#  r = 2, rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
#  conf.level = conf.level, ylim = c(0, 1), main = "")

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 1, n.geomean = 2,
#  r = 2, rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
#  conf.level = conf.level, add = TRUE, plot.col = "red", plot.lty = 2)

#plotPredIntLnormAltSimultaneousTestPowerCurve(n = 25, k = 1, m = 1, n.geomean = 3,
#  r = 2, rule="k.of.m", range.ratio.of.means = c(1, 10), pi.type = "upper",
#  conf.level = conf.level, add = TRUE, plot.col = "blue", plot.lty = 3)

#legend("topleft", c("1-of-2, Order 2", "1-of-1, Order 3", "1-of-1, Order 2"),
#  col = c("black", "blue", "red"), lty = c(1, 3, 2), lwd = 3 * par("cex"),
#  bty="n")

#title(main = paste("Power Curves for 3 Sampling Strategies Based on 25 Background",
#  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))
#mtext("Assuming Lognormal Data with CV=1", line = 0)

#==========

# Clean up
#---------
rm(nc, nw, conf.level)
graphics.off()
```

---

plotPredIntLnormAltTestPowerCurve

Power Curves for Sampling Design for Test Based on Prediction Interval for Lognormal Distribution

---

Description

Plot power vs. θ_1/θ_2 (ratio of means) for a sampling design for a test based on a prediction interval for a lognormal distribution.

Usage

```
plotPredIntLnormAltTestPowerCurve(n = 8, df = n - 1, n.geomean = 1, k = 1,
  cv = 1, range.ratio.of.means = c(1, 5), pi.type = "upper", conf.level = 0.95,
  plot.it = TRUE, add = FALSE, n.points = 20, plot.col = "black",
  plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, ...,
  main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

Arguments

n	positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is n=8.
df	positive integer indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.geomean	positive integer specifying the sample size associated with the future geometric mean(s). The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
k	positive integer specifying the number of future observations that the prediction interval should contain with confidence level conf.level. The default value is k=1.
cv	positive value specifying the coefficient of variation for both the population that was sampled to construct the prediction interval **and** the population that will be sampled to produce the future observations. The default value is cv=1.
range.ratio.of.means	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is range.ratio.of.means=c(1,5).
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".
conf.level	numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
plot.it	a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.
add	a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.
n.points	a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col	a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.
plot.lwd	a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.
plot.lty	a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.
digits	a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").
main, xlab, ylab, type, ...	additional graphical parameters (see par).

Details

See the help file for predIntLnormAltTestPower for information on how to compute the power of a hypothesis test for the ratio of two means of lognormal distributions based on a prediction interval for a lognormal distribution.

Value

plotPredIntLnormAltTestPowerCurve invisibly returns a list with components:

x.var	x-coordinates of points that have been or would have been plotted.
y.var	y-coordinates of points that have been or would have been plotted.

Note

See the help files for predIntNormTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for predIntNormTestPower and tTestLnormAltPower.

See Also

predIntLnormAltTestPower,
predIntLnormAlt,
predIntNorm,
predIntNormK,
plotPredIntNormTestPowerCurve,
predIntLnormAltSimultaneous,
predIntLnormAltSimultaneousTestPower,
Prediction Intervals, LognormalAlt.

Examples

```
# Plot power vs. ratio of means for k=1 future observation for
# various sample sizes using a 5% significance level and assuming cv=1.

dev.new()
plotPredIntLnormAltTestPowerCurve(n = 8, k = 1,
  range.ratio.of.means=c(1, 10), ylim = c(0, 1), main = "")

plotPredIntLnormAltTestPowerCurve(n = 16, k = 1,
  range.ratio.of.means = c(1, 10), add = TRUE, plot.col = "red")

plotPredIntLnormAltTestPowerCurve(n = 32, k = 1,
  range.ratio.of.means=c(1, 10), add = TRUE, plot.col = "blue")

legend("topleft", c("n=32", "n=16", "n=8"), lty = 1, lwd = 3 * par("cex"),
  col = c("blue", "red", "black"), bty = "n")

title(main = paste("Power vs. Ratio of Means for Upper Prediction Interval",
  "with k=1, Confidence=95%, and Various Sample Sizes", sep="\n"))
mtext("Assuming a Lognormal Distribution with CV = 1", line = 0)

#==========

## Not run:
# Pages 6-16 to 6-17 of USEPA (2009) present EPA Reference Power Curves (ERPC)
# for groundwater monitoring:
#
# "Since effect sizes discussed in the next section often cannot or have not been
# quantified, the Unified Guidance recommends using the ERPC as a suitable basis
# of comparison for proposed testing procedures.  Each reference power curve
# corresponds to one of three typical yearly statistical evaluation schedules -
# quarterly, semi-annual, or annual - and represents the cumulative power
# achievable during a single year at one well-constituent pair by a 99
# (normal) prediction limit based on n = 10 background measurements and one new
# measurement from the compliance well.
#
# Here we will create a variation of Figure 6-3 on page 6-17 based on
# using a lognormal distribution and plotting power versus ratio of the
# means assuming cv=1.

dev.new()
plotPredIntLnormAltTestPowerCurve(n = 10, k = 1, cv = 1, conf.level = 0.99,
  range.ratio.of.means = c(1, 10), ylim = c(0, 1), main="")

plotPredIntLnormAltTestPowerCurve(n = 10, k = 2, cv = 1, conf.level = 0.99,
  range.ratio.of.means = c(1, 10), add = TRUE, plot.col = "red", plot.lty = 2)

plotPredIntLnormAltTestPowerCurve(n = 10, k = 4, cv = 1, conf.level = 0.99,
  range.ratio.of.means = c(1, 10), add = TRUE, plot.col = "blue", plot.lty = 3)

legend("topleft", c("Quarterly", "Semi-Annual", "Annual"), lty = 3:1,
  lwd = 3 * par("cex"), col = c("blue", "red", "black"), bty = "n")
```

```
  title(main = paste("Power vs. Ratio of Means for Upper Prediction Interval with",
    "n=10, Confidence=99%, and Various Sampling Frequencies", sep="\n"))
  mtext("Assuming a Lognormal Distribution with CV = 1", line = 0)

## End(Not run)

  #==========

  # Clean up
  #---------
  graphics.off()
```

`plotPredIntNormDesign` *Plots for a Sampling Design Based on a Prediction Interval for the Next k Observations from a Normal Distribution*

Description

Create plots involving sample size, number of future observations, half-width, estimated standard deviation, and confidence level for a prediction interval for the next k observations from a normal distribution.

Usage

```
plotPredIntNormDesign(x.var = "n", y.var = "half.width", range.x.var = NULL,
  n = 25, k = 1, n.mean = 1, half.width = 4 * sigma.hat, sigma.hat = 1,
  method = "Bonferroni", conf.level = 0.95, round.up = FALSE, n.max = 5000,
  tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 100,
  plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
  digits = .Options$digits, cex.main = par("cex"), ..., main = NULL,
  xlab = NULL, ylab = NULL, type = "l")
```

Arguments

`x.var`	character string indicating what variable to use for the x-axis. Possible values are `"n"` (sample size; the default), `"half.width"` (the half-width of the confidence interval), `"k"` (number of future observations or averages), `"sigma.hat"` (the estimated standard deviation), and `"conf.level"` (the confidence level).
`y.var`	character string indicating what variable to use for the y-axis. Possible values are `"half.width"` (the half-width of the confidence interval; the default), and `"n"` (sample size).
`range.x.var`	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of `x.var`. When `x.var="n"` the default value is `c(2,50)`. When `x.var="half.width"` the default value is `c(2.5 * sigma.hat, 4 * sigma.hat)`. When `x.var="k"` the default value is `c(1, 20)`. When `x.var="sigma.hat"`, the default value is `c(0.1, 2)`. When `x.var="conf.level"`, the default value is `c(0.5, 0.99)`.

n	positive integer greater than 1 indicating the sample size upon which the prediction interval is based. The default value is n=25. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
k	positive integer specifying the number of future observations or averages the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored if x.var="k".
n.mean	positive integer specifying the sample size associated with the k future *averages*. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
half.width	positive scalar indicating the half-widths of the prediction interval. The default value is half.width=4*sigma.hat. This argument is ignored if either x.var="half.width" or y.var="half.width".
sigma.hat	numeric scalar specifying the value of the estimated standard deviation. The default value is sigma.hat=1. This argument is ignored if x.var="sigma.hat".
method	character string specifying the method to use if the number of future observations (k) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferonni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). This argument is ignored if k=1.
conf.level	numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
round.up	for the case when y.var="n", logical scalar indicating whether to round up the values of the computed sample sizes to the next smallest integer. The default value is round.up=TRUE.
n.max	for the case when y.var="n", the maximum possible sample size. The default value is n.max=5000.
tol	numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.
maxiter	positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.
plot.it	a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.
add	a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.
n.points	a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.
plot.col	a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits` a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`cex.main`, `main`, `xlab`, `ylab`, `type`, `...`
additional graphical parameters (see `par`).

Details

See the help files for `predIntNorm`, `predIntNormK`, `predIntNormHalfWidth`, and `predIntNormN` for information on how to compute a prediction interval for the next k observations or averages from a normal distribution, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

`plotPredIntNormDesign` invisibly returns a list with components:

`x.var` x-coordinates of points that have been or would have been plotted.

`y.var` y-coordinates of points that have been or would have been plotted.

Note

See the help file for `predIntNorm`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce prediction intervals. The functions `predIntNormHalfWidth`, `predIntNormN`, and `plotPredIntNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

See the help file for `predIntNorm`.

See Also

`predIntNorm`, `predIntNormK`, `predIntNormHalfWidth`, `predIntNormN`, `Normal`.

Examples

```
# Look at the relationship between half-width and sample size for a
# prediction interval for k=1 future observation, assuming an estimated
# standard deviation of 1 and a confidence level of 95%:

dev.new()
plotPredIntNormDesign()

#==========

# Plot sample size vs. the estimated standard deviation for various levels
# of confidence, using a half-width of 4:

dev.new()
plotPredIntNormDesign(x.var = "sigma.hat", y.var = "n", range.x.var = c(1, 2),
  ylim = c(0, 90), main = "")

plotPredIntNormDesign(x.var = "sigma.hat", y.var = "n", range.x.var = c(1, 2),
  conf.level = 0.9, add = TRUE, plot.col = "red")

plotPredIntNormDesign(x.var = "sigma.hat", y.var = "n", range.x.var = c(1, 2),
  conf.level = 0.8, add = TRUE, plot.col = "blue")

legend("topleft", c("95%", "90%", "80%"), lty = 1, lwd = 3 * par("cex"),
  col = c("black", "red", "blue"), bty = "n")

title(main = paste("Sample Size vs. Sigma Hat for Prediction Interval for",
  "k=1 Future Obs, Half-Width=4, and Various Confidence Levels",
  sep = "\n"))

#==========

# The data frame EPA.92c.arsenic3.df contains arsenic concentrations (ppb)
# collected quarterly for 3 years at a background well and quarterly for
# 2 years at a compliance well.  Using the data from the background well,
# plot the relationship between half-width and sample size for a two-sided
# 90% prediction interval for k=4 future observations.

EPA.92c.arsenic3.df
#   Arsenic Year  Well.type
#1     12.6    1 Background
#2     30.8    1 Background
#3     52.0    1 Background
#...
#18     3.8    5 Compliance
#19     2.6    5 Compliance
#20    51.9    5 Compliance

mu.hat <- with(EPA.92c.arsenic3.df,
  mean(Arsenic[Well.type=="Background"]))

mu.hat
```

```
#[1] 27.51667

sigma.hat <- with(EPA.92c.arsenic3.df,
  sd(Arsenic[Well.type=="Background"]))

sigma.hat
#[1] 17.10119

dev.new()
plotPredIntNormDesign(x.var = "n", y.var = "half.width", range.x.var = c(4, 50),
  k = 4, sigma.hat = sigma.hat, conf.level = 0.9)

#==========

# Clean up
#---------
rm(mu.hat, sigma.hat)
graphics.off()
```

`plotPredIntNormSimultaneousTestPowerCurve`
Power Curves for Sampling Design for Test Based on Simultaneous Prediction Interval for Normal Distribution

---

Description

Plot power vs. Δ/σ (scaled minimal detectable difference) for a sampling design for a test based on a simultaneous prediction interval for a normal distribution.

Usage

```
plotPredIntNormSimultaneousTestPowerCurve(n = 8, df = n - 1, n.mean = 1,
  k = 1, m = 2, r = 1, rule = "k.of.m", range.delta.over.sigma = c(0, 5),
  pi.type = "upper", conf.level = 0.95, r.shifted = r,
  K.tol = .Machine$double.eps^(1/2), integrate.args.list = NULL,
  plot.it = TRUE, add = FALSE, n.points = 20, plot.col = "black",
  plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits,
 cex.main = par("cex"), ..., main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

Arguments

`n`	positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is `n=8`.
`df`	positive integer indicating the degrees of freedom associated with the sample size. The default value is `df=n-1`.
`n.mean`	positive integer specifying the sample size associated with the future average(s). The default value is `n.mean=1` (i.e., individual observations). Note that all future averages must be based on the same sample size.

`k` for the k-of-m rule (`rule="k.of.m"`), positive integer specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`. This argument is ignored when the argument `rule` is not equal to `"k.of.m"`.

`m` positive integer specifying the maximum number of future observations (or averages) on one future sampling "occasion". The default value is `m=2`, except when `rule="Modified.CA"`, in which case this argument is ignored and `m` is automatically set equal to 4.

`r` positive integer specifying the number of future sampling "occasions". The default value is `r=1`.

`rule` character string specifying which rule to use. The possible values are `"k.of.m"` (k-of-m rule; the default), `"CA"` (California rule), and `"Modified.CA"` (modified California rule). See the DETAILS section below for more information.

`range.delta.over.sigma`
numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is `range.delta.over.sigma=c(0,5)`.

`pi.type` character string indicating what kind of prediction interval to compute. The possible values are `pi.type="upper"` (the default), and `pi.type="lower"`.

`conf.level` numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

`r.shifted` positive integer between 1 and `r` specifying the number of future sampling occasions for which the mean is shifted by Δ/σ. The default value is `r.shifted=r`.

`K.tol` numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is `K.tol=.Machine$double.eps^(1/2)`. For many applications, the value of K needs to be known only to the second decimal place, in which case setting `K.tol=1e-4` will speed up computation a bit.

`integrate.args.list`
a list of arguments to supply to the `integrate` function. The default value is `integrate.args.list=NULL` which means that the default values of `integrate` are used.

`plot.it` a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument `add` below) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is `plot.it=TRUE`.

`add` a logical scalar indicating whether to add the design plot to the existing plot (`add=TRUE`), or to create a plot from scratch (`add=FALSE`). The default value is `add=FALSE`. This argument is ignored if `plot.it=FALSE`.

`n.points` a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are `n.points` x-values evenly spaced between `range.x.var[1]` and `range.x.var[2]`. The default value is `n.points=100`.

`plot.col` a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col="black"`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits` a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`cex.main`, `main`, `xlab`, `ylab`, `type`, `...`
additional graphical parameters (see `par`).

Details

See the help file for `predIntNormSimultaneousTestPower` for information on how to compute the power of a hypothesis test for the difference between two means of normal distributions based on a simultaneous prediction interval for a normal distribution.

Value

`plotPredIntNormSimultaneousTestPowerCurve` invisibly returns a list with components:

`x.var` x-coordinates of points that have been or would have been plotted.

`y.var` y-coordinates of points that have been or would have been plotted.

Note

See the help file for `predIntNormSimultaneous`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions `predIntNormSimultaneousTestPower` and `plotPredIntNormSimultaneousTestPowerCurve` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

See the help file for `predIntNormSimultaneous`.

See Also

`predIntNormSimultaneous`, `predIntNormSimultaneousK`, `predIntNormSimultaneousTestPower`, `predIntNorm`, `predIntNormK`, `predIntNormTestPower`, Prediction Intervals, Normal.

Examples

```
# USEPA (2009) contains an example on page 19-23 that involves monitoring
# nw=100 compliance wells at a large facility with minimal natural spatial
# variation every 6 months for nc=20 separate chemicals.
# There are n=25 background measurements for each chemical to use to create
# simultaneous prediction intervals.  We would like to determine which kind of
# resampling plan based on normal distribution simultaneous prediction intervals to
# use (1-of-m, 1-of-m based on means, or Modified California) in order to have
# adequate power of detecting an increase in chemical concentration at any of the
# 100 wells while at the same time maintaining a site-wide false positive rate
# (SWFPR) of 10% per year over all 4,000 comparisons
# (100 wells x 20 chemicals x semi-annual sampling).

# The function predIntNormSimultaneousTestPower includes the argument "r"
# that is the number of future sampling occasions (r=2 in this case because
# we are performing semi-annual sampling), so to compute the individual test
# Type I error level alpha.test (and thus the individual test confidence level),
# we only need to worry about the number of wells (100) and the number of
# constituents (20): alpha.test = 1-(1-alpha)^(1/(nw x nc)).  The individual
# confidence level is simply 1-alpha.test.  Plugging in 0.1 for alpha,
# 100 for nw, and 20 for nc yields an individual test confidence level of
# 1-alpha.test = 0.9999473.

nc <- 20
nw <- 100
conf.level <- (1 - 0.1)^(1 / (nc * nw))
conf.level
#[1] 0.9999473

# The help file for predIntNormSimultaneousTestPower shows how to
# create the results below for various sampling plans:

#         Rule k m N.Mean    K Power Total.Samples
#1      k.of.m 1 2      1 3.16  0.39             2
#2      k.of.m 1 3      1 2.33  0.65             3
#3      k.of.m 1 4      1 1.83  0.81             4
#4 Modified.CA 1 4      1 2.57  0.71             4
#5      k.of.m 1 1      2 3.62  0.41             2
#6      k.of.m 1 2      2 2.33  0.85             4
#7      k.of.m 1 1      3 2.99  0.71             3

# The above table shows the K-multipliers for each prediction interval, along with
# the power of detecting a change in concentration of three standard deviations at
# any of the 100 wells during the course of a year, for each of the sampling
# strategies considered.  The last three rows of the table correspond to sampling
# strategies that involve using the mean of two or three observations.

# Here is the power curve for the 1-of-4 sampling strategy:

dev.new()
plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
  rule="k.of.m", pi.type = "upper",  conf.level = conf.level,
```

```
  xlab = "SD Units Above Background", main = "")

title(main = paste(
  "Power Curve for 1-of-4 Sampling Strategy Based on 25 Background",
  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))

#----------

# Here are the power curves for the first four sampling strategies.
# Because this takes several seconds to run, here we have commented out
# the R commands.  To run this example, just remove the pound signs (#)
# from in front of the R commands.

#dev.new()
#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 4, r = 2,
#  rule="k.of.m", pi.type = "upper",  conf.level = conf.level,
#  xlab = "SD Units Above Background", main = "")

#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 3, r = 2,
#  rule="k.of.m", pi.type = "upper",  conf.level = conf.level, add = TRUE,
#  plot.col = "red", plot.lty = 2)

#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 2, r = 2,
#  rule="k.of.m", pi.type = "upper",  conf.level = conf.level, add = TRUE,
#  plot.col = "blue", plot.lty = 3)

#plotPredIntNormSimultaneousTestPowerCurve(n = 25, r = 2, rule="Modified.CA",
#  pi.type = "upper", conf.level = conf.level, add = TRUE, plot.col = "green3",
#  plot.lty = 4)

#legend(0, 1, c("1-of-4", "Modified CA", "1-of-3", "1-of-2"),
#  col = c("black", "green3", "red", "blue"), lty = c(1, 4, 2, 3),
#  lwd = 3 * par("cex"), bty = "n")

#title(main = paste("Power Curves for 4 Sampling Strategies Based on 25 Background",
#  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))

#----------

# Here are the power curves for the last 3 sampling strategies.
# Because this takes several seconds to run, here we have commented out
# the R commands.  To run this example, just remove the pound signs (#)
# from in front of the R commands.

#dev.new()
#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 2, n.mean = 2,
#  r = 2, rule="k.of.m", pi.type = "upper", conf.level = conf.level,
#  xlab = "SD Units Above Background", main = "")

#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 1, n.mean = 2,
#  r = 2, rule="k.of.m", pi.type = "upper", conf.level = conf.level, add = TRUE,
#  plot.col = "red", plot.lty = 2)
```

```
#plotPredIntNormSimultaneousTestPowerCurve(n = 25, k = 1, m = 1, n.mean = 3,
#  r = 2, rule="k.of.m", pi.type = "upper", conf.level = conf.level, add = TRUE,
#  plot.col = "blue", plot.lty = 3)

#legend(0, 1, c("1-of-2, Order 2", "1-of-1, Order 3", "1-of-1, Order 2"),
#  col = c("black", "blue", "red"), lty = c(1, 3, 2), lwd = 3 * par("cex"),
#  bty="n")

#title(main = paste("Power Curves for 3 Sampling Strategies Based on 25 Background",
#  "Samples, SWFPR=10%, and 2 Future Sampling Periods", sep = "\n"))

#==========

# Clean up
#---------
rm(nc, nw, conf.level)
graphics.off()
```

---

plotPredIntNormTestPowerCurve

Power Curves for Sampling Design for Test Based on Prediction Interval for Normal Distribution

---

Description

Plot power vs. Δ/σ (scaled minimal detectable difference) for a sampling design for a test based on a prediction interval for a normal distribution.

Usage

```
plotPredIntNormTestPowerCurve(n = 8, df = n - 1, n.mean = 1, k = 1,
  range.delta.over.sigma = c(0, 5), pi.type = "upper", conf.level = 0.95,
  plot.it = TRUE, add = FALSE, n.points = 20, plot.col = "black",
  plot.lwd = 3 * par("cex"), plot.lty = 1, digits = .Options$digits, ...,
  main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

Arguments

n	positive integer greater than 2 indicating the sample size upon which the prediction interval is based. The default is value is n=8.
df	positive integer indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.mean	positive integer specifying the sample size associated with the future average(s). The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
k	positive integer specifying the number of future observations that the prediction interval should contain with confidence level conf.level. The default value is k=1.

`range.delta.over.sigma`
: numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is `range.delta.over.sigma=c(0,5)`.

`pi.type`
: character string indicating what kind of prediction interval to compute. The possible values are `pi.type="upper"` (the default), and `pi.type="lower"`.

`conf.level`
: numeric scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

`plot.it`
: a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument `add` below) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is `plot.it=TRUE`.

`add`
: a logical scalar indicating whether to add the design plot to the existing plot (`add=TRUE`), or to create a plot from scratch (`add=FALSE`). The default value is `add=FALSE`. This argument is ignored if `plot.it=FALSE`.

`n.points`
: a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are `n.points` x-values evenly spaced between `range.x.var[1]` and `range.x.var[2]`. The default value is `n.points=100`.

`plot.col`
: a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col="black"`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd`
: a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty`
: a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits`
: a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`main, xlab, ylab, type, ...`
: additional graphical parameters (see `par`).

Details

See the help file for `predIntNormTestPower` for information on how to compute the power of a hypothesis test for the difference between two means of normal distributions based on a prediction interval for a normal distribution.

Value

`plotPredIntNormTestPowerCurve` invisibly returns a list with components:

`x.var`
: x-coordinates of points that have been or would have been plotted.

`y.var`
: y-coordinates of points that have been or would have been plotted.

Note

See the help files for predIntNorm and predIntNormSimultaneous.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions predIntNormTestPower and plotPredIntNormTestPowerCurve can be used to investigate these relationships for the case of normally-distributed observations. In the case of a simple shift between the two means, the test based on a prediction interval is not as powerful as the two-sample t-test. However, the test based on a prediction interval is more efficient at detecting a shift in the tail.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for predIntNorm and predIntNormSimultaneous.

See Also

predIntNorm, predIntNormK, predIntNormTestPower, predIntNormSimultaneous, predIntNormSimultaneousK, predIntNormSimultaneousTestPower, Prediction Intervals, Normal.

Examples

```
# Pages 6-16 to 6-17 of USEPA (2009) present EPA Reference Power Curves (ERPC)
# for groundwater monitoring:
#
# "Since effect sizes discussed in the next section often cannot or have not been
# quantified, the Unified Guidance recommends using the ERPC as a suitable basis
# of comparison for proposed testing procedures.  Each reference power curve
# corresponds to one of three typical yearly statistical evaluation schedules -
# quarterly, semi-annual, or annual - and represents the cumulative power
# achievable during a single year at one well-constituent pair by a 99% upper
# (normal) prediction limit based on n = 10 background measurements and one new
# measurement from the compliance well.
#
# Here we will reproduce Figure 6-3 on page 6-17.

dev.new()
plotPredIntNormTestPowerCurve(n = 10, k = 1, conf.level = 0.99,
  ylim = c(0, 1), main="")

plotPredIntNormTestPowerCurve(n = 10, k = 2, conf.level = 0.99,
  add = TRUE, plot.col = "red", plot.lty = 2)

plotPredIntNormTestPowerCurve(n = 10, k = 4, conf.level = 0.99,
  add = TRUE, plot.col = "blue", plot.lty = 3)
```

```
  legend("topleft", c("Quarterly", "Semi-Annual", "Annual"), lty = 3:1,
    lwd = 3 * par("cex"), col = c("blue", "red", "black"), bty = "n")

  title(main = paste("Power vs. Delta/Sigma for Upper Prediction Interval with",
    "n=10, Confidence=99%, and Various Sampling Frequencies", sep="\n"))

  #==========
## Not run:
  # Plot power vs. scaled minimal detectable difference for various sample sizes
  # using a 5

  dev.new()
  plotPredIntNormTestPowerCurve(n = 8, k = 1, ylim = c(0, 1), main="")

  plotPredIntNormTestPowerCurve(n = 16, k = 1, add = TRUE, plot.col = "red")

  plotPredIntNormTestPowerCurve(n = 32, k = 1, add = TRUE, plot.col = "blue")

  legend("bottomright", c("n=32", "n=16", "n=8"), lty = 1, lwd = 3 * par("cex"),
    col = c("blue", "red", "black"), bty = "n")

  title(main = paste("Power vs. Delta/Sigma for Upper Prediction Interval with",
    "k=1, Confidence=95%, and Various Sample Sizes", sep="\n"))

  #==========

  # Clean up
  #---------
  graphics.off()

## End(Not run)
```

---

plotPredIntNparDesign *Plots for a Sampling Design Based on a Nonparametric Prediction Interval*

Description

Create plots involving sample size (n), number of future observations (m), minimum number of future observations the interval should contain (k), and confidence level ($1-\alpha$) for a nonparametric prediction interval.

Usage

```
plotPredIntNparDesign(x.var = "n", y.var = "conf.level", range.x.var = NULL,
  n = max(25, lpl.rank + n.plus.one.minus.upl.rank + 1),
  k = 1, m = ifelse(x.var == "k", ceiling(max.x), 1), conf.level = 0.95,
  pi.type = "two.sided", lpl.rank = ifelse(pi.type == "upper", 0, 1),
  n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1), n.max = 5000,
```

```
maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 100,
plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
digits = .Options$digits, cex.main = par("cex"), ..., main = NULL,
xlab = NULL, ylab = NULL, type = "l")
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "conf.level" (the confidence level), "k" (minimum number of future observations the interval should contain), and "m" (number of future observations).
y.var	character string indicating what variable to use for the y-axis. Possible values are "conf.level" (confidence level; the default), and "n" (sample size).
range.x.var	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="conf.level", the default value is c(0.5, 0.99). When x.var="k" or x.var="m", the default value is c(1, 20).
n	numeric scalar indicating the sample size. The default value is max(25, lpl.rank + n.plus.one.minus.upl.rank + 1). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".
k	positive integer specifying the minimum number of future observations out of m that should be contained in the prediction interval. The default value is k=1.
m	positive integer specifying the number of future observations. The default value is ifelse(x.var == "k", ceiling(max.x), 1). That is, if x.var="k" then the default value is the smallest integer greater than or equal to the maximum value that k will take on in the plot; otherwise the default value is m=1.
conf.level	numeric scalar between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is conf.level=0.95.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
lpl.rank	non-negative integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="two-sided" or pi.type="lower", the default value is lpl.rank=1 (implying the minimum value is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0.
n.plus.one.minus.upl.rank	non-negative integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0.
n.max	for the case when y.var="n", a positive integer greater than 2 indicating the maximum possible sample size. The default value is n.max=5000.
maxiter	positive integer indicating the maximum number of iterations to use in the uniroot search algorithm for sample size when y.var="n". The default value is maxiter=1000.

`plot.it` a logical scalar indicating whether to create a plot or add to the existing plot (see `add`) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is `plot.it=TRUE`.

`add` a logical scalar indicating whether to add the design plot to the existing plot (`add=TRUE`), or to create a plot from scratch (`add=FALSE`). The default value is `add=FALSE`. This argument is ignored if `plot.it=FALSE`.

`n.points` a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are `n.points` x-values evenly spaced between `range.x.var[1]` and `range.x.var[2]`. The default value is `n.points=100`.

`plot.col` a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col="black"`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits` a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`cex.main`, `main`, `xlab`, `ylab`, `type`, `...`
additional graphical parameters (see `par`).

Details

See the help file for `predIntNpar`, `predIntNparConfLevel`, and `predIntNparN` for information on how to compute a nonparametric prediction interval, how the confidence level is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

`plotPredIntNparDesign` invisibly returns a list with components `x.var` and `y.var`, giving coordinates of the points that have been or would have been plotted.

Note

See the help file for `predIntNpar`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNpar`.

See Also

predIntNpar, predIntNparConfLevel, predIntNparN.

Examples

```
# Look at the relationship between confidence level and sample size for a
# two-sided nonparametric prediction interval for the next m=1 future observation.

dev.new()
plotPredIntNparDesign()

#==========

# Plot confidence level vs. sample size for various values of number of
# future observations (m):

dev.new()
plotPredIntNparDesign(k = 1, m = 1, ylim = c(0, 1), main = "")

plotPredIntNparDesign(k = 2, m = 2, add = TRUE, plot.col = "red")

plotPredIntNparDesign(k = 3, m = 3, add = TRUE, plot.col = "blue")

legend("bottomright", c("m=1", "m=2", "m=3"), lty = 1, lwd = 3 * par("cex"),
  col = c("black", "red", "blue"), bty = "n")

title(main = paste("Confidence Level vs. Sample Size for Nonparametric PI",
  "with Various Values of m", sep="\n"))

#==========

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.
#
# Modify this example by creating a plot to look at confidence level versus
# sample size (i.e., number of observations at the background wells) for
# predicting the next m = 4 future observations when constructing a one-sided
# upper prediction interval based on the maximum value.

dev.new()
plotPredIntNparDesign(k = 4, m = 4, pi.type = "upper")

#==========

# Clean up
#---------
graphics.off()
```

plotPredIntNparSimultaneousDesign
Plots for a Sampling Design Based on a Simultaneous Nonparametric Prediction Interval

Description

Create plots involving sample size (n), number of future observations (m), minimum number of future observations the interval should contain (k), number of future sampling occasions (r), and confidence level $(1-\alpha)$ for a simultaneous nonparametric prediction interval.

Usage

```
plotPredIntNparSimultaneousDesign(x.var = "n", y.var = "conf.level",
  range.x.var = NULL, n = max(25, lpl.rank + n.plus.one.minus.upl.rank + 1),
  n.median = 1, k = 1, m = ifelse(x.var == "k", ceiling(max.x), 1), r = 2,
  rule = "k.of.m", conf.level = 0.95, pi.type = "upper",
  lpl.rank = ifelse(pi.type == "upper", 0, 1),
  n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1), n.max = 5000,
  maxiter = 1000, integrate.args.list = NULL, plot.it = TRUE, add = FALSE,
  n.points = 100, plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
  digits = .Options$digits, cex.main = par("cex"), ..., main = NULL,
  xlab = NULL, ylab = NULL, type = "l")
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "conf.level" (the confidence level), "k" (minimum number of future observations the interval should contain), "m" (number of future observations), and "r" (number of future sampling occasions).
y.var	character string indicating what variable to use for the y-axis. Possible values are "conf.level" (confidence level; the default), and "n" (sample size).
range.x.var	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="conf.level", the default value is c(0.5, 0.99). When x.var="k", x.var="m", or x.var="r", the default value is c(1, 20).
n	numeric scalar indicating the sample size. The default value is max(25, lpl.rank + n.plus.one.minus.upl.rank + 1). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".
n.median	positive odd integer specifying the sample size associated with the future medians. The default value is n.median=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.

k — for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling "occassion") the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".

m — positive integer specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.

r — positive integer specifying the number of future sampling "occasions". The default value is r=1.

rule — character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.

conf.level — numeric scalar between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is conf.level=0.95.

pi.type — character string indicating what kind of prediction interval to compute. The possible values are "upper" (the default) and "lower".

lpl.rank — non-negative integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="lower", the default value is lpl.rank=1 (implying the minimum value is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0.

n.plus.one.minus.upl.rank — non-negative integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0.

n.max — numeric scalar indicating the maximum sample size to consider when y.var="n". This argument is used in the search algorithm to determine the required sample size. The default value is n.max=5000.

maxiter — positive integer indicating the maximum number of iterations to use in the uniroot search algorithm when y.var="n". The default value is maxiter=1000.

integrate.args.list — list of arguments to supply to the integrate function. The default value is NULL.

plot.it — a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add — a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points — a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

`plot.col` a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col="black"`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits` a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`cex.main`, `main`, `xlab`, `ylab`, `type`, `...`
additional graphical parameters (see `par`).

Details

See the help file for `predIntNparSimultaneous`, `predIntNparSimultaneousConfLevel`, and `predIntNparSimultaneousN` for information on how to compute a simultaneous nonparametric prediction interval, how the confidence level is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

`plotPredIntNparSimultaneousDesign` invisibly returns a list with components `x.var` and `y.var`, giving coordinates of the points that have been or would have been plotted.

Note

See the help file for `predIntNparSimultaneous`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNparSimultaneous`.

See Also

`predIntNparSimultaneous`, `predIntNparSimultaneousConfLevel`, `predIntNparSimultaneousN`, `predIntNparSimultaneousTestPower`, `predIntNpar`, `tolIntNpar`.

Examples

```
# For the 1-of-3 rule with r=20 future sampling occasions, look at the
# relationship between confidence level and sample size for a one-sided
# upper simultaneous nonparametric prediction interval.
```

```
dev.new()
plotPredIntNparSimultaneousDesign(k = 1, m = 3, r = 20, range.x.var = c(2, 20))

#==========

# Plot confidence level vs. sample size for various values of number of
# future sampling occasions (r):

dev.new()
plotPredIntNparSimultaneousDesign(m = 3, r = 10, rule = "CA",
  ylim = c(0, 1), main = "")

plotPredIntNparSimultaneousDesign(m = 3, r = 20, rule = "CA", add = TRUE,
  plot.col = "red")

plotPredIntNparSimultaneousDesign(m = 3, r = 30, rule = "CA", add = TRUE,
  plot.col = "blue")

legend("bottomright", c("r=10", "r=20", "r=30"), lty = 1, lwd = 3 * par("cex"),
  col = c("black", "red", "blue"), bty = "n")

title(main = paste("Confidence Level vs. Sample Size for Simultaneous",
  "Nonparametric PI with Various Values of r", sep="\n"))

#==========

# Modifying Example 19-5 of USEPA (2009, p. 19-33), plot confidence level
# versus sample size (number of background observations requried) for
# a 1-of-3 plan assuming r = 10 compliance wells (future sampling occasions).

dev.new()
plotPredIntNparSimultaneousDesign(k = 1, m = 3, r = 10, rule = "k.of.m")

#==========

# Clean up
#---------
graphics.off()
```

`plotPredIntNparSimultaneousTestPowerCurve`
Power Curves for Sampling Design for Test Based on Nonparametric Simultaneous Prediction Interval

---

Description

Plot power vs. Δ/σ (scaled minimal detectable difference) for a sampling design for a test based on a nonparametric simultaneous prediction interval. The power is based on assuming the true distribution of the observations is normal.

Usage

```
plotPredIntNparSimultaneousTestPowerCurve(n = 8, n.median = 1, k = 1, m = 2,
  r = 1, rule = "k.of.m", lpl.rank = ifelse(pi.type == "upper", 0, 1),
 n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1), pi.type = "upper",
  r.shifted = r, integrate.args.list = NULL, method = "approx", NMC = 100,
  range.delta.over.sigma = c(0, 5), plot.it = TRUE, add = FALSE, n.points = 20,
  plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
  digits = .Options$digits, cex.main = par("cex"), ..., main = NULL,
  xlab = NULL, ylab = NULL, type = "l")
```

Arguments

n	positive integer specifying the sample sizes.
n.median	positive odd integer specifying the sample size associated with the future medians. The default value is n.median=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling "occassion") the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	positive integer specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	positive integer specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule).
lpl.rank	non-negative integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. When pi.type="lower", the default value is lpl.rank=1 (implying the minimum value of x is used as the lower bound of the prediction interval). When pi.type="upper", the argument lpl.rank is set equal to 0.
n.plus.one.minus.upl.rank	non-negative integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. When pi.type="upper", the default value is n.plus.one.minus.upl.rank=1. When pi.type="lower", the argument n.plus.one.minus.upl.rank is set equal to 0.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".

`r.shifted` integer between 1 and `r` specifying the number of future sampling occasions for which the scaled mean is shifted by Δ/σ. The default value is `r.shifted=r`.

`integrate.args.list`
list of arguments to supply to the `integrate` function. The default value is `NULL`.

`method` character string indicating what method to use to compute the power. The possible values are `"approx"` (the default) and `"simulate"` (use Monte Carlo simulation).

`NMC` positive integer indicating the number of Monte Carlo trials to run when `method="simulate"`. The default value is `NMC=100`.

`range.delta.over.sigma`
numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value is `range.delta.over.sigma=c(0,5)`.

`plot.it` a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument `add` below) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is `plot.it=TRUE`.

`add` a logical scalar indicating whether to add the design plot to the existing plot (`add=TRUE`), or to create a plot from scratch (`add=FALSE`). The default value is `add=FALSE`. This argument is ignored if `plot.it=FALSE`.

`n.points` a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are `n.points` x-values evenly spaced between `range.x.var[1]` and `range.x.var[2]`. The default value is `n.points=100`.

`plot.col` a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col="black"`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits` a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`cex.main`, `main`, `xlab`, `ylab`, `type`, `...`
additional graphical parameters (see `par`).

Details

See the help file for `predIntNparSimultaneousTestPower` for information on how to compute the power of a hypothesis test for the difference between two means of normal distributions based on a nonparametric simultaneous prediction interval.

Value

`plotPredIntNparSimultaneousTestPowerCurve` invisibly returns a list with components:

`x.var` x-coordinates of points that have been or would have been plotted.

`y.var` y-coordinates of points that have been or would have been plotted.

Note

See the help file for predIntNparSimultaneous.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions predIntNparSimultaneousTestPower and plotPredIntNparSimultaneousTestPowerCurve can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNparSimultaneous.

Gansecki, M. (2009). *Using the Optimal Rank Values Calculator*. US Environmental Protection Agency, Region 8, March 10, 2009.

See Also

predIntNparSimultaneousTestPower, predIntNparSimultaneous, predIntNparSimultaneousN, predIntNparSimultaneousConfLevel, plotPredIntNparSimultaneousDesign, predIntNpar, tolIntNpar.

Examples

```
# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells  are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 20 to use to construct the prediction limit.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually.  For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year (i.e., r = 10 * 1 = 10).

# Here we will reproduce Figure 19-2 on page 19-35.  This figure plots the
# power of the nonparametric simultaneous prediction interval for 6 different
# plans:
#          Rule Median.n k m Order.Statistic Achieved.alpha BG.Limit
#1)      k.of.m        1 1 3             Max         0.0055     0.28
#2)      k.of.m        1 1 4             Max         0.0009     0.28
#3) Modified.CA        1 1 4             Max         0.0140     0.28
```

```
#4)      k.of.m        3 1 2             Max         0.0060     0.28
#5)      k.of.m        1 1 4             2nd         0.0046     0.25
#6)      k.of.m        1 1 4             3rd         0.0135     0.24

# Here is the power curve for the 1-of-4 sampling strategy.

dev.new()
plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 4, r = 10,
  rule = "k.of.m", n.plus.one.minus.upl.rank = 3, pi.type = "upper",
  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5), main = "")

title(main = paste(
  "Power Curve for Nonparametric 1-of-4 Sampling Strategy Based on",
  "25 Background Samples, SWFPR=10%, and 2 Future Sampling Periods",
  sep = "\n"), cex.main = 1.1)

#----------

# Here are the power curves for all 6 sampling strategies.
# Because these take several seconds to create, here we have commented out
# the R commands.  To run this example, just remove the pound signs (#) from
# in front of the R commands.

#dev.new()
#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 4, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 3, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5), main = "")

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, n.median = 3, k = 1, m = 2,
#  r = 10, rule = "k.of.m", n.plus.one.minus.upl.rank = 1, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 2, plot.lty = 2)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, r = 10, rule = "Modified.CA",
#  n.plus.one.minus.upl.rank = 1, pi.type = "upper", r.shifted = 1,
#  method = "approx", range.delta.over.sigma = c(0, 5), add = TRUE,
#  plot.col = 3, plot.lty = 3)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 4, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 2, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 4, plot.lty = 4)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 3, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 1, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 5, plot.lty = 5)

#plotPredIntNparSimultaneousTestPowerCurve(n = 20, k = 1, m = 4, r = 10,
#  rule = "k.of.m", n.plus.one.minus.upl.rank = 1, pi.type = "upper",
#  r.shifted = 1, method = "approx", range.delta.over.sigma = c(0, 5),
#  add = TRUE, plot.col = 6, plot.lty = 6)
```

```
#legend("topleft", legend = c("1-of-4, 3rd", "1-of-2, Max, Median", "Mod CA",
#  "1-of-4, 2nd", "1-of-3, Max", "1-of-4, Max"), lwd = 3 * par("cex"),
#  col = 1:6, lty = 1:6, bty = "n")

#title(main = "Figure 19-2. Comparison of Full Power Curves")

#==========

# Clean up
#---------
graphics.off()
```

---

plotPropTestDesign *Plots for Sampling Design Based on One- or Two-Sample Proportion Test*

Description

Create plots involving sample size, power, difference, and significance level for a one- or two-sample proportion test.

Usage

```
plotPropTestDesign(x.var = "n", y.var = "power",
  range.x.var = NULL, n.or.n1 = 25, n2 = n.or.n1, ratio = 1,
   p.or.p1 = switch(alternative, greater = 0.6, less = 0.4,
     two.sided = ifelse(two.sided.direction == "greater", 0.6, 0.4)),
  p0.or.p2 = 0.5, alpha = 0.05, power = 0.95,
 sample.type = ifelse(!missing(n2) || !missing(ratio), "two.sample", "one.sample"),
  alternative = "two.sided", two.sided.direction = "greater",
  approx = TRUE, correct = sample.type == "two.sample", round.up = FALSE,
  warn = TRUE, n.min = 2, n.max = 10000, tol.alpha = 0.1 * alpha,
  tol = 1e-07, maxiter = 1000, plot.it = TRUE, add = FALSE, n.points = 50,
  plot.col = "black", plot.lwd = 3 * par("cex"), plot.lty = 1,
  digits = .Options$digits, cex.main = par("cex"), ..., main = NULL,
  xlab = NULL, ylab = NULL, type = "l")
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "delta" (minimal detectable difference), "power" (power of the test), and "alpha" (significance level of the test).
y.var	character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "delta" (minimal detectable difference), and "n" (sample size).

range.x.var numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(20,400). When x.var="delta" and alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is c(0.05, 0.2). When x.var="delta" and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is -c(0.2, 0.05). When x.var="power" the default value is c(alpha + .Machine$double.eps, 0.95). When x.var="alpha", the default value is c(0.01, 0.2).

n.or.n1 numeric scalar indicating the sample size. The default value is n.or.n1=25. When sample.type="one.sample", n.or.n1 denotes the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

n2 numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

ratio numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 n_2/n_1. The default value is ratio=1. All values of ratio must be greater than or equal to 1. This argument is only used when x.var="n" or y.var="n" and sample.type="two.sample".

p.or.p1 numeric vector of proportions. When sample.type="one.sample", p.or.p1 denotes the *true* value of p, the probability of "success". When sample.type="two.sample", p.or.p1 denotes the value of p_1, the probability of "success" in group 1. When alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is p.or.p1=0.6. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is p.or.p1=0.4. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when x.var="delta" or y.var="delta".

p0.or.p2 numeric vector of proportions. When sample.type="one.sample", p0.or.p2 denotes the *hypothesized* value of p, the probability of "success". When sample.type="two.sample", p0.or.p2 denotes the value of p_2, the probability of "success" in group 2. The default value is p0.or.p2=0.5. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05. This argument is ignored when x.var="alpha".

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95. This argument is ignored when x.var="power" or y.var="power".

sample.type character string indicating whether the design is based on a one-sample or two-sample proportion test. When sample.type="one.sample", the computations for the plot are based on a one-sample proportion test. When

sample.type="two.sample", the computations for the plot are based on a two-sample proportion test. The default value is sample.type="one.sample".

alternative — character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

two.sided.direction — character string indicating the direction (positive or negative) for the minimal detectable difference when alternative="two.sided". When two.sided.direction="greater" (the default), the minimal detectable difference is positive. When two.sided.direction="less", the minimal detectable difference is negative. This argument is ignored unless alternative="two.sided" and either x.var="delta" or y.var="delta".

approx — logical scalar indicating whether to compute the power, sample size, or minimal detectable difference based on the normal approximation to the binomial distribution. The default value is approx=TRUE. Currently, the exact method (approx=FALSE) is only available for the one-sample case (i.e., sample.type="one.sample").

correct — logical scalar indicating whether to use the continuity correction when approx=TRUE. The default value is correct=TRUE when sample.type="two.sample" and correct=FALSE when sample.type="one.sample". This argument is ignored when approx=FALSE.

round.up — logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=FALSE. This argument is ignored unless y.var="n".

warn — logical scalar indicating whether to issue a warning. The default value is warn=TRUE. When approx=TRUE (test based on the normal approximation) and warn=TRUE, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate. When approx=FALSE (exact test) and warn=TRUE, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of alpha.

n.min — integer relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the minimum allowed value for n to use in the search algorithm. The default value is n.min=2.

n.max — integer relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the maximum allowed value for n to use in the search algorithm. The default value is n.max=10000.

tol.alpha — numeric vector relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the tolerance on alpha to use in the search algorithm (i.e., how close the actual Type I error level is to the value prescribed by the argument alpha). The default value is tol.alpha=0.1*alpha.

tol — numeric scalar relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test), or when y.var="delta". This argument is passed to the uniroot function and indicates the tolerance to use in the search algorithm. The default value is tol=1e-7.

`maxiter` integer relevant to the case when `y.var="n"` and `approx=FALSE` (i.e., when the power is based on the exact test), or when `y.var="delta"`. This argument is passed to the `uniroot` function and indicates the maximum number of iterations to use in the search algorithm. The default value is `maxiter=1000`.

`plot.it` a logical scalar indicating whether to create a new plot or add to the existing plot (see `add`) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is `plot.it=TRUE`.

`add` a logical scalar indicating whether to add the design plot to the existing plot (`add=TRUE`), or to create a plot from scratch (`add=FALSE`). The default value is `add=FALSE`. This argument is ignored if `plot.it=FALSE`.

`n.points` a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are `n.points` x-values evenly spaced between `range.x.var[1]` and `range.x.var[2]`. The default value is `n.points=100`.

`plot.col` a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col="black"`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

`digits` a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of `options("digits")`.

`cex.main`, `main`, `xlab`, `ylab`, `type`, `...`
additional graphical parameters (see `par`).

Details

See the help files for `propTestPower`, `propTestN`, and `propTestMdd` for information on how to compute the power, sample size, or minimal detectable difference for a one- or two-sample proportion test.

Value

`plotPropTestDesign` invisibly returns a list with components `x.var` and `y.var`, giving coordinates of the points that have been or would have been plotted.

Note

See the help files for `propTestPower`, `propTestN`, and `propTestMdd`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for propTestPower, propTestN, and propTestMdd.

See Also

propTestPower, propTestN, propTestMdd, Binomial, binom.test, prop.test.

Examples

```
# Look at the relationship between power and sample size for a
# one-sample proportion test, assuming the true proportion is 0.6, the
# hypothesized proportion is 0.5, and a 5% significance level.
# Compute the power based on the normal approximation to the binomial
# distribution.

dev.new()
plotPropTestDesign()

#----------

# For a two-sample proportion test, plot sample size vs. the minimal detectable
# difference for various levels of power, using a 5% significance level and a
# two-sided alternative:

dev.new()
plotPropTestDesign(x.var = "delta", y.var = "n", sample.type = "two",
  ylim = c(0, 2800), main="")

plotPropTestDesign(x.var = "delta", y.var = "n", sample.type = "two",
  power = 0.9, add = TRUE, plot.col = "red")

plotPropTestDesign(x.var = "delta", y.var = "n", sample.type = "two",
  power = 0.8, add = TRUE, plot.col = "blue")

legend("topright", c("95%", "90%", "80%"), lty = 1,
  lwd = 3 * par("cex"), col = c("black", "red", "blue"), bty = "n")

title(main = paste("Sample Size vs. Minimal Detectable Difference for Two-Sample",
  "Proportion Test with p2=0.5, Alpha=0.05 and Various Powers", sep = "\n"))

#==========

# Example 22-3 on page 22-20 of USEPA (2009) involves determining whether more than
# 10% of chlorine gas containers are stored at pressures above a compliance limit.
# We want to test the one-sided null hypothesis that 10% or fewer of the containers
# are stored at pressures greater than the compliance limit versus the alternative
# that more than 10% are stored at pressures greater than the compliance limit.
# We want to have at least 90% power of detecting a true proportion of 30% or
# greater, using a 5% Type I error level.

# Here we will modify this example and create a plot of power versus
# sample size for various assumed minimal detactable differences,
```

```
# using a 5% Type I error level.


dev.new()
plotPropTestDesign(x.var = "n", y.var = "power",
  sample.type = "one", alternative = "greater",
  p0.or.p2 = 0.1, p.or.p1 = 0.25,
  range.x.var = c(20, 50), ylim = c(0.6, 1), main = "")

plotPropTestDesign(x.var = "n", y.var = "power",
  sample.type = "one", alternative = "greater",
  p0.or.p2 = 0.1, p.or.p1 = 0.3,
  range.x.var = c(20, 50), add = TRUE, plot.col = "red")

plotPropTestDesign(x.var = "n", y.var = "power",
  sample.type = "one", alternative = "greater",
  p0.or.p2 = 0.1, p.or.p1 = 0.35,
  range.x.var = c(20, 50), add = TRUE, plot.col = "blue")

legend("bottomright", c("p=0.35", "p=0.3", "p=0.25"), lty = 1,
  lwd = 3 * par("cex"), col = c("blue", "red", "black"), bty = "n")

title(main = paste("Power vs. Sample Size for One-Sided One-Sample Proportion",
  "Test with p0=0.1, Alpha=0.05 and Various Detectable Differences",
  sep = "\n"))

#==========

# Clean up
#---------
graphics.off()
```

`plotTolIntNormDesign` *Plots for a Sampling Design Based on a Tolerance Interval for a Normal Distribution*

Description

Create plots involving sample size, half-width, estimated standard deviation, coverage, and confidence level for a tolerance interval for a normal distribution.

Usage

```
plotTolIntNormDesign(x.var = "n", y.var = "half.width", range.x.var = NULL,
  n = 25, half.width = ifelse(x.var == "sigma.hat", 3 * max.x, 3 * sigma.hat),
  sigma.hat = 1, coverage = 0.95, conf.level = 0.95, cov.type = "content",
  round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE,
  add = FALSE, n.points = 100, plot.col = 1, plot.lwd = 3 * par("cex"),
  plot.lty = 1, digits = .Options$digits, ..., main = NULL, xlab = NULL,
  ylab = NULL, type = "l")
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "half.width" (half-width), "sigma.hat" (estimated standard deviation), "coverage" (the coverage), and "conf.level" (the confidence level).
y.var	character string indicating what variable to use for the y-axis. Possible values are "half.width" (the half-width; the default), and "n" (sample size).
range.x.var	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="half.width" the default value is c(2.5 * sigma.hat, 4 * sigma.hat). When x.var="sigma.hat", the default value is c(0.1, 2). When x.var="coverage" or x.var="conf.level", the default value is c(0.5, 0.99).
n	positive integer greater than 1 indicating the sample size upon which the tolerance interval is based. The default value is n=25. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
half.width	positive scalar indicating the half-width of the prediction interval. The default value depends on the value of x.var. When x.var="sigma.hat" the default value is 3 times the second value of range.x.var. When x.var is not equal to "sigma.hat" the default value is half.width=4*sigma.hat. This argument is ignored if either x.var="half.width" or y.var="half.width".
sigma.hat	numeric scalar specifying the value of the estimated standard deviation. The default value is sigma.hat=1. This argument is ignored if x.var="sigma.hat".
coverage	numeric scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.
conf.level	numeric scalar between 0 and 1 indicating the confidence level of the tolerance interval. The default value is conf.level=0.95.
cov.type	character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation).
round.up	for the case when y.var="n", logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=TRUE.
n.max	for the case when y.var="n", positive integer greater than 1 specifying the maximum possible sample size. The default value is n.max=5000.
tol	for the case when y.var="n", numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.
maxiter	for the case when y.var="n", positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.
plot.it	a logical scalar indicating whether to create a plot or add to the existing plot (see explanation of the argument add below) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see the section VALUE). The default value is plot.it=TRUE.

add	a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.
n.points	a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.
plot.col	a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.
plot.lwd	a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.
plot.lty	a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.
digits	a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").
main, xlab, ylab, type, ...	additional graphical parameters (see par).

Details

See the help files for tolIntNorm, tolIntNormK, tolIntNormHalfWidth, and tolIntNormN for information on how to compute a tolerance interval for a normal distribution, how the half-width is computed when other quantities are fixed, and how the sample size is computed when other quantities are fixed.

Value

plotTolIntNormDesign invisibly returns a list with components:

x.var	x-coordinates of points that have been or would have been plotted.
y.var	y-coordinates of points that have been or would have been plotted.

Note

See the help file for tolIntNorm.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce tolerance intervals. The functions tolIntNormHalfWidth, tolIntNormN, and plotTolIntNormDesign can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for tolIntNorm.

See Also

tolIntNorm, tolIntNormK, tolIntNormN, plotTolIntNormDesign, Normal.

Examples

```
# Look at the relationship between half-width and sample size for a
# 95% beta-content tolerance interval, assuming an estimated standard
# deviation of 1 and a confidence level of 95%:

dev.new()
plotTolIntNormDesign()

#==========

# Plot half-width vs. coverage for various levels of confidence:

dev.new()
plotTolIntNormDesign(x.var = "coverage", y.var = "half.width",
  ylim = c(0, 3.5), main="")

plotTolIntNormDesign(x.var = "coverage", y.var = "half.width",
  conf.level = 0.9, add = TRUE, plot.col = "red")

plotTolIntNormDesign(x.var = "coverage", y.var = "half.width",
  conf.level = 0.8, add = TRUE, plot.col = "blue")

legend("topleft", c("95%", "90%", "80%"), lty = 1, lwd = 3 * par("cex"),
  col = c("black", "red", "blue"), bty = "n")

title(main = paste("Half-Width vs. Coverage for Tolerance Interval",
  "with Sigma Hat=1 and Various Confidence Levels", sep = "\n"))

#==========

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence  using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5).  The tolerance limit
# is based on the data from the background wells.

# Here we will first take the log of the data and then estimate the
# standard deviation based on the two background wells.  We will use this
# estimate of standard deviation to plot the half-widths of
# future tolerance intervals on the log-scale for various sample sizes.
```

```
head(EPA.09.Ex.17.3.chrysene.df)
#  Month   Well  Well.type Chrysene.ppb
#1     1 Well.1 Background         19.7
#2     2 Well.1 Background         39.2
#3     3 Well.1 Background          7.8
#4     4 Well.1 Background         12.8
#5     1 Well.2 Background         10.2
#6     2 Well.2 Background          7.2

longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene.ppb", "Month", "Well")
#  Well.1 Well.2 Well.3 Well.4 Well.5
#1   19.7   10.2   68.0   26.8   47.0
#2   39.2    7.2   48.9   17.7   30.5
#3    7.8   16.1   30.1   31.9   15.0
#4   12.8    5.7   38.1   22.2   23.4

summary.stats <- summaryStats(log(Chrysene.ppb) ~ Well.type,
  data = EPA.09.Ex.17.3.chrysene.df)

summary.stats
#            N   Mean     SD Median    Min    Max
#Background  8 2.5086 0.6279 2.4359 1.7405 3.6687
#Compliance 12 3.4173 0.4361 3.4111 2.7081 4.2195

sigma.hat <- summary.stats["Background", "SD"]
sigma.hat
#[1] 0.6279

dev.new()
plotTolIntNormDesign(x.var = "n", y.var = "half.width",
  range.x.var = c(5, 40), sigma.hat = sigma.hat, cex.main = 1)

#==========

# Clean up
#---------
rm(summary.stats, sigma.hat)
graphics.off()
```

`plotTolIntNparDesign` *Plots for a Sampling Design Based on a Nonparametric Tolerance Interval*

Description

Create plots involving sample size (n), coverage (β), and confidence level $(1 - \alpha)$ for a nonparametric tolerance interval.

Usage

```
plotTolIntNparDesign(x.var = "n", y.var = "conf.level", range.x.var = NULL, n = 25,
  coverage = 0.95, conf.level = 0.95, ti.type = "two.sided", cov.type = "content",
    ltl.rank = ifelse(ti.type == "upper", 0, 1),
   n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1), plot.it = TRUE,
    add = FALSE, n.points = 100, plot.col = "black", plot.lwd = 3 * par("cex"),
  plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ..., main = NULL,
    xlab = NULL, ylab = NULL, type = "l")
```

Arguments

`x.var`	character string indicating what variable to use for the x-axis. Possible values are `"n"` (sample size; the default), `"coverage"` (the coverage), and `"conf.level"` (the confidence level).
`y.var`	character string indicating what variable to use for the y-axis. Possible values are `"conf.level"` (the confidence level; the default), `"n"` (sample size), and `"coverage"` (the coverage).
`range.x.var`	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of `x.var`. When `x.var="n"` the default value is `c(2,50)`. When `x.var="coverage"` or `x.var="conf"`, the default value is `c(0.5, 0.99)`.
`n`	numeric scalar indicating the sample size. The default value is `max(25, lpl.rank + n.plus.one.minus.upl.rank + 1)`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed. This argument is ignored if either `x.var="n"` or `y.var="n"`.
`coverage`	numeric scalar between 0 and 1 specifying the coverage of the tolerance interval. The default value is `coverage=0.95`. This argument is ignored if `x.var="coverage"` or `y.var="coverage"`.
`conf.level`	a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level=0.95`. This argument is ignored if `x.var="conf.level"` or `y.var="conf.level"`, or if `cov.type="expectation"`.
`ti.type`	character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation).
`ltl.rank`	vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If `ti.type="two-sided"` or `ti.type="lower"`, the default value is `ltl.rank=1` (implying the minimum value of `x` is used as the lower bound of the tolerance interval). If `ti.type="upper"`, this argument is set equal to `0`.
`n.plus.one.minus.utl.rank`	vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of `n.plus.one.minus.utl.rank=1` (the default) means use the first largest value, and in general a value of

n.plus.one.minus.utl.rank=i means use the i'th largest value. If ti.type="lower", this argument is set equal to 0.

plot.it a logical scalar indicating whether to create a plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.

add a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.

n.points a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=100.

plot.col a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.

plot.lwd a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.

plot.lty a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ... additional graphical parameters (see par).

Details

See the help file for tolIntNpar, tolIntNparConfLevel, tolIntNparCoverage, and tolIntNparN for information on how to compute a nonparametric tolerance interval, how the confidence level is computed when other quantities are fixed, how the coverage is computed when other quantites are fixed, and and how the sample size is computed when other quantities are fixed.

Value

plotTolIntNparDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note

See the help file for tolIntNpar.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions tolIntNparN, tolIntNparCoverage, tolIntNparConfLevel, and plotTolIntNparDesign can be used to investigate these relationships for constructing nonparametric tolerance intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for tolIntNpar.

See Also

tolIntNpar, tolIntNparConfLevel, tolIntNparCoverage, tolIntNparN.

Examples

```
# Look at the relationship between confidence level and sample size for a two-sided
# nonparametric tolerance interval.

dev.new()
plotTolIntNparDesign()

#==========

# Plot confidence level vs. sample size for various values of coverage:

dev.new()
plotTolIntNparDesign(coverage = 0.7, ylim = c(0,1), main = "")

plotTolIntNparDesign(coverage = 0.8, add = TRUE, plot.col = "red")

plotTolIntNparDesign(coverage = 0.9, add = TRUE, plot.col = "blue")

legend("bottomright", c("coverage = 70%", "coverage = 80%", "coverage = 90%"), lty=1,
  lwd = 3 * par("cex"), col = c("black", "red", "blue"), bty = "n")

title(main = paste("Confidence Level vs. Sample Size for Nonparametric TI",
  "with Various Levels of Coverage", sep = "\n"))

#==========

# Example 17-4 on page 17-21 of USEPA (2009) uses copper concentrations (ppb) from 3
# background wells to set an upper limit for 2 compliance wells.  There are 6 observations
# per well, and the maximum value from the 3 wells is set to the 95% confidence upper
# tolerance limit, and we need to determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
#[1] 0.8826538

# Here we will modify the example and look at confidence level versus coverage for
# a set sample size of n = 24.

dev.new()
plotTolIntNparDesign(x.var = "coverage", y.var = "conf.level", n = 24, ti.type = "upper")
```

```
#==========

# Clean up
#---------
graphics.off()
```

---

plotTTestDesign *Plots for a Sampling Design Based on a One- or Two-Sample t-Test*

Description

Create plots involving sample size, power, scaled difference, and significance level for a one- or two-sample t-test.

Usage

```
plotTTestDesign(x.var = "n", y.var = "power", range.x.var = NULL,
  n.or.n1 = 25, n2 = n.or.n1,
  delta.over.sigma = switch(alternative, greater = 0.5, less = -0.5,
    two.sided = ifelse(two.sided.direction == "greater", 0.5, -0.5)),
  alpha = 0.05, power = 0.95,
  sample.type = ifelse(!missing(n2), "two.sample", "one.sample"),
  alternative = "two.sided", two.sided.direction = "greater", approx = FALSE,
  round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE,
  add = FALSE, n.points = 50, plot.col = "black", plot.lwd = 3 * par("cex"),
  plot.lty = 1, digits = .Options$digits, ..., main = NULL, xlab = NULL,
  ylab = NULL, type = "l")
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "delta.over.sigma" (scaled minimal detectable difference), "power" (power of the test), and "alpha" (significance level of the test).
y.var	character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "delta.over.sigma" (scaled minimal detectable difference), and "n" (sample size).
range.x.var	numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var. When x.var="n" the default value is c(2,50). When x.var="delta.over.sigma" and alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is c(0.5, 2). When x.var="delta.over.sigma" and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is -c(2, 0.5). When x.var="power" the default value is c(alpha + .Machine$double.eps, 0.95). When x.var="alpha", the default value is c(0.01, 0.2).

n.or.n1 numeric scalar indicating the sample size. The default value is n.or.n1=25. When sample.type="one.sample", n.or.n1 denotes the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

n2 numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

delta.over.sigma
numeric scalar specifying the ratio of the true difference (δ) to the population standard deviation (σ). This is also called the "scaled difference". When alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is delta.over.sigma=0.5. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is delta.over.sigma=-0.5. This argument is ignored when x.var="delta.over.sigma" or y.var="delta.over.sigma".

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05. This argument is ignored when x.var="alpha".

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95. This argument is ignored when x.var="power" or y.var="power".

sample.type character string indicating whether the design is based on a one-sample or two-sample t-test. When sample.type="one.sample", the computations for the plot are based on a one-sample t-test. When sample.type="two.sample", the computations for the plot are based on a two-sample t-test. The default value is sample.type="one.sample".

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

two.sided.direction
character string indicating the direction (positive or negative) for the scaled minimal detectable difference when alternative="two.sided". When two.sided.direction="greater" (the default), the scaled minimal detectable difference is positive. When two.sided.direction="less", the scaled minimal detectable difference is negative. This argument is ignored unless alternative="two.sided" and either x.var="delta" or y.var="delta".

approx logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is approx=FALSE.

round.up logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=FALSE. This argument is ignored unless y.var="n".

n.max for the case when y.var="n", a positive integer greater than 1 indicating the maximum sample size when sample.type="one.sample" or the maximum sample size for group 1 when sample.type="two.sample". The default value is n.max=5000.

tol	numeric scalar relevant to the case when y.var="n" or y.var="delta.over.sigma". This argument is passed to the uniroot function and indicates the tolerance to use in the search algorithm. The default value is tol=1e-7.
maxiter	numeric scalar relevant to the case when y.var="n" and approx=FALSE (i.e., when the power is based on the exact test), or when y.var="delta.over.sigma". This argument is passed to the uniroot function and is a positive integer indicating the maximum number of iterations. The default value is maxiter=1000.
plot.it	a logical scalar indicating whether to create a new plot or add to the existing plot (see add) on the current graphics device. If plot.it=FALSE, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is plot.it=TRUE.
add	a logical scalar indicating whether to add the design plot to the existing plot (add=TRUE), or to create a plot from scratch (add=FALSE). The default value is add=FALSE. This argument is ignored if plot.it=FALSE.
n.points	a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are n.points x-values evenly spaced between range.x.var[1] and range.x.var[2]. The default value is n.points=50.
plot.col	a numeric scalar or character string determining the color of the plotted line or points. The default value is plot.col="black". See the entry for col in the help file for par for more information.
plot.lwd	a numeric scalar determining the width of the plotted line. The default value is 3*par("cex"). See the entry for lwd in the help file for par for more information.
plot.lty	a numeric scalar determining the line type of the plotted line. The default value is plot.lty=1. See the entry for lty in the help file for par for more information.
digits	a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").
main, xlab, ylab, type, ...	additional graphical parameters (see par).

Details

See the help files for tTestPower, tTestN, and tTestScaledMdd for information on how to compute the power, sample size, or scaled minimal detectable difference for a one- or two-sample t-test.

Value

plotTTestDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note

See the help files for tTestPower, tTestN, and tTestScaledMdd.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for tTestPower, tTestN, and tTestScaledMdd.

See Also

tTestPower, tTestN, tTestScaledMdd, t.test.

Examples

```
# Look at the relationship between power and sample size for a two-sample t-test,
# assuming a scaled difference of 0.5 and a 5% significance level:

dev.new()
plotTTestDesign(sample.type = "two")

#----------

# For a two-sample t-test, plot sample size vs. the scaled minimal detectable
# difference for various levels of power, using a 5% significance level:

dev.new()
plotTTestDesign(x.var = "delta.over.sigma", y.var = "n", sample.type = "two",
  ylim = c(0, 110), main="")

plotTTestDesign(x.var = "delta.over.sigma", y.var = "n", sample.type = "two",
  power = 0.9, add = TRUE, plot.col = "red")

plotTTestDesign(x.var = "delta.over.sigma", y.var = "n", sample.type = "two",
  power = 0.8, add = TRUE, plot.col = "blue")

legend("topright", c("95%", "90%", "80%"), lty = 1,
  lwd = 3 * par("cex"), col = c("black", "red", "blue"), bty = "n")

title(main = paste("Sample Size vs. Scaled Difference for",
  "Two-Sample t-Test, with Alpha=0.05 and Various Powers",
  sep="\n"))

#==========

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009), look at
# power versus scaled minimal detectable difference for various sample
# sizes in the context of the problem of using a one-sample t-test to
# compare the mean for the well with the MCL of 7 ppb.  Use alpha = 0.01,
# assume an upper one-sided alternative (i.e., compliance well mean larger
# than 7 ppb).

dev.new()
plotTTestDesign(x.var = "delta.over.sigma", y.var = "power",
  range.x.var = c(0.5, 2), n.or.n1 = 8, alpha = 0.01,
  alternative = "greater", ylim = c(0, 1), main = "")

plotTTestDesign(x.var = "delta.over.sigma", y.var = "power",
```

```
  range.x.var = c(0.5, 2), n.or.n1 = 6, alpha = 0.01,
  alternative = "greater", add = TRUE, plot.col = "red")

plotTTestDesign(x.var = "delta.over.sigma", y.var = "power",
  range.x.var = c(0.5, 2), n.or.n1 = 4, alpha = 0.01,
  alternative = "greater", add = TRUE, plot.col = "blue")

legend("topleft", paste("N =", c(8, 6, 4)), lty = 1, lwd = 3 * par("cex"),
  col = c("black", "red", "blue"), bty = "n")

title(main = paste("Power vs. Scaled Difference for One-Sample t-Test",
  "with Alpha=0.01 and Various Sample Sizes", sep="\n"))

#==========

# Clean up
#---------
graphics.off()
```

---

plotTTestLnormAltDesign *Plots for a Sampling Design Based on a One- or Two-Sample t-Test, Assuming Lognormal Data*

Description

Create plots involving sample size, power, ratio of means, coefficient of variation, and significance level for a one- or two-sample t-test, assuming lognormal data.

Usage

```
plotTTestLnormAltDesign(x.var = "n", y.var = "power", range.x.var = NULL,
  n.or.n1 = 25, n2 = n.or.n1,
  ratio.of.means = switch(alternative, greater = 2, less = 0.5,
    two.sided = ifelse(two.sided.direction == "greater", 2, 0.5)),
  cv = 1, alpha = 0.05, power = 0.95,
  sample.type = ifelse(!missing(n2), "two.sample", "one.sample"),
  alternative = "two.sided", two.sided.direction = "greater", approx = FALSE,
  round.up = FALSE, n.max = 5000, tol = 1e-07, maxiter = 1000, plot.it = TRUE,
  add = FALSE, n.points = 50, plot.col = "black", plot.lwd = 3 * par("cex"),
  plot.lty = 1, digits = .Options$digits, cex.main = par("cex"), ...,
  main = NULL, xlab = NULL, ylab = NULL, type = "l")
```

Arguments

x.var	character string indicating what variable to use for the x-axis. Possible values are "n" (sample size; the default), "ratio.of.means" (minimal or maximal detectable ratio of means), "cv" (coefficient of variaiton), "power" (power of the test), and "alpha" (significance level of the test).

y.var character string indicating what variable to use for the y-axis. Possible values are "power" (power of the test; the default), "ratio.of.means" (minimal or maximal detectable ratio of means), and "n" (sample size).

range.x.var numeric vector of length 2 indicating the range of the x-variable to use for the plot. The default value depends on the value of x.var:

- When x.var="n" the default value is c(2,50).
- When x.var="ratio.of.means" and alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is c(1, 2).
- When x.var="delta" and alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is c(0.5, 1).
- When x.var="cv" the default value is c(0.5, 2).
- When x.var="power" the default value is c(alpha + .Machine$double.eps, 0.95).
- When x.var="alpha", the default value is c(0.01, 0.2).

n.or.n1 numeric scalar indicating the sample size. The default value is n.or.n1=25. When sample.type="one.sample", n.or.n1 denotes the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored if either x.var="n" or y.var="n".

n2 numeric scalar indicating the sample size for group 2. The default value is the value of n.or.n1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed. This argument is ignored when sample.type="one.sample".

ratio.of.means numeric scalar specifying the ratio of the first mean to the second mean. When sample.type="one.sample", this is the ratio of the population mean to the hypothesized mean. When sample.type="two.sample", this is the ratio of the mean of the first population to the mean of the second population.

When alternative="greater" or alternative="two.sided" and two.sided.direction="greater", the default value is ratio.of.means=2. When alternative="less" or alternative="two.sided" and two.sided.direction="less", the default value is ratio.of.means=0.5. This argument is ignored when x.var="ratio.of.means" or y.var="ratio.of.means".

cv numeric scalar: a positive value specifying the coefficient of variation. When sample.type="one.sample", this is the population coefficient of variation. When sample.type="two.sample", this is the coefficient of variation for both the first and second population. The default value is cv=1.

alpha numeric scalar between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

power numeric scalar between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.

sample.type character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed

power is based on a hypothesis test for a single mean. When `sample.type="two.sample"`, the computed power is based on a hypothesis test for the difference between two means. The default value is `sample.type="one.sample"` unless the argument `n2` is supplied.

`alternative` character string indicating the kind of alternative hypothesis. The possible values are `"two.sided"` (the default), `"greater"`, and `"less"`.

`two.sided.direction`
character string indicating the direction (greater than 1 or less than 1) for the detectable ratio of means when `alternative="two.sided"`. When `two.sided.direction="greater"` (the default), the detectable ratio of means is greater than 1. When `two.sided.direction="less"`, the detectable ratio of means is less than 1 (but greater than 0). This argument is ignored if `alternative="less"` or `alternative="greater"`.

`approx` logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `approx=FALSE`.

`round.up` logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is `TRUE`.

`n.max` for the case when `y.var="n"`, a positive integer greater than 1 indicating the maximum sample size when `sample.type="one.sample"` or the maximum sample size for group 1 when `sample.type="two.sample"`. The default value is `n.max=5000`.

`tol` numeric scalar indicating the toloerance to use in the `uniroot` search algorithm. The default value is `tol=1e-7`.

`maxiter` positive integer indicating the maximum number of iterations argument to pass to the `uniroot` function. The default value is `maxiter=1000`.

`plot.it` a logical scalar indicating whether to create a new plot or add to the existing plot (see `add`) on the current graphics device. If `plot.it=FALSE`, no plot is produced, but a list of (x,y) values is returned (see VALUE). The default value is `plot.it=TRUE`.

`add` a logical scalar indicating whether to add the design plot to the existing plot (`add=TRUE`), or to create a plot from scratch (`add=FALSE`). The default value is `add=FALSE`. This argument is ignored if `plot.it=FALSE`.

`n.points` a numeric scalar specifying how many (x,y) pairs to use to produce the plot. There are `n.points` x-values evenly spaced between `range.x.var[1]` and `range.x.var[2]`. The default value is `n.points=100`.

`plot.col` a numeric scalar or character string determining the color of the plotted line or points. The default value is `plot.col="black"`. See the entry for `col` in the help file for `par` for more information.

`plot.lwd` a numeric scalar determining the width of the plotted line. The default value is `3*par("cex")`. See the entry for `lwd` in the help file for `par` for more information.

`plot.lty` a numeric scalar determining the line type of the plotted line. The default value is `plot.lty=1`. See the entry for `lty` in the help file for `par` for more information.

digits a scalar indicating how many significant digits to print out on the plot. The default value is the current setting of options("digits").

cex.main, main, xlab, ylab, type, ... additional graphical parameters (see par).

Details

See the help files for tTestLnormAltPower, tTestLnormAltN, and tTestLnormAltRatioOfMeans for information on how to compute the power, sample size, or ratio of means for a one- or two-sample t-test assuming lognormal data.

Value

plotTTestLnormAltDesign invisibly returns a list with components x.var and y.var, giving coordinates of the points that have been or would have been plotted.

Note

See the help files for tTestLnormAltPower, tTestLnormAltN, and tTestLnormAltRatioOfMeans.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for tTestLnormAltPower, tTestLnormAltN, and tTestLnormAltRatioOfMeans.

See Also

tTestLnormAltPower, tTestLnormAltN, tTestLnormAltRatioOfMeans, t.test.

Examples

```
# Look at the relationship between power and sample size for a two-sample t-test,
# assuming lognormal data, a ratio of means of 2, a coefficient of variation
# of 1, and a 5% significance level:

dev.new()
plotTTestLnormAltDesign(sample.type = "two")

#----------

# For a two-sample t-test based on lognormal data, plot sample size vs. the
# minimal detectable ratio for various levels of power, assuming a coefficient
# of variation of 1 and using a 5% significance level:

dev.new()
plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "n",
  range.x.var = c(1.5, 2), sample.type = "two", ylim = c(20, 120), main="")
```

```
plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "n",
  range.x.var = c(1.5, 2), sample.type="two", power = 0.9,
  add = TRUE, plot.col = "red")

plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "n",
  range.x.var = c(1.5, 2), sample.type="two", power = 0.8,
  add = TRUE, plot.col = "blue")

legend("topright", c("95%", "90%", "80%"), lty=1, lwd = 3*par("cex"),
  col = c("black", "red", "blue"), bty = "n")

title(main = paste("Sample Size vs. Ratio of Lognormal Means for",
  "Two-Sample t-Test, with CV=1, Alpha=0.05 and Various Powers",
  sep="\n"))

#==========

# The guidance document Soil Screening Guidance: Technical Background Document
# (USEPA, 1996c, Part 4) discusses sampling design and sample size calculations
# for studies to determine whether the soil at a potentially contaminated site
# needs to be investigated for possible remedial action. Let 'theta' denote the
# average concentration of the chemical of concern.  The guidance document
# establishes the following goals for the decision rule (USEPA, 1996c, p.87):
#
#     Pr[Decide Don't Investigate | theta > 2 * SSL] = 0.05
#
#     Pr[Decide to Investigate | theta <= (SSL/2)] = 0.2
#
# where SSL denotes the pre-established soil screening level.
#
# These goals translate into a Type I error of 0.2 for the null hypothesis
#
#     H0: [theta / (SSL/2)] <= 1
#
# and a power of 95% for the specific alternative hypothesis
#
#     Ha: [theta / (SSL/2)] = 4
#
# Assuming a lognormal distribution, a coefficient of variation of 2, and the above
# values for Type I error and power, create a performance goal diagram
# (USEPA, 1996c, p.89) showing the power of a one-sample test versus the minimal
# detectable ratio of theta/(SSL/2) when the sample size is 6 and the exact power
# calculations are used.

dev.new()
plotTTestLnormAltDesign(x.var = "ratio.of.means", y.var = "power",
  range.x.var = c(1, 5), n.or.n1 = 6, cv = 2, alpha = 0.2,
  alternative = "greater", approx = FALSE, ylim = c(0.2, 1),
  xlab = "theta / (SSL/2)")

#==========

# Clean up
```

```
#---------
graphics.off()
```

---

pointwise *Pointwise Confidence Limits for Predictions*

Description

Computes pointwise confidence limits for predictions computed by the function predict.

Usage

```
pointwise(results.predict, coverage = 0.99,
  simultaneous = FALSE, individual = FALSE)
```

Arguments

results.predict	output from a call to predict with se.fit=TRUE.
coverage	optional numeric scalar between 0 and 1 indicating the confidence level associated with the confidence limits. The default value is coverage=0.99.
simultaneous	optional logical scalar indicating whether to base the confidence limits for the predicted values on simultaneous or non-simultaneous prediction limits. The default value is simultaneous=FALSE.
individual	optional logical scalar indicating whether to base the confidence intervals for the predicted values on prediction limits for the mean (individual=FALSE) or prediction limits for an individual observation (individual=TRUE). The default value is individual=FALSE.

Details

This function computes pointwise confidence limits for predictions computed by the function predict. The limits are computed at those points specified by the argument newdata of predict.

The predict function is a generic function with methods for several different classes. The funciton pointwise was part of the S language. The modifications to pointwise in the package **EnvStats** involve confidence limits for predictions for a linear model (i.e., an object of class "lm").

Confidence Limits for a Predicted Mean Value (individual=FALSE). Consider a standard linear model with p predictor variables. Often, one of the major goals of regression analysis is to predict a future value of the response variable given known values of the predictor variables. The equations for the predicted mean value of the response given fixed values of the predictor variables as well as the equation for a two-sided (1-α)100% confidence interval for the mean value of the response can be found in Draper and Smith (1998, p.80) and Millard and Neerchal (2001, p.547).

Technically, this formula is a confidence interval for the mean of the response for one set of fixed values of the predictor variables and corresponds to the case when simultaneous=FALSE. To create simultaneous confidence intervals over the range of of the predictor variables, the critical t-value in the equation has to be replaced with a critical F-value and the modified formula is given in Draper

and Smith (1998, p. 83), Miller (1981a, p. 111), and Millard and Neerchal (2001, p. 547). This formula is used in the case when `simultaneous=TRUE`.

Confidence Limits for a Predicted Individual Value (`individual=TRUE`). In the above section we discussed how to create a confidence interval for the mean of the response given fixed values for the predictor variables. If instead we want to create a prediction interval for a single future observation of the response variable, the fomula is given in Miller (1981a, p. 115) and Millard and Neerchal (2001, p. 551).

Technically, this formula is a prediction interval for a single future observation for one set of fixed values of the predictor variables and corresponds to the case when `simultaneous=FALSE`. Miller (1981a, p. 115) gives a formula for simultaneous prediction intervals for k future observations. If we are interested in creating an interval that will encompass *all* possible future observations over the range of the preictor variables with some specified probability however, we need to create simultaneous tolerance intervals. A formula for such an interval was developed by Lieberman and Miller (1963) and is given in Miller (1981a, p. 124). This formula is used in the case when `simultaneous=TRUE`.

Value

a list with the following components:

`upper`	upper limits of pointwise confidence intervals.
`fit`	surface values. This is the same as the component `fit` of the argument `results.predict`.
`lower`	lower limits of pointwise confidence intervals.

Note

The function `pointwise` is called by the functions `detectionLimitCalibrate` and `inversePredictCalibrate`, which are used in **calibration**.

Almost always the process of determining the concentration of a chemical in a soil, water, or air sample involves using some kind of machine that produces a signal, and this signal is related to the concentration of the chemical in the physical sample. The process of relating the machine signal to the concentration of the chemical is called **calibration** (see `calibrate`). Once calibration has been performed, estimated concentrations in physical samples with unknown concentrations are computed using inverse regression. The uncertainty in the process used to estimate the concentration may be quantified with decision, detection, and quantitation limits.

In practice, only the point estimate of concentration is reported (along with a possible qualifier), without confidence bounds for the true concentration C. This is most unfortunate because it gives the impression that there is no error associated with the reported concentration. Indeed, both the International Organization for Standardization (ISO) and the International Union of Pure and Applied Chemistry (IUPAC) recommend always reporting both the estimated concentration and the uncertainty associated with this estimate (Currie, 1997).

Author(s)

Authors of S (for code for `pointwise` in S).

Steven P. Millard (for modification to allow the arguments `simultaneous` and `individual`); <EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., and Hastie, T.J., eds. (1992). *Statistical Models in S*. Chapman and Hall/CRC, Boca Raton, FL.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, Chapter 3.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.546-553.

Miller, R.G. (1981a). *Simultaneous Statistical Inference*. Springer-Verlag, New York, pp.111, 124.

See Also

predict, predict.lm, lm, calibrate, inversePredictCalibrate, detectionLimitCalibrate.

Examples

```
# Using the data in the built-in data frame Air.df,
# fit the cube root of ozone as a function of temperature.
# Then compute predicted values for ozone at 70 and 90
# degrees F, and compute 95% confidence intervals for the
# mean value of ozone at these temperatures.

# First create the lm object
#---------------------------

ozone.fit <- lm(ozone ~ temperature, data = Air.df)


# Now get predicted values and CIs at 70 and 90 degrees
#------------------------------------------------------

predict.list <- predict(ozone.fit,
  newdata = data.frame(temperature = c(70, 90)), se.fit = TRUE)

pointwise(predict.list, coverage = 0.95)
# $upper
#        1        2
# 2.839145 4.278533

# $fit
#        1        2
# 2.697810 4.101808

# $lower
#        1        2
# 2.556475 3.925082

#------------------------------------------------------------------------

# Continuing with the above example, create a scatterplot of ozone
# vs. temperature, and add the fitted line along with simultaneous
```

```
# 95% confidence bands.

x <- Air.df$temperature

y <- Air.df$ozone

dev.new()
plot(x, y, xlab="Temperature (degrees F)",
  ylab = expression(sqrt("Ozone (ppb)", 3)))

abline(ozone.fit, lwd = 2)

new.x <- seq(min(x), max(x), length=100)

predict.ozone <- predict(ozone.fit,
  newdata = data.frame(temperature = new.x), se.fit = TRUE)

ci.ozone <- pointwise(predict.ozone, coverage=0.95,
  simultaneous=TRUE)

lines(new.x, ci.ozone$lower, lty=2, lwd = 2, col = 2)

lines(new.x, ci.ozone$upper, lty=2, lwd = 2, col = 2)

title(main=paste("Cube Root Ozone vs. Temperature with Fitted Line",
  "and Simultaneous 95% Confidence Bands",
  sep="\n"))

#------------------------------------------------------------------------

# Redo the last example by creating non-simultaneous
# confidence bounds and prediction bounds as well.

dev.new()
plot(x, y, xlab = "Temperature (degrees F)",
  ylab = expression(sqrt("Ozone (ppb)", 3)))

abline(ozone.fit, lwd = 2)

new.x <- seq(min(x), max(x), length=100)

predict.ozone <- predict(ozone.fit,
  newdata = data.frame(temperature = new.x), se.fit = TRUE)

ci.ozone <- pointwise(predict.ozone, coverage=0.95)

lines(new.x, ci.ozone$lower, lty=2, col = 2, lwd = 2)

lines(new.x, ci.ozone$upper, lty=2, col = 2, lwd = 2)

pi.ozone <- pointwise(predict.ozone, coverage = 0.95,
  individual = TRUE)
```

```
lines(new.x, pi.ozone$lower, lty=4, col = 4, lwd = 2)

lines(new.x, pi.ozone$upper, lty=4, col = 4, lwd = 2)

title(main=paste("Cube Root Ozone vs. Temperature with Fitted Line",
  "and 95% Confidence and Prediction Bands",
  sep="\n"))

#----------------------------------------------------------------------

# Clean up
rm(predict.list, ozone.fit, x, y, new.x, predict.ozone, ci.ozone,
  pi.ozone)
```

---

ppointsCensored *Plotting Positions for Type I Censored Data*

Description

Returns a list of "ordered" observations and associated plotting positions based on Type I left-censored or right-censored data. These plotting positions may be used to construct empirical cumulative distribution plots or quantile-quantile plots, or to estimate distribution parameters.

Usage

```
ppointsCensored(x, censored, censoring.side = "left",
  prob.method = "michael-schucany", plot.pos.con = 0.375)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
prob.method	character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are: "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "modified kaplan-meier" (modification of Kaplan-Meier method), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and

"hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)).

The default value is prob.method="michael-schucany".

The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" method is only available for censoring.side="left". See the DETAILS section for more explanation.

plot.pos.con numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".

Details

Methods for computing plotting positions for complete data sets (no censored observations) are discussed in D'Agostino, R.B. (1986a) and Cleveland (1993). For data sets with censored observations, these methods must be modified. The function ppointsCensored allows you to compute plotting positions based on any of the following methods:

- Product-limit method of Kaplan and Meier (1958) (prob.method="kaplan-meier").
- Hazard plotting method of Nelson (1972) (prob.method="nelson").
- Generalization of the product-limit method due to Michael and Schucany (1986) (prob.method="michael-schucany") (the default).
- Generalization of the product-limit method due to Hirsch and Stedinger (1987) (prob.method="hirsch-stedinger").

Let $\underline{x}$ denote a random sample of N observations from some distribution. Assume n ($0 < n < N$) of these observations are known and c ($c = N - n$) of these observations are all censored below (left-censored) or all censored above (right-censored) at k fixed censoring levels

$$T_1, T_2, \ldots, T_K;\ K \geq 1 \qquad (1)$$

For the case when $K \geq 2$, the data are said to be Type I ***multiply censored***. For the case when $K = 1$, set $T = T_1$. If the data are left-censored and all n known observations are greater than or equal to T, or if the data are right-censored and all n known observations are less than or equal to T, then the data are said to be Type I ***singly censored*** (Nelson, 1982, p.7), otherwise they are considered to be Type I multiply censored.

Let c_j denote the number of observations censored below or above censoring level T_j for $j = 1, 2, \ldots, K$, so that

$$\sum_{i=1}^{K} c_j = c \qquad (2)$$

Let $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ denote the "ordered" observations, where now "observation" means either the actual observation (for uncensored observations) or the censoring level (for censored observations). For right-censored data, if a censored observation has the same value as an uncensored one, the uncensored observation should be placed first. For left-censored data, if a censored observation has the same value as an uncensored one, the censored observation should be placed first.

Note that in this case the quantity $x_{(i)}$ does not necessarily represent the i'th "largest" observation from the (unknown) complete sample.

Finally, let Ω (omega) denote the set of n subscripts in the "ordered" sample that correspond to uncensored observations.

Product-Limit Method of Kaplan and Meier (`prob.method="kaplan-meier"`)
For complete data sets (no censored observations), the ***empirical probabilities*** estimator of the cumulative distribution function evaluated at the i'th ordered observation is given by (D'Agostino, 1986a, p.8):

$$\hat{F}[x_{(i)}] = \hat{p}_i = \frac{\#[x_j \leq x_{(i)}]}{n} \qquad (3)$$

where $\#[x_j \leq x_{(i)}]$ denotes the number of observations less than or equal to $x_{(i)}$ (see the help file for `ecdfPlot`). Kaplan and Meier (1958) extended this method of computing the empirical cdf to the case of right-censored data.

Right-Censored Data (`censoring.side="right"`)
Let $S(t)$ denote the survival function evaluated at t, that is:

$$S(t) = 1 - F(t) = Pr(X > t) \qquad (4)$$

Kaplan and Meier (1958) show that a nonparametric estimate of the survival function at the i'th ordered observation that is not censored (i.e., $i \in \Omega$), is given by:

$$\begin{aligned} \hat{S}[x_{(i)}] &= \widehat{Pr}[X > x_{(i)}] \\ &= \widehat{Pr}[X > x_{(1)}] \\ &\quad \widehat{Pr}[X > x_{(2)} | X > x_{(1)}] \quad \cdots \\ &\quad \widehat{Pr}[X > x_{(i)} | X > x_{(i-1)}] \\ &= \prod_{j \in \Omega, j \leq i} \frac{n_j - d_j}{n_j}, \quad i \in \Omega \qquad (5) \end{aligned}$$

where n_j is the number of observations (uncensored or censored) with values greater than or equal to $x_{(j)}$, and d_j denotes the number of uncensored observations exactly equal to $x_{(j)}$ (if there are no tied uncensored observations then d_j will equal 1 for all values of j). (See also Lee and Wang, 2003, pp. 64–69; Michael and Schucany, 1986). By convention, the estimate of the survival function at a censored observation is set equal to the estimated value of the survival function at the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than or equal to a particular censoring level, the estimate of the survival function is set to 1 for that censoring level.

Thus the Kaplan-Meier plotting position at the i'th ordered observation that is not censored (i.e., $i \in \Omega$), is given by:

$$\hat{p}_i = \hat{F}[x_{(i)}] = 1 - \prod_{j \in \Omega, j \leq i} \frac{n_j - d_j}{n_j} \qquad (6)$$

The plotting position for a censored observation is set equal to the plotting position associated with the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than or equal to a particular censoring level, the plotting position is set to 0 for that censoring level.

As an example, consider the following right-censored data set:

$$3, \geq 4, \geq 4, 5, 5, 6$$

The table below shows how the plotting positions are computed.

i	$x_{(i)}$	n_i	d_i	$\frac{n_i - d_i}{n_i}$	Plotting Position
1	3	6	1	$5/6$	$1 - (5/6) = 0.167$
2	≥ 4				
3	≥ 4				
4	5	3	2	$1/3$	$1 - (5/6)(1/3) = 0.722$
5	5				0.722
6	6	1	1	$0/1$	$1 - (5/6)(1/3)(0/1) = 1$

Note that for complete data sets, Equation (6) reduces to Equation (3).

Left-Censored Data (`censoring.side="left"`)
Gillespie et al. (2010) give formulas for the Kaplan-Meier estimator for the case of left-cesoring (`censoring.side="left"`). In this case, the plotting position for the i'th ordered observation, assuming it is not censored, is computed as:

$$\hat{p}_i = \hat{F}[x_{(i)}] = \prod_{j \in \Omega, j > i} \frac{n_j - d_j}{n_j} \qquad (7)$$

where n_j is the number of observations (uncensored or censored) with values less than or equal to $x_{(j)}$, and d_j denotes the number of uncensored observations exactly equal to $x_{(j)}$ (if there are no tied uncensored observations then d_j will equal 1 for all values of j). The plotting position is equal to 1 for the largest uncensored order statistic.

As an example, consider the following left-censored data set:

$$3, < 4, < 4, 5, 5, 6$$

The table below shows how the plotting positions are computed.

i	$x_{(i)}$	n_i	d_i	$\frac{n_i - d_i}{n_i}$	Plotting Position
1	3	1	1	$0/1$	$1(5/6)(3/5) = 0.5$
2	< 4				
3	< 4				
4	5	5	2	$3/5$	0.833
5	5				$1(5/6) = 0.833$
6	6	6	1	$5/6$	1

Note that for complete data sets, Equation (7) reduces to Equation (3).

Modified Kaplan-Meier Method (`prob.method="modified kaplan-meier"`)
(Left-Censored Data Only.) For left-censored data, the modified Kaplan-Meier method is the same as the Kaplan-Meier method, except that for the largest uncensored order statistic, the plotting position is not set to 1 but rather is set equal to the Blom plotting position: $(N - 0.375)/(N + 0.25)$.

This method is useful, for example, when creating Quantile-Quantile plots.

Hazard Plotting Method of Nelson (`prob.method="nelson"`)
(Right-Censored Data Only.) For right-censored data, Equation (5) can be re-written as:

$$\hat{S}[x_{(i)}] = \prod_{j \in \Omega, j \le i} \frac{N-j}{N-j+1}, \quad i \in \Omega \qquad (8)$$

Nelson (1972) proposed the following formula for plotting positions for the uncensored observations in the context of estimating the hazard function (see Michael and Schucany,1986, p.469):

$$\hat{p}_i = \hat{F}[x_{(i)}] = 1 - \prod_{j \in \Omega, j \le i} exp(\frac{-1}{N-j+1}) \qquad (9)$$

See Lee and Wang (2003) for more information about the hazard function.

As for the Kaplan and Meier (1958) method, the plotting position for a censored observation is set equal to the plotting position associated with the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than or equal to a particular censoring level, the plotting position is set to 0 for that censoring level.

Generalization of Product-Limit Method, Michael and Schucany
(`prob.method="michael-schucany"`)
For complete data sets, the disadvantage of using Equation (3) above to define plotting positions is that it implies the largest observed value is the maximum possible value of the distribution (the 100'th percentile). This may be satisfactory if the underlying distribution is known to be discrete, but it is usually not satisfactory if the underlying distribution is known to be continuous.

A more frequently used formula for plotting positions for complete data sets is given by:

$$\hat{F}[x_{(i)}] = \hat{p}_i = \frac{i-a}{N-2a+1} \qquad (10)$$

where $0 \le a \le 1$ (Cleveland, 1993, p. 18; D'Agostino, 1986a, pp. 8,25). The value of a is usually chosen so that the plotting positions are approximately unbiased (i.e., approximate the mean of their distribution) or else approximate the median value of their distribution (see the help file for `ecdfPlot`). Michael and Schucany (1986) extended this method for both left- and right-censored data sets.

Right-Censored Data (`censoring.side="right"`)
For right-censored data sets, the plotting positions for the uncensored observations are computed as:

$$\hat{p}_i = 1 - \frac{N-a+1}{N-2a+1} \prod_{j \in \Omega, j \le i} \frac{N-j-a+1}{N-j-a+2} \quad i \in \Omega \qquad (11)$$

Note that the plotting positions proposed by Herd (1960) and Johnson (1964) are a special case of Equation (11) with $a = 0$. Equation (11) reduces to Equation (10) in the case of complete data sets. Note that unlike the Kaplan-Meier method, plotting positions associated with tied uncensored observations are not the same (just as in the case for complete data using Equation (10)).

As for the Kaplan and Meier (1958) method, for right-censored data the plotting position for a censored observation is set equal to the plotting position associated with the largest uncensored observation less than or equal to that censoring level. If there are no uncensored observations less than

or equal to a particular censoring level, the plotting position is set to 0 for that censoring level.

Left-Censored Data (`censoring.side="left"`)
For left-censored data sets the plotting positions are computed as:

$$\hat{p}_i = \frac{N-a+1}{N-2a+1} \prod_{j \in \Omega, j \geq i} \frac{j-a}{j-a+1} \quad i \in \Omega \qquad (12)$$

Equation (12) reduces to Equation (10) in the case of complete data sets. Note that unlike the Kaplan-Meier method, plotting positions associated with tied uncensored observations are not the same (just as in the case for complete data using Equation (10)).

For left-censored data, the plotting position for a censored observation is set equal to the plotting position associated with the smallest uncensored observation greater than or equal to that censoring level. If there are no uncensored observations greater than or equal to a particular censoring level, the plotting position is set to 1 for that censoring level.

Generalization of Product-Limit Method, Hirsch and Stedinger
(`prob.method="hirsch-stedinger"`)
Hirsch and Stedinger (1987) use a slightly different approach than Kaplan and Meier (1958) and Michael and Schucany (1986) to derive a nonparametric estimate of the survival function (probability of exceedance) in the context of left-censored data. First they estimate the value of the survival function at each of the censoring levels. The value of the survival function for an uncensored observation between two adjacent censoring levels is then computed by linear interpolation (in the form of a plotting position). See also Helsel and Cohn (1988).

The discussion below presents an extension of the method of Hirsch and Stedinger (1987) to the case of right-censored data, and then presents the original derivation due to Hirsch and Stedinger (1987) for left-censored data.

Right-Censored Data (`censoring.side="right"`)
For right-censored data, the survival function is estimated as follows. For the j'th censoring level ($j = 0, 1, \ldots, K$), write the value of the survival function as:

$$\begin{aligned} S(T_j) &= Pr[X > T_j] \\ &= Pr[X > T_{j+1}] + Pr[T_j < X \leq T_{j+1}] \\ &= S(T_{j+1}) + Pr[T_j < X \leq T_{j+1} | X > T_j] Pr[X > T_j] \\ &= S(T_{j+1}) + Pr[T_j < X \leq T_{j+1} | X > T_j] S(T_j) \qquad (13) \end{aligned}$$

where

$$T_0 = -\infty, \qquad (14)$$

$$T_{K+1} = \infty \qquad (15)$$

Now set

$$\begin{aligned} A_j &= \text{\# uncensored observations in } (T_j, T_{j+1}] \qquad (16) \\ B_j &= \text{\# observations in } (T_{j+1}, \infty) \qquad (17) \end{aligned}$$

for $j = 0, 1, \ldots, K$. Then the method of moments estimator of the conditional probability in Equation (13) is given by:

$$\widehat{Pr}[T_j < X \leq T_{j+1} | X > T_j] = \frac{A_j}{A_j + B_j} \qquad (18)$$

Hence, by equations (13) and (18) we have

$$\hat{S}(T_j) = \hat{S}(T_{j+1}) + (\frac{A_j}{A_j + B_j})\hat{S}(T_j) \qquad (19)$$

which can be rewritten as:

$$\hat{S}(T_{j+1}) = \hat{S}(T_j)[1 - (\frac{A_j}{A_j + B_j})] \qquad (20)$$

Equation (20) can be solved interatively for $j = 1, 2, \ldots, K$. Note that

$$\hat{S}(T_0) = \hat{S}(-\infty) = S(-\infty) = 1 \qquad (21)$$

$$\hat{S}(T_{K+1}) = \hat{S}(\infty) = S(\infty) = 0 \qquad (22)$$

Once the values of the survival function at the censoring levels are computed, the plotting positions for the A_j uncensored observations in the interval $(T_J, T_{j+1}]$ $(j = 0, 1, \ldots, K)$ are computed as

$$\hat{p}_i = [1 - \hat{S}(T_j)] + [\hat{S}(T_j) - \hat{S}(T_{j+1})]\frac{r - a}{A_j - 2a + 1} \qquad (23)$$

where a denotes the plotting position constant, $0 \leq a \leq 1$, and r denotes the rank of the i'th observation among the A_j uncensored observations in the interval $(T_J, T_{j+1}]$. (Tied observations are given distinct ranks.)

For the c_j observations censored at censoring level T_j $(j = 1, 2, \ldots, K)$, the plotting positions are computed as:

$$\hat{p}_i = 1 - [\hat{S}(T_j)\frac{r - a}{c_j - 2a + 1}] \qquad (24)$$

where r denotes the rank of the i'th observation among the c_j observations censored at censoring level T_j. Note that all the observations censored at the same censoring level are given distinct ranks, even though there is no way to distinguish between them.

Left-Censored Data (`censoring.side="left"`)
For left-censored data, Hirsch and Stedinger (1987) modify the definition of the survival function as follows:

$$S^*(t) = Pr[X \geq t] \qquad (25)$$

For continuous distributions, the functions in Equations (4) and (25) are identical.

Hirsch and Stedinger (1987) show that for the j'th censoring level $(j = 0, 1, \ldots, K)$, the value of the survival function can be written as:

$$\begin{aligned} S(T_j) &= Pr[X \geq T_j] \\ &= Pr[X \geq T_{j+1}] + Pr[T_j \leq X < T_{j+1}] \\ &= S^*(T_{j+1}) + Pr[T_j \leq X < T_{j+1} | X < T_{j+1}] Pr[X < T_{j+1}] \\ &= S^*(T_{j+1}) + Pr[T_j \leq X < T_{j+1} | X < T_j][1 - S^*(T_j)] \end{aligned} \qquad (26)$$

where T_0 and T_{K+1} are defined in Equations (14) and (15).

Now set

$$A_j = \text{\# uncensored observations in } [T_j, T_{j+1}) \quad (27)$$
$$B_j = \text{\# observations in } (-\infty, T_j) \quad (28)$$

for $j = 0, 1, \ldots, K$. Then the method of moments estimator of the conditional probability in Equation (26) is given by:

$$Pr[T_j \le X < T_{j+1} | X < T_{j+1}] = \frac{A_j}{A_j + B_j} \quad (29)$$

Hence, by Equations (26) and (29) we have

$$\hat{S}(T_j) = \hat{S}(T_{j+1}) + (\frac{A_j}{A_j + B_j})\hat{S}(T_j) \quad (30)$$

which can be solved interatively for $j = 1, 2, \ldots, K$. Note that

$$\widehat{S^*}(T_{K+1}) = \widehat{S^*}(\infty) = S^*(\infty) = 0 \quad (31)$$

$$\widehat{S^*}(T_0) = \widehat{S^*}(-\infty) = S^*(-\infty) = 1 \quad (32)$$

Once the values of the survival function at the censoring levels are computed, the plotting positions for the A_j uncensored observations in the interval $[T_J, T_{j+1})$ $(j = 0, 1, \ldots, K)$ are computed as

$$\hat{p}_i = [1 - \widehat{S^*}(T_j)] + [\widehat{S^*}(T_j) - \widehat{S^*}(T_{j+1})]\frac{r - a}{A_j - 2a + 1} \quad (33)$$

where a denotes the plotting position constant, $0 \le a \le 0.5$, and r denotes the rank of the i'th observation among the A_j uncensored observations in the interval $[T_J, T_{j+1})$. (Tied observations are given distinct ranks.)

For the c_j observations censored at censoring level T_j $(j = 1, 2, \ldots, K)$, the plotting positions are computed as:

$$\hat{p}_i = [1 - \widehat{S^*}(T_j)]\frac{r - a}{c_j - 2a + 1} \quad (34)$$

where r denotes the rank of the i'th observation among the c_j observations censored at censoring level T_j. Note that all the observations censored at the same censoring level are given distinct ranks, even though there is no way to distinguish between them.

Value

`ppointsCensored` returns a list with the following components:

`Order.Statistics`
: numeric vector of the "ordered" observations.

`Cumulative.Probabilities`
: numeric vector of the associated plotting positions.

`Censored`
: logical vector indicating which of the ordered observations are censored.

`Censoring.Side`
: character string indicating whether the data are left- or right-censored. This is same value as the argument `censoring.side`.

`Prob.Method`
: character string indicating what method was used to compute the plotting positions. This is the same value as the argument `prob.method`.

Optional Component (only present when `prob.method="michael-schucany"` or `prob.method="hirsch-stedinger"`):

`Plot.Pos.Con` numeric scalar containing the value of the plotting position constant that was used. This is the same as the argument `plot.pos.con`.

Note

For censored data sets, plotting positions may be used to construct empirical cumulative distribution plots (see `ecdfPlotCensored`), construct quantile-quantile plots (see `qqPlotCensored`), or to estimate distribution parameters (see `FcnsByCatCensoredData`).

The function `survfit` in the built-in R library **survival** computes the survival function for right-censored, left-censored, or interval-censored data. Calling `survfit` with `type="kaplan-meier"` will produce similar results to calling `ppointsCensored` with `prob.method="kaplan-meier"`. Also, calling `survfit` with `type="fh2"` will produce similar results to calling `ppointsCensored` with `prob.method="nelson"`.

Helsel and Cohn (1988, p.2001) found very little effect of changing the value of the plotting position constant when using the method of Hirsch and Stedinger (1987) to compute plotting positions for multiply left-censored data. In general, there will be very little difference between plotting positions computed by the different methods except in the case of very small samples and a large amount of censoring.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11-16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7-62.

Gillespie, B.W., Q. Chen, H. Reichert, A. Franzblau, E. Hedgeman, J. Lepkowski, P. Adriaens, A. Demond, W. Luksemburg, and D.H. Garabrant. (2010). Estimating Population Distributions When Some Data Are Below a Limit of Detection by Using a Reverse Kaplan-Meier Estimator. *Epidemiology* **21**(4), S64–S70.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997-2004.

Hirsch, R.M., and J.R. Stedinger. (1987). Plotting Positions for Historical Floods and Their Precision. *Water Resources Research* **23**(4), 715-727.

Kaplan, E.L., and P. Meier. (1958). Nonparametric Estimation From Incomplete Observations. *Journal of the American Statistical Association* **53**, 457-481.

Lee, E.T., and J. Wang. (2003). *Statistical Methods for Survival Data Analysis, Third Edition*. John Wiley and Sons, New York.

Michael, J.R., and W.R. Schucany. (1986). Analysis of Data from Censored Samples. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, 560pp, Chapter 11, 461-496.

Nelson, W. (1972). Theory and Applications of Hazard Plotting for Censored Failure Data. *Technometrics* **14**, 945-966.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. Chapter 15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

ppoints, ecdfPlot, qqPlot, ecdfPlotCensored, qqPlotCensored, survfit.

Examples

```
# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then compute plotting positions for
# this data set.  Compare the plotting positions to the plotting positions
# for the uncensored data set.  Note that the plotting positions for the
# censored data set start at the first ordered uncensored observation and
# that for values of x > 18 the plotting positions for the two data sets are
# exactly the same. This is because there is only one censoring level and
# no uncensored observations fall below the censored observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(333)
x <- rnorm(20, mean=20, sd=5)
censored <- x < 18
censored
# [1] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
#[13] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE

sum(censored)
#[1] 7

new.x <- x
new.x[censored] <- 18
round(sort(new.x),1)
# [1] 18.0 18.0 18.0 18.0 18.0 18.0 18.0 18.1 18.7 19.6 20.2 20.3 20.6 21.4
#[15] 21.8 21.8 23.2 26.2 26.8 29.7

p.list <- ppointsCensored(new.x, censored)
p.list
#$Order.Statistics
# [1] 18.00000 18.00000 18.00000 18.00000 18.00000 18.00000 18.00000 18.09771
# [9] 18.65418 19.58594 20.21931 20.26851 20.55296 21.38869 21.76359 21.82364
#[17] 23.16804 26.16527 26.84336 29.67340
```

```
#
#$Cumulative.Probabilities
# [1] 0.3765432 0.3765432 0.3765432 0.3765432 0.3765432 0.3765432 0.3765432
# [8] 0.3765432 0.4259259 0.4753086 0.5246914 0.5740741 0.6234568 0.6728395
#[15] 0.7222222 0.7716049 0.8209877 0.8703704 0.9197531 0.9691358
#
#$Censored
# [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
#[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#
#$Censoring.Side
#[1] "left"
#
#$Prob.Method
#[1] "michael-schucany"
#
#$Plot.Pos.Con
#[1] 0.375

#----------

# Round off plotting positions to two decimal places
# and compare to plotting positions that ignore censoring
#-------------------------------------------------------

round(p.list$Cum, 2)
# [1] 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.43 0.48 0.52 0.57 0.62 0.67
#[15] 0.72 0.77 0.82 0.87 0.92 0.97

round(ppoints(x, a=0.375), 2)
# [1] 0.03 0.08 0.13 0.18 0.23 0.28 0.33 0.38 0.43 0.48 0.52 0.57 0.62 0.67
#[15] 0.72 0.77 0.82 0.87 0.92 0.97

#----------

# Clean up
#---------
rm(x, censored, new.x, p.list)

#--------------------------------------------------------------------------

# Reproduce the example in Appendix B of Helsel and Cohn (1988).  The data
# are stored in Helsel.Cohn.88.appb.df.  This data frame contains 18
# observations, of which 9 are censored below one of 2 distinct censoring
# levels.

Helsel.Cohn.88.app.b.df
#   Conc.orig Conc Censored
#1         <1    1     TRUE
#2         <1    1     TRUE
#...
#17        33   33    FALSE
#18        50   50    FALSE
```

```
p.list <- with(Helsel.Cohn.88.app.b.df,
  ppointsCensored(Conc, Censored, prob.method="hirsch-stedinger", plot.pos.con=0))
lapply(p.list[1:2], round, 3)
#$Order.Statistics
# [1]  1  1  1  1  1  1  3  7  9 10 10 10 12 15 20 27 33 50
#
#$Cumulative.Probabilities
# [1] 0.063 0.127 0.190 0.254 0.317 0.381 0.500 0.556 0.611 0.167 0.333 0.500
#[13] 0.714 0.762 0.810 0.857 0.905 0.952

# Clean up
#---------
rm(p.list)

#-------------------------------------------------------------------------

# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# computing plotting positions based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring
# wells.  The data for this example are stored in
# EPA.09.Ex.15.1.manganese.df.

EPA.09.Ex.15.1.manganese.df
#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#4       4 Well.1               21.6          21.6    FALSE
#5       5 Well.1                 <2           2.0     TRUE
#...
#21      1 Well.5               17.9          17.9    FALSE
#22      2 Well.5               22.7          22.7    FALSE
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

p.list.EPA <- with(EPA.09.Ex.15.1.manganese.df,
  ppointsCensored(Manganese.ppb, Censored,
    prob.method = "kaplan-meier"))
data.frame(Mn = p.list.EPA$Order.Statistics, Censored = p.list.EPA$Censored,
  CDF = p.list.EPA$Cumulative.Probabilities)
#      Mn Censored  CDF
#1    2.0     TRUE 0.21
#2    2.0     TRUE 0.21
#3    2.0     TRUE 0.21
#4    3.3    FALSE 0.28
#5    5.0     TRUE 0.28
#6    5.0     TRUE 0.28
#7    5.0     TRUE 0.28
#8    5.3    FALSE 0.32
#9    6.3    FALSE 0.36
#10   7.7    FALSE 0.40
```

```
#11   8.4    FALSE 0.44
#12   9.5    FALSE 0.48
#13  10.0    FALSE 0.52
#14  11.9    FALSE 0.56
#15  12.1    FALSE 0.60
#16  12.6    FALSE 0.64
#17  16.9    FALSE 0.68
#18  17.9    FALSE 0.72
#19  21.6    FALSE 0.76
#20  22.7    FALSE 0.80
#21  34.5    FALSE 0.84
#22  45.9    FALSE 0.88
#23  53.6    FALSE 0.92
#24  77.2    FALSE 0.96
#25 106.3    FALSE 1.00

#----------

# Clean up
#---------
rm(p.list.EPA)
```

`predict` *Model Predictions*

Description

The **EnvStats** function `predict` is a generic function for predictions from the results of various model fitting functions. The function invokes particular `methods` which depend on the `class` of the first argument. The **EnvStats** function `predict.default` simply calls the R generic function `predict`.

The **EnvStats** functions `predict` and `predict.default` have been created in order to comply with CRAN policies, because **EnvStats** contains a modified version of the R function `predict.lm`.

Usage

```
predict(object, ...)

## Default S3 method:
predict(object, ...)
```

Arguments

`object`	a model object for which prediction is desired.
`...`	Further arguments passed to or from other methods. See the R help file for `predict` for more information.

Details

See the R help file for predict.

Value

See the R help file for predict.

Author(s)

R Development Core Team for code for R version of predict.

Steven P. Millard for **EnvStats** version of predict.default; <EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., and Hastie, T.J., eds. (1992). *Statistical Models in S*. Chapman and Hall/CRC, Boca Raton, FL.

See Also

R help file for predict, predict.lm.

Examples

```
# Using the data from the built-in data frame Air.df,
# fit the cube-root of ozone as a function of temperature,
# then compute predicted values for ozone at 70 and 90 degrees F,
# along with the standard errors of these predicted values.

# First look at the data
#-----------------------
with(Air.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
    ylab = "Cube-Root Ozone (ppb)"))


# Now create the lm object
#-------------------------
ozone.fit <- lm(ozone ~ temperature, data = Air.df)


# Now get predicted values and CIs at 70 and 90 degrees.
# Note the presence of the last component called n.coefs.
#-------------------------------------------------------
predict.list <- predict(ozone.fit,
  newdata = data.frame(temperature = c(70, 90)), se.fit = TRUE)

predict.list
#$fit
#       1        2
#2.697810 4.101808
#
```

```
#$se.fit
#         1          2
#0.07134554 0.08921071
#
#$df
#[1] 114
#
#$residual.scale
#[1] 0.5903046
#
#$n.coefs
#[1] 2


#----------

#Continuing with the above example, create a scatterplot of
# cube-root ozone vs. temperature, and add the fitted line
# along with simultaneous 95% confidence bands.

with(Air.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
    ylab = "Cube-Root Ozone (ppb)"))

abline(ozone.fit, lwd = 3, col = "blue")

new.temp <- with(Air.df,
  seq(min(temperature), max(temperature), length = 100))

predict.list <- predict(ozone.fit,
  newdata = data.frame(temperature = new.temp),
  se.fit = TRUE)

ci.ozone <- pointwise(predict.list, coverage = 0.95,
  simultaneous = TRUE)

lines(new.temp, ci.ozone$lower, lty = 2, lwd = 3, col = "magenta")

lines(new.temp, ci.ozone$upper, lty = 2, lwd = 3, col = "magenta")

title(main=paste("Scatterplot of Cube-Root Ozone vs. Temperature",
  "with Fitted Line and Simultaneous 95% Confidence Bands",
  sep="\n"))

#----------

# Clean up
#---------
rm(ozone.fit, predict.list, new.temp, ci.ozone)
graphics.off()
```

predict.lm *Predict Method for Linear Model Fits*

Description

The function predict.lm in **EnvStats** is a modified version of the built-in R function predict.lm. The only modification is that for the **EnvStats** function predict.lm, if se.fit=TRUE, the list returned includes a component called n.coefs. The component n.coefs is used by the function pointwise to create simultaneous confidence or prediction limits.

Usage

```
## S3 method for class 'lm'
predict(object, ...)
```

Arguments

object	Object of class inheriting from "lm".
...	Further arguments passed to the R function predict.lm. See the R help file for the R function predict.lm.

Details

See the R help file for predict.lm.

The function predict.lm in **EnvStats** is a modified version of the built-in R function predict.lm. The only modification is that for the **EnvStats** function predict.lm, if se.fit=TRUE, the list returned includes a component called n.coefs. The component n.coefs is used by the function pointwise to create simultaneous confidence or prediction limits.

Value

See the R help file for predict.lm.

The only modification is that for the **EnvStats** function predict.lm, if se.fit=TRUE, the list returned includes a component called n.coefs, i.e., the function returns a list with the following components:

fit	vector or matrix as above
se.fit	standard error of predicted means
residual.scale	residual standard deviations
df	degrees of freedom for residual
n.coefs	numeric scalar denoting the number of predictor variables used in the model

Author(s)

R Development Core Team (for code for R version of predict.lm).

Steven P. Millard (for modification to add compenent n.coefs; <EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., and Hastie, T.J., eds. (1992). *Statistical Models in S*. Chapman and Hall/CRC, Boca Raton, FL.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, Chapter 3.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.546-553.

Miller, R.G. (1981a). *Simultaneous Statistical Inference*. Springer-Verlag, New York, pp.111, 124.

See Also

Help file for R function predict, Help file for R function predict.lm, lm, calibrate, calibrate, inversePredictCalibrate, detectionLimitCalibrate.

Examples

```
# Using the data from the built-in data frame Air.df,
# fit the cube-root of ozone as a function of temperature,
# then compute predicted values for ozone at 70 and 90 degrees F,
# along with the standard errors of these predicted values.

# First look at the data
#-----------------------
with(Air.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
    ylab = "Cube-Root Ozone (ppb)"))


# Now create the lm object
#-------------------------
ozone.fit <- lm(ozone ~ temperature, data = Air.df)


# Now get predicted values and CIs at 70 and 90 degrees.
# Note the presence of the last component called n.coefs.
#-------------------------------------------------------
predict.list <- predict(ozone.fit,
  newdata = data.frame(temperature = c(70, 90)), se.fit = TRUE)

predict.list
#$fit
#       1        2
#2.697810 4.101808
#
#$se.fit
#         1          2
#0.07134554 0.08921071
#
#$df
#[1] 114
```

```
#
#$residual.scale
#[1] 0.5903046
#
#$n.coefs
#[1] 2


#----------

#Continuing with the above example, create a scatterplot of
# cube-root ozone vs. temperature, and add the fitted line
# along with simultaneous 95% confidence bands.

with(Air.df,
  plot(temperature, ozone, xlab = "Temperature (degrees F)",
    ylab = "Cube-Root Ozone (ppb)"))

abline(ozone.fit, lwd = 3, col = "blue")

new.temp <- with(Air.df,
  seq(min(temperature), max(temperature), length = 100))

predict.list <- predict(ozone.fit,
  newdata = data.frame(temperature = new.temp),
  se.fit = TRUE)

ci.ozone <- pointwise(predict.list, coverage = 0.95,
  simultaneous = TRUE)

lines(new.temp, ci.ozone$lower, lty = 2, lwd = 3, col = "magenta")

lines(new.temp, ci.ozone$upper, lty = 2, lwd = 3, col = "magenta")

title(main=paste("Scatterplot of Cube-Root Ozone vs. Temperature",
  "with Fitted Line and Simultaneous 95% Confidence Bands",
  sep="\n"))

#----------

# Clean up
#---------
rm(ozone.fit, predict.list, new.temp, ci.ozone)
graphics.off()
```

---

predIntGamma *Prediction Interval for Gamma Distribution*

Description

Construct a prediction interval for the next k observations or next set of k transformed means for a gamma distribution.

Usage

```
predIntGamma(x, n.transmean = 1, k = 1, method = "Bonferroni",
  pi.type = "two-sided", conf.level = 0.95, est.method = "mle",
  normal.approx.transform = "kulkarni.powar")

predIntGammaAlt(x, n.transmean = 1, k = 1, method = "Bonferroni",
  pi.type = "two-sided", conf.level = 0.95, est.method = "mle",
  normal.approx.transform = "kulkarni.powar")
```

Arguments

`x`	numeric vector of non-negative observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`n.transmean`	positive integer specifying the sample size associated with the `k` future transformed means (see the DETAILS section for an explanation of what the transformation is). The default value is `n.transmean=1` (i.e., predicting future observations). Note that all future transformed means must be based on the same sample size.
`k`	positive integer specifying the number of future observations or means the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`.
`method`	character string specifying the method to use if the number of future observations or averages (`k`) is greater than 1. The possible values are `"Bonferroni"` (approximate method based on Bonferonni inequality; the default), and `"exact"` (exact method due to Dunnett, 1955). See the DETAILS section for more information. This argument is ignored if `k=1`.
`pi.type`	character string indicating what kind of prediction interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.
`conf.level`	a scalar between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is `conf.level=0.95`.
`est.method`	character string specifying the method of estimation for the shape and scale distribution parameters. The possible values are `"mle"` (maximum likelihood; the default), `"bcmle"` (bias-corrected mle), `"mme"` (method of moments), and `"mmue"` (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for `egamma` for more information.
`normal.approx.transform`	character string indicating which power transformation to use. Possible values are `"kulkarni.powar"` (the default), `"cube.root"`, and `"fourth.root"`. See the DETAILS section for more informaiton.

Details

If x contains any missing (NA), undefined (NaN) or infinite (Inf, -Inf) values, they will be removed prior to performing the estimation.

The function predIntGamma returns a prediction interval as well as estimates of the shape and scale parameters. The function predIntGammaAlt returns a prediction interval as well as estimates of the mean and coefficient of variation.

Following Krishnamoorthy et al. (2008), the prediction interval is computed by:

1. using a power transformation on the original data to induce approximate normality,
2. calling predIntNorm with the transformed data to compute the prediction interval, and then
3. back-transforming the interval to create a prediction interval on the original scale.

The argument normal.approx.transform determines which transformation is used. The value normal.approx.transform="cube.root" uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value normal.approx.transform="fourth.root" uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value normal.approx.transform="kulkarni.powar" uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010). The "optimum" power p is determined by:

$$\begin{aligned} p &= -0.0705 - 0.178\, shape + 0.475\, \sqrt{shape} \quad \text{if } shape \leq 1.5 \\ p &= 0.246 \quad \text{if } shape > 1.5 \end{aligned}$$

where $shape$ denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power p, for the functions predIntGamma and predIntGammaAlt the power p is based on whatever estimate of shape is used (e.g., est.method="mle", est.method="bcmle", etc.).

When the argument n.transmean is larger than 1 (i.e., you are constructing a prediction interval for future means, not just single observations), in order to properly compare a future mean with the prediction limits, you must follow these steps:

1. Take the observations that will be used to compute the mean and transform them by raising them to the power given by the value in the component interval$normal.transform.power (see the section VALUE below).
2. Compute the mean of the transformed observations.
3. Take the mean computed in step 2 above and raise it to the inverse of the power originally used to transform the observations.

Value

A list of class "estimate" containing the estimated parameters, the prediction interval, and other information. See estimate.object for details.

In addition to the usual components contained in an object of class "estimate", the returned value also includes two additional components within the "interval" component:

n.transmean the value of n.transmean supplied in the call to predIntGamma or predIntGammaAlt.

`normal.transform.power`
: the value of the power used to transform the original data to approximate normality.

Warning

It is possible for the lower prediction limit based on the transformed data to be less than 0. In this case, the lower prediction limit on the original scale is set to 0 and a warning is issued stating that the normal approximation is not accurate in this case.

Note

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter a tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribuiton instead.

Prediction intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973), and are often discussed in the context of linear regression (Draper and Smith, 1998; Zar, 2010). Prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities. References that discuss prediction intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Evans, M., N. Hastings, and B. Peacock. (1993). *Statistical Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 18.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hawkins, D. M., and R.A.J. Wixley. (1986). A Note on the Transformation of Chi-Squared Variables to Normality. *The American Statistician*, **40**, 296–298.

Johnson, N.L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, Chapter 17.

Krishnamoorthy K., T. Mathew, and S. Mukherjee. (2008). Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability. *Technometrics*, **50**(1), 69–78.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Kulkarni, H.V., and S.K. Powar. (2010). A New Method for Interval Estimation of the Mean of the Gamma Distribution. *Lifetime Data Analysis*, **16**, 431–447.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Singh, A., A.K. Singh, and R.J. Iaci. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Wilson, E.B., and M.M. Hilferty. (1931). The Distribution of Chi-Squares. *Proceedings of the National Academy of Sciences*, **17**, 684–688.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, New Jersey.

See Also

`GammaDist`, `GammaAlt`, `estimate.object`, `egamma`, `predIntNorm`, `tolIntGamma`.

Examples

```
# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then create a prediciton interval for the
# next observation.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
predIntGamma(dat)
```

```
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 2.203862
#                                 scale = 2.174928
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      exact using
#                                 Kulkarni & Powar (2010)
#                                 transformation to Normality
#                                 based on mle of 'shape'
#
#Normal Transform Power:          0.246
#
#Prediction Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Number of Future Observations:   1
#
#Prediction Interval:             LPL =  0.5371569
#                                 UPL = 15.5313783

#----------------------------------------------------------------------

# Using the same data as in the previous example, create an upper
# one-sided prediction interval for the next three averages of
# order 2 (i.e., each mean is based on 2 future observations), and
# use the bias-corrected estimate of shape.

pred.list <- predIntGamma(dat, n.transmean = 2, k = 3,
  pi.type = "upper", est.method = "bcmle")

pred.list

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 1.906616
#                                 scale = 2.514005
#
#Estimation Method:               bcmle
#
#Data:                            dat
```

```
#
#Sample Size:                     20
#
#Prediction Interval Method:      Bonferroni using
#                                 Kulkarni & Powar (2010)
#                                 transformation to Normality
#                                 based on bcmle of 'shape'
#
#Normal Transform Power:          0.246
#
#Prediction Interval Type:        upper
#
#Confidence Level:                95%
#
#Number of Future
#Transformed Means:               3
#
#Sample Size for
#Transformed Means:               2
#
#Prediction Interval:             LPL =  0.00000
#                                 UPL = 12.17404

#----------------------------------------------------------------------

# Continuing with the above example, assume the distribution shifts
# in the future to a gamma distribution with shape = 5 and scale = 2.
# Create 6 future observations from this distribution, and create 3
# means by pairing the observations sequentially.  Note we must first
# transform these observations using the power 0.246, then compute the
# means based on the transformed data, and then transform the means
# back to the original scale and compare them to the upper prediction
# limit of 12.17

set.seed(427)
new.dat <- rgamma(6, shape = 5, scale = 2)

p <- pred.list$interval$normal.transform.power
p
#[1] 0.246

new.dat.trans <- new.dat^p
means.trans <- c(mean(new.dat.trans[1:2]), mean(new.dat.trans[3:4]),
  mean(new.dat.trans[5:6]))
means <- means.trans^(1/p)
means
#[1] 11.74214 17.05299 11.65272

any(means > pred.list$interval$limits["UPL"])
#[1] TRUE

#----------
```

```
# Clean up
rm(dat, pred.list, new.dat, p, new.dat.trans, means.trans, means)

#--------------------------------------------------------------------

# Reproduce part of the example on page 73 of
# Krishnamoorthy et al. (2008), which uses alkalinity concentrations
# reported in Gibbons (1994) and Gibbons et al. (2009) to construct a
# one-sided upper 90% prediction limit.

predIntGamma(Gibbons.et.al.09.Alkilinity.vec, pi.type = "upper",
  conf.level = 0.9, normal.approx.transform = "cube.root")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 9.375013
#                                 scale = 6.202461
#
#Estimation Method:               mle
#
#Data:                            Gibbons.et.al.09.Alkilinity.vec
#
#Sample Size:                     27
#
#Prediction Interval Method:      exact using
#                                 Wilson & Hilferty (1931) cube-root
#                                 transformation to Normality
#
#Normal Transform Power:          0.3333333
#
#Prediction Interval Type:        upper
#
#Confidence Level:                90%
#
#Number of Future Observations:   1
#
#Prediction Interval:             LPL =  0.0000
#                                 UPL = 85.3495
```

`predIntGammaSimultaneous` *Simultaneous Prediction Interval for a Gamma Distribution*

Description

Estimate the shape and scale parameters for a gamma distribution, or estimate the mean and coefficient of variation for a gamma distribution (alternative parameterization), and construct a simul-

taneous prediction interval for the next r sampling occasions, based on one of three possible rules: k-of-m, California, or Modified California.

Usage

```
predIntGammaSimultaneous(x, n.transmean = 1, k = 1, m = 2, r = 1,
  rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
 K.tol = 1e-07, est.method = "mle", normal.approx.transform = "kulkarni.powar")

predIntGammaAltSimultaneous(x, n.transmean = 1, k = 1, m = 2, r = 1,
  rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
 K.tol = 1e-07, est.method = "mle", normal.approx.transform = "kulkarni.powar")
```

Arguments

x	numeric vector of non-negative observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n.transmean	positive integer specifying the sample size associated with future transformed means (see the DETAILS section for an explanation of what the transformation is). The default value is n.transmean=1 (i.e., individual observations). Note that all future transformed means must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or transformed means) out of m observations (or transformed means) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	positive integer specifying the maximum number of future observations (or transformed means) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	positive integer specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.
delta.over.sigma	numeric scalar indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population (on the transformed scale) that was sampled to construct the prediction interval, and the mean of the population (on the transformed scale) that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation (on the transformed scale) for both populations. See the DETAILS section below for more information. The default value is delta.over.sigma=0.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".

`conf.level` a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

`K.tol` numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is `K.tol=.Machine$double.eps^(1/2)`. For many applications, the value of K needs to be known only to the second decimal place, in which case setting `K.tol=1e-4` will speed up computation a bit.

`est.method` character string specifying the method of estimation for the shape and scale distribution parameters. The possible values are `"mle"` (maximum likelihood; the default), `"bcmle"` (bias-corrected mle), `"mme"` (method of moments), and `"mmue"` (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for `egamma` for more information.

`normal.approx.transform` character string indicating which power transformation to use. Possible values are `"kulkarni.powar"` (the default), `"cube.root"`, and `"fourth.root"`. See the DETAILS section for more informaiton.

Details

The function `predIntGammaSimultaneous` returns a simultaneous prediction interval as well as estimates of the shape and scale parameters. The function `predIntGammaAltSimultaneous` returns a simultaneous prediction interval as well as estimates of the mean and coefficient of variation.

Following Krishnamoorthy et al. (2008), the simultaneous prediction interval is computed by:

1. using a power transformation on the original data to induce approximate normality,
2. calling `predIntNormSimultaneous` with the transformed data to compute the simultaneous prediction interval, and then
3. back-transforming the interval to create a simultaneous prediction interval on the original scale.

The argument `normal.approx.transform` determines which transformation is used. The value `normal.approx.transform="cube.root"` uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value `normal.approx.transform="fourth.root"` uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value `normal.approx.transform="kulkarni.powar"` uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010). The "optimum" power p is determined by:

$$\begin{aligned} p &= -0.0705 - 0.178\, shape + 0.475\, \sqrt{shape} \quad \text{if } shape \le 1.5 \\ p &= 0.246 \quad \text{if } shape > 1.5 \end{aligned}$$

where $shape$ denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power p, for the functions `predIntGammaSimultaneous` and `predIntGammaAltSimultaneous` the power p is based on whatever estimate of shape is used (e.g., `est.method="mle"`, `est.method="bcmle"`, etc.).

When the argument `n.transmean` is larger than 1 (i.e., you are constructing a prediction interval for future means, not just single observations), in order to properly compare a future mean with the prediction limits, you must follow these steps:

1. Take the observations that will be used to compute the mean and transform them by raising them to the power given by the value in the component `interval$normal.transform.power` (see the section VALUE below).
2. Compute the mean of the transformed observations.
3. Take the mean computed in step 2 above and raise it to the inverse of the power originally used to transform the observations.

Value

A list of class `"estimate"` containing the estimated parameters, the simultaneous prediction interval, and other information. See `estimate.object` for details.

In addition to the usual components contained in an object of class `"estimate"`, the returned value also includes two additional components within the `"interval"` component:

`n.transmean` the value of `n.transmean` supplied in the call to `predIntGammaSimultaneous` or `predIntGammaAltSimultaneous`.

`normal.transform.power` the value of the power used to transform the original data to approximate normality.

Warning

It is possible for the lower prediction limit based on the transformed data to be less than 0. In this case, the lower prediction limit on the original scale is set to 0 and a warning is issued stating that the normal approximation is not accurate in this case.

Note

The Gamma Distribution

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter a tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribuiton instead.

Motivation

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional α-level (e.g., 0.05 or 0.01) for each test leads to a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others.

Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual α-level at each well is equal to the FWFRP divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-m rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.

The k-of-m Rule

For the k-of-m rule, Davis and McNichols (1987) give tables with "optimal" choices of k (in terms of best power for a given overall confidence level) for selected values of m, r, and n. They found that the optimal ratios of k to m (i.e., k/m) are generally small, in the range of 15-50%.

The California Rule

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay's Code of California Regulations, 1991). The California code mandates a "California" rule with $m \geq 3$. The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a k-of-m rule with $k = 1$ and $m = 3$, a monitoring location will pass if the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with $m = 3$, either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPR is kept constant, then the California rule offers little increased power compared to the k-of-m rule, and can actually decrease the power of detecting contamination.

The Modified California Rule

The Modified California Rule was proposed as a compromise between a 1-of-m rule and the California rule. For a given FWFPR, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-m rule.

Different Notations Between Different References

For the k-of-m rule described in this help file, both Davis and McNichols (1987) and USEPA (2009, Chapter 19) use the variable p instead of k to represent the minimum number of future observations the interval should contain on each of the r sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of K for both k-of-m rules and California rules. Gibbons et al.'s notation reverses the meaning of k and r compared to the notation used in this help file. That is, in Gibbons et al.'s notation, k represents the number of future sampling occasions or monitoring wells, and r represents the minimum number of observations the interval should contain on each sampling occasion.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Barclay's California Code of Regulations. (1991). Title 22, Section 66264.97 [concerning hazardous waste facilities] and Title 23, Section 2550.7(e)(8) [concerning solid waste facilities]. Barclay's Law Publishers, San Francisco, CA.

Davis, C.B. (1998a). *Ground-Water Statistics & Regulations: Principles, Progress and Problems*. Second Edition. Environmetrics & Statistics Limited, Henderson, NV.

Davis, C.B. (1998b). Personal Communication, September 3, 1998.

Davis, C.B., and R.J. McNichols. (1987). One-sided Intervals for at Least p of m Observations from a Lognormal Population on Each of r Future Occasions. *Technometrics* **29**, 359–370.

Evans, M., N. Hastings, and B. Peacock. (1993). *Statistical Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 18.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Fertig, K.W., and N.R. Mann. (1977). One-Sided Prediction Intervals for at Least p Out of m Future Observations From a Lognormal Population. *Technometrics* **19**, 167–177.

Hahn, G.J. (1969). Factors for Calculating Two-Sided Prediction Intervals for Samples from a Lognormal Distribution. *Journal of the American Statistical Association* **64**(327), 878-898.

Hahn, G.J. (1970a). Additional Factors for Calculating Prediction Intervals for Samples from a Lognormal Distribution. *Journal of the American Statistical Association* **65**(332), 1668-1676.

Hahn, G.J. (1970b). Statistical Intervals for a Lognormal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Lognormal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178-188.

Hall, I.J., and R.R. Prairie. (1973). One-Sided Prediction Intervals to Contain at Least m Out of k Future Observations. *Technometrics* **15**, 897–914.

Hawkins, D. M., and R.A.J. Wixley. (1986). A Note on the Transformation of Chi-Squared Variables to Normality. *The American Statistician*, **40**, 296–298.

Johnson, N.L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, Chapter 17.

Krishnamoorthy K., T. Mathew, and S. Mukherjee. (2008). Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability. *Technometrics*, **50**(1), 69–78.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Kulkarni, H.V., and S.K. Powar. (2010). A New Method for Interval Estimation of the Mean of the Gamma Distribution. *Lifetime Data Analysis*, **16**, 431–447.

Millard, S.P. (1987). Environmental Monitoring, Statistics, and the Law: Room for Improvement (with Comment). *The American Statistician* **41**(4), 249–259.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Singh, A., A.K. Singh, and R.J. Iaci. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Wilson, E.B., and M.M. Hilferty. (1931). The Distribution of Chi-Squares. *Proceedings of the National Academy of Sciences*, **17**, 684–688.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

GammaDist, GammaAlt, predIntNorm, predIntNormSimultaneous, predIntNormSimultaneousTestPower, tolIntGamma, egamma, egammaAlt, estimate.object.

Examples

```
# Generate 8 observations from a gamma distribution with parameters
# mean=10 and cv=1, then use predIntGammaAltSimultaneous to estimate the
# mean and coefficient of variation of the true distribution and construct an
# upper 95% prediction interval to contain at least 1 out of the next
# 3 observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(479)
```

```
dat <- rgammaAlt(8, mean = 10, cv = 1)

predIntGammaAltSimultaneous(dat, k = 1, m = 3)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          mean = 13.875825
#                                 cv   =  1.049504
#
#Estimation Method:               MLE
#
#Data:                            dat
#
#Sample Size:                     8
#
#Prediction Interval Method:      exact using
#                                 Kulkarni & Powar (2010)
#                                 transformation to Normality
#                                 based on MLE of 'shape'
#
#Normal Transform Power:          0.2204908
#
#Prediction Interval Type:        upper
#
#Confidence Level:                95%
#
#Minimum Number of
#Future Observations
#Interval Should Contain:         1
#
#Total Number of
#Future Observations:             3
#
#Prediction Interval:             LPL =  0.00000
#                                 UPL = 15.87101

#----------

# Compare the 95% 1-of-3 upper prediction limit to the California and
# Modified California upper prediction limits.  Note that the upper
# prediction limit for the Modified California rule is between the limit
# for the 1-of-3 rule and the limit for the California rule.

predIntGammaAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
#     UPL
#15.87101

predIntGammaAltSimultaneous(dat, m = 3, rule = "CA")$interval$limits["UPL"]
#     UPL
#34.11499
```

```
predIntGammaAltSimultaneous(dat, rule = "Modified.CA")$interval$limits["UPL"]
#     UPL
#22.58809

#----------

# Show how the upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions r increases.
# Here, we'll use the 1-of-3 rule.

predIntGammaAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
#     UPL
#15.87101

predIntGammaAltSimultaneous(dat, k = 1, m = 3, r = 10)$interval$limits["UPL"]
#     UPL
#37.86825

#----------

# Compare the upper simultaneous prediction limit for the 1-of-3 rule
# based on individual observations versus based on transformed means of
# order 4.

predIntGammaAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
#     UPL
#15.87101

predIntGammaAltSimultaneous(dat, n.transmean = 4, k = 1,
  m = 3)$interval$limits["UPL"]
#     UPL
#14.76528

#==========

# Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an
# upper simultaneous prediction limit for the 1-of-3 rule for
# r = 2 future sampling occasions.  The data for this example are
# stored in EPA.09.Ex.19.1.sulfate.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 25 to use to construct the prediction limit.

# There are 50 compliance wells and we will monitor 10 different
# constituents at each well at each of the r=2 future sampling
# occasions.  To determine the confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the individual Type I Error level at each well to

# 1 - (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))
```

```
# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))

# where SWFPR = site-wide false positive rate.  For this example, we
# will set SWFPR = 0.1.  Thus, the confidence level is given by:

nc <- 10
nw <- 50
SWFPR <- 0.1
conf.level <- (1 - SWFPR)^(1 / (nc * nw))

conf.level
#[1] 0.9997893

#----------

# Look at the data:

names(EPA.09.Ex.19.1.sulfate.df)
#[1] "Well"                "Month"               "Day"
#[4] "Year"                "Date"                "Sulfate.mg.per.l"
#[7] "log.Sulfate.mg.per.l"

EPA.09.Ex.19.1.sulfate.df[,
  c("Well", "Date", "Sulfate.mg.per.l", "log.Sulfate.mg.per.l")]

#    Well       Date Sulfate.mg.per.l log.Sulfate.mg.per.l
#1  GW-01 1999-07-08             63.0             4.143135
#2  GW-01 1999-09-12             51.0             3.931826
#3  GW-01 1999-10-16             60.0             4.094345
#4  GW-01 1999-11-02             86.0             4.454347
#5  GW-04 1999-07-09            104.0             4.644391
#6  GW-04 1999-09-14            102.0             4.624973
#7  GW-04 1999-10-12             84.0             4.430817
#8  GW-04 1999-11-15             72.0             4.276666
#9  GW-08 1997-10-12             31.0             3.433987
#10 GW-08 1997-11-16             84.0             4.430817
#11 GW-08 1998-01-28             65.0             4.174387
#12 GW-08 1999-04-20             41.0             3.713572
#13 GW-08 2002-06-04             51.8             3.947390
#14 GW-08 2002-09-16             57.5             4.051785
#15 GW-08 2002-12-02             66.8             4.201703
#16 GW-08 2003-03-24             87.1             4.467057
#17 GW-09 1997-10-16             59.0             4.077537
#18 GW-09 1998-01-28             85.0             4.442651
#19 GW-09 1998-04-12             75.0             4.317488
#20 GW-09 1998-07-12             99.0             4.595120
#21 GW-09 2000-01-30             75.8             4.328098
#22 GW-09 2000-04-24             82.5             4.412798
#23 GW-09 2000-10-24             85.5             4.448516
#24 GW-09 2002-12-01            188.0             5.236442
#25 GW-09 2003-03-24            150.0             5.010635
```

```
# The EPA guidance document constructs the upper simultaneous
# prediction limit for the 1-of-3 plan assuming a lognormal
# distribution for the sulfate data.  Here we will compare
# the value of the limit based on assuming a lognormal distribution
# versus assuming a gamma distribution.

Sulfate <- EPA.09.Ex.19.1.sulfate.df$Sulfate.mg.per.l

pred.int.list.lnorm <-
  predIntLnormSimultaneous(x = Sulfate, k = 1, m = 3, r = 2,
    rule = "k.of.m", pi.type = "upper", conf.level = conf.level)

pred.int.list.gamma <-
  predIntGammaSimultaneous(x = Sulfate, k = 1, m = 3, r = 2,
    rule = "k.of.m", pi.type = "upper", conf.level = conf.level)


pred.int.list.lnorm$interval$limits["UPL"]
#     UPL
#159.5497

pred.int.list.gamma$interval$limits["UPL"]
#     UPL
#153.3232

#==========

# Cleanup
#--------
rm(dat, nc, nw, SWFPR, conf.level, Sulfate, pred.int.list.lnorm,
  pred.int.list.gamma)
```

---

predIntLnorm *Prediction Interval for a Lognormal Distribution*

Description

Estimate the mean and standard deviation on the log-scale for a lognormal distribution, or estimate the mean and coefficient of variation for a lognormal distribution (alternative parameterization), and construct a prediction interval for the next k observations or next set of k geometric means.

Usage

```
predIntLnorm(x, n.geomean = 1, k = 1, method = "Bonferroni",
  pi.type = "two-sided", conf.level = 0.95)

predIntLnormAlt(x, n.geomean = 1, k = 1, method = "Bonferroni",
  pi.type = "two-sided", conf.level = 0.95, est.arg.list = NULL)
```

Arguments

x	For predIntLnorm, x can be a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., elnorm or elnormCensored). You *cannot* supply objects resulting from a call to estimating functions that use the alternative parameterization such as elnormAlt or elnormAltCensored. For predIntLnormAlt, a numeric vector of positive observations. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n.geomean	positive integer specifying the sample size associated with the k future geometric means. The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
k	positive integer specifying the number of future observations or geometric means the prediction interval should contain with confidence level conf.level. The default value is k=1.
method	character string specifying the method to use if the number of future observations (k) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferonni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). See the DETAILS section of predIntNormK for more information. This argument is ignored if k=1.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="two-sided" (the default), pi.type="lower", and pi.type="upper".
conf.level	a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
est.arg.list	for predIntLnormAlt, a list containing arguments to pass to the function elnormAlt for estimating the mean and coefficient of variation. The default value is est.arg.list=NULL, which implies the default values will be used in the call to elnormAlt.

Details

The function predIntLnorm returns a prediction interval as well as estimates of the meanlog and sdlog parameters. The function predIntLnormAlt returns a prediction interval as well as estimates of the mean and coefficient of variation.

A prediction interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a prediction interval based on the normal (Gaussian) distribution by calling predIntNorm. These prediction limits are then exponentiated to produce a prediction interval on the original scale of the data.

Value

If x is a numeric vector, a list of class "estimate" containing the estimated parameters, the prediction interval, and other information. See the help file for estimate.object for details.

If x is the result of calling an estimation function, predIntLnorm returns a list whose class is the same as x. The list contains the same components as x, as well as a component called interval containing the prediction interval information. If x already has a component called interval, this component is replaced with the prediction interval information.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Dunnett, C.W. (1955). A Multiple Comparisons Procedure for Comparing Several Treatments with a Control. *Journal of the American Statistical Association* **50**, 1096-1121.

Dunnett, C.W. (1964). New Tables for Multiple Comparisons with a Control. *Biometrics* **20**, 482-491.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J. (1969). Factors for Calculating Two-Sided Prediction Intervals for Samples from a Lognormal Distribution. *Journal of the American Statistical Association* **64**(327), 878-898.

Hahn, G.J. (1970a). Additional Factors for Calculating Prediction Intervals for Samples from a Lognormal Distribution. *Journal of the American Statistical Association* **65**(332), 1668-1676.

Hahn, G.J. (1970b). Statistical Intervals for a Lognormal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Lognormal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178-188.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York.

Helsel, D.R., and R.M. Hirsch. (2002). *Statistical Methods in Water Resources*. Techniques of Water Resources Investigations, Book 4, chapter A3. U.S. Geological Survey. (available on-line at: https://pubs.usgs.gov/tm/04/a03/tm4a3.pdf).

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Miller, R.G. (1981a). *Simultaneous Statistical Inference*. McGraw-Hill, New York.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

elnorm, elnormAlt, predIntNorm, predIntNormK, predIntLnormSimultaneous, predIntLnormAltSimultaneous, tolIntLnorm, tolIntLnormAlt, Lognormal, estimate.object.

Examples

```
# Generate 20 observations from a lognormal distribution with parameters
# meanlog=0 and sdlog=1.  The exact two-sided 90% prediction interval for
# k=1 future observation is given by: [exp(-1.645), exp(1.645)] = [0.1930, 5.181].
# Use predIntLnorm to estimate the distribution parameters, and construct a
# two-sided 90% prediction interval.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(47)
dat <- rlnorm(20, meanlog = 0, sdlog = 1)
predIntLnorm(dat, conf = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = -0.1035722
#                                 sdlog   =  0.9106429
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        two-sided
#
#Confidence Level:                90%
#
#Number of Future Observations:   1
#
```

```
#Prediction Interval:             LPL = 0.1795898
#                                 UPL = 4.5264399

#----------

# Repeat the above example, but do it in two steps.
# First create a list called est.list containing information about the
# estimated parameters, then create the prediction interval.

est.list <- elnorm(dat)

est.list
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = -0.1035722
#                                 sdlog   =  0.9106429
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20

predIntLnorm(est.list, conf = 0.9)
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = -0.1035722
#                                 sdlog   =  0.9106429
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        two-sided
#
#Confidence Level:                90%
#
#Number of Future Observations:   1
#
#Prediction Interval:             LPL = 0.1795898
#                                 UPL = 4.5264399

#----------
```

```
# Using the same data from the first example, create a one-sided
# upper 99% prediction limit for the next 3 geometric means of order 2
# (i.e., each of the 3 future geometric means is based on a sample size
# of 2 future observations).

predIntLnorm(dat, n.geomean = 2, k = 3, conf.level = 0.99,
  pi.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = -0.1035722
#                                 sdlog   =  0.9106429
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      Bonferroni
#
#Prediction Interval Type:        upper
#
#Confidence Level:                99%
#
#Number of Future
#Geometric Means:                 3
#
#Sample Size for
#Geometric Means:                 2
#
#Prediction Interval:             LPL = 0.000000
#                                 UPL = 7.047571

#----------

# Compare the result above that is based on the Bonferroni method
# with the exact method

predIntLnorm(dat, n.geomean = 2, k = 3, conf.level = 0.99,
  pi.type = "upper", method = "exact")$interval$limits["UPL"]

#    UPL
#7.00316

#----------

# Clean up
rm(dat, est.list)
```

```
#----------------------------------------------------------------------

# Example 18-2 of USEPA (2009, p.18-15) shows how to construct a 99%
# upper prediction interval for the log-scale mean of 4 future observations
# (future mean of order 4) assuming a lognormal distribution based on
# chrysene concentrations (ppb) in groundwater at 2 background wells.
# Data were collected once per month over 4 months at the 2 background
# wells, and also at a compliance well.
# The question to be answered is whether there is evidence of
# contamination at the compliance well.

# Here we will follow the example, but look at the geometric mean
# instead of the log-scale mean.

#----------

# The data for this example are stored in EPA.09.Ex.18.2.chrysene.df.

EPA.09.Ex.18.2.chrysene.df

#   Month   Well  Well.type Chrysene.ppb
#1      1 Well.1 Background          6.9
#2      2 Well.1 Background         27.3
#3      3 Well.1 Background         10.8
#4      4 Well.1 Background          8.9
#5      1 Well.2 Background         15.1
#6      2 Well.2 Background          7.2
#7      3 Well.2 Background         48.4
#8      4 Well.2 Background          7.8
#9      1 Well.3 Compliance         68.0
#10     2 Well.3 Compliance         48.9
#11     3 Well.3 Compliance         30.1
#12     4 Well.3 Compliance         38.1

Chrysene.bkgd <- with(EPA.09.Ex.18.2.chrysene.df,
  Chrysene.ppb[Well.type == "Background"])
Chrysene.cmpl <- with(EPA.09.Ex.18.2.chrysene.df,
  Chrysene.ppb[Well.type == "Compliance"])


#----------

# A Shapiro-Wilks goodness-of-fit test for normality indicates
# we should reject the assumption of normality and assume a
# lognormal distribution for the background well data:

gofTest(Chrysene.bkgd)

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
```

```
#
#Hypothesized Distribution:       Normal
#
#Estimated Parameter(s):          mean = 16.55000
#                                 sd   = 14.54441
#
#Estimation Method:               mvue
#
#Data:                            Chrysene.bkgd
#
#Sample Size:                     8
#
#Test Statistic:                  W = 0.7289006
#
#Test Statistic Parameter:        n = 8
#
#P-value:                         0.004759859
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Normal Distribution.

gofTest(Chrysene.bkgd, dist = "lnorm")

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#
#Hypothesized Distribution:       Lognormal
#
#Estimated Parameter(s):          meanlog = 2.5533006
#                                 sdlog   = 0.7060038
#
#Estimation Method:               mvue
#
#Data:                            Chrysene.bkgd
#
#Sample Size:                     8
#
#Test Statistic:                  W = 0.8546352
#
#Test Statistic Parameter:        n = 8
#
#P-value:                         0.1061057
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Lognormal Distribution.

#----------

# Here is the one-sided 99% upper prediction limit for
# a geometric mean based on 4 future observations:
```

```
predIntLnorm(Chrysene.bkgd, n.geomean = 4, k = 1,
  conf.level = 0.99, pi.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = 2.5533006
#                                 sdlog   = 0.7060038
#
#Estimation Method:               mvue
#
#Data:                            Chrysene.bkgd
#
#Sample Size:                     8
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                99%
#
#Number of Future
#Geometric Means:                 1
#
#Sample Size for
#Geometric Means:                 4
#
#Prediction Interval:             LPL =  0.00000
#                                 UPL = 46.96613

UPL <- predIntLnorm(Chrysene.bkgd, n.geomean = 4, k = 1,
  conf.level = 0.99, pi.type = "upper")$interval$limits["UPL"]

UPL
#     UPL
#46.96613

# Is there evidence of contamination at the compliance well?

geoMean(Chrysene.cmpl)
#[1] 44.19034

# Since the geometric mean at the compliance well is less than
# the upper prediction limit, there is no evidence of contamination.

#----------

# Cleanup
#--------

rm(Chrysene.bkgd, Chrysene.cmpl, UPL)
```

predIntLnormAltSimultaneousTestPower

Probability That at Least One Set of Future Observations Violates the Given Rule Based on a Simultaneous Prediction Interval for a Lognormal Distribution

Description

Compute the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for the next r future sampling occasions for a lognormal distribution. The three possible rules are: k-of-m, California, or Modified California.

Usage

```
predIntLnormAltSimultaneousTestPower(n, df = n - 1, n.geomean = 1, k = 1,
  m = 2, r = 1, rule = "k.of.m", ratio.of.means = 1, cv = 1, pi.type = "upper",
  conf.level = 0.95, r.shifted = r, K.tol = .Machine$double.eps^0.5,
  integrate.args.list = NULL)
```

Arguments

n	vector of positive integers greater than 2 indicating the sample size upon which the prediction interval is based.
df	vector of positive integers indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.geomean	positive integer specifying the sample size associated with the future geometric means. The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), vector of positive integers specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	vector of positive integers specifying the maximum number of future observations (or averages) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	vector of positive integers specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.
ratio.of.means	numeric vector specifying the ratio of the mean of the population that will be sampled to produce the future observations vs. the mean of the population that was sampled to construct the prediction interval. See the DETAILS section below for more information. The default value is ratio.of.means=1.

`cv` numeric vector of positive values specifying the coefficient of variation for both the population that was sampled to construct the prediction interval **and** the population that will be sampled to produce the future observations. The default value is `cv=1`.

`pi.type` character string indicating what kind of prediction interval to compute. The possible values are `pi.type="upper"` (the default), and `pi.type="lower"`.

`conf.level` vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

`r.shifted` vector of positive integers specifying the number of future sampling occasions for which the mean is shifted. All values must be integeters between 1 and the corresponding element of `r`. The default value is `r.shifted=r`.

`K.tol` numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is `K.tol=.Machine$double.eps^(1/2)`. For many applications, the value of K needs to be known only to the second decimal place, in which case setting `K.tol=1e-4` will speed up computation a bit.

`integrate.args.list` a list of arguments to supply to the `integrate` function. The default value is `integrate.args.list=NULL` which means that the default values of `integrate` are used.

Details

What is a Simultaneous Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNorm` computes a standard prediction interval based on a sample from a normal distribution.

The function `predIntLnormAltSimultaneous` computes a simultaneous prediction interval (assuming lognormal observations) that will contain a certain number of future observations with probability $(1-\alpha)100\%$ for each of r future sampling "occasions", where r is some pre-specified positive integer. The quantity r may refer to r distinct future sampling occasions in time, or it may for example refer to sampling at r distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the r distinct locations.

The function `predIntLnormAltSimultaneous` computes a simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (`rule="k.of.m"`), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If obserations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: When $k = m$ and $r = 1$, the results of `predIntNormSimultaneous` are equivalent to the results of `predIntNorm`.
- For the California rule (`rule="CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.

- For the Modified California rule (`rule="Modified.CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Computing Power

The function `predIntNormSimultaneousTestPower` computes the probability that at least one set of future observations or averages will violate the given rule based on a simultaneous prediction interval for the next r future sampling occasions for a normal distribution, based on the assumption of normally distributed observations, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval.

The function `predIntLnormAltSimultaneousTestPower` assumes all observations are from a lognormal distribution. The observations used to construct the prediction interval are assumed to come from a lognormal distribution with mean θ_2 and coefficient of variation τ. The future observations are assumed to come from a lognormal distribution with mean θ_1 and coefficient of variation τ; that is, the means are allowed to differ between the two populations, but not the coefficient of variation.

The function `predIntLnormAltSimultaneousTestPower` calls the function `predIntNormSimultaneousTestPower`, with the argument `delta.over.sigma` given by:

$$\frac{\delta}{\sigma} = \frac{log(R)}{\sqrt{log(\tau^2+1)}} \qquad (1)$$

where R is given by:

$$R = \frac{\theta_1}{\theta_2} \qquad (2)$$

and corresponds to the argument `ratio.of.means` for the function `predIntLnormAltSimultaneousTestPower`, and τ corresponds to the argument `cv`.

Value

vector of values between 0 and 1 equal to the probability that the rule will be violated.

Note

See the help files for `predIntLnormAltSimultaneous` and `predIntNormSimultaneousTestPower`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntLnormAltSimultaneous`.

See Also

predIntLnormAltSimultaneous,
plotPredIntLnormAltSimultaneousTestPowerCurve,
predIntNormSimultaneous,
plotPredIntNormSimultaneousTestPowerCurve,
Prediction Intervals, LognormalAlt.

Examples

```
# For the k-of-m rule with n=4, k=1, m=3, and r=1, show how the power increases
# as ratio.of.means increases.  Assume a 95% upper prediction interval.

predIntLnormAltSimultaneousTestPower(n = 4, m = 3, ratio.of.means = 1:3)
#[1] 0.0500000 0.2356914 0.4236723

#----------

# Look at how the power increases with sample size for an upper one-sided
# prediction interval using the k-of-m rule with k=1, m=3, r=20,
# ratio.of.means=4, and a confidence level of 95%.

predIntLnormAltSimultaneousTestPower(n = c(4, 8), m = 3, r = 20, ratio.of.means = 4)
#[1] 0.4915743 0.8218175

#----------

# Compare the power for the 1-of-3 rule with the power for the California and
# Modified California rules, based on a 95% upper prediction interval and
# ratio.of.means=4.  Assume a sample size of n=8.  Note that in this case the
# power for the Modified California rule is greater than the power for the
# 1-of-3 rule and California rule.

predIntLnormAltSimultaneousTestPower(n = 8, k = 1, m = 3, ratio.of.means = 4)
#[1] 0.6594845

predIntLnormAltSimultaneousTestPower(n = 8, m = 3, rule = "CA", ratio.of.means = 4)
#[1] 0.5864311

predIntLnormAltSimultaneousTestPower(n = 8, rule = "Modified.CA", ratio.of.means = 4)
#[1] 0.691135

#----------

# Show how the power for an upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions r increases.  Here, we'll use the
# 1-of-3 rule with n=8 and ratio.of.means=4.

predIntLnormAltSimultaneousTestPower(n = 8, k = 1, m = 3, r = c(1, 2, 5, 10),
  ratio.of.means = 4)
#[1] 0.6594845 0.7529576 0.8180814 0.8302302
```

predIntLnormAltTestPower
Probability That at Least One Future Observation Falls Outside a Prediction Interval for a Lognormal Distribution

Description

Compute the probability that at least one out of k future observations (or geometric means) falls outside a prediction interval for k future observations (or geometric means) for a normal distribution.

Usage

```
predIntLnormAltTestPower(n, df = n - 1, n.geomean = 1, k = 1,
  ratio.of.means = 1, cv = 1, pi.type = "upper", conf.level = 0.95)
```

Arguments

n	vector of positive integers greater than 2 indicating the sample size upon which the prediction interval is based.
df	vector of positive integers indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.geomean	positive integer specifying the sample size associated with the future geometric means. The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
k	vector of positive integers specifying the number of future observations that the prediction interval should contain with confidence level conf.level. The default value is k=1.
ratio.of.means	numeric vector specifying the ratio of the mean of the population that will be sampled to produce the future observations vs. the mean of the population that was sampled to construct the prediction interval. See the DETAILS section below for more information. The default value is ratio.of.means=1.
cv	numeric vector of positive values specifying the coefficient of variation for both the population that was sampled to construct the prediction interval **and** the population that will be sampled to produce the future observations. The default value is cv=1.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".
conf.level	numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

Details

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations or averages from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is call the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNorm` computes a standard prediction interval based on a sample from a normal distribution.

The function `predIntNormTestPower` computes the probability that at least one out of k future observations or averages will **not** be contained in a prediction interval based on the assumption of normally distributed observations, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval.

The function `predIntLnormAltTestPower` assumes all observations are from a lognormal distribution. The observations used to construct the prediction interval are assumed to come from a lognormal distribution with mean θ_2 and coefficient of variation τ. The future observations are assumed to come from a lognormal distribution with mean θ_1 and coefficient of variation τ; that is, the means are allowed to differ between the two populations, but not the coefficient of variation.

The function `predIntLnormAltTestPower` calls the function `predIntNormTestPower`, with the argument `delta.over.sigma` given by:

$$\frac{\delta}{\sigma} = \frac{log(R)}{\sqrt{log(\tau^2 + 1)}} \qquad (1)$$

where R is given by:

$$R = \frac{\theta_1}{\theta_2} \qquad (2)$$

and corresponds to the argument `ratio.of.means` for the function `predIntLnormAltTestPower`, and τ corresponds to the argument `cv`.

Value

vector of numbers between 0 and 1 equal to the probability that at least one of k future observations or geometric means will fall outside the prediction interval.

Note

See the help files for `predIntNormTestPower`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for `predIntNormTestPower` and `tTestLnormAltPower`.

See Also

`plotPredIntLnormAltTestPowerCurve`, `predIntLnormAlt`, `predIntNorm`, `predIntNormK`, `plotPredIntNormTestPowerCurve`, `predIntLnormAltSimultaneous`, `predIntLnormAltSimultaneousTestPower`, Prediction Intervals, LognormalAlt.

Examples

```
# Show how the power increases as ratio.of.means increases.  Assume a
# 95% upper prediction interval.

predIntLnormAltTestPower(n = 4, ratio.of.means = 1:3)
#[1] 0.0500000 0.1459516 0.2367793

#----------

# Look at how the power increases with sample size for an upper one-sided
# prediction interval with k=3, ratio.of.means=4, and a confidence level of 95%.

predIntLnormAltTestPower(n = c(4, 8), k = 3, ratio.of.means = 4)
#[1] 0.2860952 0.4533567

#----------

# Show how the power for an upper 95% prediction limit increases as the
# number of future observations k increases.  Here, we'll use n=20 and
# ratio.of.means=2.

predIntLnormAltTestPower(n = 20, k = 1:3, ratio.of.means = 2)
#[1] 0.1945886 0.2189538 0.2321562
```

---

predIntLnormSimultaneous

Simultaneous Prediction Interval for a Lognormal Distribution

---

Description

Estimate the mean and standard deviation on the log-scale for a lognormal distribution, or estimate the mean and coefficient of variation for a lognormal distribution (alternative parameterization), and construct a simultaneous prediction interval for the next r sampling occasions, based on one of three possible rules: k-of-m, California, or Modified California.

Usage

```
predIntLnormSimultaneous(x, n.geomean = 1, k = 1, m = 2, r = 1, rule = "k.of.m",
  delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
  K.tol = .Machine$double.eps^0.5)

predIntLnormAltSimultaneous(x, n.geomean = 1, k = 1, m = 2, r = 1, rule = "k.of.m",
  delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
  K.tol = .Machine$double.eps^0.5, est.arg.list = NULL)
```

Arguments

x	For predIntLnormSimultaneous, x can be a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., elnorm or elnormCensored). You *cannot* supply objects resulting from a call to estimating functions that use the alternative parameterization such as elnormAlt or elnormAltCensored. For predIntLnormAltSimultaneous, a numeric vector of positive observations. If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n.geomean	positive integer specifying the sample size associated with future geometric means. The default value is n.geomean=1 (i.e., individual observations). Note that all future geometric means must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or geometric means) out of m observations (or geometric means) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	positive integer specifying the maximum number of future observations (or geometric means) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	positive integer specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.
delta.over.sigma	numeric scalar indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population (on the log-scale) that was sampled to construct the prediction interval, and the mean of the population (on the log-scale) that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation (on the log-scale) for both populations. See the DETAILS section below for more information. The default value is delta.over.sigma=0.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), pi.type="lower" and pi.type="two-sided".
conf.level	a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
K.tol	numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is K.tol=.Machine$double.eps^(1/2). For many applications, the value of K needs to be known only to the second decimal place, in which case setting K.tol=1e-4 will speed up computation a bit.

`est.arg.list` a list containing arguments to pass to the function `elnormAlt` for estimating the mean and coefficient of variation. The default value is `est.arg.list=NULL`, which implies the default values will be used in the call to `elnormAlt`.

Details

The function `predIntLnormSimultaneous` returns a simultaneous prediction interval as well as estimates of the meanlog and sdlog parameters. The function `predIntLnormAltSimultaneous` returns a prediction interval as well as estimates of the mean and coefficient of variation.

A simultaneous prediction interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a prediction interval based on the normal (Gaussian) distribution by calling `predIntNormSimultaneous`. These prediction limits are then exponentiated to produce a prediction interval on the original scale of the data.

Value

If `x` is a numeric vector, `predIntLnormSimultaneous` returns a list of class `"estimate"` containing the estimated parameters, the prediction interval, and other information. See the help file for `estimate.object` for details.

If `x` is the result of calling an estimation function, `predIntLnormSimultaneous` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as a component called `interval` containing the prediction interval information. If `x` already has a component called `interval`, this component is replaced with the prediction interval information.

Note

Motivation

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional α-level (e.g., 0.05 or 0.01) for each test leads to a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others.

Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual α-level at each well is equal to the FWFRP divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If

spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-m rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.

The k-of-m Rule

For the k-of-m rule, Davis and McNichols (1987) give tables with "optimal" choices of k (in terms of best power for a given overall confidence level) for selected values of m, r, and n. They found that the optimal ratios of k to m (i.e., k/m) are generally small, in the range of 15-50%.

The California Rule

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay's Code of California Regulations, 1991). The California code mandates a "California" rule with $m \geq 3$. The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a k-of-m rule with $k = 1$ and $m = 3$, a monitoring location will pass if the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with $m = 3$, either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPR is kept constant, then the California rule offers little increased power compared to the k-of-m rule, and can actually decrease the power of detecting contamination.

The Modified California Rule

The Modified California Rule was proposed as a compromise between a 1-of-m rule and the California rule. For a given FWFPR, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-m rule.

Different Notations Between Different References

For the k-of-m rule described in this help file, both Davis and McNichols (1987) and USEPA (2009, Chapter 19) use the variable p instead of k to represent the minimum number of future observations the interval should contain on each of the r sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of K for both k-of-m rules and California rules. Gibbons et al.'s notation reverses the meaning of k and r compared to the notation used in this help file. That is, in Gibbons et al.'s notation, k represents the number of future sampling occasions or monitoring wells, and r represents the minimum number of observations the interval should contain on each sampling occasion.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Barclay's California Code of Regulations. (1991). Title 22, Section 66264.97 [concerning hazardous waste facilities] and Title 23, Section 2550.7(e)(8) [concerning solid waste facilities]. Barclay's Law Publishers, San Francisco, CA.

Davis, C.B. (1998a). *Ground-Water Statistics & Regulations: Principles, Progress and Problems*. Second Edition. Environmetrics & Statistics Limited, Henderson, NV.

Davis, C.B. (1998b). Personal Communication, September 3, 1998.

Davis, C.B., and R.J. McNichols. (1987). One-sided Intervals for at Least p of m Observations from a Lognormal Population on Each of r Future Occasions. *Technometrics* **29**, 359–370.

Fertig, K.W., and N.R. Mann. (1977). One-Sided Prediction Intervals for at Least p Out of m Future Observations From a Lognormal Population. *Technometrics* **19**, 167–177.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J. (1969). Factors for Calculating Two-Sided Prediction Intervals for Samples from a Lognormal Distribution. *Journal of the American Statistical Association* **64**(327), 878-898.

Hahn, G.J. (1970a). Additional Factors for Calculating Prediction Intervals for Samples from a Lognormal Distribution. *Journal of the American Statistical Association* **65**(332), 1668-1676.

Hahn, G.J. (1970b). Statistical Intervals for a Lognormal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Lognormal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178-188.

Hall, I.J., and R.R. Prairie. (1973). One-Sided Prediction Intervals to Contain at Least m Out of k Future Observations. *Technometrics* **15**, 897–914.

Millard, S.P. (1987). Environmental Monitoring, Statistics, and the Law: Room for Improvement (with Comment). *The American Statistician* **41**(4), 249–259.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

`predIntLnormAltSimultaneousTestPower`, `predIntNorm`, `predIntNormSimultaneous`, `predIntNormSimultaneousTestPower`, `tolIntLnorm`, `Lognormal`, `LognormalAlt`, `estimate.object`, `elnorm`, `elnormAlt`.

Examples

```
# Generate 8 observations from a lognormal distribution with parameters
# mean=10 and cv=1, then use predIntLnormAltSimultaneous to estimate the
# mean and coefficient of variation of the true distribution and construct an
# upper 95% prediction interval to contain at least 1 out of the next
# 3 observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(479)
dat <- rlnormAlt(8, mean = 10, cv = 1)

predIntLnormAltSimultaneous(dat, k = 1, m = 3)

# Compare the 95% 1-of-3 upper prediction limit to the California and
# Modified California upper prediction limits.  Note that the upper
# prediction limit for the Modified California rule is between the limit
# for the 1-of-3 rule and the limit for the California rule.

predIntLnormAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]

predIntLnormAltSimultaneous(dat, m = 3, rule = "CA")$interval$limits["UPL"]

predIntLnormAltSimultaneous(dat, rule = "Modified.CA")$interval$limits["UPL"]

# Show how the upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions r increases.
# Here, we'll use the 1-of-3 rule.

predIntLnormAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]

predIntLnormAltSimultaneous(dat, k = 1, m = 3, r = 10)$interval$limits["UPL"]

# Compare the upper simultaneous prediction limit for the 1-of-3 rule
# based on individual observations versus based on geometric means of
# order 4.

predIntLnormAltSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]

predIntLnormAltSimultaneous(dat, n.geomean = 4, k = 1,
  m = 3)$interval$limits["UPL"]

#==========

# Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an
# upper simultaneous prediction limit for the 1-of-3 rule for
# r = 2 future sampling occasions.  The data for this example are
# stored in EPA.09.Ex.19.1.sulfate.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 25 to use to construct the prediction limit.
```

```
  # There are 50 compliance wells and we will monitor 10 different
  # constituents at each well at each of the r=2 future sampling
  # occasions.  To determine the confidence level we require for
  # the simultaneous prediction interval, USEPA (2009) recommends
  # setting the individual Type I Error level at each well to

  # 1 - (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))

  # which translates to setting the confidence limit to

  # (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))

  # where SWFPR = site-wide false positive rate.  For this example, we
  # will set SWFPR = 0.1.  Thus, the confidence level is given by:

  nc <- 10
  nw <- 50
  SWFPR <- 0.1
  conf.level <- (1 - SWFPR)^(1 / (nc * nw))

  conf.level

  #----------

  # Look at the data:

  names(EPA.09.Ex.19.1.sulfate.df)

  EPA.09.Ex.19.1.sulfate.df[,
    c("Well", "Date", "Sulfate.mg.per.l", "log.Sulfate.mg.per.l")]



  # Construct the upper simultaneous prediction limit for the
  # 1-of-3 plan assuming a lognormal distribution for the
  # sulfate data

  Sulfate <- EPA.09.Ex.19.1.sulfate.df$Sulfate.mg.per.l

  predIntLnormSimultaneous(x = Sulfate, k = 1, m = 3, r = 2,
    rule = "k.of.m", pi.type = "upper", conf.level = conf.level)

  #==========

  # NOTE: Two-sided simultaneous prediction intervals computed using
  # Versions 2.4.0 - 2.8.1 of EnvStats are *NOT* valid.
  ## Not run:
  predIntLnormSimultaneous(x = Sulfate, k = 1, m = 3, r = 2,
    rule = "k.of.m", pi.type = "two-sided", conf.level = conf.level)

## End(Not run)
```

predIntNorm *Prediction Interval for a Normal Distribution*

Description

Estimate the mean and standard deviation of a normal distribution, and construct a prediction interval for the next k observations or next set of k means.

Usage

```
predIntNorm(x, n.mean = 1, k = 1, method = "Bonferroni",
  pi.type = "two-sided", conf.level = 0.95)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (e.g., enorm, eqnorm, enormCensored, etc.). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n.mean	positive integer specifying the sample size associated with the k future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
k	positive integer specifying the number of future observations or averages the prediction interval should contain with confidence level conf.level. The default value is k=1.
method	character string specifying the method to use if the number of future observations (k) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferonni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). See the DETAILS section of predIntNormK for more information. This argument is ignored if k=1.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="two-sided" (the default), pi.type="lower", and pi.type="upper".
conf.level	a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

Details

What is a Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations or averages from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval.

The Form of a Prediction Interval

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with parameters `mean`=μ and `sd`=σ. Also, let m denote the sample size associated with the k future averages (i.e., `n.mean`=m). When $m = 1$, each average is really just a single observation, so in the rest of this help file the term "averages" will replace the phrase "observations or averages".

For a normal distribution, the form of a two-sided $(1 - \alpha)100\%$ prediction interval is:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future averages k, and the sample size associated with the future averages, m. Do not confuse the constant K (uppercase K) with the number of future averages k (lowercase k). The symbol K is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \qquad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \qquad (5)$$

but K differs for one-sided versus two-sided prediction intervals. The derivation of the constant K is explained in the help file for `predIntNormK`.

A Prediction Interval is a Random Interval

A prediction interval is a *random* interval; that is, the lower and/or upper bounds are random variables computed based on sample statistics in the baseline sample. Prior to taking one specific baseline sample, the probability that the prediction interval will contain the next k averages is $(1 - \alpha)100\%$. Once a specific baseline sample is taken and the prediction interval based on that sample is computed, the probability that that prediction interval will contain the next k averages is not necessarily $(1 - \alpha)100\%$, but it should be close.

If an experiment is repeated N times, and for each experiment:

1. A sample is taken and a $(1 - \alpha)100\%$ prediction interval for $k = 1$ future observation is computed, and
2. One future observation is generated and compared to the prediction interval,

then the number of prediction intervals that actually contain the future observation generated in step 2 above is a binomial random variable with parameters `size`=N and `prob`=$(1 - \alpha)100\%$.

If, on the other hand, only one baseline sample is taken and only one prediction interval for $k = 1$ future observation is computed, then the number of future observations out of a total of N future observations that will be contained in that one prediction interval is a binomial random variable with parameters `size`=N and `prob`=$(1 - \alpha^*)100\%$, where α^* depends on the true population parameters and the computed bounds of the prediction interval.

Value

If `x` is a numeric vector, `predIntNorm` returns a list of class `"estimate"` containing the estimated parameters, the prediction interval, and other information. See the help file for `estimate.object` for details.

If `x` is the result of calling an estimation function, `predIntNorm` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as a component called `interval` containing the prediction interval information. If `x` already has a component called `interval`, this component is replaced with the prediction interval information.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Dunnett, C.W. (1955). A Multiple Comparisons Procedure for Comparing Several Treatments with a Control. *Journal of the American Statistical Association* **50**, 1096-1121.

Dunnett, C.W. (1964). New Tables for Multiple Comparisons with a Control. *Biometrics* **20**, 482-491.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J. (1969). Factors for Calculating Two-Sided Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **64**(327), 878-898.

Hahn, G.J. (1970a). Additional Factors for Calculating Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **65**(332), 1668-1676.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178-188.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York.

Helsel, D.R., and R.M. Hirsch. (2002). *Statistical Methods in Water Resources*. Techniques of Water Resources Investigations, Book 4, chapter A3. U.S. Geological Survey. (available on-line at: https://pubs.usgs.gov/tm/04/a03/tm4a3.pdf).

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Miller, R.G. (1981a). *Simultaneous Statistical Inference*. McGraw-Hill, New York.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

predIntNormK, predIntNormSimultaneous, predIntLnorm, tolIntNorm, Normal, estimate.object, enorm, eqnorm.

Examples

```
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=2, then create a two-sided 95% prediction interval for
# the next observation.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(47)
dat <- rnorm(20, mean = 10, sd = 2)
predIntNorm(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 9.792856
#                                 sd   = 1.821286
#
#Estimation Method:               mvue
#
```

```
#Data:                             dat
#
#Sample Size:                      20
#
#Prediction Interval Method:       exact
#
#Prediction Interval Type:         two-sided
#
#Confidence Level:                 95%
#
#Number of Future Observations:    1
#
#Prediction Interval:              LPL =  5.886723
#                                  UPL = 13.698988

#----------

# Using the same data from the last example, create a one-sided
# upper 99% prediction limit for the next 3 averages of order 2
# (i.e., each of the 3 future averages is based on a sample size
# of 2 future observations).

predIntNorm(dat, n.mean = 2, k = 3, conf.level = 0.99,
  pi.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:             Normal
#
#Estimated Parameter(s):           mean = 9.792856
#                                  sd   = 1.821286
#
#Estimation Method:                mvue
#
#Data:                             dat
#
#Sample Size:                      20
#
#Prediction Interval Method:       Bonferroni
#
#Prediction Interval Type:         upper
#
#Confidence Level:                 99%
#
#Number of Future Averages:        3
#
#Sample Size for Averages:         2
#
#Prediction Interval:              LPL =     -Inf
#                                  UPL = 13.90537

#----------
```

```
# Compare the result above that is based on the Bonferroni method
# with the exact method

predIntNorm(dat, n.mean = 2, k = 3, conf.level = 0.99,
  pi.type = "upper", method = "exact")$interval$limits["UPL"]

#     UPL
#13.89272

#----------

# Clean up
rm(dat)

#--------------------------------------------------------------------

# Example 18-1 of USEPA (2009, p.18-9) shows how to construct a 95%
# prediction interval for 4 future observations assuming a
# normal distribution based on arsenic concentrations (ppb) in
# groundwater at a solid waste landfill.  There were 4 years of
# quarterly monitoring, and years 1-3 are considered background.
# The question to be answered is whether there is evidence of
# contamination in year 4.

# The data for this example is stored in EPA.09.Ex.18.1.arsenic.df.

EPA.09.Ex.18.1.arsenic.df

#   Year Sampling.Period Arsenic.ppb
#1     1      Background        12.6
#2     1      Background        30.8
#3     1      Background        52.0
#4     1      Background        28.1
#5     2      Background        33.3
#6     2      Background        44.0
#7     2      Background         3.0
#8     2      Background        12.8
#9     3      Background        58.1
#10    3      Background        12.6
#11    3      Background        17.6
#12    3      Background        25.3
#13    4      Compliance        48.0
#14    4      Compliance        30.3
#15    4      Compliance        42.5
#16    4      Compliance        15.0

As.bkgd <- with(EPA.09.Ex.18.1.arsenic.df,
  Arsenic.ppb[Sampling.Period == "Background"])
As.cmpl <- with(EPA.09.Ex.18.1.arsenic.df,
  Arsenic.ppb[Sampling.Period == "Compliance"])

# A Shapiro-Wilks goodness-of-fit test for normality indicates
```

```
# there is no evidence to reject the assumption of normality
# for the background data:

gofTest(As.bkgd)

#Results of Goodness-of-Fit Test
#--------------------------------
#
#Test Method:                     Shapiro-Wilk GOF
#
#Hypothesized Distribution:       Normal
#
#Estimated Parameter(s):          mean = 27.51667
#                                 sd   = 17.10119
#
#Estimation Method:               mvue
#
#Data:                            As.bkgd
#
#Sample Size:                     12
#
#Test Statistic:                  W = 0.94695
#
#Test Statistic Parameter:        n = 12
#
#P-value:                         0.5929102
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Normal Distribution.

# Here is the one-sided 95% upper prediction limit:

UPL <- predIntNorm(As.bkgd, k = 4,
  pi.type = "upper")$interval$limits["UPL"]
UPL
#     UPL
#73.67237

# Are any of the compliance observations above the prediction limit?

any(As.cmpl > UPL)
#[1] FALSE

#==========

# Cleanup
#--------

rm(As.bkgd, As.cmpl, UPL)
```

`predIntNormHalfWidth` *Half-Width of a Prediction Interval for the next k Observations from a Normal Distribution*

Description

Compute the half-width of a prediction interval for the next k observations from a normal distribution.

Usage

```
predIntNormHalfWidth(n, df = n - 1, n.mean = 1, k = 1, sigma.hat = 1,
  method = "Bonferroni", conf.level = 0.95)
```

Arguments

`n`	numeric vector of positive integers greater than 1 indicating the sample size upon which the prediction interval is based. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`df`	numeric vector of positive integers indicating the degrees of freedom associated with the prediction interval. The default is `df=n-1`.
`n.mean`	numeric vector of positive integers specifying the sample size associated with the k future *averages*. The default value is `n.mean=1` (i.e., individual observations). Note that all future averages must be based on the same sample size.
`k`	numeric vector of positive integers specifying the number of future observations or averages the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`.
`sigma.hat`	numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is `sigma.hat=1`.
`method`	character string specifying the method to use if the number of future observations (`k`) is greater than 1. The possible values are `method="Bonferroni"` (approximate method based on Bonferonni inequality; the default), and `method="exact"` (exact method due to Dunnett, 1955). This argument is ignored if `k=1`.
`conf.level`	numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

Details

If the arguments `n`, `k`, `n.mean`, `sigma.hat`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help files for `predIntNorm` and `predIntNormK` give formulas for a two-sided prediction interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level. Specifically, the two-sided prediction interval is given by:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future averages k, and the sample size associated with the future averages, m (see the help file for predIntNormK). Thus, the half-width of the prediction interval is given by:

$$HW = Ks \qquad (4)$$

Value

numeric vector of half-widths.

Note

See the help file for predIntNorm.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNorm.

See Also

predIntNorm, predIntNormK, predIntNormN, plotPredIntNormDesign.

Examples

```
# Look at how the half-width of a prediction interval increases with
# increasing number of future observations:

1:5
#[1] 1 2 3 4 5

hw <- predIntNormHalfWidth(n = 10, k = 1:5)

round(hw, 2)
#[1] 2.37 2.82 3.08 3.26 3.41

#----------
```

```
# Look at how the half-width of a prediction interval decreases with
# increasing sample size:

2:5
#[1] 2 3 4 5

hw <- predIntNormHalfWidth(n = 2:5)

round(hw, 2)
#[1] 15.56  4.97  3.56  3.04

#----------

# Look at how the half-width of a prediction interval increases with
# increasing estimated standard deviation for a fixed sample size:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

hw <- predIntNormHalfWidth(n = 10, sigma.hat = seq(0.5, 2, by = 0.5))

round(hw, 2)
#[1] 1.19 2.37 3.56 4.75

#----------

# Look at how the half-width of a prediction interval increases with
# increasing confidence level for a fixed sample size:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

hw <- predIntNormHalfWidth(n = 5, conf = seq(0.5, 0.9, by = 0.1))

round(hw, 2)
#[1] 0.81 1.03 1.30 1.68 2.34

#==========

# The data frame EPA.92c.arsenic3.df contains arsenic concentrations (ppb)
# collected quarterly for 3 years at a background well and quarterly for
# 2 years at a compliance well.  Using the data from the background well, compute
# the half-width associated with sample sizes of 12 (3 years of quarterly data),
# 16 (4 years of quarterly data), and 20 (5 years of quarterly data) for a
# two-sided 90% prediction interval for k=4 future observations.

EPA.92c.arsenic3.df
#   Arsenic Year  Well.type
#1     12.6    1 Background
#2     30.8    1 Background
#3     52.0    1 Background
#...
#18     3.8    5 Compliance
```

```
#19     2.6    5 Compliance
#20    51.9    5 Compliance

mu.hat <- with(EPA.92c.arsenic3.df,
  mean(Arsenic[Well.type=="Background"]))

mu.hat
#[1] 27.51667

sigma.hat <- with(EPA.92c.arsenic3.df,
  sd(Arsenic[Well.type=="Background"]))

sigma.hat
#[1] 17.10119

hw <- predIntNormHalfWidth(n = c(12, 16, 20), k = 4, sigma.hat = sigma.hat,
  conf.level = 0.9)

round(hw, 2)
#[1] 46.16 43.89 42.64

#==========

# Clean up
#---------
rm(hw, mu.hat, sigma.hat)
```

---

predIntNormK *Compute the Value of K for a Prediction Interval for a Normal Distribution*

Description

Compute the value of K (the multiplier of estimated standard deviation) used to construct a prediction interval for the next k observations or next set of k means based on data from a normal distribution. The function predIntNormK is called by predIntNorm.

Usage

```
predIntNormK(n, df = n - 1, n.mean = 1, k = 1,
  method = "Bonferroni", pi.type = "two-sided",
  conf.level = 0.95)
```

Arguments

n	a positive integer greater than 2 indicating the sample size upon which the prediction interval is based.
df	the degrees of freedom associated with the prediction interval. The default is df=n-1.

n.mean	positive integer specifying the sample size associated with the k future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
k	positive integer specifying the number of future observations or averages the prediction interval should contain with confidence level conf.level. The default value is k=1.
method	character string specifying the method to use if the number of future observations (k) is greater than 1. The possible values are method="Bonferroni" (approximate method based on Bonferonni inequality; the default), and method="exact" (exact method due to Dunnett, 1955). See the DETAILS section for more information. This argument is ignored if k=1.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="two-sided" (the default), pi.type="lower", and pi.type="upper".
conf.level	a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.

Details

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations or averages from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval.

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with parameters mean=μ and sd=σ. Also, let m denote the sample size associated with the k future averages (i.e., n.mean=m). When $m = 1$, each average is really just a single observation, so in the rest of this help file the term "averages" will replace the phrase "observations or averages".

For a normal distribution, the form of a two-sided $(1-\alpha)100\%$ prediction interval is:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future averages k, and the sample size associated with the future averages, m. Do not confuse the constant K (uppercase K) with the number of future averages k (lowercase k). The symbol K is used here to be consistent with the notation used for tolerance intervals (see tolIntNorm).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \qquad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \qquad (5)$$

but K differs for one-sided versus two-sided prediction intervals. The derivation of the constant K is explained below. The function `predIntNormK` computes the value of K and is called by `predIntNorm`.

The Derivation of K for One Future Observation or Average (k = 1)

Let X denote a random variable from a normal distribution with parameters `mean`=μ and `sd`=σ, and let x_p denote the p'th quantile of X.

A true two-sided $(1-\alpha)100\%$ prediction interval for the next $k = 1$ observation of X is given by:

$$[x_{\alpha/2}, x_{1-\alpha/2}] = [\mu - z_{1-\alpha/2}\sigma, \mu + z_{1-\alpha/2}\sigma] \qquad (6)$$

where z_p denotes the p'th quantile of a standard normal distribution.

More generally, a true two-sided $(1-\alpha)100\%$ prediction interval for the next $k = 1$ average based on a sample of size m is given by:

$$[\mu - z_{1-\alpha/2}\frac{\sigma}{\sqrt{m}}, \mu + z_{1-\alpha/2}\frac{\sigma}{\sqrt{m}}] \qquad (7)$$

Because the values of μ and σ are unknown, they must be estimated, and a prediction interval then constructed based on the estimated values of μ and σ.

For a two-sided prediction interval (`pi.type="two-sided"`), the constant K for a $(1-\alpha)100\%$ prediction interval for the next $k = 1$ average based on a sample size of m is computed as:

$$K = t_{n-1,1-\alpha/2}\sqrt{\frac{1}{m} + \frac{1}{n}} \qquad (8)$$

where $t_{\nu,p}$ denotes the p'th quantile of the Student's t-distribution with ν degrees of freedom. For a one-sided prediction interval (`pi.type="lower"` or `pi.type="lower"`), the prediction interval is given by:

$$K = t_{n-1,1-\alpha}\sqrt{\frac{1}{m} + \frac{1}{n}} \qquad (9)$$

.

The formulas for these prediction intervals are derived as follows. Let $\bar{y}$ denote the future average based on m observations. Then the quantity $\bar{y} - \bar{x}$ has a normal distribution with expectation and variance given by:

$$E(\bar{y} - \bar{x}) = 0 \qquad (10)$$

$$Var(\bar{y} - \bar{x}) = Var(\bar{y}) + Var(\bar{x}) = \frac{\sigma^2}{m} + \frac{\sigma^2}{n} = \sigma^2(\frac{1}{m} + \frac{1}{n}) \qquad (11)$$

so the quantity

$$t = \frac{\bar{y} - \bar{x}}{s\sqrt{\frac{1}{m} + \frac{1}{n}}} \qquad (12)$$

has a Student's t-distribution with $n - 1$ degrees of freedom.

The Derivation of K for More than One Future Observation or Average (k >1)

When $k > 1$, the function `predIntNormK` allows for two ways to compute K: an exact method due to Dunnett (1955) (`method="exact"`), and an approximate (conservative) method based on the Bonferroni inequality (`method="Bonferroni"`; see Miller, 1981a, pp.8, 67-70; Gibbons et al., 2009, p.4). Each of these methods is explained below.

Exact Method Due to Dunnett (1955) (`method="exact"`)

Dunnett (1955) derived the value of K in the context of the multiple comparisons problem of comparing several treatment means to one control mean. The value of K is computed as:

$$K = c\sqrt{\frac{1}{m} + \frac{1}{n}} \qquad (13)$$

where c is a constant that depends on the sample size n, the number of future observations (averages) k, the sample size associated with the k future averages m, and the confidence level $(1-\alpha)100\%$.

When `pi.type="lower"` or `pi.type="upper"`, the value of c is the number that satisfies the following equation (Gupta and Sobel, 1957; Hahn, 1970a):

$$1 - \alpha = \int_0^\infty F_1(cs, k, \rho) h(s\sqrt{n-1}, n-1)\sqrt{n-1} ds \qquad (14)$$

where

$$F_1(x, k, \rho) = \int_\infty^\infty [\Phi(\frac{x + \rho^{1/2} y}{\sqrt{1-\rho}})]^k \phi(y) dy \qquad (15)$$

$$\rho = 1/(\frac{n}{m} + 1) \qquad (16)$$

$$h(x, \nu) = \frac{x^{\nu-1} e^{-x^2/2}}{2^{(\nu/2)-1}\Gamma(\frac{\nu}{2})} \qquad (17)$$

and $\Phi()$ and $\phi()$ denote the cumulative distribution function and probability density function, respectively, of the standard normal distribution. Note that the function $h(x, \nu)$ is the probability density function of a chi random variable with ν degrees of freedom.

When `pi.type="two-sided"`, the value of c is the number that satisfies the following equation:

$$1 - \alpha = \int_0^\infty F_2(cs, k, \rho) h(s\sqrt{n-1}, n-1)\sqrt{n-1} ds \qquad (18)$$

where

$$F_2(x, k, \rho) = \int_\infty^\infty [\Phi(\frac{x + \rho^{1/2} y}{\sqrt{1-\rho}}) - \Phi(\frac{-x + \rho^{1/2} y}{\sqrt{1-\rho}})]^k \phi(y) dy \qquad (19)$$

Approximate Method Based on the Bonferroni Inequality (`method="Bonferroni"`)

As shown above, when $k = 1$, the value of K is given by Equation (8) or Equation (9) for two-sided or one-sided prediction intervals, respectively. When $k > 1$, a conservative way to construct a $(1-\alpha^*)100\%$ prediction interval for the next k observations or averages is to use a Bonferroni correction (Miller, 1981a, p.8) and set $\alpha = \alpha^*/k$ in Equation (8) or (9) (Chew, 1968). This value of K will be conservative in that the computed prediction intervals will be wider than the exact predictions intervals. Hahn (1969, 1970a) compared the exact values of K with those based on the

Bonferroni inequality for the case of $m = 1$ and found the approximation to be quite satisfactory except when n is small, k is large, and α is large. For example, Gibbons (1987a) notes that for a 99% prediction interval (i.e., $\alpha = 0.01$) for the next k observations, if $n > 4$, the bias of K is never greater than 1% no matter what the value of k.

Value

A numeric scalar equal to K, the multiplier of estimated standard deviation that is used to construct the prediction interval.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Dunnett, C.W. (1955). A Multiple Comparisons Procedure for Comparing Several Treatments with a Control. *Journal of the American Statistical Association* **50**, 1096-1121.

Dunnett, C.W. (1964). New Tables for Multiple Comparisons with a Control. *Biometrics* **20**, 482-491.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J. (1969). Factors for Calculating Two-Sided Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **64**(327), 878-898.

Hahn, G.J. (1970a). Additional Factors for Calculating Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **65**(332), 1668-1676.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178-188.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York.

Helsel, D.R., and R.M. Hirsch. (2002). *Statistical Methods in Water Resources*. Techniques of Water Resources Investigations, Book 4, chapter A3. U.S. Geological Survey. (available on-line at: https://pubs.usgs.gov/tm/04/a03/tm4a3.pdf).

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

Miller, R.G. (1981a). *Simultaneous Statistical Inference*. McGraw-Hill, New York.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

predIntNorm, predIntNormSimultaneous, predIntLnorm, tolIntNorm, Normal, estimate.object, enorm, eqnorm.

Examples

```
# Compute the value of K for a two-sided 95% prediction interval
# for the next observation given a sample size of n=20.

predIntNormK(n = 20)
#[1] 2.144711

#----------------------------------------------------------------------

# Compute the value of K for a one-sided upper 99% prediction limit
# for the next 3 averages of order 2 (i.e., each of the 3 future
# averages is based on a sample size of 2 future observations) given a
# samle size of n=20.

predIntNormK(n = 20, n.mean = 2, k = 3, pi.type = "upper",
  conf.level = 0.99)
#[1] 2.258026

#----------

# Compare the result above that is based on the Bonferroni method
# with the exact method.

predIntNormK(n = 20, n.mean = 2, k = 3, method = "exact",
  pi.type = "upper", conf.level = 0.99)
#[1] 2.251084

#----------------------------------------------------------------------

# Example 18-1 of USEPA (2009, p.18-9) shows how to construct a 95%
```

```
# prediction interval for 4 future observations assuming a
# normal distribution based on arsenic concentrations (ppb) in
# groundwater at a solid waste landfill.  There were 4 years of
# quarterly monitoring, and years 1-3 are considered background,

# So the sample size for the prediciton limit is n = 12,
# and the number of future samples is k = 4.

predIntNormK(n = 12, k = 4, pi.type = "upper")
#[1] 2.698976
```

`predIntNormN` *Sample Size for a Specified Half-Width of a Prediction Interval for the next k Observations from a Normal Distribution*

Description

Compute the sample size necessary to achieve a specified half-width of a prediction interval for the next k observations from a normal distribution.

Usage

```
predIntNormN(half.width, n.mean = 1, k = 1, sigma.hat = 1,
  method = "Bonferroni", conf.level = 0.95, round.up = TRUE,
  n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

`half.width`	numeric vector of (positive) half-widths. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`n.mean`	numeric vector of positive integers specifying the sample size associated with the k future *averages*. The default value is `n.mean=1` (i.e., individual observations). Note that all future averages must be based on the same sample size.
`k`	numeric vector of positive integers specifying the number of future observations or averages the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`.
`sigma.hat`	numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is `sigma.hat=1`.
`method`	character string specifying the method to use if the number of future observations (`k`) is greater than 1. The possible values are `method="Bonferroni"` (approximate method based on Bonferonni inequality; the default), and `method="exact"` (exact method due to Dunnett, 1955). This argument is ignored if `k=1`.
`conf.level`	numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

round.up	logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=TRUE.
n.max	positive integer greater than 1 indicating the maximum possible sample size. The default value is n.max=5000.
tol	numeric scalar indicating the tolerance to use in the uniroot search algorithm. The default value is tol=1e-7.
maxiter	positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.

Details

If the arguments half.width, k, n.mean, sigma.hat, and conf.level are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help files for predIntNorm and predIntNormK give formulas for a two-sided prediction interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level. Specifically, the two-sided prediction interval is given by:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future averages k, and the sample size associated with the future averages, m (see the help file for predIntNormK). Thus, the half-width of the prediction interval is given by:

$$HW = Ks \qquad (4)$$

The function predIntNormN uses the uniroot search algorithm to determine the sample size for specified values of the half-width, number of observations used to create a single future average, number of future observations or averages, the sample standard deviation, and the confidence level. **Note that unlike a confidence interval, the half-width of a prediction interval does *not* approach 0 as the sample size increases.**

Value

numeric vector of sample sizes.

Note

See the help file for predIntNorm.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNorm.

See Also

predIntNorm, predIntNormK, predIntNormHalfWidth, plotPredIntNormDesign.

Examples

```
# Look at how the required sample size for a prediction interval increases
# with increasing number of future observations:

1:5
#[1] 1 2 3 4 5

predIntNormN(half.width = 3, k = 1:5)
#[1]  6  9 11 14 18

#----------

# Look at how the required sample size for a prediction interval decreases
# with increasing half-width:

2:5
#[1] 2 3 4 5

predIntNormN(half.width = 2:5)
#[1] 86  6  4  3

predIntNormN(2:5, round = FALSE)
#[1] 85.567387  5.122911  3.542393  2.987861

#----------

# Look at how the required sample size for a prediction interval increases
# with increasing estimated standard deviation for a fixed half-width:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

predIntNormN(half.width = 4, sigma.hat = seq(0.5, 2, by = 0.5))
#[1]  3  4  7 86

#----------

# Look at how the required sample size for a prediction interval increases
# with increasing confidence level for a fixed half-width:
```

```
seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

predIntNormN(half.width = 2, conf.level = seq(0.5, 0.9, by = 0.1))
#[1] 2 2 3 4 9

#==========

# The data frame EPA.92c.arsenic3.df contains arsenic concentrations (ppb)
# collected quarterly for 3 years at a background well and quarterly for
# 2 years at a compliance well.  Using the data from the background well,
# compute the required sample size in order to achieve a half-width of
# 2.25, 2.5, or 3 times the estimated standard deviation for a two-sided
# 90% prediction interval for k=4 future observations.
#
# For a half-width of 2.25 standard deviations, the required sample size is 526,
# or about 131 years of quarterly observations!  For a half-width of 2.5
# standard deviations, the required sample size is 20, or about 5 years of
# quarterly observations.  For a half-width of 3 standard deviations, the required
# sample size is 9, or about 2 years of quarterly observations.

EPA.92c.arsenic3.df
#   Arsenic Year  Well.type
#1     12.6    1 Background
#2     30.8    1 Background
#3     52.0    1 Background
#...
#18     3.8    5 Compliance
#19     2.6    5 Compliance
#20    51.9    5 Compliance

mu.hat <- with(EPA.92c.arsenic3.df,
  mean(Arsenic[Well.type=="Background"]))

mu.hat
#[1] 27.51667

sigma.hat <- with(EPA.92c.arsenic3.df,
  sd(Arsenic[Well.type=="Background"]))

sigma.hat
#[1] 17.10119

predIntNormN(half.width=c(2.25, 2.5, 3) * sigma.hat, k = 4,
  sigma.hat = sigma.hat, conf.level = 0.9)
#[1] 526  20   9

#==========

# Clean up
#---------
rm(mu.hat, sigma.hat)
```

predIntNormSimultaneous
Simultaneous Prediction Interval for a Normal Distribution

Description

Estimate the mean and standard deviation of a normal distribution, and construct a simultaneous prediction interval for the next r sampling "occasions", based on one of three possible rules: k-of-m, California, or Modified California.

Usage

```
predIntNormSimultaneous(x, n.mean = 1, k = 1, m = 2, r = 1, rule = "k.of.m",
  delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
  K.tol = .Machine$double.eps^0.5)
```

Arguments

x	a numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (e.g., enorm, eqnorm, enormCensored, etc.). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n.mean	positive integer specifying the sample size associated with the future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	positive integer specifying the maximum number of future observations (or averages) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	positive integer specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.
delta.over.sigma	numeric scalar indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is delta.over.sigma=0.

`pi.type`	character string indicating what kind of prediction interval to compute. The possible values are `pi.type="upper"` (the default) and `pi.type="lower"`. **NOTE:** In Versions 2.4.0 - 2.8.1 of *EnvStats*, the value `pi.type="two-sided"` was allowed, but these two-sided simultaneous prediction intervals were based on faulty assumptions and were **NOT** valid.
`conf.level`	a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.
`K.tol`	numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is `K.tol=.Machine$double.eps^(1/2)`. For many applications, the value of K needs to be known only to the second decimal place, in which case setting `K.tol=1e-4` will speed up computation a bit.

Details

What is a Simultaneous Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNorm` computes a standard prediction interval based on a sample from a normal distribution.

The function `predIntNormSimultaneous` computes a simultaneous prediction interval that will contain a certain number of future observations with probability $(1-\alpha)100\%$ for each of r future sampling "occasions", where r is some pre-specified positive integer. The quantity r may refer to r distinct future sampling occasions in time, or it may for example refer to sampling at r distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the r distinct locations.

The function `predIntNormSimultaneous` computes a simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (`rule="k.of.m"`), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If obserations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: When $k = m$ and $r = 1$, the results of `predIntNormSimultaneous` are equivalent to the results of `predIntNorm`.
- For the California rule (`rule="CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.
- For the Modified California rule (`rule="Modified.CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Simultaneous prediction intervals can be extended to using averages (means) in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a simultaneous prediction interval

that will contain a specified number of averages (based on which rule you choose) on each of r future sampling occassions, where each each average is based on w individual observations. For the function `predIntNormSimultaneous`, the argument `n.mean` corresponds to w.

The Form of a Prediction Interval for 1 Future Observation

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with parameters `mean`=μ and `sd`=σ. Also, let w denote the sample size associated with the future averages (i.e., `n.mean`=w). When $w = 1$, each average is really just a single observation, so in the rest of this help file the term "averages" will replace the phrase "observations or averages".

For a normal distribution, the form of a two-sided $(1 - \alpha)100\%$ prediction interval is:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future sampling occassions r, and the sample size associated with the future averages, w. Do not confuse the constant K (uppercase K) with the number of future averages k (lowercase k) in the k-of-m rule. The symbol K is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \qquad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \qquad (5)$$

The derivation of the constant K is explained in the help file for `predIntNormK`.

The Form of a Simultaneous Prediction Interval

For *simultaneous* prediction intervals, only lower (`pi.type="lower"`) and upper (`pi.type="upper"`) prediction intervals are available. Two-sided simultaneous prediction intervals were available in Versions 2.4.0 - 2.8.1 of *EnvStats*, but these prediction intervals were based on an incorrect algorithm for K.

Equations (4) and (5) above hold for simultaneous prediction intervals, but the derivation of the constant K is more difficult, and is explained in the help file for `predIntNormSimultaneousK`.

Prediction Intervals are Random Intervals

A prediction interval is a *random* interval; that is, the lower and/or upper bounds are random variables computed based on sample statistics in the baseline sample. Prior to taking one specific baseline sample, the probability that the prediction interval will perform according to the rule chosen is

$(1 - \alpha)100\%$. Once a specific baseline sample is taken and the prediction interval based on that sample is computed, the probability that that prediction interval will perform according to the rule chosen is not necessarily $(1 - \alpha)100\%$, but it should be close. See the help file for `predIntNorm` for more information.

Value

If `x` is a numeric vector, `predIntNormSimultaneous` returns a list of class `"estimate"` containing the estimated parameters, the prediction interval, and other information. See the help file for `estimate.object` for details.

If `x` is the result of calling an estimation function, `predIntNormSimultaneous` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as a component called `interval` containing the prediction interval information. If `x` already has a component called `interval`, this component is replaced with the prediction interval information.

Note

Motivation

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional α-level (e.g., 0.05 or 0.01) for each test leads to a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others.

Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual α-level at each well is equal to the FWFRP divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-m rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.

The k-of-m Rule

For the k-of-m rule, Davis and McNichols (1987) give tables with "optimal" choices of k (in terms of best power for a given overall confidence level) for selected values of m, r, and n. They found that the optimal ratios of k to m (i.e., k/m) are generally small, in the range of 15-50%.

The California Rule

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay's Code of California Regulations, 1991). The California code mandates a "California" rule with $m \geq 3$. The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a k-of-m rule with $k = 1$ and $m = 3$, a monitoring location will pass if the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with $m = 3$, either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPR is kept constant, then the California rule offers little increased power compared to the k-of-m rule, and can actually decrease the power of detecting contamination.

The Modified California Rule

The Modified California Rule was proposed as a compromise between a 1-of-m rule and the California rule. For a given FWFPR, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-m rule.

Different Notations Between Different References

For the k-of-m rule described in this help file, both Davis and McNichols (1987) and USEPA (2009, Chapter 19) use the variable p instead of k to represent the minimum number of future observations the interval should contain on each of the r sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of K for both k-of-m rules and California rules. Gibbons et al.'s notation reverses the meaning of k and r compared to the notation used in this help file. That is, in Gibbons et al.'s notation, k represents the number of future sampling occasions or monitoring wells, and r represents the minimum number of observations the interval should contain on each sampling occasion.

USEPA (2009, Chapter 19) uses p in place of k.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Barclay's California Code of Regulations. (1991). Title 22, Section 66264.97 [concerning hazardous waste facilities] and Title 23, Section 2550.7(e)(8) [concerning solid waste facilities]. Barclay's Law Publishers, San Francisco, CA.

Davis, C.B. (1998a). *Ground-Water Statistics & Regulations: Principles, Progress and Problems*. Second Edition. Environmetrics & Statistics Limited, Henderson, NV.

Davis, C.B. (1998b). Personal Communication, September 3, 1998.

Davis, C.B., and R.J. McNichols. (1987). One-sided Intervals for at Least p of m Observations from a Normal Population on Each of r Future Occasions. *Technometrics* **29**, 359–370.

Fertig, K.W., and N.R. Mann. (1977). One-Sided Prediction Intervals for at Least p Out of m Future Observations From a Normal Population. *Technometrics* **19**, 167–177.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J. (1969). Factors for Calculating Two-Sided Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **64**(327), 878-898.

Hahn, G.J. (1970a). Additional Factors for Calculating Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **65**(332), 1668-1676.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178-188.

Hall, I.J., and R.R. Prairie. (1973). One-Sided Prediction Intervals to Contain at Least m Out of k Future Observations. *Technometrics* **15**, 897–914.

Millard, S.P. (1987). Environmental Monitoring, Statistics, and the Law: Room for Improvement (with Comment). *The American Statistician* **41**(4), 249–259.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

`predIntNormSimultaneousK`, `predIntNormSimultaneousTestPower`, `predIntNorm`, `predIntLnormSimultaneous`, `tolIntNorm`, Normal, `estimate.object`, `enorm`

Examples

```
# Generate 8 observations from a normal distribution with parameters
# mean=10 and sd=2, then use predIntNormSimultaneous to estimate the
# mean and standard deviation of the true distribution and construct an
# upper 95% prediction interval to contain at least 1 out of the next
# 3 observations.
```

```
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(479)
dat <- rnorm(8, mean = 10, sd = 2)

predIntNormSimultaneous(dat, k = 1, m = 3)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 10.269773
#                                 sd   =  2.210246
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     8
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                95%
#
#Minimum Number of
#Future Observations
#Interval Should Contain:         1
#
#Total Number of
#Future Observations:             3
#
#Prediction Interval:             LPL =    -Inf
#                                 UPL = 11.4021

#----------

# Repeat the above example, but do it in two steps.  First create a list called
# est.list containing information about the estimated parameters, then create the
# prediction interval.

est.list <- enorm(dat)
est.list

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 10.269773
#                                 sd   =  2.210246
```

```
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     8


predIntNormSimultaneous(est.list, k = 1, m = 3)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 10.269773
#                                 sd   =  2.210246
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     8
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                95%
#
#Minimum Number of
#Future Observations
#Interval Should Contain:         1
#
#Total Number of
#Future Observations:             3
#
#Prediction Interval:             LPL =    -Inf
#                                 UPL = 11.4021

#----------

# Compare the 95% 1-of-3 upper prediction interval to the California and
# Modified California prediction intervals.  Note that the upper prediction
# bound for the Modified California rule is between the bound for the
# 1-of-3 rule bound and the bound for the California rule.

predIntNormSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
#    UPL
#11.4021

predIntNormSimultaneous(dat, m = 3, rule = "CA")$interval$limits["UPL"]
#     UPL
```

```
#13.03717

predIntNormSimultaneous(dat, rule = "Modified.CA")$interval$limits["UPL"]
#     UPL
#12.12201

#----------

# Show how the upper bound on an upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions r increases.  Here, we'll use the
# 1-of-3 rule.

predIntNormSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
#    UPL
#11.4021

predIntNormSimultaneous(dat, k = 1, m = 3, r = 10)$interval$limits["UPL"]
#     UPL
#13.28234

#----------

# Compare the upper simultaneous prediction limit for the 1-of-3 rule
# based on individual observations versus based on means of order 4.

predIntNormSimultaneous(dat, k = 1, m = 3)$interval$limits["UPL"]
#    UPL
#11.4021

predIntNormSimultaneous(dat, n.mean = 4, k = 1,
  m = 3)$interval$limits["UPL"]
#     UPL
#11.26157

#==========

# Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an
# upper simultaneous prediction limit for the 1-of-3 rule for
# r = 2 future sampling occasions.  The data for this example are
# stored in EPA.09.Ex.19.1.sulfate.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 25 to use to construct the prediction limit.

# There are 50 compliance wells and we will monitor 10 different
# constituents at each well at each of the r=2 future sampling
# occasions.  To determine the confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the individual Type I Error level at each well to

# 1 - (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))
```

```
# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))

# where SWFPR = site-wide false positive rate.  For this example, we
# will set SWFPR = 0.1.  Thus, the confidence level is given by:

nc <- 10
nw <- 50
SWFPR <- 0.1
conf.level <- (1 - SWFPR)^(1 / (nc * nw))

conf.level
#[1] 0.9997893

#----------

# Look at the data:

names(EPA.09.Ex.19.1.sulfate.df)
#[1] "Well"                "Month"               "Day"
#[4] "Year"                "Date"                "Sulfate.mg.per.l"
#[7] "log.Sulfate.mg.per.l"

EPA.09.Ex.19.1.sulfate.df[,
  c("Well", "Date", "Sulfate.mg.per.l", "log.Sulfate.mg.per.l")]

#    Well       Date Sulfate.mg.per.l log.Sulfate.mg.per.l
#1  GW-01 1999-07-08             63.0             4.143135
#2  GW-01 1999-09-12             51.0             3.931826
#3  GW-01 1999-10-16             60.0             4.094345
#4  GW-01 1999-11-02             86.0             4.454347
#5  GW-04 1999-07-09            104.0             4.644391
#6  GW-04 1999-09-14            102.0             4.624973
#7  GW-04 1999-10-12             84.0             4.430817
#8  GW-04 1999-11-15             72.0             4.276666
#9  GW-08 1997-10-12             31.0             3.433987
#10 GW-08 1997-11-16             84.0             4.430817
#11 GW-08 1998-01-28             65.0             4.174387
#12 GW-08 1999-04-20             41.0             3.713572
#13 GW-08 2002-06-04             51.8             3.947390
#14 GW-08 2002-09-16             57.5             4.051785
#15 GW-08 2002-12-02             66.8             4.201703
#16 GW-08 2003-03-24             87.1             4.467057
#17 GW-09 1997-10-16             59.0             4.077537
#18 GW-09 1998-01-28             85.0             4.442651
#19 GW-09 1998-04-12             75.0             4.317488
#20 GW-09 1998-07-12             99.0             4.595120
#21 GW-09 2000-01-30             75.8             4.328098
#22 GW-09 2000-04-24             82.5             4.412798
#23 GW-09 2000-10-24             85.5             4.448516
#24 GW-09 2002-12-01            188.0             5.236442
#25 GW-09 2003-03-24            150.0             5.010635
```

```
# Construct the upper simultaneous prediction limit for the
# 1-of-3 plan based on the log-transformed sulfate data

log.Sulfate <- EPA.09.Ex.19.1.sulfate.df$log.Sulfate.mg.per.l

pred.int.list.log <-
  predIntNormSimultaneous(x = log.Sulfate, k = 1, m = 3, r = 2,
    rule = "k.of.m", pi.type = "upper", conf.level = conf.level)

pred.int.list.log

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 4.3156194
#                                 sd   = 0.3756697
#
#Estimation Method:               mvue
#
#Data:                            log.Sulfate
#
#Sample Size:                     25
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                99.97893%
#
#Minimum Number of
#Future Observations
#Interval Should Contain
#(per Sampling Occasion):         1
#
#Total Number of
#Future Observations
#(per Sampling Occasion):         3
#
#Number of Future
#Sampling Occasions:              2
#
#Prediction Interval:             LPL =     -Inf
#                                 UPL = 5.072355


# Now exponentiate the prediction interval to get the limit on
# the original scale

exp(pred.int.list.log$interval$limits["UPL"])
#     UPL
```

```
#159.5497

#==========

## Not run:
# Try to compute a two-sided simultaneous prediction interval:

predIntNormSimultaneous(x = log.Sulfate, k = 1, m = 3, r = 2,
    rule = "k.of.m", pi.type = "two-sided", conf.level = conf.level)
#Error in predIntNormSimultaneous(x = log.Sulfate, k = 1, m = 3, r = 2,  :
#  Two-sided simultaneous prediction intervals are not currently available.
# NOTE: Two-sided simultaneous prediction intervals computed using
# Versions 2.4.0 - 2.8.1 of EnvStats are *NOT* valid.

## End(Not run)

#==========

# Cleanup
#--------

rm(dat, est.list, nc, nw, SWFPR, conf.level, log.Sulfate,
  pred.int.list.log)
```

---

predIntNormSimultaneousK

Compute the Value of K for a Simultaneous Prediction Interval for a Normal Distribution

---

Description

Compute the value of K (the multiplier of estimated standard deviation) used to construct a simultaneous prediction interval based on data from a normal distribution. The function predIntNormSimultaneousK is called by predIntNormSimultaneous.

Usage

```
predIntNormSimultaneousK(n, df = n - 1, n.mean = 1, k = 1, m = 2, r = 1,
  rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
  K.tol = .Machine$double.eps^0.5, integrate.args.list = NULL)
```

Arguments

n	a positive integer greater than 2 indicating the sample size upon which the prediction interval is based.
df	the degrees of freedom associated with the prediction interval. The default is df=n-1.

`n.mean` positive integer specifying the sample size associated with the future averages. The default value is `n.mean=1` (i.e., individual observations). Note that all future averages must be based on the same sample size.

`k` for the k-of-m rule (`rule="k.of.m"`), a positive integer specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`. This argument is ignored when the argument `rule` is not equal to `"k.of.m"`.

`m` positive integer specifying the maximum number of future observations (or averages) on one future sampling "occasion". The default value is `m=2`, except when `rule="Modified.CA"`, in which case this argument is ignored and `m` is automatically set equal to `4`.

`r` positive integer specifying the number of future sampling "occasions". The default value is `r=1`.

`rule` character string specifying which rule to use. The possible values are `"k.of.m"` (k-of-m rule; the default), `"CA"` (California rule), and `"Modified.CA"` (modified California rule). See the DETAILS section below for more information.

`delta.over.sigma`
numeric scalar indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is `delta.over.sigma=0`.

`pi.type` character string indicating what kind of prediction interval to compute. The possible values are `pi.type="upper"` (the default), and `pi.type="lower"`. **NOTE:** In Versions 2.4.0 - 2.8.1 of *EnvStats*, the value `pi.type="two-sided"` was allowed, but these two-sided simultaneous prediction intervals were based on faulty assumptions and were **NOT** valid.

`conf.level` a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

`K.tol` numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is `K.tol=.Machine$double.eps^(1/2)`. For many applications, the value of K needs to be known only to the second decimal place, in which case setting `K.tol=1e-4` will speed up computation a bit.

`integrate.args.list`
a list of arguments to supply to the `integrate` function. The default value is `integrate.args.list=NULL` which means that the default values of `integrate` are used.

Details

What is a Simultaneous Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is called

the confidence coefficient or confidence level associated with the prediction interval. The function predIntNorm computes a standard prediction interval based on a sample from a normal distribution.

The function predIntNormSimultaneous computes a simultaneous prediction interval that will contain a certain number of future observations with probability $(1 - \alpha)100\%$ for each of r future sampling "occasions", where r is some pre-specified positive integer. The quantity r may refer to r distinct future sampling occasions in time, or it may for example refer to sampling at r distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the r distinct locations.

The function predIntNormSimultaneous computes a simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (rule="k.of.m"), at least k of the next m future observations will fall in the prediction interval with probability $(1 - \alpha)100\%$ on each of the r future sampling occasions. If obserations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: When $k = m$ and $r = 1$, the results of predIntNormSimultaneous are equivalent to the results of predIntNorm.
- For the California rule (rule="CA"), with probability $(1 - \alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m - 1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m - 1$ more observations must be taken.
- For the Modified California rule (rule="Modified.CA"), with probability $(1 - \alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Simultaneous prediction intervals can be extended to using averages (means) in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a simultaneous prediction interval that will contain a specified number of averages (based on which rule you choose) on each of r future sampling occassions, where each each average is based on w individual observations. For the functions predIntNormSimultaneous and predIntNormSimultaneousK, the argument n.mean corresponds to w.

The Form of a Prediction Interval for 1 Future Observation

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with parameters mean=μ and sd=σ. Also, let w denote the sample size associated with the future averages (i.e., n.mean=w). When $w = 1$, each average is really just a single observation, so in the rest of this help file the term "averages" will sometimes replace the phrase "observations or averages".

For a normal distribution, the form of a two-sided $(1 - \alpha)100\%$ simultaneous prediction interval is:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future sampling occassions r, and the sample size associated with the future averages, w. Do not confuse the constant K (uppercase K) with the number of future averages k (lowercase k) in the k-of-m rule. The symbol K is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \qquad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \qquad (5)$$

The derivation of the constant K for 1 future observation is explained in the help file for `predIntNormK`.

The Form of a Simultaneous Prediction Interval

For **simultaneous** prediction intervals, only lower (`pi.type="lower"`) and upper (`pi.type="upper"`) prediction intervals are available. Two-sided simultaneous prediction intervals were available in Versions 2.4.0 - 2.8.1 of *EnvStats*, but these prediction intervals were based on an incorrect algorithm for K.

Equations (4) and (5) above hold for simultaneous prediction intervals, but the derivation of the constant K is more difficult, and is explained below.

The Derivation of K for Future Observations

First we will show the derivation based on future observations (i.e., $w = 1$, `n.mean=1`), and then extend the formulas to future averages.

The Derivation of K for the k-of-m Rule (`rule="k.of.m"`)

For the k-of-m rule (`rule="k.of.m"`) with $w = 1$ (i.e., `n.mean=1`), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: When $k = m$ and $r = 1$, this kind of simultaneous prediction interval becomes the same as a standard prediction interval for the next k observations (see `predIntNorm`).

For the case when $r = 1$ future sampling occasion, both Hall and Prairie (1973) and Fertig and Mann (1977) discuss the derivation of K. Davis and McNichols (1987) extend the derivation to the case where r is a positive integer. They show that for a one-sided upper prediction interval (`pi.type="upper"`), the probability p that at least k of the next m future observations will be contained in the interval given in Equation (5) above, for each of r future sampling occasions, is given by:

$$p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}[\Phi^{-1}(v) + \frac{\Delta}{\sigma}])r[I(v; k, m+1-k)]^{r-1}[\frac{v^{k-1}(1-v)^{m-k}}{B(k, m+1-k)}]dv \qquad (6)$$

where $T(x;\nu,\delta)$ denotes the cdf of the non-central Student's t-distribution with parameters df=ν and ncp=δ evaluated at x; $\Phi(x)$ denotes the cdf of the standard normal distribution evaluated at x; $I(x;\nu,\omega)$ denotes the cdf of the beta distribution with parameters shape1=ν and shape2=ω; and $B(\nu,\omega)$ denotes the value of the beta function with parameters a=ν and b=ω.

The quantity Δ (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation of both of these populations. Usually you assume $\Delta = 0$ unless you are interested in computing the power of the rule to detect a change in means between the populations (see predIntNormSimultaneousTestPower).

For given values of the confidence level (p), sample size (n), minimum number of future observations to be contained in the interval per sampling occasion (k), number of future observations per sampling occasion (m), and number of future sampling occasions (r), Equation (6) can be solved for K. The function predIntNormSimultaneousK uses the R function nlminb to solve Equation (6) for K.

When pi.type="lower", the same value of K is used as when pi.type="upper", but Equation (4) is used to construct the prediction interval.

The Derivation of K for the California Rule (rule="CA")

For the California rule (rule="CA"), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.

The formula for K is the same as for the k-of-m rule, except that Equation (6) becomes the following (Davis, 1998b):

$$p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}[\Phi^{-1}(v)+\frac{\Delta}{\sigma}])r\{v[1+v^{m-2}(1-v)]\}^{r-1}[1+v^{m-2}(m-1-mv)]dv \qquad (7)$$

The Derivation of K for the Modified California Rule (rule="Modified.CA")

For the Modified California rule (rule="Modified.CA"), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

The formula for K is the same as for the k-of-m rule, except that Equation (6) becomes the following (Davis, 1998b):

$$p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}[\Phi^{-1}(v)+\frac{\Delta}{\sigma}])r\{v[1+v(3-v[5-2v])]\}^{r-1}\{1+v[6-v(15-8v)]\}dv \qquad (8)$$

The Derivation of K for Future Means

For each of the above rules, if we are interested in using averages instead of single observations,

with $w \geq 1$ (i.e., `n.mean`≥ 1), the first term in the integral in Equations (6)-(8) that involves the cdf of the non-central Student's t-distribution becomes:

$$T(\sqrt{n}K; n-1, \frac{\sqrt{n}}{\sqrt{w}}[\Phi^{-1}(v) + \frac{\sqrt{w}\Delta}{\sigma}]) \qquad (9)$$

Value

A numeric scalar equal to K, the multiplier of estimated standard deviation that is used to construct the simultaneous prediction interval.

Note

Motivation

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities.

One of the main statistical problems that plague groundwater monitoring programs at hazardous and solid waste facilities is the requirement of testing several wells and several constituents at each well on each sampling occasion. This is an obvious multiple comparisons problem, and the naive approach of using a standard t-test at a conventional α-level (e.g., 0.05 or 0.01) for each test leads to a very high probability of at least one significant result on each sampling occasion, when in fact no contamination has occurred. This problem was pointed out years ago by Millard (1987) and others.

Davis and McNichols (1987) proposed simultaneous prediction intervals as a way of controlling the facility-wide false positive rate (FWFPR) while maintaining adequate power to detect contamination in the groundwater. Because of the ubiquitous presence of spatial variability, it is usually best to use simultaneous prediction intervals at each well (Davis, 1998a). That is, by constructing prediction intervals based on background (pre-landfill) data on each well, and comparing future observations at a well to the prediction interval for that particular well. In each of these cases, the individual α-level at each well is equal to the FWFRP divided by the product of the number of wells and constituents.

Often, observations at downgradient wells are not available prior to the construction and operation of the landfill. In this case, upgradient well data can be combined to create a background prediction interval, and observations at each downgradient well can be compared to this prediction interval. If spatial variability is present and a major source of variation, however, this method is not really valid (Davis, 1994; Davis, 1998a).

Chapter 19 of USEPA (2009) contains an extensive discussion of using the 1-of-m rule and the Modified California rule.

Chapters 1 and 3 of Gibbons et al. (2009) discuss simultaneous prediction intervals for the normal and lognormal distributions, respectively.

The k-of-m Rule

For the k-of-m rule, Davis and McNichols (1987) give tables with "optimal" choices of k (in terms of best power for a given overall confidence level) for selected values of m, r, and n. They found that the optimal ratios of k to m (i.e., k/m) are generally small, in the range of 15-50%.

The California Rule

The California rule was mandated in that state for groundwater monitoring at waste disposal facilities when resampling verification is part of the statistical program (Barclay's Code of California Regulations, 1991). The California code mandates a "California" rule with $m \geq 3$. The motivation for this rule may have been a desire to have a majority of the observations in bounds (Davis, 1998a). For example, for a k-of-m rule with $k = 1$ and $m = 3$, a monitoring location will pass if the first observation is out of bounds, the second resample is out of bounds, but the last resample is in bounds, so that 2 out of 3 observations are out of bounds. For the California rule with $m = 3$, either the first observation must be in bounds, or the next 2 observations must be in bounds in order for the monitoring location to pass.

Davis (1998a) states that if the FWFPR is kept constant, then the California rule offers little increased power compared to the k-of-m rule, and can actually decrease the power of detecting contamination.

The Modified California Rule

The Modified California Rule was proposed as a compromise between a 1-of-m rule and the California rule. For a given FWFPR, the Modified California rule achieves better power than the California rule, and still requires at least as many observations in bounds as out of bounds, unlike a 1-of-m rule.

Different Notations Between Different References

For the k-of-m rule described in this help file, both Davis and McNichols (1987) and USEPA (2009, Chapter 19) use the variable p instead of k to represent the minimum number of future observations the interval should contain on each of the r sampling occasions.

Gibbons et al. (2009, Chapter 1) presents extensive lists of the value of K for both k-of-m rules and California rules. Gibbons et al.'s notation reverses the meaning of k and r compared to the notation used in this help file. That is, in Gibbons et al.'s notation, k represents the number of future sampling occasions or monitoring wells, and r represents the minimum number of observations the interval should contain on each sampling occasion.

USEPA (2009, Chapter 19) uses p in place of k.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Barclay's California Code of Regulations. (1991). Title 22, Section 66264.97 [concerning hazardous waste facilities] and Title 23, Section 2550.7(e)(8) [concerning solid waste facilities]. Barclay's Law Publishers, San Francisco, CA.

Davis, C.B. (1998a). *Ground-Water Statistics & Regulations: Principles, Progress and Problems*. Second Edition. Environmetrics & Statistics Limited, Henderson, NV.

Davis, C.B. (1998b). Personal Communication, September 3, 1998.

Davis, C.B., and R.J. McNichols. (1987). One-sided Intervals for at Least p of m Observations from a Normal Population on Each of r Future Occasions. *Technometrics* **29**, 359–370.

Fertig, K.W., and N.R. Mann. (1977). One-Sided Prediction Intervals for at Least p Out of m Future Observations From a Normal Population. *Technometrics* **19**, 167–177.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J. (1969). Factors for Calculating Two-Sided Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **64**(327), 878-898.

Hahn, G.J. (1970a). Additional Factors for Calculating Prediction Intervals for Samples from a Normal Distribution. *Journal of the American Statistical Association* **65**(332), 1668-1676.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178-188.

Hall, I.J., and R.R. Prairie. (1973). One-Sided Prediction Intervals to Contain at Least m Out of k Future Observations. *Technometrics* **15**, 897–914.

Millard, S.P. (1987). Environmental Monitoring, Statistics, and the Law: Room for Improvement (with Comment). *The American Statistician* **41**(4), 249–259.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

predIntNormSimultaneous, predIntNormSimultaneousTestPower, predIntNorm, predIntNormK, predIntLnormSimultaneous, tolIntNorm, Normal, estimate.object, enorm

Examples

```
# Compute the value of K for an upper 95% simultaneous prediction
# interval to contain at least 1 out of the next 3 observations
# given a background sample size of n=8.

predIntNormSimultaneousK(n = 8, k = 1, m = 3)
#[1] 0.5123091

#----------

# Compare the value of K for a 95% 1-of-3 upper prediction interval to
# the value for the California and Modified California rules.
```

```
# Note that the value of K for the Modified California rule is between
# the value of K for the 1-of-3 rule and the California rule.

predIntNormSimultaneousK(n = 8, k = 1, m = 3)
#[1] 0.5123091

predIntNormSimultaneousK(n = 8, m = 3, rule = "CA")
#[1] 1.252077

predIntNormSimultaneousK(n = 8, rule = "Modified.CA")
#[1] 0.8380233

#----------

# Show how the value of K for an upper 95% simultaneous prediction
# limit increases as the number of future sampling occasions r increases.
# Here, we'll use the 1-of-3 rule.

predIntNormSimultaneousK(n = 8, k = 1, m = 3)
#[1] 0.5123091


predIntNormSimultaneousK(n = 8, k = 1, m = 3, r = 10)
#[1] 1.363002

#==========

# Example 19-1 of USEPA (2009, p. 19-17) shows how to compute an
# upper simultaneous prediction limit for the 1-of-3 rule for
# r = 2 future sampling occasions.  The data for this example are
# stored in EPA.09.Ex.19.1.sulfate.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 25 to use to construct the prediction limit.

# There are 50 compliance wells and we will monitor 10 different
# constituents at each well at each of the r=2 future sampling
# occasions.  To determine the confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the individual Type I Error level at each well to

# 1 - (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / (Number of Constituents * Number of Wells))

# where SWFPR = site-wide false positive rate.  For this example, we
# will set SWFPR = 0.1.  Thus, the confidence level is given by:

nc <- 10
nw <- 50
```

```
SWFPR <- 0.1
conf.level <- (1 - SWFPR)^(1 / (nc * nw))

conf.level
#[1] 0.9997893

#----------

# Compute the value of K for the upper simultaneous prediction
# limit for the 1-of-3 plan.

predIntNormSimultaneousK(n = 25, k = 1, m = 3, r = 2,
  rule = "k.of.m", pi.type = "upper", conf.level = conf.level)
#[1] 2.014365


#==========

## Not run:
# Try to compute K for a two-sided simultaneous prediction interval:

predIntNormSimultaneousK(n = 25, k = 1, m = 3, r = 2,
 rule = "k.of.m", pi.type = "two-sided", conf.level = conf.level)
#Error in predIntNormSimultaneousK(n = 25, k = 1, m = 3, r = 2, rule = "k.of.m",  :
#  Two-sided simultaneous prediction intervals are not currently available.
# NOTE: Two-sided simultaneous prediction intervals computed using
# Versions 2.4.0 - 2.8.1 of EnvStats are *NOT* valid.

## End(Not run)

#==========

# Cleanup
#--------

rm(nc, nw, SWFPR, conf.level)
```

---

predIntNormSimultaneousTestPower

Probability That at Least One Set of Future Observations Violates the Given Rule Based on a Simultaneous Prediction Interval for a Normal Distribution

---

Description

Compute the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for the next r future sampling occasions for a normal distribution. The three possible rules are: k-of-m, California, or Modified California.

Usage

```
predIntNormSimultaneousTestPower(n, df = n - 1, n.mean = 1, k = 1, m = 2, r = 1,
  rule = "k.of.m", delta.over.sigma = 0, pi.type = "upper", conf.level = 0.95,
  r.shifted = r, K.tol = .Machine$double.eps^0.5, integrate.args.list = NULL)
```

Arguments

n	vector of positive integers greater than 2 indicating the sample size upon which the prediction interval is based.
df	vector of positive integers indicating the degrees of freedom associated with the sample size. The default value is df=n-1.
n.mean	positive integer specifying the sample size associated with the future averages. The default value is n.mean=1 (i.e., individual observations). Note that all future averages must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), vector of positive integers specifying the minimum number of observations (or averages) out of m observations (or averages) (all obtained on one future sampling "occassion") the prediction interval should contain with confidence level conf.level. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	vector of positive integers specifying the maximum number of future observations (or averages) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	vector of positive integers specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.
delta.over.sigma	numeric vector indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is delta.over.sigma=0.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are pi.type="upper" (the default), and pi.type="lower".
conf.level	vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is conf.level=0.95.
r.shifted	vector of positive integers specifying the number of future sampling occasions for which the scaled mean is shifted by Δ/σ. All values must be integeters between 1 and the corresponding element of r. The default value is r.shifted=r.
K.tol	numeric scalar indicating the tolerance to use in the nonlinear search algorithm to compute K. The default value is K.tol=.Machine$double.eps^(1/2). For many applications, the value of K needs to be known only to the second decimal place, in which case setting K.tol=1e-4 will speed up computation a bit.

integrate.args.list
: a list of arguments to supply to the integrate function. The default value is integrate.args.list=NULL which means that the default values of integrate are used.

Details

What is a Simultaneous Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function predIntNorm computes a standard prediction interval based on a sample from a normal distribution.

The function predIntNormSimultaneous computes a simultaneous prediction interval that will contain a certain number of future observations with probability $(1-\alpha)100\%$ for each of r future sampling "occasions", where r is some pre-specified positive integer. The quantity r may refer to r distinct future sampling occasions in time, or it may for example refer to sampling at r distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the r distinct locations.

The function predIntNormSimultaneous computes a simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (rule="k.of.m"), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If obserations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: When $k = m$ and $r = 1$, the results of predIntNormSimultaneous are equivalent to the results of predIntNorm.
- For the California rule (rule="CA"), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.
- For the Modified California rule (rule="Modified.CA"), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Simultaneous prediction intervals can be extended to using averages (means) in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a simultaneous prediction interval that will contain a specified number of averages (based on which rule you choose) on each of r future sampling occassions, where each each average is based on w individual observations. For the function predIntNormSimultaneous, the argument n.mean corresponds to w.

The Form of a Prediction Interval

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with parameters mean=μ and sd=σ. Also, let w denote the sample size associated with the future averages (i.e.,

n.mean=w). When $w = 1$, each average is really just a single observation, so in the rest of this help file the term "averages" will replace the phrase "observations or averages".

For a normal distribution, the form of a two-sided $(1 - \alpha)100\%$ prediction interval is:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future sampling occassions r, and the sample size associated with the future averages, w. Do not confuse the constant K (uppercase K) with the number of future averages k (lowercase k) in the k-of-m rule. The symbol K is used here to be consistent with the notation used for tolerance intervals (see tolIntNorm).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \qquad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \qquad (5)$$

Note: For simultaneous prediction intervals, only lower (pi.type="lower") and upper (pi.type="upper") prediction intervals are available.

The derivation of the constant K is explained in the help file for predIntNormSimultaneousK.

Computing Power

The "power" is defined as the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for the next r future sampling occasions, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval.

The quantity Δ (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation of both of these populations. The argument delta.over.sigma corresponds to the quantity Δ/σ.

Power Based on the k-of-m Rule (rule="k.of.m")

For the k-of-m rule (rule="k.of.m") with $w = 1$ (i.e., n.mean=1), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion,

up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: When $k = m$ and $r = 1$, this kind of simultaneous prediction interval becomes the same as a standard prediction interval for the next k observations (see `predIntNorm`).

Davis and McNichols (1987) show that for a one-sided upper prediction interval (`pi.type="upper"`), the probability p that at least k of the next m future observations will be contained in the interval given in Equation (5) above, for each of r future sampling occasions, is given by:

$$p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}[\Phi^{-1}(v) + \frac{\Delta}{\sigma}])r[I(v; k, m+1-k)]^{r-1}[\frac{v^{k-1}(1-v)^{m-k}}{B(k, m+1-k)}]dv \quad (6)$$

where $T(x; \nu, \delta)$ denotes the cdf of the non-central Student's t-distribution with parameters `df`=ν and `ncp`=δ evaluated at x; $\Phi(x)$ denotes the cdf of the standard normal distribution evaluated at x; $I(x; \nu, \omega)$ denotes the cdf of the beta distribution with parameters `shape1`=ν and `shape2`=ω; and $B(\nu, \omega)$ denotes the value of the beta function with parameters `a`=ν and `b`=ω.

The quantity Δ (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation of both of these populations. Usually you assume $\Delta = 0$ unless you are interested in computing the power of the rule to detect a change in means between the populations, as we are here.

If we are interested in using averages instead of single observations, with $w \geq 1$ (i.e., `n.mean`≥ 1), the first term in the integral in Equation (6) that involves the cdf of the non-central Student's t-distribution becomes:

$$T(\sqrt{n}K; n-1, \frac{\sqrt{n}}{\sqrt{w}}[\Phi^{-1}(v) + \frac{\sqrt{w}\Delta}{\sigma}]) \quad (7)$$

For a given confidence level $(1-\alpha)100\%$, the power of the rule to detect a change in means is simply given by:

$$Power = 1 - p \quad (8)$$

where p is defined in Equation (6) above using the value of K that corresponds to $\Delta/\sigma = 0$. Thus, when the argument `delta.over.sigma=0`, the value of p is $1-\alpha$ and the power is simply $\alpha 100\%$. As `delta.over.sigma` increases above 0, the power increases.

When `pi.type="lower"`, the same value of `K` is used as when `pi.type="upper"`, but Equation (4) is used to construct the prediction interval. Thus, the power increases as `delta.over.sigma` decreases below 0.

Power Based on the California Rule (`rule="CA"`)

For the California rule (`rule="CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.

The derivation of the power is the same as for the k-of-m rule, except that Equation (6) becomes the following (Davis, 1998b):

$$p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}[\Phi^{-1}(v)+\frac{\Delta}{\sigma}])r\{v[1+v^{m-2}(1-v)]\}^{r-1}[1+v^{m-2}(m-1-mv)]dv \quad (9)$$

Power Based on the Modified California Rule (`rule="Modified.CA"`)
For the Modified California rule (`rule="Modified.CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

The derivation of the power is the same as for the k-of-m rule, except that Equation (6) becomes the following (Davis, 1998b):

$$p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}[\Phi^{-1}(v)+\frac{\Delta}{\sigma}])r\{v[1+v(3-v[5-2v])]\}^{r-1}\{1+v[6-v(15-8v)]\}dv \qquad (10)$$

Value

vector of values between 0 and 1 equal to the probability that the rule will be violated.

Note

See the help file for `predIntNormSimultaneous`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions `predIntNormSimultaneousTestPower` and `plotPredIntNormSimultaneousTestPowerCurve` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNormSimultaneous`.

See Also

`predIntNormSimultaneous`, `predIntNormSimultaneousK`, `plotPredIntNormSimultaneousTestPowerCurve`, `predIntNorm`, `predIntNormK`, `predIntNormTestPower`, Prediction Intervals, Normal.

Examples

```
# For the k-of-m rule with n=4, k=1, m=3, and r=1, show how the power increases
# as delta.over.sigma increases. Assume a 95% upper prediction interval.

predIntNormSimultaneousTestPower(n = 4, m = 3, delta.over.sigma = 0:2)
#[1] 0.0500000 0.2954156 0.7008558
```

```
#----------

# Look at how the power increases with sample size for an upper one-sided
# prediction interval using the k-of-m rule with k=1, m=3, r=20,
# delta.over.sigma=2, and a confidence level of 95%.

predIntNormSimultaneousTestPower(n = c(4, 8), m = 3, r = 20, delta.over.sigma = 2)
#[1] 0.6075972 0.9240924

#----------

# Compare the power for the 1-of-3 rule with the power for the California and
# Modified California rules, based on a 95% upper prediction interval and
# delta.over.sigma=2.  Assume a sample size of n=8.  Note that in this case the
# power for the Modified California rule is greater than the power for the
# 1-of-3 rule and California rule.

predIntNormSimultaneousTestPower(n = 8, k = 1, m = 3, delta.over.sigma = 2)
#[1] 0.788171

predIntNormSimultaneousTestPower(n = 8, m = 3, rule = "CA", delta.over.sigma = 2)
#[1] 0.7160434

predIntNormSimultaneousTestPower(n = 8, rule = "Modified.CA", delta.over.sigma = 2)
#[1] 0.8143687

#----------

# Show how the power for an upper 95% simultaneous prediction limit increases
# as the number of future sampling occasions r increases.  Here, we'll use the
# 1-of-3 rule with n=8 and delta.over.sigma=1.

predIntNormSimultaneousTestPower(n = 8, k = 1, m = 3, r=c(1, 2, 5, 10),
  delta.over.sigma = 1)
#[1] 0.3492512 0.4032111 0.4503603 0.4633773

#==========

# USEPA (2009) contains an example on page 19-23 that involves monitoring
# nw=100 compliance wells at a large facility with minimal natural spatial
# variation every 6 months for nc=20 separate chemicals.
# There are n=25 background measurements for each chemical to use to create
# simultaneous prediction intervals.  We would like to determine which kind of
# resampling plan based on normal distribution simultaneous prediction intervals to
# use (1-of-m, 1-of-m based on means, or Modified California) in order to have
# adequate power of detecting an increase in chemical concentration at any of the
# 100 wells while at the same time maintaining a site-wide false positive rate
# (SWFPR) of 10% per year over all 4,000 comparisons
# (100 wells x 20 chemicals x semi-annual sampling).

# The function predIntNormSimultaneousTestPower includes the argument "r"
# that is the number of future sampling occasions (r=2 in this case because
# we are performing semi-annual sampling), so to compute the individual test
```

```
# Type I error level alpha.test (and thus the individual test confidence level),
# we only need to worry about the number of wells (100) and the number of
# constituents (20): alpha.test = 1-(1-alpha)^(1/(nw x nc)).  The individual
# confidence level is simply 1-alpha.test.  Plugging in 0.1 for alpha,
# 100 for nw, and 20 for nc yields an individual test confidence level of
# 1-alpha.test = 0.9999473.

nc <- 20
nw <- 100
conf.level <- (1 - 0.1)^(1 / (nc * nw))
conf.level
#[1] 0.9999473

# Now we can compute the power of any particular sampling strategy using
# predIntNormSimultaneousTestPower.  For example, here is the power of
# detecting an increase of three standard deviations in concentration using
# the prediction interval based on the "1-of-2" resampling rule:

predIntNormSimultaneousTestPower(n = 25, k = 1, m = 2, r = 2, rule = "k.of.m",
  delta.over.sigma = 3,  pi.type = "upper", conf.level = conf.level)
#[1] 0.3900202

# The following commands will reproduce the table shown in Step 2 on page
# 19-23 of USEPA (2009).  Because these commands can take more than a few
# seconds to execute, we have commented them out here.  To run this example,
# just remove the pound signs (#) that are in front of R commands.

#rule.vec <- c(rep("k.of.m", 3), "Modified.CA",  rep("k.of.m", 3))

#m.vec <- c(2, 3, 4, 4, 1, 2, 1)

#n.mean.vec <- c(rep(1, 4), 2, 2, 3)

#n.scenarios <- length(rule.vec)

#K.vec <- numeric(n.scenarios)

#Power.vec <- numeric(n.scenarios)

#K.vec <- predIntNormSimultaneousK(n = 25, k = 1, m = m.vec,  n.mean = n.mean.vec,
#  r = 2, rule = rule.vec,  pi.type = "upper", conf.level = conf.level)

#Power.vec <- predIntNormSimultaneousTestPower(n = 25, k = 1, m = m.vec,
#  n.mean = n.mean.vec, r = 2, rule = rule.vec,  delta.over.sigma = 3,
#  pi.type = "upper",  conf.level = conf.level)

#Power.df <- data.frame(Rule = rule.vec, k = rep(1, n.scenarios),  m = m.vec,
#  N.Mean = n.mean.vec, K = round(K.vec, 2),  Power = round(Power.vec, 2),
#  Total.Samples = m.vec * n.mean.vec)

#Power.df

#         Rule k m N.Mean    K Power Total.Samples
```

```
#1      k.of.m 1 2      1 3.16  0.39            2
#2      k.of.m 1 3      1 2.33  0.65            3
#3      k.of.m 1 4      1 1.83  0.81            4
#4 Modified.CA 1 4      1 2.57  0.71            4
#5      k.of.m 1 1      2 3.62  0.41            2
#6      k.of.m 1 2      2 2.33  0.85            4
#7      k.of.m 1 1      3 2.99  0.71            3

# The above table shows the K-multipliers for each prediction interval, along with
# the power of detecting a change in concentration of three standard deviations at
# any of the 100 wells during the course of a year, for each of the sampling
# strategies considered.  The last three rows of the table correspond to sampling
# strategies that involve using the mean of two or three observations.

#==========

# Clean up
#---------
rm(nc, nw, conf.level, rule.vec, m.vec, n.mean.vec, n.scenarios, K.vec,
  Power.vec, Power.df)
```

`predIntNormTestPower` *Probability That at Least One Future Observation Falls Outside a Prediction Interval for a Normal Distribution*

Description

Compute the probability that at least one out of k future observations (or means) falls outside a prediction interval for k future observations (or means) for a normal distribution.

Usage

```
predIntNormTestPower(n, df = n - 1, n.mean = 1, k = 1, delta.over.sigma = 0,
  pi.type = "upper", conf.level = 0.95)
```

Arguments

`n`	vector of positive integers greater than 2 indicating the sample size upon which the prediction interval is based.
`df`	vector of positive integers indicating the degrees of freedom associated with the sample size. The default value is `df=n-1`.
`n.mean`	positive integer specifying the sample size associated with the future averages. The default value is `n.mean=1` (i.e., individual observations). Note that all future averages must be based on the same sample size.
`k`	vector of positive integers specifying the number of future observations that the prediction interval should contain with confidence level `conf.level`. The default value is `k=1`.

`delta.over.sigma`
: vector of numbers indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation for both populations. See the DETAILS section below for more information. The default value is `delta.over.sigma=0`.

`pi.type`
: character string indicating what kind of prediction interval to compute. The possible values are `pi.type="upper"` (the default), and `pi.type="lower"`.

`conf.level`
: numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.

Details

What is a Prediction Interval?

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations or averages from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is call the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNorm` computes a standard prediction interval based on a sample from a normal distribution. The function `predIntNormTestPower` computes the probability that at least one out of k future observations or averages will **not** be contained in the prediction interval, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval.

The Form of a Prediction Interval

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with parameters `mean`=μ and `sd`=σ. Also, let m denote the sample size associated with the k future averages (i.e., `n.mean`=m). When $m = 1$, each average is really just a single observation, so in the rest of this help file the term "averages" will replace the phrase "observations or averages".

For a normal distribution, the form of a two-sided $(1-\alpha)100\%$ prediction interval is:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, the number of future averages k, and the sample size associated with the future averages, m. Do not confuse the constant K (uppercase K) with the number of future averages k (lowercase k). The symbol K is used here to be consistent with the notation used for tolerance intervals (see `tolIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

$$[\bar{x} - Ks, \infty] \qquad (4)$$

and the form of a one-sided upper prediction interval is:

$$[-\infty, \bar{x} + Ks] \qquad (5)$$

but K differs for one-sided versus two-sided prediction intervals. The derivation of the constant K is explained in the help file for predIntNormK.

Computing Power

The "power" of the prediction interval is defined as the probability that at least one out of the k future observations or averages will **not** be contained in the prediction interval, where the population mean for the future observations is allowed to differ from the population mean for the observations used to construct the prediction interval. The probability p that all k future observations will be contained in a one-sided upper prediction interval (pi.type="upper") is given in Equation (6) of the help file for predIntNormSimultaneousK, where $k = m$ and $r = 1$:

$$p = \int_0^1 T(\sqrt{n}K; n-1, \sqrt{n}[\Phi^{-1}(v) + \frac{\Delta}{\sigma}])[\frac{v^{k-1}}{B(k,1)}]dv \qquad (6)$$

where $T(x; \nu, \delta)$ denotes the cdf of the non-central Student's t-distribution with parameters df=ν and ncp=δ evaluated at x; $\Phi(x)$ denotes the cdf of the standard normal distribution evaluated at x; and $B(\nu, \omega)$ denotes the value of the beta function with parameters a=ν and b=ω.

The quantity Δ (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation of both of these populations. Usually you assume $\Delta = 0$ unless you are interested in computing the power of the rule to detect a change in means between the populations, as we are here.

If we are interested in using averages instead of single observations, with $w \geq 1$ (i.e., n.mean≥ 1), the first term in the integral in Equation (6) that involves the cdf of the non-central Student's t-distribution becomes:

$$T(\sqrt{n}K; n-1, \frac{\sqrt{n}}{\sqrt{w}}[\Phi^{-1}(v) + \frac{\sqrt{w}\Delta}{\sigma}]) \qquad (7)$$

For a given confidence level $(1-\alpha)100\%$, the power of the rule to detect a change in means is simply given by:

$$Power = 1 - p \qquad (8)$$

where p is defined in Equation (6) above using the value of K that corresponds to $\Delta/\sigma = 0$. Thus, when the argument delta.over.sigma=0, the value of p is $1 - \alpha$ and the power is simply $\alpha 100\%$. As delta.over.sigma increases above 0, the power increases.

When pi.type="lower", the same value of K is used as when pi.type="upper", but Equation (4) is used to construct the prediction interval. Thus, the power increases as delta.over.sigma decreases below 0.

Value

vector of values between 0 and 1 equal to the probability that at least one of k future observations or averages will fall outside the prediction interval.

Note

See the help files for predIntNorm and predIntNormSimultaneous.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions predIntNormTestPower and plotPredIntNormTestPowerCurve can be used to investigate these relationships for the case of normally-distributed observations. In the case of a simple shift between the two means, the test based on a prediction interval is not as powerful as the two-sample t-test. However, the test based on a prediction interval is more efficient at detecting a shift in the tail.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help files for predIntNorm and predIntNormSimultaneous.

See Also

predIntNorm, predIntNormK, plotPredIntNormTestPowerCurve, predIntNormSimultaneous, predIntNormSimultaneousK, predIntNormSimultaneousTestPower, Prediction Intervals, Normal.

Examples

```
# Show how the power increases as delta.over.sigma increases.
# Assume a 95% upper prediction interval.

predIntNormTestPower(n = 4, delta.over.sigma = 0:2)
#[1] 0.0500000 0.1743014 0.3990892

#----------

# Look at how the power increases with sample size for a one-sided upper
# prediction interval with k=3, delta.over.sigma=2, and a confidence level
# of 95%.

predIntNormTestPower(n = c(4, 8), k = 3, delta.over.sigma = 2)
#[1] 0.3578250 0.5752113

#----------

# Show how the power for an upper 95% prediction limit increases as the
```

```
# number of future observations k increases.  Here, we'll use n=20 and
# delta.over.sigma=1.

predIntNormTestPower(n = 20, k = 1:3, delta.over.sigma = 1)
#[1] 0.2408527 0.2751074 0.2936486
```

---

predIntNpar *Nonparametric Prediction Interval for a Continuous Distribution*

Description

Construct a nonparametric prediction interval to contain at least k out of the next m future observations with probability $(1-\alpha)100\%$ for a continuous distribution.

Usage

```
predIntNpar(x, k = m, m = 1,  lpl.rank = ifelse(pi.type == "upper", 0, 1),
  n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
  lb = -Inf, ub = Inf, pi.type = "two-sided")
```

Arguments

x	a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
k	positive integer specifying the minimum number of future observations out of m that should be contained in the prediction interval. The default value is k=m.
m	positive integer specifying the number of future observations. The default value is m=1.
lpl.rank	positive integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="two-sided" or pi.type="lower", the default value is lpl.rank=1 (implying the minimum value of x is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0 and the value of lb is used as the lower bound of the tolerance interval.
n.plus.one.minus.upl.rank	positive integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default when pi.type="two.sided" or pi.type="upper") means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0 and the value of ub is used as the upper bound of the prediction interval.
lb, ub	scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a prediction interval for a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value. The argument lb is ignored if pi.type="two-sided" or pi.type="lower". The argument ub is ignored if pi.type="two-sided" or pi.type="upper".

`pi.type` character string indicating what kind of prediction interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

Details

What is a Nonparametric Prediction Interval?

A nonparametric prediction interval for some population is an interval on the real line constructed so that it will contain at least k of m future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k and m are pre-specified positive integer where $k \leq m$. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval.

The Form of a Nonparametric Prediction Interval

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n independent observations from some continuous distribution, and let $x_{(i)}$ denote the the i'th order statistics in $\underline{x}$. A two-sided nonparametric prediction interval is constructed as:

$$[x_{(u)}, x_{(v)}] \qquad (1)$$

where u and v are positive integers between 1 and n, and $u < v$. That is, u denotes the rank of the lower prediction limit, and v denotes the rank of the upper prediction limit. To make it easier to write some equations later on, we can also write the prediction interval (1) in a slightly different way as:

$$[x_{(u)}, x_{(n+1-w)}] \qquad (2)$$

where

$$w = n + 1 - v \qquad (3)$$

so that w is a positive integer between 1 and $n-1$, and $u < n+1-w$. In terms of the arguments to the function `predIntNpar`, the argument `lpl.rank` corresponds to u, and the argument `n.plus.one.minus.upl.rank` corresponds to w.

If we allow $u = 0$ and $w = 0$ and define lower and upper bounds as:

$$x_{(0)} = lb \qquad (4)$$

$$x_{(n+1)} = ub \qquad (5)$$

then Equation (2) above can also represent a one-sided lower or one-sided upper prediction interval as well. That is, a one-sided lower nonparametric prediction interval is constructed as:

$$[x_{(u)}, x_{(n+1)}] = [x_{(u)}, ub] \qquad (6)$$

and a one-sided upper nonparametric prediction interval is constructed as:

$$[x_{(0)}, x_{(n+1-w)}] = [lb, x_{(n+1-w)}] \qquad (7)$$

Usually, $lb = -\infty$ or $lb = 0$ and $ub = \infty$.

Constructing Nonparametric Prediction Intervals for Future Observations

Danziger and Davis (1964) show that the probability that at least k out of the next m observations will fall in the interval defined in Equation (2) is given by:

$$(1-\alpha) = [\sum_{i=k}^{m} \binom{m-i+u+w-1}{m-i} \binom{i+n-u-w}{i}] / \binom{n+m}{m} \qquad (8)$$

(Note that computing a nonparametric prediction interval for the case $k = m = 1$ is equivalent to computing a nonparametric β-expectation tolerance interval with coverage $(1 - \alpha)100\%$; see `tolIntNpar`).

The Special Case of Using the Minimum and the Maximum

Setting $u = w = 1$ implies using the smallest and largest observed values as the prediction limits. In this case, it can be shown that the probability that at least k out of the next m observations will fall in the interval

$$[x_{(1)}, x_{(n)}] \qquad (9)$$

is given by:

$$(1 - \alpha) = [\sum_{i=k}^{m} (m - i - 1)\binom{n + i - 2}{i}]/\binom{n + m}{m} \qquad (10)$$

Setting $k = m$ in Equation (10), the probability that all of the next m observations will fall in the interval defined in Equation (9) is given by:

$$(1 - \alpha) = \frac{n(n - 1)}{(n + m)(n + m - 1)} \qquad (11)$$

For one-sided prediction limits, the probability that all m future observations will fall below $x_{(n)}$ (upper prediction limit; `pi.type="upper"`) and the probabilitiy that all m future observations will fall above $x_{(1)}$ (lower prediction limit; `pi.type="lower"`) are both given by:

$$(1 - \alpha) = \frac{n}{n + m} \qquad (12)$$

Constructing Nonparametric Prediction Intervals for Future Medians

To construct a nonparametric prediction interval for a future median based on s future observations, where s is odd, note that this is equivalent to constructing a nonparametric prediction interval that must hold at least $k = (s + 1)/2$ of the next $m = s$ future observations.

Value

a list of class `"estimate"` containing the prediction interval and other information. See the help file for `estimate.object` for details.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Danziger, L., and S. Davis. (1964). Tables of Distribution-Free Tolerance Limits. *Annals of Mathematical Statistics* **35**(5), 1361–1365.

Davis, C.B. (1994). Environmental Regulatory Statistics. In Patil, G.P., and C.R. Rao, eds., *Handbook of Statistics, Vol. 12: Environmental Statistics*. North-Holland, Amsterdam, a division of Elsevier, New York, NY, Chapter 26, 817–865.

Davis, C.B., and R.J. McNichols. (1987). One-sided Intervals for at Least p of m Observations from a Normal Population on Each of r Future Occasions. *Technometrics* **29**, 359–370.

Davis, C.B., and R.J. McNichols. (1994a). Ground Water Monitoring Statistics Update: Part I: Progress Since 1988. *Ground Water Monitoring and Remediation* **14**(4), 148–158.

Davis, C.B., and R.J. McNichols. (1994b). Ground Water Monitoring Statistics Update: Part II: Nonparametric Prediction Limits. *Ground Water Monitoring and Remediation* **14**(4), 159–175.

Davis, C.B., and R.J. McNichols. (1999). Simultaneous Nonparametric Prediction Limits (with Discusson). *Technometrics* **41**(2), 89–112.

Gibbons, R.D. (1987a). Statistical Prediction Intervals for the Evaluation of Ground-Water Quality. *Ground Water* **25**, 455–465.

Gibbons, R.D. (1991b). Statistical Tolerance Limits for Ground-Water Monitoring. *Ground Water* **29**, 563–570.

Gibbons, R.D., and J. Baker. (1991). The Properties of Various Statistical Prediction Intervals for Ground-Water Detection Monitoring. *Journal of Environmental Science and Health* **A26**(4), 535–553.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York, 392pp.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178–188.

Hall, I.J., R.R. Prairie, and C.K. Motlagh. (1975). Non-Parametric Prediction Intervals. *Journal of Quality Technology* **7**(3), 109–114.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

`estimate.object`, `predIntNparN`, `predIntNparConfLevel`, `plotPredIntNparDesign`.

Examples

```
# Generate 20 observations from a lognormal mixture distribution with
# parameters mean1=1, cv1=0.5, mean2=5, cv2=1, and p.mix=0.1.  Use
# predIntNpar to construct a two-sided prediction interval using the
# minimum and maximum observed values.  Note that the associated confidence
# level is 90%.  A larger sample size is required to obtain a larger
# confidence level (see the help file for predIntNparN).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormMixAlt(n = 20, mean1 = 1, cv1 = 0.5,
  mean2 = 5, cv2 = 1, p.mix = 0.1)

predIntNpar(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      Exact
#
#Prediction Interval Type:        two-sided
#
#Confidence Level:                90.47619%
#
#Prediction Limit Rank(s):        1 20
#
#Number of Future Observations:   1
#
#Prediction Interval:             LPL = 0.3647875
#                                 UPL = 1.8173115

#----------

# Repeat the above example, but specify m=5 future observations should be
# contained in the prediction interval.  Note that the confidence level is
# now only 63%.

predIntNpar(dat, m = 5)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            dat
#
```

```
#Sample Size:                     20
#
#Prediction Interval Method:      Exact
#
#Prediction Interval Type:        two-sided
#
#Confidence Level:                63.33333%
#
#Prediction Limit Rank(s):        1 20
#
#Number of Future Observations:   5
#
#Prediction Interval:             LPL = 0.3647875
#                                 UPL = 1.8173115

#----------

# Repeat the above example, but specify that a minimum of k=3 observations
# out of a total of m=5 future observations should be contained in the
# prediction interval.  Note that the confidence level is now 98%.

predIntNpar(dat, k = 3, m = 5)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      Exact
#
#Prediction Interval Type:        two-sided
#
#Confidence Level:                98.37945%
#
#Prediction Limit Rank(s):        1 20
#
#Minimum Number of
#Future Observations
#Interval Should Contain:         3
#
#Total Number of
#Future Observations:             5
#
#Prediction Interval:             LPL = 0.3647875
#                                 UPL = 1.8173115

#==========

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct
```

```
# a one-sided upper nonparametric prediction interval for the next
# 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.

# Look at the data
#-----------------

EPA.09.Ex.18.3.TCE.df

#   Month Well  Well.type TCE.ppb.orig TCE.ppb Censored
#1      1 BW-1 Background           <5     5.0     TRUE
#2      2 BW-1 Background           <5     5.0     TRUE
#3      3 BW-1 Background            8     8.0    FALSE
#...
#22     4 CW-4 Compliance           <5     5.0     TRUE
#23     5 CW-4 Compliance            8     8.0    FALSE
#24     6 CW-4 Compliance           14    14.0    FALSE


longToWide(EPA.09.Ex.18.3.TCE.df, "TCE.ppb.orig", "Month", "Well",
  paste.row.name = TRUE)

#        BW-1 BW-2 BW-3 CW-4
#Month.1   <5    7   <5
#Month.2   <5  6.5   <5
#Month.3    8   <5 10.5  7.5
#Month.4   <5    6   <5   <5
#Month.5    9   12   <5    8
#Month.6   10   <5    9   14


# Construct the prediction limit based on the background well data
# using the maximum value as the upper prediction limit.
# Note that since all censored observations are censored at one
# censoring level and the censoring level is less than all of the
# uncensored observations, we can just supply the censoring level
# to predIntNpar.
#------------------------------------------------------------------

with(EPA.09.Ex.18.3.TCE.df,
  predIntNpar(TCE.ppb[Well.type == "Background"],
    m = 4, pi.type = "upper", lb = 0))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            TCE.ppb[Well.type == "Background"]
#
#Sample Size:                     18
```

```
#
#Prediction Interval Method:      Exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                81.81818%
#
#Prediction Limit Rank(s):        18
#
#Number of Future Observations:   4
#
#Prediction Interval:             LPL =  0
#                                 UPL = 12

# Since the value of 14 ppb for Month 6 at the compliance well exceeds
# the upper prediction limit of 12, we might conclude that there is
# statistically significant evidence of an increase over background
# at CW-4.  However, the confidence level associated with this
# prediction limit is about 82%, which implies a Type I error level of
# 18%.  This means there is nearly a one in five chance of a false positive.
# Only additional background data and/or use of a retesting strategy
# (see predIntNparSimultaneous) would lower the false positive rate.

#==========

# Example 18-4 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# median of order 3 of xylene at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.4.xylene.df.
# There are 8 monthly observations of xylene (ppb) at 3 background wells,
# and 3 montly observations of TCE at a compliance well.

# Look at the data
#-----------------

EPA.09.Ex.18.4.xylene.df

#   Month   Well  Well.type Xylene.ppb.orig Xylene.ppb Censored
#1      1 Well.1 Background              <5        5.0     TRUE
#2      2 Well.1 Background              <5        5.0     TRUE
#3      3 Well.1 Background             7.5        7.5    FALSE
#...
#30     6 Well.4 Compliance              <5        5.0     TRUE
#31     7 Well.4 Compliance             7.8        7.8    FALSE
#32     8 Well.4 Compliance            10.4       10.4    FALSE

longToWide(EPA.09.Ex.18.4.xylene.df, "Xylene.ppb.orig", "Month", "Well",
  paste.row.name = TRUE)

#        Well.1 Well.2 Well.3 Well.4
#Month.1     <5    9.2     <5
#Month.2     <5     <5    5.4
#Month.3    7.5     <5    6.7
```

```
#Month.4     <5    6.1     <5
#Month.5     <5      8     <5
#Month.6     <5    5.9     <5     <5
#Month.7    6.4     <5     <5    7.8
#Month.8      6     <5     <5   10.4

# Construct the prediction limit based on the background well data
# using the maximum value as the upper prediction limit.
# Note that since all censored observations are censored at one
# censoring level and the censoring level is less than all of the
# uncensored observations, we can just supply the censoring level
# to predIntNpar.
#
# To compute a prediction interval for a median of order 3 (i.e.,
# a median based on 3 observations), this is equivalent to
# constructing a nonparametric prediction interval that must hold
# at least 2 of the next 3 future observations.
#----------------------------------------------------------------

with(EPA.09.Ex.18.4.xylene.df,
  predIntNpar(Xylene.ppb[Well.type == "Background"],
    k = 2, m = 3, pi.type = "upper", lb = 0))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            Xylene.ppb[Well.type == "Background"]
#
#Sample Size:                     24
#
#Prediction Interval Method:      Exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                99.1453%
#
#Prediction Limit Rank(s):        24
#
#Minimum Number of
#Future Observations
#Interval Should Contain:         2
#
#Total Number of
#Future Observations:             3
#
#Prediction Interval:             LPL = 0.0
#                                 UPL = 9.2

# The Month 8 observation at the Complance well is 10.4 ppb of Xylene,
# which is greater than the upper prediction limit of 9.2 ppb, so
# conclude there is evidence of contamination at the
```

```
# 100% - 99% = 1% Type I Error Level

#==========

# Cleanup
#--------

rm(dat)
```

---

predIntNparConfLevel *Confidence Level for Nonparametric Prediction Interval for Continuous Distribution*

Description

Compute the confidence level associated with a nonparametric prediction interval that should contain at least k out of the next m future observations for a continuous distribution.

Usage

```
predIntNparConfLevel(n, k = m, m = 1, lpl.rank = ifelse(pi.type == "upper", 0, 1),
   n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
   pi.type = "two.sided")
```

Arguments

n	vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
k	vector of positive integers specifying the minimum number of future observations out of m that should be contained in the prediction interval. The default value is k=m.
m	vector of positive integers specifying the number of future observations. The default value is m=1.
lpl.rank	vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="two-sided" or pi.type="lower", the default value is lpl.rank=1 (implying the minimum value is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0.
n.plus.one.minus.upl.rank	vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0.
pi.type	character string indicating what kind of prediction interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".

Details

If the arguments n, k, m, lpl.rank, and n.plus.one.minus.upl.rank are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for predIntNpar explains how nonparametric prediction intervals are constructed and how the confidence level associated with the prediction interval is computed based on specified values for the sample size and the ranks of the order statistics used for the bounds of the prediction interval.

Value

vector of values between 0 and 1 indicating the confidence level associated with the specified nonparametric prediction interval.

Note

See the help file for predIntNpar.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNpar.

See Also

predIntNpar, predIntNparN, plotPredIntNparDesign.

Examples

```
# Look at how the confidence level of a nonparametric prediction interval
# increases with increasing sample size:

seq(5, 25, by = 5)
#[1] 5 10 15 20 25

round(predIntNparConfLevel(n = seq(5, 25, by = 5)), 2)
#[1] 0.67 0.82 0.87 0.90 0.92

#---------

# Look at how the confidence level of a nonparametric prediction interval
# decreases as the number of future observations increases:

round(predIntNparConfLevel(n = 10, m = 1:5), 2)
#[1] 0.82 0.68 0.58 0.49 0.43

#----------

# Look at how the confidence level of a nonparametric prediction interval
```

```
# decreases with minimum number of observations that must be contained within
# the interval (k):

round(predIntNparConfLevel(n = 10, k = 1:5, m = 5), 2)
#[1] 1.00 0.98 0.92 0.76 0.43

#----------

# Look at how the confidence level of a nonparametric prediction interval
# decreases with the rank of the lower prediction limit:

round(predIntNparConfLevel(n = 10, lpl.rank = 1:5), 2)
#[1] 0.82 0.73 0.64 0.55 0.45

#==========

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.

# Look at the data
#-----------------

EPA.09.Ex.18.3.TCE.df

#   Month Well  Well.type TCE.ppb.orig TCE.ppb Censored
#1      1 BW-1 Background          <5     5.0     TRUE
#2      2 BW-1 Background          <5     5.0     TRUE
#3      3 BW-1 Background           8     8.0    FALSE
#...
#22     4 CW-4 Compliance          <5     5.0     TRUE
#23     5 CW-4 Compliance           8     8.0    FALSE
#24     6 CW-4 Compliance          14    14.0    FALSE


longToWide(EPA.09.Ex.18.3.TCE.df, "TCE.ppb.orig", "Month", "Well",
  paste.row.name = TRUE)

#        BW-1 BW-2 BW-3 CW-4
#Month.1   <5    7   <5
#Month.2   <5  6.5   <5
#Month.3    8   <5 10.5  7.5
#Month.4   <5    6   <5   <5
#Month.5    9   12   <5    8
#Month.6   10   <5    9   14


# If we construct the prediction limit based on the background well
# data using the maximum value as the upper prediction limit,
# the associated confidence level is only 82%.
```

```
#----------------------------------------------------------------

predIntNparConfLevel(n = 18, m = 4, pi.type = "upper")
#[1] 0.8181818

# We would have to collect an additional 18 observations to achieve a
# confidence level of at least 90%:

predIntNparN(m = 4, pi.type = "upper", conf.level = 0.9)
#[1] 36

predIntNparConfLevel(n = 36, m = 4, pi.type = "upper")
#[1] 0.9
```

---

predIntNparN *Sample Size for a Nonparametric Prediction Interval for a Continuous Distribution*

Description

Compute the sample size necessary for a nonparametric prediction interval to contain at least k out of the next m future observations with probability $(1 - \alpha)100\%$ for a continuous distribution.

Usage

```
predIntNparN(k = m, m = 1, lpl.rank = ifelse(pi.type == "upper", 0, 1),
  n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
  pi.type = "two.sided", conf.level = 0.95, n.max = 5000, maxiter = 1000)
```

Arguments

k	vector of positive integers specifying the minimum number of future observations out of m that should be contained in the prediction interval. The default value is k=m.
m	vector of positive integers specifying the number of future observations. The default value is m=1.
lpl.rank	vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. If pi.type="two-sided" or pi.type="lower", the default value is lpl.rank=1 (implying the minimum value is used as the lower bound of the prediction interval). If pi.type="upper", this argument is set equal to 0.
n.plus.one.minus.upl.rank	vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. If pi.type="lower", this argument is set equal to 0.

pi.type	character string indicating what kind of prediction interval to compute. The possible values are "two.sided" (the default), "lower", and "upper".
conf.level	numeric vector of values between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is conf.level=0.95.
n.max	positive integer greater than 1 indicating the maximum possible sample size. The default value is n.max=5000.
maxiter	positive integer indicating the maximum number of iterations to use in the uniroot search algorithm. The default value is maxiter=1000.

Details

If the arguments k, m, lpl.rank, and n.plus.one.minus.upl.rank are not all the same length, they are replicated to be the same length as the length of the longest argument.

The function predIntNparN initially computes the required sample size n by solving Equation (11) or (12) in the help file for predIntNpar for n, depending on the value of the argument pi.type. If k < m, lpl.rank > 1 (two-sided and lower prediction intervals only), or n.plus.one.minus.upl.rank > 1 (two-sided and upper prediction intervals only), then this initial value of n is used as the upper bound in a binary search based on Equation (8) in the help file for predIntNpar and is implemented via the R function uniroot with the argument tolerance set to 1.

Value

vector of positive integers indicating the required sample size(s) for the specified nonparametric prediction interval(s).

Note

See the help file for predIntNpar.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNpar.

See Also

predIntNpar, predIntNparConfLevel, plotPredIntNparDesign.

Examples

```
# Look at how the required sample size for a nonparametric prediction interval
# increases with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9
```

```
predIntNparN(conf.level = seq(0.5, 0.9, by = 0.1))
#[1] 3 4 6 9 19

#----------

# Look at how the required sample size for a nonparametric prediction interval
# increases with number of future observations (m):

1:5
#[1] 1 2 3 4 5

predIntNparN(m = 1:5)
#[1] 39 78 116 155 193

#----------

# Look at how the required sample size for a nonparametric prediction interval
# increases with minimum number of observations that must be contained within
# the interval (k):

predIntNparN(k = 1:5, m = 5)
#[1] 4 7 13 30 193

#----------

# Look at how the required sample size for a nonparametric prediction interval
# increases with the rank of the lower prediction limit:

predIntNparN(lpl.rank = 1:5)
#[1]  39  59  79 100 119

#==========

# Example 18-3 of USEPA (2009, p.18-19) shows how to construct
# a one-sided upper nonparametric prediction interval for the next
# 4 future observations of trichloroethylene (TCE) at a downgradient well.
# The data for this example are stored in EPA.09.Ex.18.3.TCE.df.
# There are 6 monthly observations of TCE (ppb) at 3 background wells,
# and 4 monthly observations of TCE at a compliance well.

# Look at the data
#-----------------

EPA.09.Ex.18.3.TCE.df

#   Month Well  Well.type TCE.ppb.orig TCE.ppb Censored
#1      1 BW-1 Background          <5     5.0     TRUE
#2      2 BW-1 Background          <5     5.0     TRUE
#3      3 BW-1 Background           8     8.0    FALSE
#...
#22     4 CW-4 Compliance          <5     5.0     TRUE
#23     5 CW-4 Compliance           8     8.0    FALSE
```

```
#24     6 CW-4 Compliance           14    14.0    FALSE


longToWide(EPA.09.Ex.18.3.TCE.df, "TCE.ppb.orig", "Month", "Well",
  paste.row.name = TRUE)

#        BW-1 BW-2 BW-3 CW-4
#Month.1   <5    7   <5
#Month.2   <5  6.5   <5
#Month.3    8   <5 10.5  7.5
#Month.4   <5    6   <5   <5
#Month.5    9   12   <5    8
#Month.6   10   <5    9   14


# If we construct the prediction limit based on the background well
# data using the maximum value as the upper prediction limit,
# the associated confidence level is only 82%.
#----------------------------------------------------------------

predIntNparConfLevel(n = 18, m = 4, pi.type = "upper")
#[1] 0.8181818

# We would have to collect an additional 18 observations to achieve a
# confidence level of at least 90%:

predIntNparN(m = 4, pi.type = "upper", conf.level = 0.9)
#[1] 36

predIntNparConfLevel(n = 36, m = 4, pi.type = "upper")
#[1] 0.9
```

`predIntNparSimultaneous` *Nonparametric Simultaneous Prediction Interval for a Continuous Distribution*

Description

Construct a nonparametric simultaneous prediction interval for the next r sampling "occasions" based on one of three possible rules: k-of-m, California, or Modified California. The simultaneous prediction interval assumes the observations from from a continuous distribution.

Usage

```
predIntNparSimultaneous(x, n.median = 1, k = 1, m = 2, r = 1, rule = "k.of.m",
  lpl.rank = ifelse(pi.type == "upper", 0, 1),
  n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
  lb = -Inf, ub = Inf, pi.type = "upper", integrate.args.list = NULL)
```

Arguments

x	a numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n.median	positive odd integer specifying the sample size associated with the future medians. The default value is n.median=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), a positive integer specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling "occassion") the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	positive integer specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.
r	positive integer specifying the number of future sampling "occasions". The default value is r=1.
rule	character string specifying which rule to use. The possible values are "k.of.m" (k-of-m rule; the default), "CA" (California rule), and "Modified.CA" (modified California rule). See the DETAILS section below for more information.
lpl.rank	positive integer indicating the rank of the order statistic to use for the lower bound of the prediction interval. When pi.type="lower", the default value is lpl.rank=1 (implying the minimum value of x is used as the lower bound of the prediction interval). When pi.type="upper", the argument lpl.rank is set equal to 0 and the value of lb is used as the lower bound of the tolerance interval.
n.plus.one.minus.upl.rank	positive integer related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of n.plus.one.minus.upl.rank=1 (the default) means use the first largest value, and in general a value of n.plus.one.minus.upl.rank=i means use the i'th largest value. When pi.type="lower", the argument n.plus.one.minus.upl.rank is set equal to 0 and the value of ub is used as the upper bound of the prediction interval.
lb, ub	scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a prediction interval for a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value. The argument lb is ignored if pi.type="two-sided" or pi.type="lower". The argument ub is ignored if pi.type="two-sided" or pi.type="upper".
pi.type	character string indicating what kind of prediction interval to compute. The possible values are "upper" (the default) and "lower".
integrate.args.list	a list of arguments to supply to the integrate function. The default value is integrate.args.list=NULL which means that the default values of integrate are used.

Details

What is a Nonparametric Simultaneous Prediction Interval?
A nonparametric prediction interval for some population is an interval on the real line constructed so that it will contain at least k of m future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k and m are some pre-specified positive integers and $k \leq m$. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNpar` computes a standard nonparametric prediction interval.

The function `predIntNparSimultaneous` computes a nonparametric simultaneous prediction interval that will contain a certain number of future observations with probability $(1-\alpha)100\%$ for each of r future sampling "occasions", where r is some pre-specified positive integer. The quantity r may refer to r distinct future sampling occasions in time, or it may for example refer to sampling at r distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the r distinct locations.

The function `predIntNparSimultaneous` computes a nonparametric simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (`rule="k.of.m"`), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If obserations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: For this rule, when $r = 1$, the results of `predIntNparSimultaneous` are equivalent to the results of `predIntNpar`.
- For the California rule (`rule="CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.
- For the Modified California rule (`rule="Modified.CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Nonparametric simultaneous prediction intervals can be extended to using medians in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a nonparametric simultaneous prediction interval that will contain a specified number of medians (based on which rule you choose) on each of r future sampling occassions, where each each median is based on b individual observations. For the function `predIntNparSimultaneous`, the argument `n.median` corresponds to b.

The Form of a Nonparametric Prediction Interval

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n independent observations from some continuous distribution, and let $x_{(i)}$ denote the the i'th order statistics in $\underline{x}$. A two-sided nonparametric prediction interval is constructed as:

$$[x_{(u)}, x_{(v)}] \qquad (1)$$

where u and v are positive integers between 1 and n, and $u < v$. That is, u denotes the rank of the lower prediction limit, and v denotes the rank of the upper prediction limit. To make it easier

to write some equations later on, we can also write the prediction interval (1) in a slightly different way as:

$$[x_{(u)}, x_{(n+1-w)}] \qquad (2)$$

where

$$w = n + 1 - v \qquad (3)$$

so that w is a positive integer between 1 and $n - 1$, and $u < n + 1 - w$. In terms of the arguments to the function `predIntNparSimultaneous`, the argument `lpl.rank` corresponds to u, and the argument `n.plus.one.minus.upl.rank` corresponds to w.

If we allow $u = 0$ and $w = 0$ and define lower and upper bounds as:

$$x_{(0)} = lb \qquad (4)$$

$$x_{(n+1)} = ub \qquad (5)$$

then Equation (2) above can also represent a one-sided lower or one-sided upper prediction interval as well. That is, a one-sided lower nonparametric prediction interval is constructed as:

$$[x_{(u)}, x_{(n+1)}] = [x_{(u)}, ub] \qquad (6)$$

and a one-sided upper nonparametric prediction interval is constructed as:

$$[x_{(0)}, x_{(n+1-w)}] = [lb, x_{(n+1-w)}] \qquad (7)$$

Usually, $lb = -\infty$ or $lb = 0$ and $ub = \infty$.

Note: For nonparametric simultaneous prediction intervals, only lower (`pi.type="lower"`) and upper (`pi.type="upper"`) prediction intervals are available.

Constructing Nonparametric Simultaneous Prediction Intervals for Future Observations
First we will show how to construct a nonparametric simultaneous prediction interval based on future observations (i.e., $b = 1$, `n.median=1`), and then extend the formulas to future medians.

Simultaneous Prediction Intervals for the k-of-m Rule (`rule="k.of.m"`)
For the k-of-m rule (`rule="k.of.m"`) with $w = 1$ (i.e., `n.median=1`), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If observations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: When $r = 1$, this kind of simultaneous prediction interval becomes the same as a standard nonparametric prediction interval (see `predIntNpar`).

Chou and Owen (1986) developed the theory for nonparametric simultaneous prediction limits for various rules, including the 1-of-m rule. Their theory, however, does not cover the California or Modified California rules, and uses an r-fold summation involving a minimum of 2^r terms. Davis and McNichols (1994b; 1999) extended the results of Chou and Owen (1986) to include the California and Modified California rule, and developed algorithms that involve summing far fewer terms.

Davis and McNichols (1999) give formulas for the probabilities associated with the one-sided upper simultaneous prediction interval shown in Equation (7). For the k-of-m rule, the probability that at least k of the next m future observations will be contained in the interval given in Equation (7) for each of r future sampling occasions is given by:

$$\begin{aligned} 1-\alpha &= E[\sum_{i=0}^{m-k} \binom{k-1+i}{k-1} Y^k (1-Y)^i]^r \\ &= \int_0^1 [\sum_{i=0}^{m-k} \binom{k-1+i}{k-1} y^k (1-y)^i]^r f(y) dy \qquad (8) \end{aligned}$$

where Y denotes a random variable with a beta distribution with parameters v and $n+1-v$, and $f()$ denotes the pdf of this distribution. Note that v denotes the rank of the order statistic used as the upper prediction limit (i.e., `n.plus.one.minus.upl.rank`=$n+1-v$), and that v is usually equal to n.

Also note that the summation term in Equation (8) corresponds to the cumulative distribution function of a Negative Binomial distribution with parameters `size`=k and `prob`=y evaluated at `q`=$m-k$.

When `pi.type="lower"`, Y denotes a random variable with a beta distribution with parameters $n+1-u$ and u. Note that u denotes the rank of the order statistic used as the lower prediction limit (i.e., `lpl.rank`=u), and that u is usually equal to 1.

Simultaneous Prediction Intervals for the California Rule (`rule="CA"`)
For the California rule (`rule="CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.

In this case, the probability is given by:

$$\begin{aligned} 1-\alpha &= E[\sum_{i=0}^{r} \binom{r}{i} Y^{r-i+(m-1)i} (1-Y)^i] \\ &= \int_0^1 [\sum_{i=0}^{r} \binom{r}{i} y^{r-i+(m-1)i} (1-y)^i] f(y) dy \qquad (9) \end{aligned}$$

Simultaneous Prediction Intervals for the Modified California Rule (`rule="Modified.CA"`)
For the Modified California rule (`rule="Modified.CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

In this case, the probability is given by:

$$\begin{aligned} 1-\alpha &= E[Y^r (1+Q+Q^2-2Q^3)^r] \\ &= \int_0^1 [y^r (1+q+q^2-2q^3)^r] f(y) dy \qquad (10) \end{aligned}$$

where $Q = 1-Y$ and $q = 1-y$.

Davis and McNichols (1999) provide algorithms for computing the probabilities based on expanding polynomials and the formula for the expected value of a beta random variable. In the discussion section of Davis and McNichols (1999), however, Vangel points out that numerical integration is adequate, and this is how these probabilities are computed in the function `predIntNparSimultaneous`.

Constructing Nonparametric Simultaneous Prediction Intervals for Future Medians
USEPA (2009, Chapter 19; Cameron, 2011) extends nonparametric simultaneous prediction intervals to testing future medians for the case of the 1-of-1 and 1-of-2 plans for medians of order 3. In

general, each of the rules (k-of-m, California, and Modified California) can be easily extended to the case of using medians as long as the medians are based on an odd (as opposed to even) sample size.

For each of the above rules, if we are interested in using medians instead of single observations (i.e., $b \geq 1$; `n.median`≥ 1), and we force b to be odd, then a median will be less than a prediction limit once $(b+1)/2$ observations are less than the prediction limit. Thus, Equations (8) - (10) are modified by replacing y with the term:

$$\sum_{i=0}^{b-b'} \binom{b'-1+i}{b'-1} y^{b'}(1-y)^i \qquad (11)$$

where

$$b' = \frac{b+1}{2} \qquad (12)$$

Value

a list of class `"estimate"` containing the simultaneous prediction interval and other information. See the help file for `estimate.object` for details.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Nelson, 1973; Krishnamoorthy and Mathew, 2009). In the context of environmental statistics, prediction intervals are useful for analyzing data from groundwater detection monitoring programs at hazardous and solid waste facilities (e.g., Gibbons et al., 2009; Millard and Neerchal, 2001; USEPA, 2009).

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

Cameron, Kirk. (2011). Personal communication, February 16, 2011. MacStat Consulting, Ltd., Colorado Springs, Colorado.

Chew, V. (1968). Simultaneous Prediction Intervals. *Technometrics* **10**(2), 323–331.

Danziger, L., and S. Davis. (1964). Tables of Distribution-Free Tolerance Limits. *Annals of Mathematical Statistics* **35**(5), 1361–1365.

Davis, C.B. (1994). Environmental Regulatory Statistics. In Patil, G.P., and C.R. Rao, eds., *Handbook of Statistics, Vol. 12: Environmental Statistics*. North-Holland, Amsterdam, a division of Elsevier, New York, NY, Chapter 26, 817–865.

Davis, C.B., and R.J. McNichols. (1987). One-sided Intervals for at Least p of m Observations from a Normal Population on Each of r Future Occasions. *Technometrics* **29**, 359–370.

Davis, C.B., and R.J. McNichols. (1994a). Ground Water Monitoring Statistics Update: Part I: Progress Since 1988. *Ground Water Monitoring and Remediation* **14**(4), 148–158.

Davis, C.B., and R.J. McNichols. (1994b). Ground Water Monitoring Statistics Update: Part II: Nonparametric Prediction Limits. *Ground Water Monitoring and Remediation* **14**(4), 159–175.

Davis, C.B., and R.J. McNichols. (1999). Simultaneous Nonparametric Prediction Limits (with Discusson). *Technometrics* **41**(2), 89–112.

Gibbons, R.D. (1987a). Statistical Prediction Intervals for the Evaluation of Ground-Water Quality. *Ground Water* **25**, 455–465.

Gibbons, R.D. (1991b). Statistical Tolerance Limits for Ground-Water Monitoring. *Ground Water* **29**, 563–570.

Gibbons, R.D., and J. Baker. (1991). The Properties of Various Statistical Prediction Intervals for Ground-Water Detection Monitoring. *Journal of Environmental Science and Health* **A26**(4), 535–553.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York, 392pp.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178–188.

Hall, I.J., R.R. Prairie, and C.K. Motlagh. (1975). Non-Parametric Prediction Intervals. *Journal of Quality Technology* **7**(3), 109–114.

Millard, S.P., and Neerchal, N.K. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, Florida.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

predIntNparSimultaneousConfLevel, predIntNparSimultaneousN, plotPredIntNparSimultaneousDesign, predIntNparSimultaneousTestPower, predIntNpar, tolIntNpar, estimate.object.

Examples

```
# Generate 20 observations from a lognormal mixture distribution with
# parameters mean1=1, cv1=0.5, mean2=5, cv2=1, and p.mix=0.1.  Use
# predIntNparSimultaneous to construct an upper one-sided prediction interval
# using the maximum observed value using the 1-of-3 rule.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnormMixAlt(n = 20, mean1 = 1, cv1 = 0.5,
  mean2 = 5, cv2 = 1, p.mix = 0.1)

predIntNparSimultaneous(dat, k = 1, m = 3, lb = 0)
```

```
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                99.94353%
#
#Prediction Limit Rank(s):        20
#
#Minimum Number of
#Future Observations
#Interval Should Contain:         1
#
#Total Number of
#Future Observations:             3
#
#Prediction Interval:             LPL = 0.000000
#                                 UPL = 1.817311

#----------

# Compare the confidence levels for the 1-of-3 rule, California Rule, and
# Modified California Rule.

predIntNparSimultaneous(dat, k = 1, m = 3, lb = 0)$interval$conf.level
#[1] 0.9994353

predIntNparSimultaneous(dat, m = 3, rule = "CA", lb = 0)$interval$conf.level
#[1] 0.9919066

predIntNparSimultaneous(dat, rule = "Modified.CA", lb = 0)$interval$conf.level
#[1] 0.9984943

#=========

# Repeat the above example, but create the baseline data using just
# n=8 observations and set r to 4 future sampling occasions

set.seed(598)
dat <- rlnormMixAlt(n = 8, mean1 = 1, cv1 = 0.5,
  mean2 = 5, cv2 = 1, p.mix = 0.1)

predIntNparSimultaneous(dat, k = 1, m = 3, r = 4, lb = 0)

#Results of Distribution Parameter Estimation
```

```
#------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            dat
#
#Sample Size:                     8
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                97.7599%
#
#Prediction Limit Rank(s):        8
#
#Minimum Number of
#Future Observations
#Interval Should Contain
#(per Sampling Occasion):         1
#
#Total Number of
#Future Observations
#(per Sampling Occasion):         3
#
#Number of Future
#Sampling Occasions:              4
#
#Prediction Interval:             LPL = 0.000000
#                                 UPL = 5.683453

#----------

# Compare the confidence levels for the 1-of-3 rule, California Rule, and
# Modified California Rule.

predIntNparSimultaneous(dat, k = 1, m = 3, r = 4, lb = 0)$interval$conf.level
#[1] 0.977599

predIntNparSimultaneous(dat, m = 3, r = 4, rule = "CA", lb = 0)$interval$conf.level
#[1] 0.8737798

predIntNparSimultaneous(dat, r = 4, rule = "Modified.CA", lb = 0)$interval$conf.level
#[1] 0.9510178

#==========

# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells  are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.
```

```
# Here we will compute the confidence level associated with two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the background maximum and
# 2) the 1-of-4 plan on individual observations using the 3rd highest background value.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 20 to use to construct the prediction limit.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually.  For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year.

# To determine the minimum confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the maximum allowed individual Type I Error level per constituent to:

# 1 - (1 - SWFPR)^(1 / Number of Constituents)

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / Number of Constituents)

# where SWFPR = site-wide false positive rate.  For this example, we
# will set SWFPR = 0.1.  Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:

# n  = 20 based on 4 Background Wells
# nw = 10 Compliance Wells
# nc =  5 Constituents
# ne =  1 Evaluation per year

n  <- 20
nw <- 10
nc <-  5
ne <-  1

# Set number of future sampling occasions r to
# Number Compliance Wells x Number Evaluations per Year
r  <-  nw * ne

conf.level <- (1 - 0.1)^(1 / nc)
conf.level
#[1] 0.9791484

alpha <- 1 - conf.level
alpha
#[1] 0.02085164

#----------
```

```
# Look at the data:

head(EPA.09.Ex.19.5.mercury.df)
#  Event Well  Well.type Mercury.ppb.orig Mercury.ppb Censored
#1     1 BG-1 Background             0.21        0.21    FALSE
#2     2 BG-1 Background              <.2        0.20     TRUE
#3     3 BG-1 Background              <.2        0.20     TRUE
#4     4 BG-1 Background              <.2        0.20     TRUE
#5     5 BG-1 Background              <.2        0.20     TRUE
#6     6 BG-1 Background                            NA    FALSE

longToWide(EPA.09.Ex.19.5.mercury.df, "Mercury.ppb.orig",
  "Event", "Well", paste.row.name = TRUE)
#        BG-1 BG-2 BG-3 BG-4 CW-1 CW-2
#Event.1 0.21  <.2  <.2  <.2 0.22 0.36
#Event.2  <.2  <.2 0.23 0.25  0.2 0.41
#Event.3  <.2  <.2  <.2 0.28  <.2 0.28
#Event.4  <.2 0.21 0.23  <.2 0.25 0.45
#Event.5  <.2  <.2 0.24  <.2 0.24 0.43
#Event.6                     <.2 0.54


# Construct the upper simultaneous prediction limit using the 1-of-2
# retesting plan for a median of order 3 based on the background maximum

Hg.Back <- with(EPA.09.Ex.19.5.mercury.df,
  Mercury.ppb[Well.type == "Background"])

pred.int.1.of.2.med.3 <- predIntNparSimultaneous(Hg.Back, n.median = 3,
  k = 1, m = 2, r = r, lb = 0)

pred.int.1.of.2.med.3

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            Hg.Back
#
#Sample Size:                     20
#
#Number NA/NaN/Inf's:             4
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                99.40354%
#
#Prediction Limit Rank(s):        20
#
#Minimum Number of
```

```
#Future Medians
#Interval Should Contain
#(per Sampling Occasion):         1
#
#Total Number of
#Future Medians
#(per Sampling Occasion):         2
#
#Number of Future
#Sampling Occasions:              10
#
#Sample Size for Medians:         3
#
#Prediction Interval:             LPL = 0.00
#                                 UPL = 0.28

# Note that the achieved confidence level of 99.4% is greater than the
# required confidence level of 97.9%.

# Now determine whether either compliance well indicates evidence of
# Mercury contamination.

# Compliance Well 1
#------------------
Hg.CW.1 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-1"])

Hg.CW.1
#[1] "0.22" "0.2"  "<.2"  "0.25" "0.24" "<.2"

# The median of the first 3 observations is 0.2, which is less than
# the UPL of 0.28, so there is no evidence of contamination.

# Compliance Well 2
#------------------
Hg.CW.2 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-2"])

Hg.CW.2
#[1] "0.36" "0.41" "0.28" "0.45" "0.43" "0.54"

# The median of the first 3 observations is 0.36, so 3 more observations have to
# be looked at.  The median of the second 3 observations is 0.45, which is
# larger than the UPL of 0.28, so there is evidence of contamination.

#----------

# Now create the upper simultaneous prediction limit using the 1-of-4 plan
# on individual observations using the 3rd highest background value.

pred.int.1.of.4.3rd <- predIntNparSimultaneous(Hg.Back, k = 1, m = 4,
  r = r, lb = 0, n.plus.one.minus.upl.rank = 3)

pred.int.1.of.4.3rd
```

```
#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            Hg.Back
#
#Sample Size:                     20
#
#Number NA/NaN/Inf's:             4
#
#Prediction Interval Method:      exact
#
#Prediction Interval Type:        upper
#
#Confidence Level:                98.64909%
#
#Prediction Limit Rank(s):        18
#
#Minimum Number of
#Future Observations
#Interval Should Contain
#(per Sampling Occasion):         1
#
#Total Number of
#Future Observations
#(per Sampling Occasion):         4
#
#Number of Future
#Sampling Occasions:              10
#
#Prediction Interval:             LPL = 0.00
#                                 UPL = 0.24

# Note that the achieved confidence level of 98.6% is greater than the
# required confidence level of 97.9%.


# Now determine whether either compliance well indicates evidence of
# Mercury contamination.

# Compliance Well 1
#------------------
Hg.CW.1 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-1"])

Hg.CW.1
#[1] "0.22" "0.2"  "<.2"  "0.25" "0.24" "<.2"

# The first observation is less than the UPL of 0.24, which is less than
# the UPL of 0.28, so there is no evidence of contamination.


# Compliance Well 2
```

```
#------------------
Hg.CW.2 <- with(EPA.09.Ex.19.5.mercury.df, Mercury.ppb.orig[Well == "CW-2"])

Hg.CW.2
#[1] "0.36" "0.41" "0.28" "0.45" "0.43" "0.54"

# All of the first 4 observations are greater than the UPL of 0.24, so there
# is evidence of contamination.

 #==========

# Cleanup
#--------
rm(dat, n, nw, nc, ne, r, conf.level, alpha, Hg.Back, pred.int.1.of.2.med.3,
  pred.int.1.of.4.3rd, Hg.CW.1, Hg.CW.2)
```

---

predIntNparSimultaneousConfLevel
Confidence Level of Simultaneous Nonparametric Prediction Interval for Continuous Distribution

---

Description

Compute the confidence level associated with a nonparametric simultaneous prediction interval based on one of three possible rules: k-of-m, California, or Modified California. Observations are assumed to come from from a continuous distribution.

Usage

```
predIntNparSimultaneousConfLevel(n, n.median = 1, k = 1, m = 2, r = 1,
  rule = "k.of.m", lpl.rank = ifelse(pi.type == "upper", 0, 1),
  n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
  pi.type = "upper", integrate.args.list = NULL)
```

Arguments

n	vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
n.median	vector of positive odd integers specifying the sample size associated with the future medians. The default value is n.median=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), a vector of positive integers specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling "occassion") the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".

`m` vector of positive integers specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is `m=2`, except when `rule="Modified.CA"`, in which case this argument is ignored and `m` is automatically set equal to `4`.

`r` vector of positive integers specifying the number of future sampling "occasions". The default value is `r=1`.

`rule` character string specifying which rule to use. The possible values are `"k.of.m"` (k-of-m rule; the default), `"CA"` (California rule), and `"Modified.CA"` (modified California rule).

`lpl.rank` vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. When `pi.type="lower"`, the default value is `lpl.rank=1` (implying the minimum value of `x` is used as the lower bound of the prediction interval). When `pi.type="upper"`, the argument `lpl.rank` is set equal to `0`.

`n.plus.one.minus.upl.rank`
vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of `n.plus.one.minus.upl.rank=1` (the default) means use the first largest value, and in general a value of `n.plus.one.minus.upl.rank=`i means use the i'th largest value. If `pi.type="lower"`, this argument is set equal to `0`.

`pi.type` character string indicating what kind of prediction interval to compute. The possible values are `"two.sided"` (the default), `"lower"`, and `"upper"`.

`integrate.args.list`
list of arguments to supply to the `integrate` function. The default value is `NULL`.

Details

If the arguments `n`, `k`, `m`, `r`, `lpl.rank`, and `n.plus.one.minus.upl.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The function `predIntNparSimultaneousConfLevel` computes the confidence level based on Equation (8), (9), or (10) in the help file for `predIntNparSimultaneous`, depending on the value of the argument `rule`.

Note that when `rule="k.of.m"` and `r=1`, this is equivalent to a standard nonparametric prediction interval and you can use the function `predIntNparConfLevel` instead.

Value

vector of values between 0 and 1 indicating the confidence level associated with the specified simultaneous nonparametric prediction interval.

Note

See the help file for `predIntNparSimultaneous`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for predIntNparSimultaneous.

See Also

predIntNparSimultaneous, predIntNparSimultaneousN, plotPredIntNparSimultaneousDesign, predIntNparSimultaneousTestPower, predIntNpar, tolIntNpar.

Examples

```
# For the 1-of-3 rule with r=20 future sampling occasions, look at how the
# confidence level of a simultaneous nonparametric prediction interval
# increases with increasing sample size:

seq(5, 25, by = 5)
#[1] 5 10 15 20 25

conf <- predIntNparSimultaneousConfLevel(n = seq(5, 25, by = 5),
  k = 1, m = 3, r = 20)
round(conf, 2)
#[1] 0.82 0.95 0.98 0.99 0.99

#----------

# For the 1-of-m rule with r=20 future sampling occasions, look at how the
# confidence level of a simultaneous nonparametric prediction interval
# increases as the number of future observations increases:

1:5
#[1] 1 2 3 4 5

conf <- predIntNparSimultaneousConfLevel(n = 10, k = 1, m = 1:5, r = 20)
round(conf, 2)
#[1] 0.33 0.81 0.95 0.98 0.99

#----------

# For the 1-of-3 rule, look at how the confidence level of a simultaneous
# nonparametric prediction interval decreases with number of future sampling
# occasions (r):

seq(5, 20, by = 5)
#[1]  5 10 15 20

conf <- predIntNparSimultaneousConfLevel(n = 10, k = 1, m = 3,
  r = seq(5, 20, by = 5))

round(conf, 2)
#[1] 0.98 0.97 0.96 0.95

#----------
```

```
# For the 1-of-3 rule with r=20 future sampling occasions, look at how the
# confidence level of a simultaneous nonparametric prediction interval
# decreases as the rank of the upper prediction limit decreases:

conf <- predIntNparSimultaneousConfLevel(n = 10, k = 1, m = 3, r = 20,
  n.plus.one.minus.upl.rank = 1:5)

round(conf, 2)
#[1] 0.95 0.82 0.63 0.43 0.25

#----------

# Clean up
#---------
rm(conf)

#==========

# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells  are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# Here we will compute the confidence level associated with two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the background maximum and
# 2) the 1-of-4 plan on individual observations using the 3rd highest background value.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 20 to use to construct the prediction limit.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually.  For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year.

# To determine the minimum confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the maximum allowed individual Type I Error level per constituent to:

# 1 - (1 - SWFPR)^(1 / Number of Constituents)

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / Number of Constituents)

# where SWFPR = site-wide false positive rate.  For this example, we
# will set SWFPR = 0.1.  Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:
```

```
# n  = 20 based on 4 Background Wells
# nw = 10 Compliance Wells
# nc =  5 Constituents
# ne =  1 Evaluation per year

n  <- 20
nw <- 10
nc <-  5
ne <-  1

# Set number of future sampling occasions r to
# Number Compliance Wells x Number Evaluations per Year
r  <-  nw * ne

conf.level <- (1 - 0.1)^(1 / nc)
conf.level
#[1] 0.9791484

# So the required confidence level is 0.98, or 98%.
# Now determine the confidence level associated with each plan.
# Note that both plans achieve the required confidence level.

# 1) the 1-of-2 retesting plan for a median of order 3 using the
#    background maximum

predIntNparSimultaneousConfLevel(n = 20, n.median = 3, k = 1, m = 2, r = r)
#[1] 0.9940354


# 2) the 1-of-4 plan on individual observations using the 3rd highest
#    background value.

predIntNparSimultaneousConfLevel(n = 20, k = 1, m = 4, r = r,
  n.plus.one.minus.upl.rank = 3)
#[1] 0.9864909

#==========

# Cleanup
#--------
rm(n, nw, nc, ne, r, conf.level)
```

---

predIntNparSimultaneousN *Sample Size for Simultaneous Nonparametric Prediction Interval for Continuous Distribution*

Description

Compute the sample size necessary for a nonparametric simultaneous prediction interval to achieve a specified confidence level based on one of three possible rules: k-of-m, California, or Modified

California. Observations are assumed to come from from a continuous distribution.

Usage

```
predIntNparSimultaneousN(n.median = 1, k = 1, m = 2, r = 1, rule = "k.of.m",
  lpl.rank = ifelse(pi.type == "upper", 0, 1),
 n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1), pi.type = "upper",
  conf.level = 0.95, n.max = 5000, integrate.args.list = NULL, maxiter = 1000)
```

Arguments

`n.median`	vector of positive odd integers specifying the sample size associated with the future medians. The default value is `n.median=1` (i.e., individual observations). Note that all future medians must be based on the same sample size.
`k`	for the k-of-m rule (`rule="k.of.m"`), a vector of positive integers specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling "occassion") the prediction interval should contain. The default value is `k=1`. This argument is ignored when the argument `rule` is not equal to `"k.of.m"`.
`m`	vector of positive integers specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is `m=2`, except when `rule="Modified.CA"`, in which case this argument is ignored and `m` is automatically set equal to `4`.
`r`	vector of positive integers specifying the number of future sampling "occasions". The default value is `r=1`.
`rule`	character string specifying which rule to use. The possible values are `"k.of.m"` (k-of-m rule; the default), `"CA"` (California rule), and `"Modified.CA"` (modified California rule).
`lpl.rank`	vector of positive integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. When `pi.type="lower"`, the default value is `lpl.rank=1` (implying the minimum value of `x` is used as the lower bound of the prediction interval). When `pi.type="upper"`, the argument `lpl.rank` is set equal to `0`.
`n.plus.one.minus.upl.rank`	vector of positive integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of `n.plus.one.minus.upl.rank=1` (the default) means use the first largest value, and in general a value of `n.plus.one.minus.upl.rank=`i means use the i'th largest value. If `pi.type="lower"`, this argument is set equal to `0`.
`pi.type`	character string indicating what kind of prediction interval to compute. The possible values are `"two.sided"` (the default), `"lower"`, and `"upper"`.
`conf.level`	numeric vector of values between 0 and 1 indicating the confidence level associated with the prediction interval. The default value is `conf=0.95`.
`n.max`	numeric scalar indicating the maximum sample size to consider. This argument is used in the search algorithm to determine the required sample size. The default value is `n.max=5000`.

`integrate.args.list`
: list of arguments to supply to the `integrate` function. The default value is `NULL`.

`maxiter`
: positive integer indicating the maximum number of iterations to use in the `uniroot` search algorithm. The default value is `maxiter=1000`.

Details

If the arguments `k`, `m`, `r`, `lpl.rank`, and `n.plus.one.minus.upl.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The function `predIntNparSimultaneousN` computes the required sample size n by solving Equation (8), (9), or (10) in the help file for `predIntNparSimultaneous` for n, depending on the value of the argument `rule`.

Note that when `rule="k.of.m"` and `r=1`, this is equivalent to a standard nonparametric prediction interval and you can use the function `predIntNparN` instead.

Value

vector of positive integers indicating the required sample size(s) for the specified nonparametric simultaneous prediction interval(s).

Note

See the help file for `predIntNparSimultaneous`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNparSimultaneous`.

See Also

`predIntNparSimultaneous`, `predIntNparSimultaneousConfLevel`, `plotPredIntNparSimultaneousDesign`, `predIntNparSimultaneousTestPower`, `predIntNpar`, `tolIntNpar`.

Examples

```
# For the 1-of-2 rule, look at how the required sample size for a one-sided
# upper simultaneous nonparametric prediction interval for r=20 future
# sampling occasions increases with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

predIntNparSimultaneousN(r = 20, conf.level = seq(0.5, 0.9, by = 0.1))
#[1]  4  5  7 10 17
```

```
#----------

# For the 1-of-m rule, look at how the required sample size for a one-sided
# upper simultaneous nonparametric prediction interval decreases with increasing
# number of future observations (m), given r=20 future sampling occasions:

predIntNparSimultaneousN(k = 1, m = 1:5, r = 20)
#[1] 380  26  11   7   5

#----------

# For the 1-of-3 rule, look at how the required sample size for a one-sided
# upper simultaneous nonparametric prediction interval increases with number
# of future sampling occasions (r):

predIntNparSimultaneousN(k = 1, m = 3, r = c(5, 10, 15, 20))
#[1]  7  8 10 11

#----------

# For the 1-of-3 rule, look at how the required sample size for a one-sided
# upper simultaneous nonparametric prediction interval increases as the rank
# of the upper prediction limit decreases, given r=20 future sampling occasions:

predIntNparSimultaneousN(k = 1, m = 3, r = 20, n.plus.one.minus.upl.rank = 1:5)
#[1] 11 19 26 34 41

#----------

# Compare the required sample size for r=20 future sampling occasions based
# on the 1-of-3 rule, the CA rule with m=3, and the Modified CA rule.

predIntNparSimultaneousN(k = 1, m = 3, r = 20, rule = "k.of.m")
#[1] 11

predIntNparSimultaneousN(m = 3, r = 20, rule = "CA")
#[1] 36

predIntNparSimultaneousN(r = 20, rule = "Modified.CA")
#[1] 15

#==========

# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells  are shown in the guidance document).
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# Here we will modify the example to compute the required number of background
# observations for two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the background maximum and
```

```
# 2) the 1-of-4 plan on individual observations using the 3rd highest background value.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.df.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually.  For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year.

# To determine the minimum confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the maximum allowed individual Type I Error level per constituent to:

# 1 - (1 - SWFPR)^(1 / Number of Constituents)

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / Number of Constituents)

# where SWFPR = site-wide false positive rate.  For this example, we
# will set SWFPR = 0.1.  Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:

# nw = 10 Compliance Wells
# nc =  5 Constituents
# ne =  1 Evaluation per year

nw <- 10
nc <-  5
ne <-  1

# Set number of future sampling occasions r to
# Number Compliance Wells x Number Evaluations per Year
r  <-  nw * ne

conf.level <- (1 - 0.1)^(1 / nc)
conf.level
#[1] 0.9791484

# So the required confidence level is 0.98, or 98%.

# Now determine the required number of background observations for  each plan.

# 1) the 1-of-2 retesting plan for a median of order 3 using the
#    background maximum

predIntNparSimultaneousN(n.median = 3, k = 1, m = 2, r = r,
  conf.level = conf.level)
#[1] 14


# 2) the 1-of-4 plan on individual observations using the 3rd highest
#    background value.
```

```
predIntNparSimultaneousN(k = 1, m = 4, r = r,
  n.plus.one.minus.upl.rank = 3, conf.level = conf.level)
#[1] 18

#==========

# Cleanup
#--------
rm(nw, nc, ne, r, conf.level)
```

---

predIntNparSimultaneousTestPower *Probability That at Least One Set of Future Observations Violates the Given Rule Based on a Nonparametric Simultaneous Prediction Interval*

Description

Compute the probability that at least one set of future observations violates the given rule based on a nonparametric simultaneous prediction interval for the next r future sampling occasions. The three possible rules are: k-of-m, California, or Modified California. The probability is based on assuming the true distribution of the observations is normal.

Usage

```
predIntNparSimultaneousTestPower(n, n.median = 1, k = 1, m = 2, r = 1,
  rule = "k.of.m", lpl.rank = ifelse(pi.type == "upper", 0, 1),
  n.plus.one.minus.upl.rank = ifelse(pi.type == "lower", 0, 1),
  delta.over.sigma = 0, pi.type = "upper", r.shifted = r,
  method = "approx", NMC = 100, ci = FALSE, ci.conf.level = 0.95,
  integrate.args.list = NULL, evNormOrdStats.method = "royston")
```

Arguments

n	vector of positive integers specifying the sample sizes. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.
n.median	vector of positive odd integers specifying the sample size associated with the future medians. The default value is n.median=1 (i.e., individual observations). Note that all future medians must be based on the same sample size.
k	for the k-of-m rule (rule="k.of.m"), a vector of positive integers specifying the minimum number of observations (or medians) out of m observations (or medians) (all obtained on one future sampling "occassion") the prediction interval should contain. The default value is k=1. This argument is ignored when the argument rule is not equal to "k.of.m".
m	vector of positive integers specifying the maximum number of future observations (or medians) on one future sampling "occasion". The default value is m=2, except when rule="Modified.CA", in which case this argument is ignored and m is automatically set equal to 4.

`r` vector of positive integers specifying the number of future sampling "occasions". The default value is `r=1`.

`rule` character string specifying which rule to use. The possible values are `"k.of.m"` (k-of-m rule; the default), `"CA"` (California rule), and `"Modified.CA"` (modified California rule).

`lpl.rank` vector of non-negative integers indicating the rank of the order statistic to use for the lower bound of the prediction interval. When `pi.type="lower"`, the default value is `lpl.rank=1` (implying the minimum value of `x` is used as the lower bound of the prediction interval). When `pi.type="upper"`, the argument `lpl.rank` is set equal to `0`.

`n.plus.one.minus.upl.rank`
vector of non-negative integers related to the rank of the order statistic to use for the upper bound of the prediction interval. A value of `n.plus.one.minus.upl.rank=1` (the default) means use the first largest value, and in general a value of `n.plus.one.minus.upl.rank=`i means use the i'th largest value. When `pi.type="lower"`, the argument `n.plus.one.minus.upl.rank` is set equal to `0`.

`delta.over.sigma`
numeric vector indicating the ratio Δ/σ. The quantity Δ (delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation for both populations. The default value is `delta.over.sigma=0`.

`pi.type` character string indicating what kind of prediction interval to compute. The possible values are `"two.sided"` (the default), `"lower"`, and `"upper"`.

`r.shifted` vector of positive integers specifying the number of future sampling occasions for which the scaled mean is shifted by Δ/σ. All values must be integeters between 1 and the corresponding element of `r`. The default value is `r.shifted=r`.

`method` character string indicating what method to use to compute the power. The possible values are `"approx"` (approximation based on `predIntNormSimultaneousTestPower`; the default) and `"simulate"` (Monte Carlo simulation).

`NMC` positive integer indicating the number of Monte Carlo trials to run when `method="simulate"`. The default value is `NMC=100`.

`ci` logical scalar indicating whether to compute a confidence interval for the power when `method="simulate"`. The default value is `ci=FALSE`.

`ci.conf.level` numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the power. The argument is ignored if `ci=FALSE` or `method="approx"`.

`integrate.args.list`
list of arguments to supply to the `integrate` function. The default value is `NULL`.

`evNormOrdStats.method`
character string indicating which method to use in the call to `evNormOrdStatsScalar` when `method="approx"`. The default value is `evNormOrdStats.method="royston"`. See the DETAILS section for more information.

Details

What is a Nonparametric Simultaneous Prediction Interval?
A nonparametric prediction interval for some population is an interval on the real line constructed so that it will contain at least k of m future observations from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k and m are some pre-specified positive integers and $k \leq m$. The quantity $(1-\alpha)100\%$ is called the confidence coefficient or confidence level associated with the prediction interval. The function `predIntNpar` computes a standard nonparametric prediction interval.

The function `predIntNparSimultaneous` computes a nonparametric simultaneous prediction interval that will contain a certain number of future observations with probability $(1-\alpha)100\%$ for each of r future sampling "occasions", where r is some pre-specified positive integer. The quantity r may refer to r distinct future sampling occasions in time, or it may for example refer to sampling at r distinct locations on one future sampling occasion, assuming that the population standard deviation is the same at all of the r distinct locations.

The function `predIntNparSimultaneous` computes a nonparametric simultaneous prediction interval based on one of three possible rules:

- For the k-of-m rule (`rule="k.of.m"`), at least k of the next m future observations will fall in the prediction interval with probability $(1-\alpha)100\%$ on each of the r future sampling occasions. If obserations are being taken sequentially, for a particular sampling occasion, up to m observations may be taken, but once k of the observations fall within the prediction interval, sampling can stop. Note: For this rule, when $r = 1$, the results of `predIntNparSimultaneous` are equivalent to the results of `predIntNpar`.
- For the California rule (`rule="CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else all of the next $m-1$ observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, $m-1$ more observations must be taken.
- For the Modified California rule (`rule="Modified.CA"`), with probability $(1-\alpha)100\%$, for each of the r future sampling occasions, either the first observation will fall in the prediction interval, or else at least 2 out of the next 3 observations will fall in the prediction interval. That is, if the first observation falls in the prediction interval then sampling can stop. Otherwise, up to 3 more observations must be taken.

Nonparametric simultaneous prediction intervals can be extended to using medians in place of single observations (USEPA, 2009, Chapter 19). That is, you can create a nonparametric simultaneous prediction interval that will contain a specified number of medians (based on which rule you choose) on each of r future sampling occassions, where each each median is based on b individual observations. For the function `predIntNparSimultaneous`, the argument `n.median` corresponds to b.

The Form of a Nonparametric Prediction Interval

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n independent observations from some continuous distribution, and let $x_{(i)}$ denote the the i'th order statistics in $\underline{x}$. A two-sided nonparametric prediction interval is constructed as:

$$[x_{(u)}, x_{(v)}] \qquad (1)$$

where u and v are positive integers between 1 and n, and $u < v$. That is, u denotes the rank of the lower prediction limit, and v denotes the rank of the upper prediction limit. To make it easier

to write some equations later on, we can also write the prediction interval (1) in a slightly different way as:

$$[x_{(u)}, x_{(n+1-w)}] \qquad (2)$$

where

$$w = n + 1 - v \qquad (3)$$

so that w is a positive integer between 1 and $n - 1$, and $u < n + 1 - w$. In terms of the arguments to the function `predIntNparSimultaneous`, the argument `lpl.rank` corresponds to u, and the argument `n.plus.one.minus.upl.rank` corresponds to w.

If we allow $u = 0$ and $w = 0$ and define lower and upper bounds as:

$$x_{(0)} = lb \qquad (4)$$

$$x_{(n+1)} = ub \qquad (5)$$

then Equation (2) above can also represent a one-sided lower or one-sided upper prediction interval as well. That is, a one-sided lower nonparametric prediction interval is constructed as:

$$[x_{(u)}, x_{(n+1)}] = [x_{(u)}, ub] \qquad (6)$$

and a one-sided upper nonparametric prediction interval is constructed as:

$$[x_{(0)}, x_{(n+1-w)}] = [lb, x_{(n+1-w)}] \qquad (7)$$

Usually, $lb = -\infty$ or $lb = 0$ and $ub = \infty$.

Note: For nonparametric simultaneous prediction intervals, only lower (`pi.type="lower"`) and upper (`pi.type="upper"`) prediction intervals are available.

Computing Power

The "power" of the prediction interval is defined as the probability that at least one set of future observations violates the given rule based on a simultaneous prediction interval for the next r future sampling occasions, where the population for the future observations is allowed to differ from the population for the observations used to construct the prediction interval.

For the function `predIntNparSimultaneousTestPower`, power is computed assuming both the background and future the observations come from normal distributions with the same standard deviation, but the means of the distributions are allowed to differ. The quantity Δ (upper case delta) denotes the difference between the mean of the population that was sampled to construct the prediction interval, and the mean of the population that will be sampled to produce the future observations. The quantity σ (sigma) denotes the population standard deviation of both of these populations. The argument `delta.over.sigma` corresponds to the quantity Δ/σ.

Approximate Power (`method="approx"`)

Based on Gansecki (2009), the power of a nonparametric simultaneous prediction interval when the underlying observations come from a nomral distribution can be approximated by the power of a normal simultaneous prediction interval (see `predIntNormSimultaneousTestPower`) where the multiplier K is replaced with the expected value of the normal order statistic that corresponds to the rank of the order statistic used for the upper or lower bound of the prediction interval. Gansecki (2009) uses the approximation:

$$K = \Phi^{-1}(\frac{i - 0.5}{n}) \qquad (8)$$

where Φ denotes the cumulative distribution function of the standard normal distribution and i denotes the rank of the order statistic used as the prediction limit. By default, the value of the argument `evNormOrdStats.method="royston"`, so the function `predIntNparSimultaneousTestPower` uses the exact value of the expected value of the normal order statistic in the call to `evNormOrdStatsScalar`. You can change the method of computing the expected value of the normal order statistic by changing the value of the argument `evNormOrdStats.method`.

Power Based on Monte Carlo Simulation (`method="simulate"`)
When `method="simulate"`, the power of the nonparametric simultaneous prediction interval is estimated based on a Monte Carlo simulation. The argument `NMC` determines the number of Monte Carlo trials. If `ci=TRUE`, a confidence interval for the power is created based on the `NMC` Monte Carlo estimates of power.

Value

vector of values between 0 and 1 equal to the probability that the rule will be violated.

Note

See the help file for `predIntNparSimultaneous`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether two distributions differ from each other. The functions `predIntNparSimultaneousTestPower` and `plotPredIntNparSimultaneousTestPowerCurve` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `predIntNparSimultaneous`.

Gansecki, M. (2009). *Using the Optimal Rank Values Calculator*. US Environmental Protection Agency, Region 8, March 10, 2009.

See Also

`plotPredIntNparSimultaneousTestPowerCurve`, `predIntNparSimultaneous`, `predIntNparSimultaneousN`, `predIntNparSimultaneousConfLevel`, `plotPredIntNparSimultaneousDesign`, `predIntNpar`, `tolIntNpar`.

Examples

```
# Example 19-5 of USEPA (2009, p. 19-33) shows how to compute nonparametric upper
# simultaneous prediction limits for various rules based on trace mercury data (ppb)
# collected in the past year from a site with four background wells and 10 compliance
# wells (data for two of the compliance wells  are shown in the guidance document).
```

```
# The facility must monitor the 10 compliance wells for five constituents
# (including mercury) annually.

# Here we will compute the confidence levels and powers associated with
# two different sampling plans:
# 1) the 1-of-2 retesting plan for a median of order 3 using the
#    background maximum and
# 2) the 1-of-4 plan on individual observations using the 3rd highest
#    background value.
# Power will be computed assuming a normal distribution and setting
# delta.over.sigma equal to 2, 3, and 4.
# The data for this example are stored in EPA.09.Ex.19.5.mercury.df.

# We will pool data from 4 background wells that were sampled on
# a number of different occasions, giving us a sample size of
# n = 20 to use to construct the prediction limit.

# There are 10 compliance wells and we will monitor 5 different
# constituents at each well annually.  For this example, USEPA (2009)
# recommends setting r to the product of the number of compliance wells and
# the number of evaluations per year.

# To determine the minimum confidence level we require for
# the simultaneous prediction interval, USEPA (2009) recommends
# setting the maximum allowed individual Type I Error level per constituent to:

# 1 - (1 - SWFPR)^(1 / Number of Constituents)

# which translates to setting the confidence limit to

# (1 - SWFPR)^(1 / Number of Constituents)

# where SWFPR = site-wide false positive rate.  For this example, we
# will set SWFPR = 0.1.  Thus, the required individual Type I Error level
# and confidence level per constituent are given as follows:

# n  = 20 based on 4 Background Wells
# nw = 10 Compliance Wells
# nc =  5 Constituents
# ne =  1 Evaluation per year

n  <- 20
nw <- 10
nc <-  5
ne <-  1

# Set number of future sampling occasions r to
# Number Compliance Wells x Number Evaluations per Year
r  <-  nw * ne

conf.level <- (1 - 0.1)^(1 / nc)
conf.level
#[1] 0.9791484
```

```
# So the required confidence level is 0.98, or 98%.
# Now determine the confidence level associated with each plan.
# Note that both plans achieve the required confidence level.

# 1) the 1-of-2 retesting plan for a median of order 3 using the
#    background maximum

predIntNparSimultaneousConfLevel(n = 20, n.median = 3, k = 1, m = 2, r = r)
#[1] 0.9940354


# 2) the 1-of-4 plan based on individual observations using the 3rd highest
#    background value.

predIntNparSimultaneousConfLevel(n = 20, k = 1, m = 4, r = r,
  n.plus.one.minus.upl.rank = 3)
#[1] 0.9864909

#----------------------------------------------------------------------------
# Compute approximate power of each plan to detect contamination at just 1 well
# assuming true underying distribution of Hg is Normal at all wells and
# using delta.over.sigma equal to 2, 3, and 4.
#----------------------------------------------------------------------------

# Computer aproximate power for
# 1) the 1-of-2 retesting plan for a median of order 3 using the
#    background maximum

predIntNparSimultaneousTestPower(n = 20, n.median = 3, k = 1, m = 2, r = r,
  delta.over.sigma = 2:4, r.shifted = 1)
#[1] 0.3953712 0.9129671 0.9983054


# Compute approximate power for
# 2) the 1-of-4 plan based on individual observations using the 3rd highest
#    background value.

predIntNparSimultaneousTestPower(n = 20, k = 1, m = 4, r = r,
  n.plus.one.minus.upl.rank = 3, delta.over.sigma = 2:4, r.shifted = 1)
#[1] 0.4367972 0.8694664 0.9888779


#----------

## Not run:
# Compare estimated power using approximation method with estimated power
# using Monte Carlo simulation for the 1-of-4 plan based on individual
# observations using the 3rd highest background value.

predIntNparSimultaneousTestPower(n = 20, k = 1, m = 4, r = r,
  n.plus.one.minus.upl.rank = 3, delta.over.sigma = 2:4, r.shifted = 1,
  method = "simulate", ci = TRUE, NMC = 1000)
```

```
  #[1] 0.437 0.863 0.989
  #attr(,"conf.int")
  #         [,1]      [,2]      [,3]
  #LCL 0.4111999 0.8451148 0.9835747
  #UCL 0.4628001 0.8808852 0.9944253

## End(Not run)

  #==========

  # Cleanup
  #--------
  rm(n, nw, nc, ne, r, conf.level)
```

---

predIntPois *Prediction Interval for a Poisson Distribution*

Description

Estimate the mean of a Poisson distribution, and construct a prediction interval for the next k observations or next set of k sums.

Usage

```
predIntPois(x, k = 1, n.sum = 1, method = "conditional",
  pi.type = "two-sided", conf.level = 0.95, round.limits = TRUE)
```

Arguments

x	numeric vector of observations, or an object resulting from a call to an estimating function that assumes a Poisson distribution (i.e., epois or epoisCensored). If x is a numeric vector, missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
k	positive integer specifying the number of future observations or sums the prediction interval should contain with confidence level conf.level. The default value is k=1.
n.sum	positive integer specifying the sample size associated with the k future sums. The default value is n.sum=1 (i.e., individual observations). Note that all future sums must be based on the same sample size.
method	character string specifying the method to use. The possible values are: "conditional" (based on a conditional distribution; the default), "conditional.approx.normal" (method based on approximating a conditional distribution with the standard normal distribution), "conditional.approx.t" (method based on approximating a conditional distribution with Student's t-distribution), and "normal.approx" (approximate method based on the fact that the mean and varaince of a Poisson distribution are the same).

	See the DETAILS section for more information on these methods. The `"conditional"` method is only implemented for `k=1`; when `k` is bigger than 1, the value of `method` cannot be `"conditional"`.
`pi.type`	character string indicating what kind of prediction interval to compute. The possible values are `pi.type="two-sided"` (the default), `pi.type="lower"`, and `pi.type="upper"`.
`conf.level`	a scalar between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.
`round.limits`	logical scalar indicating whether to round the computed prediction limits to the nearest integer. The default value is `round.limits=TRUE`.

Details

A prediction interval for some population is an interval on the real line constructed so that it will contain k future observations or averages from that population with some specified probability $(1-\alpha)100\%$, where $0 < \alpha < 1$ and k is some pre-specified positive integer. The quantity $(1-\alpha)100\%$ is call the confidence coefficient or confidence level associated with the prediction interval.

In the case of a Poisson distribution, we have modified the usual meaning of a prediction interval and instead construct an interval that will contain k future observations or k future *sums* with a certain confidence level.

A prediction interval is a *random* interval; that is, the lower and/or upper bounds are random variables computed based on sample statistics in the baseline sample. Prior to taking one specific baseline sample, the probability that the prediction interval will contain the next k averages is $(1-\alpha)100\%$. Once a specific baseline sample is taken and the prediction interval based on that sample is computed, the probability that that prediction interval will contain the next k averages is not necessarily $(1-a)100\%$, but it should be close.

If an experiment is repeated N times, and for each experiment:

1. A sample is taken and a $(1-a)100\%$ prediction interval for $k = 1$ future observation is computed, and
2. One future observation is generated and compared to the prediction interval,

then the number of prediction intervals that actually contain the future observation generated in step 2 above is a binomial random variable with parameters `size=`N and `prob=`$(1-\alpha)100\%$ (see Binomial).

If, on the other hand, only one baseline sample is taken and only one prediction interval for $k = 1$ future observation is computed, then the number of future observations out of a total of N future observations that will be contained in that one prediction interval is a binomial random variable with parameters `size=`N and `prob=`$(1-\alpha^*)100\%$, where α^* depends on the true population parameters and the computed bounds of the prediction interval.

Because of the discrete nature of the Poisson distribution, even if the true mean of the distribution λ were known exactly, the actual confidence level associated with a prediction limit will usually not be exactly equal to $(1-\alpha)100\%$. For example, for the Poisson distribution with parameter `lambda=2`, the interval [0, 4] contains 94.7% of this distribution and the interval [0,5] contains 98.3% of this distribution. Thus, no interval can contain exactly 95% of this distribution, so it is impossible to construct an exact 95% prediction interval for the next $k = 1$ observation for a Poisson distribution

with parameter `lambda=2`.

The Form of a Poisson Prediction Interval

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a Poisson distribution with parameter `lambda=`λ. Also, let X denote the sum of these n random variables, i.e.,

$$X = \sum_{i=1}^{n} x_i \qquad (1)$$

Finally, let m denote the sample size associated with the k future sums (i.e., `n.sum=`m). When $m = 1$, each sum is really just a single observation, so in the rest of this help file the term "sums" replaces the phrase "observations or sums".

Let $\underline{y} = y_1, y_2, \ldots, y_m$ denote a vector of m future observations from a Poisson distribution with parameter `lambda=`λ^*, and set Y equal to the sum of these m random variables, i.e.,

$$Y = \sum_{i=1}^{m} y_i \qquad (2)$$

Then Y has a Poisson distribution with parameter `lambda=`$m\lambda^*$ (Johnson et al., 1992, p.160). We are interested in constructing a prediction limit for the next value of Y, or else the next k sums of m Poisson random variables, based on the observed value of X and assuming $\lambda^* = \lambda$.

For a Poisson distribution, the form of a two-sided prediction interval is:

$$[m\bar{x} - K, m\bar{x} + K] = [cX - K, cX + K] \qquad (3)$$

where

$$\bar{x} = \frac{X}{n} = \sum_{i=1}^{n} x_i \qquad (4)$$

$$c = \frac{m}{n} \qquad (5)$$

and K is a constant that depends on the sample size n, the confidence level $(1 - \alpha)100\%$, the number of future sums k, and the sample size associated with the future sums m. Do not confuse the constant K (uppercase K) with the number of future sums k (lowercase k). The symbol K is used here to be consistent with the notation used for prediction intervals for the normal distribution (see `predIntNorm`).

Similarly, the form of a one-sided lower prediction interval is:

$$[m\bar{x} - K, \infty] = [cX - K, \infty] \qquad (6)$$

and the form of a one-sided upper prediction interval is:

$$[0, m\bar{x} + K] = [0, cX + K] \qquad (7)$$

The derivation of the constant K is explained below.

Conditional Distribution (`method="conditional"`)

Nelson (1970) derives a prediction interval for the case $k = 1$ based on the conditional distribution of Y given $X + Y$. He notes that the conditional distribution of Y given the quantity $X + Y = w$

is binomial with parameters size=w and prob=$[m\lambda^*/(m\lambda^* + n\lambda)]$ (Johnson et al., 1992, p.161). When $k = 1$, the prediction limits are computed as those most extreme values of Y that still yield a non-significant test of the hypothesis $H_0 : \lambda^* = \lambda$, which for the conditional distribution of Y is equivalent to the hypothesis H_0: prob=[m /(m + n)].

Using the relationship between the binomial and F-distribution (see the explanation of exact confidence intervals in the help file for ebinom), Nelson (1982, p. 203) states that exact two-sided $(1 - \alpha)100\%$ prediction limits [LPL, UPL] are the closest integer solutions to the following equations:

$$\frac{m}{LPL + 1} = \frac{n}{X} F(2LPL + 2, 2X, 1 - \alpha/2) \qquad (8)$$

$$\frac{UPL}{n} = \frac{X + 1}{n} F(2X + 2, 2UPL, 1 - \alpha/2) \qquad (9)$$

where $F(\nu_1, \nu_2, p)$ denotes the p'th quantile of the F-distribution with ν_1 and ν_2 degrees of freedom.

If ci.type="lower", $\alpha/2$ is replaced with α in Equation (8) above for LPL, and UPL is set to ∞.

If ci.type="upper", $\alpha/2$ is replaced with α in Equation (9) above for UPL, and LPL is set to 0.

NOTE: This method is not extended to the case $k > 1$.

Conditional Distribution Approximation Based on Normal Distribution
(method="conditional.approx.normal")
Cox and Hinkley (1974, p.245) derive an approximate prediction interval for the case $k = 1$. Like Nelson (1970), they note that the conditional distribution of Y given the quantity $X + Y = w$ is binomial with parameters size=w and prob=$[m\lambda^*/(m\lambda^* + n\lambda)]$, and that the hypothesis $H_0 : \lambda^* = \lambda$ is equivalent to the hypothesis H_0: prob=[m /(m + n)].

Cox and Hinkley (1974, p.245) suggest using the normal approximation to the binomial distribution (in this case, without the continuity correction; see Zar, 2010, pp.534-536 for information on the continuity correction associated with the normal approximation to the binomial distribution). Under the null hypothesis $H_0 : \lambda^* = \lambda$, the quantity

$$z = [Y - \frac{c(X + Y)}{1 + c}]/\{[\frac{c(X + Y)}{(1 + c)^2}]^{1/2}\} \qquad (10)$$

is approximately distributed as a standard normal random variable.

The Case When k = 1
When $k = 1$ and pi.type="two-sided", the prediction limits are computed by solving the equation

$$z^2 \leq z^2_{1-\alpha/2} \qquad (11)$$

where z_p denotes the p'th quantile of the standard normal distribution. In this case, Gibbons (1987b) notes that the quantity K in Equation (3) above is given by:

$$K = \frac{t^2 c}{2} tc[X(1 + \frac{1}{c}) + \frac{t^2}{4}]^{1/2} \qquad (12)$$

where $t = z_{1-\alpha/2}$.

When pi.type="lower" or pi.type="upper", K is computed exactly as above, except t is set to $t = z_{1-\alpha}$.

The Case When k > 1

When $k > 1$, Gibbons (1987b) suggests using the Bonferroni inequality. That is, the value of K is computed exactly as for the case $k = 1$ described above, except that the Bonferroni value of t is used in place of the usual value of t:

When `pi.type="two-side"`, $t = z_{1-(\alpha/k)/2}$.

When `pi.type="lower"` or `pi.type="upper"`, $t = z_{1-\alpha/k}$.

Conditional Distribution Approximation Based on Student's t-Distribution (`method="conditional.approx.t"`)

When `method="conditional.approx.t"`, the exact same procedure is used as when `method="conditional.approx.normal"`, except that the quantity in Equation (10) is assumed to follow a Student's t-distribution with $n - 1$ degrees of freedom. Thus, all occurrences of z_p are replaced with $t_{n-1,p}$, where $t_{\nu,p}$ denotes the p'th quantile of Student's t-distribution with ν degrees of freedom.

Normal Approximation (`method="normal.approx"`)

The normal approximation for Poisson prediction limits was given by Nelson (1970; 1982, p.203) and is based on the fact that the mean and variance of a Poisson distribution are the same (Johnson et al, 1992, p.157), and for "large" values of n and m, both X and Y are approximately normally distributed.

The Case When k = 1

The quantity $Y - cX$ is approximately normally distributed with expectation and variance given by:

$$E(Y - cX) = E(Y) - cE(X) = m\lambda - cn\lambda = 0 \qquad (13)$$

$$Var(Y - cX) = Var(Y) + c^2 Var(X) = m\lambda + c^2 n\lambda = m\lambda(1 + \frac{m}{n}) \qquad (14)$$

so the quantity

$$z = \frac{Y - cX}{\sqrt{m\hat{\lambda}(1 + \frac{m}{n})}} = \frac{Y - cX}{\sqrt{m\bar{x}(1 + \frac{m}{n})}} \qquad (15)$$

is approximately distributed as a standard normal random variable. The function `predIntPois`, however, assumes this quantity is distributed as approximately a Student's t-distribution with $n - 1$ degrees of freedom.

Thus, following the idea of prediction intervals for a normal distribution (see `predIntNorm`), when `pi.type="two-sided"`, the constant K for a $(1 - \alpha)100\%$ prediction interval for the next $k = 1$ sum of m observations is computed as:

$$K = t_{n-1,1-\alpha/2}\sqrt{m\bar{x}(1 + \frac{m}{n})} \qquad (16)$$

where $t_{\nu,p}$ denotes the p'th quantile of a Student's t-distribution with ν degrees of freedom.

Similarly, when `pi.type="lower"` or `pi.type="upper"`, the constant K is computed as:

$$K = t_{n-1,1-\alpha}\sqrt{m\bar{x}(1 + \frac{m}{n})} \qquad (17)$$

The Case When k > 1

When $k > 1$, the value of K is computed exactly as for the case $k = 1$ described above, except that the Bonferroni value of t is used in place of the usual value of t:

When `pi.type="two-sided"`,

$$K = t_{n-1,1-(\alpha/k)/2}\sqrt{m\bar{x}(1 + \frac{m}{n})} \quad (18)$$

When `pi.type="lower"` or `pi.type="upper"`,

$$K = t_{n-1,1-(\alpha/k)}\sqrt{m\bar{x}(1 + \frac{m}{n})} \quad (19)$$

Hahn and Nelson (1973, p.182) discuss another method of computing K when $k > 1$, but this method is not implemented here.

Value

If `x` is a numeric vector, `predIntPois` returns a list of class `"estimate"` containing the estimated parameter, the prediction interval, and other information. See the help file for `estimate.object` for details.

If `x` is the result of calling an estimation function, `predIntPois` returns a list whose class is the same as `x`. The list contains the same components as `x`, as well as a component called `interval` containing the prediction interval information. If `x` already has a component called `interval`, this component is replaced with the prediction interval information.

Note

Prediction and tolerance intervals have long been applied to quality control and life testing problems. Nelson (1970) notes that his development of confidence and prediction limits for the Poisson distribution is based on well-known results dating back to the 1950's. Hahn and Nelson (1973) review predicion intervals for several distributions, including Poisson prediction intervals. The mongraph by Hahn and Meeker (1991) includes a discussion of Poisson prediction intervals.

Gibbons (1987b) uses the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical concentration (in ppb), and presents formulas for prediction and tolerance intervals. The formulas for prediction intervals are based on Cox and Hinkley (1974, p.245). Gibbons (1987b) only deals with the case where `n.sum=1`.

Gibbons et al. (2009, pp. 72–76) discuss methods for Poisson prediction limits.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Cox, D.R., and D.V. Hinkley. (1974). *Theoretical Statistics*. Chapman and Hall, New York, pp.242–245.

Gibbons, R.D. (1987b). Statistical Models for the Analysis of Volatile Organic Compounds in Waste Disposal Sites. *Ground Water* **25**, 572–580.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken, pp. 72–76.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hahn, G., and W. Nelson. (1973). A Survey of Prediction Intervals and Their Applications. *Journal of Quality Technology* **5**, 178–188.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 4.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Miller, R.G. (1981a). *Simultaneous Statistical Inference*. McGraw-Hill, New York, pp.8, 76–81.

Nelson, W.R. (1970). Confidence Intervals for the Ratio of Two Poisson Means and Poisson Predictor Intervals. *IEEE Transactions of Reliability* **R-19**, 42–49.

Nelson, W.R. (1982). *Applied Life Data Analysis*. John Wiley and Sons, New York, pp.200–204.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, pp. 585–586.

See Also

Poisson, epois, estimate.object, Prediction Intervals, tolIntPois, Estimating Distribution Parameters.

Examples

```
# Generate 20 observations from a Poisson distribution with parameter
# lambda=2.  The interval [0, 4] contains 94.7% of this distribution and
# the interval [0,5] contains 98.3% of this distribution.  Thus, because
# of the discrete nature of the Poisson distribution, no interval contains
# exactly 95% of this distribution.  Use predIntPois to estimate the mean
# parameter of the true distribution, and construct a one-sided upper
# 95% prediction interval for the next single observation from this distribution.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rpois(20, lambda = 2)

predIntPois(dat, pi.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Poisson
#
#Estimated Parameter(s):          lambda = 1.8
#
#Estimation Method:               mle/mme/mvue
```

```
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      conditional
#
#Prediction Interval Type:        upper
#
#Confidence Level:                95%
#
#Number of Future Observations:   1
#
#Prediction Interval:             LPL = 0
#                                 UPL = 5

#----------

# Compare results above with the other approximation methods:

predIntPois(dat, method = "conditional.approx.normal",
  pi.type = "upper")$interval$limits
#LPL UPL
#  0   4


predIntPois(dat, method = "conditional.approx.t",
  pi.type = "upper")$interval$limits
#LPL UPL
#  0   4


predIntPois(dat, method = "normal.approx",
  pi.type = "upper")$interval$limits
#LPL UPL
#  0   4
#Warning message:
#In predIntPois(dat, method = "normal.approx", pi.type = "upper") :
#  Estimated value of 'lambda' and/or number of future observations
#  is/are probably too small for the normal approximation to work well.

#==========

# Using the same data as in the previous example, compute a one-sided
# upper 95% prediction limit for k=10 future observations.

# Using conditional approximation method based on the normal distribution.

predIntPois(dat, k = 10, method = "conditional.approx.normal",
  pi.type = "upper")

#Results of Distribution Parameter Estimation
#--------------------------------------------
```

```
#
#Assumed Distribution:            Poisson
#
#Estimated Parameter(s):          lambda = 1.8
#
#Estimation Method:               mle/mme/mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Prediction Interval Method:      conditional.approx.normal
#
#Prediction Interval Type:        upper
#
#Confidence Level:                95%
#
#Number of Future Observations:   10
#
#Prediction Interval:             LPL = 0
#                                 UPL = 6


# Using method based on approximating conditional distribution with
# Student's t-distribution

predIntPois(dat, k = 10, method = "conditional.approx.t",
  pi.type = "upper")$interval$limits
#LPL UPL
#  0   6

#==========

# Repeat the above example, but set k=5 and n.sum=3.  Thus, we want a
# 95% upper prediction limit for the next 5 sets of sums of 3 observations.

predIntPois(dat, k = 5, n.sum = 3, method = "conditional.approx.t",
  pi.type = "upper")$interval$limits
#LPL UPL
#  0  12

#==========

# Reproduce Example 3.6 in Gibbons et al. (2009, p. 75)
# A 32-constituent VOC scan was performed for n=16 upgradient
# samples and there were 5 detections out of these 16.  We
# want to construct a one-sided upper 95% prediction limit
# for 20 monitoring wells (so k=20 future observations) based
# on these data.

# First we need to create a data set that will yield a mean
# of 5/16 based on a sample size of 16.  Any number of data
# sets will do.  Here are two possible ones:
```

```
dat <- c(rep(1, 5), rep(0, 11))
dat <- c(2, rep(1, 3), rep(0, 12))

# Now call predIntPois.  Don't round the limits so we can
# compare to the example in Gibbons et al. (2009).

predIntPois(dat, k = 20, method = "conditional.approx.t",
  pi.type = "upper", round.limits = FALSE)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Poisson
#
#Estimated Parameter(s):          lambda = 0.3125
#
#Estimation Method:               mle/mme/mvue
#
#Data:                            dat
#
#Sample Size:                     16
#
#Prediction Interval Method:      conditional.approx.t
#
#Prediction Interval Type:        upper
#
#Confidence Level:                95%
#
#Number of Future Observations:   20
#
#Prediction Interval:             LPL = 0.000000
#                                 UPL = 2.573258

#==========

# Cleanup
#--------
rm(dat)
```

---

print *Print Values*

Description

The **EnvStats** function print is a generic function for printing its argument and returning it invisibly (via invisible(x)). The function invokes particular methods which depend on the class of the first argument. When given an argument of class "htest", the **EnvStats** function print.default calls the **EnvStats** function link{print.htest}, otherwise it calls the R function print.default.

The **EnvStats** functions print and print.default have been created in order to comply with CRAN policies, because **EnvStats** contains a modified version of the R function print.htest.

Usage

```
print(x, ...)

## Default S3 method:
print(x, ...)
```

Arguments

x	an object to be printed and used to select a printing method.
...	further arguments passed to or from other methods. See the help files for the R functions print and print.default for more information.

Details

The **EnvStats** function print is a generic function for printing its argument and returning it invisibly (via invisible(x)). The function invokes particular methods which depend on the class of the first argument. When given an argument of class "htest", the **EnvStats** function print.default calls the **EnvStats** function link{print.htest}, otherwise it calls the R function print.default.

The **EnvStats** functions print and print.default have been created in order to comply with CRAN policies, because **EnvStats** contains a modified version of the R function print.htest. When **EnvStats** is loaded, objects of class "htest" will be printed using the **EnvStats** version of print.htest.

See the help files for the R functions print and print.default.

Value

See the help files for the R functions print and print.default.

Author(s)

R Development Core Team for code for R versions of print and print.default.

Steven P. Millard for **EnvStats** version of print.default; <EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., and Hastie, T.J., eds. (1992). *Statistical Models in S*. Chapman and Hall/CRC, Boca Raton, FL.

See Also

R help file for print, R help file for print.default, print.htest.

print.boxcox *Print Output of Objective for Box-Cox Power Transformations*

Description

Formats and prints the results of calling the function boxcox. This method is automatically called by print when given an object of class "boxcox". The names of other functions involved in Box-Cox transformations are listed under Data Transformations.

Usage

```
## S3 method for class 'boxcox'
print(x, ...)
```

Arguments

x	an object of class "boxcox". See boxcox.object for details.
...	arguments that can be supplied to the format function.

Details

This is the "boxcox" method for the generic function print. Prints the objective name, the name of the data object used, the sample size, the values of the powers, and the values of the objective. In the case of optimization, also prints the range of powers over which the optimization took place.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

boxcox, boxcox.object, plot.boxcox, Data Transformations, print.

`print.boxcoxCensored` *Print Output of Objective for Box-Cox Power Transformations Based on Type I Censored Data*

Description

Formats and prints the results of calling the function `boxcoxCensored`. This method is automatically called by `print` when given an object of class `"boxcoxCensored"`.

Usage

```
## S3 method for class 'boxcoxCensored'
print(x, ...)
```

Arguments

`x`	an object of class `"boxcoxCensored"`. See `boxcoxCensored.object` for details.
`...`	arguments that can be supplied to the `format` function.

Details

This is the `"boxcoxCensored"` method for the generic function `print`. Prints the objective name, the name of the data object used, the sample size, the percentage of censored observations, the values of the powers, and the values of the objective. In the case of optimization, also prints the range of powers over which the optimization took place.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`boxcoxCensored`, `boxcoxCensored.object`, `plot.boxcoxCensored`, Data Transformations, `print`.

`print.boxcoxLm` *Print Output of Objective for Box-Cox Power Transformations for an "lm" Object*

Description

Formats and prints the results of calling the function `boxcox` when the argument `x` supplied to `boxcox` is an object of class `"lm"`. This method is automatically called by `print` when given an object of class `"boxcoxLm"`. The names of other functions involved in Box-Cox transformations are listed under Data Transformations.

Usage

```
## S3 method for class 'boxcoxLm'
print(x, ...)
```

Arguments

`x`	an object of class `"boxcoxLm"`. See `boxcoxLm.object` for details.
`...`	arguments that can be supplied to the `format` function.

Details

This is the `"boxcoxLm"` method for the generic function `print`. Prints the objective name, the details of the `"lm"` object used, the sample size, the values of the powers, and the values of the objective. In the case of optimization, also prints the range of powers over which the optimization took place.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`boxcox`, `boxcoxLm.object`, `plot.boxcoxLm`, Data Transformations, `print`.

`print.distChoose` *Print Output of Goodness-of-Fit Tests*

Description

Formats and prints the results of calling the function `distChoose`, which uses a series of goodness-of-fit tests to choose among candidate distributions. This method is automatically called by `print` when given an object of class `"distChoose"`.

Usage

```
## S3 method for class 'distChoose'
print(x, ...)
```

Arguments

`x`	an object of class `"distChoose"`. See `distChoose.object` for details.
`...`	arguments that can be supplied to the `format` function.

Details

This is the `"distChoose"` method for the generic function `print`. Prints the candidate distributions, method used to choose among the candidate distributions, chosen distribution, Type I error associated with each goodness-of-fit test, estimated population parameter(s) associated with the chosen distribution, estimation method, goodness-of-fit test results for each candidate distribution, and the data name.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`distChoose`, `distChoose.object`, Goodness-of-Fit Tests, `print`.

print.distChooseCensored
Print Output of Goodness-of-Fit Tests

Description

Formats and prints the results of calling the function distChooseCensored, which uses a series of goodness-of-fit tests to choose among candidate distributions based on censored data. This method is automatically called by print when given an object of class "distChooseCensored".

Usage

```
## S3 method for class 'distChooseCensored'
print(x, show.cen.levels = TRUE,
  pct.censored.digits = .Options$digits, ...)
```

Arguments

x	an object of class "distChoose". See distChoose.object for details.
show.cen.levels	logical scalar indicating whether to print the censoring levels. The default is show.cen.levels=TRUE.
pct.censored.digits	numeric scalar indicating the number of significant digits to print for the percent of censored observations.
...	arguments that can be supplied to the format function.

Details

This is the "distChooseCensored" method for the generic function print. Prints the candidate distributions, method used to choose among the candidate distributions, chosen distribution, Type I error associated with each goodness-of-fit test, estimated population parameter(s) associated with the chosen distribution, estimation method, goodness-of-fit test results for each candidate distribution, and the data name and censoring variable.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

distChooseCensored, distChooseCensored.object, Censored Data, print.

---

print.estimate *Print Objects of Class "estimate"*

Description

Formats and prints the results of **EnvStats** functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of data assumed to come from that distribution. This method is automatically called by print when given an object of class "estimate".

See the help files Estimating Distribution Parameters and Estimating Distribution Quantiles for lists of functions that estimate distribution parameters and quantiles. See the help files Prediction Intervals and Tolerance Intervals for lists of functions that create prediction and tolerance intervals.

Usage

```
## S3 method for class 'estimate'
print(x, conf.cov.sig.digits = .Options$digits,
  limits.sig.digits = .Options$digits, ...)
```

Arguments

x	an object of class "estimate". See estimate.object for details.
conf.cov.sig.digits	numeric scalar indicating the number of significant digits to print for the confidence level or coverage of a confidence, prediction, or tolerance interval.
limits.sig.digits	numeric scalar indicating the number of significant digits to print for the upper and lower limits of a confidence, prediction, or tolerance interval.
...	arguments that can be supplied to the format function.

Details

This is the "estimate" method for the generic function print. Prints estimated parameters and, if present in the object, information regarding confidence, prediction, or tolerance intervals.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

estimate.object, Estimating Distribution Parameters, Estimating Distribution Quantiles, Prediction Intervals, Tolerance Intervals, print.

---

print.estimateCensored

Print Objects of Class "estimateCensored"

Description

Formats and prints the results of **EnvStats** functions that estimate the parameters or quantiles of a probability distribution and optionally construct confidence, prediction, or tolerance intervals based on a sample of Tyep I censored data assumed to come from that distribution. This method is automatically called by print when given an object of class "estimateCensored".

See the subsections *Estimating Distribution Parameters* and *Estimating Distribution Quantiles* in the help file Censored Data for lists of functions that estimate distribution parameters and quantiles based on Type I censored data.

See the subsection *Prediction and Tolerance Intervals* in the help file Censored Data for lists of functions that create prediction and tolerance intervals.

Usage

```
## S3 method for class 'estimateCensored'
print(x, show.cen.levels = TRUE,
  pct.censored.digits = .Options$digits,
  conf.cov.sig.digits = .Options$digits, limits.sig.digits = .Options$digits,
  ...)
```

Arguments

x	an object of class "estimateCensored". See estimateCensored.object for details.
show.cen.levels	logical scalar indicating whether to print the censoring levels. The default is show.cen.levels=TRUE.
pct.censored.digits	numeric scalar indicating the number of significant digits to print for the percent of censored observations.
conf.cov.sig.digits	numeric scalar indicating the number of significant digits to print for the confidence level or coverage of a confidence, prediction, or tolerance interval.

limits.sig.digits
: numeric scalar indicating the number of significant digits to print for the upper and lower limits of a confidence, prediction, or tolerance interval.

... : arguments that can be supplied to the format function.

Details

This is the "estimateCensored" method for the generic function print. Prints estimated parameters and, if present in the object, information regarding confidence, prediction, or tolerance intervals.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

estimateCensored.object, Censored Data, print.

---

print.gof *Print Output of Goodness-of-Fit Tests*

Description

Formats and prints the results of performing a goodness-of-fit test. This method is automatically called by print when given an object of class "gof". The names of the functions that perform goodness-of-fit tests and that produce objects of class "gof" are listed under Goodness-of-Fit Tests.

Usage

```
## S3 method for class 'gof'
print(x, ...)
```

Arguments

x
: an object of class "gof". See gof.object for details.

... : arguments that can be supplied to the format function.

Details

This is the "gof" method for the generic function print. Prints name of the test, hypothesized distribution, estimated population parameter(s), estimation method, data name, sample size, value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and the alternative hypothesis.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

Goodness-of-Fit Tests, gof.object, print.

---

print.gofCensored *Print Output of Goodness-of-Fit Tests Based on Censored Data*

Description

Formats and prints the results of performing a goodness-of-fit test. This method is automatically called by print when given an object of class "gofCensored". Currently, the only function that produces an object of this class is gofTestCensored.

Usage

```
## S3 method for class 'gofCensored'
print(x, show.cen.levels = TRUE,
  pct.censored.digits = .Options$digits, ...)
```

Arguments

x	an object of class "gofCensored". See gofCensored.object for details.
show.cen.levels	logical scalar indicating whether to print the censoring levels. The default is show.cen.levels=TRUE.
pct.censored.digits	numeric scalar indicating the number of significant digits to print for the percent of censored observations.
...	arguments that can be supplied to the format function.

Details

This is the "gofCensored" method for the generic function print. Prints name of the test, hypothesized distribution, estimated population parameter(s), estimation method, data name, sample size, censoring information, value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and the alternative hypothesis.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

Censored Data, gofCensored.object, print.

---

print.gofGroup *Print Output of Group Goodness-of-Fit Tests*

Description

Formats and prints the results of performing a group goodness-of-fit test. This method is automatically called by print when given an object of class "gofGroup". Currently, the only **EnvStats** function that performs a group goodness-of-fit test that produces an object of class "gofGroup" is gofGroupTest.

Usage

```
## S3 method for class 'gofGroup'
print(x, ...)
```

Arguments

x	an object of class "gofGroup". See gofGroup.object for details.
...	arguments that can be supplied to the format function.

Details

This is the "gofGroup" method for the generic function print. See the help file for gofGroup.object for information on the information contained in this kind of object.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

Goodness-of-Fit Tests, `gofGroup.object`, `print`.

`print.gofOutlier` *Print Output of Goodness-of-Fit Outlier Tests*

Description

Formats and prints the results of performing a goodness-of-fit test for outliers. This method is automatically called by `print` when given an object of class `"gofOutlier"`. The names of the functions that perform goodness-of-fit tests for outliers and that produce objects of class `"gofOutlier"` are listed under Tests for Outliers.

Usage

```
## S3 method for class 'gofOutlier'
print(x, ...)
```

Arguments

`x`	an object of class `"gofOutlier"`. See `gofOutlier.object` for details.
`...`	arguments that can be supplied to the `format` function.

Details

This is the `"gofOutlier"` method for the generic function `print`. Prints name of the test, hypothesized distribution, data name, sample size, value of the test statistic, parameters associated with the null distribution of the test statistic, Type I error, critical values, and the alternative hypothesis.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

Tests for Outliers, `gofOutlier.object`, `print`.

`print.gofTwoSample` *Print Output of Two-Sample Goodness-of-Fit Tests*

Description

Formats and prints the results of performing a two-sample goodness-of-fit test. This method is automatically called by `print` when given an object of class `"gofTwoSample"`. Currently, the only **EnvStats** function that performs a two-sample goodness-of-fit test that produces an object of class `"gofTwoSample"` is `gofTest`.

Usage

```
## S3 method for class 'gofTwoSample'
print(x, ...)
```

Arguments

`x`	an object of class `"gofTwoSample"`. See `gofTwoSample.object` for details.
`...`	arguments that can be supplied to the `format` function.

Details

This is the `"gofTwoSample"` method for the generic function `print`. See the help file for `gofTwoSample.object` for information on the information contained in this kind of object.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

Goodness-of-Fit Tests, `gofTwoSample.object`, `print`.

print.htest.htestEnvStats
Print Output of Hypothesis Tests

Description

Print objects of class "htest" or "htestEnvStats", respectively, by simple print methods.

The functions print.htest and print.htestEnvStats are identical, and are a modification of the R function print.htest.

The function print.htestEnvStats formats and prints the results of performing a hypothesis test that was performed using one of the functions listed in the help file for htestEnvStats.object. This method is automatically called by the **EnvStats** generic function print when given an object of class "htestEnvStats".

The function print.htest formats and prints the results of performing a hypothesis test that was performed using a function that returns an object of class "htest". Because of the design of R, the **EnvStats** method print.htest is NOT automatically called when the user types a command at the command prompt that returns an object of class "htest". Instead, the **EnvStats** print function must be explicitly called to invoke print.htest. See the EXAMPLES section below.

Usage

```
  ## S3 method for class 'htest'
print(x, ...)
  ## S3 method for class 'htestEnvStats'
print(x, ...)
```

Arguments

x	an object of class "htest" or "htestEnvStats". See htest.object and htestEnvStats.object for details.
...	arguments that can be supplied to the format function.

Details

The function print.htest is the "htest" method for the **EnvStats** generic function print, and the function print.htestEnvStats is the "htestEnvStats" method for the **EnvStats** generic function print. These functions print null and alternative hypotheses, name of the test, estimated population parameter(s) involved in the null hypothesis, estimation method (if present), data name, sample size (if present), number of missing observations removed prior to performing the test (if present), value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and confidence interval for the population parameter (if present).

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

Hypothesis Tests, `htest.object`, `htestEnvStats.object`, `print`.

Examples

```
# Create an object of class "htestEnvStats", then print it out.
#-------------------------------------------------------------
htestEnvStats.obj <- chenTTest(EPA.02d.Ex.9.mg.per.L.vec, mu = 30)

mode(htestEnvStats.obj)
#[1] "list"

class(htestEnvStats.obj)
#[1] "htestEnvStats"

names(htestEnvStats.obj)
# [1] "statistic"   "parameters"  "p.value"     "estimate"
# [5] "null.value"  "alternative" "method"      "sample.size"
# [9] "data.name"   "bad.obs"     "interval"

htestEnvStats.obj

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 mean = 30
#
#Alternative Hypothesis:          True mean is greater than 30
#
#Test Name:                       One-sample t-Test
#                                 Modified for
#                                 Positively-Skewed Distributions
#                                 (Chen, 1995)
#
#Estimated Parameter(s):          mean = 34.566667
#                                 sd   = 27.330598
#                                 skew =  2.365778
#
#Data:                            EPA.02d.Ex.9.mg.per.L.vec
#
#Sample Size:                     60
#
#Test Statistic:                  t = 1.574075
#
```

```
#Test Statistic Parameter:        df = 59
#
#P-values:                        z               = 0.05773508
#                                 t               = 0.06040889
#                                 Avg. of z and t = 0.05907199
#
#Confidence Interval for:         mean
#
#Confidence Interval Method:      Based on z
#
#Confidence Interval Type:        Lower
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 29.82
#                                 UCL =   Inf

#==========================================================================

# Now create an object of class "htest" and note the difference in how it is
# printed out depending on whether or not you explicitly use the print() command.
#--------------------------------------------------------------------------------

htest.obj <- t.test(EPA.02d.Ex.9.mg.per.L.vec, mu = 30, alternative = "greater")

class(htest.obj)
#[1] "htest"

names(htest.obj)
# [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
# [6] "null.value"  "stderr"      "alternative" "method"      "data.name"

htest.obj

#        One Sample t-test
#
#data:  EPA.02d.Ex.9.mg.per.L.vec
#t = 1.2943, df = 59, p-value = 0.1003
#alternative hypothesis: true mean is greater than 30
#95 percent confidence interval:
# 28.67044      Inf
#sample estimates:
#mean of x
# 34.56667

print(htest.obj)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 mean = 30
#
#Alternative Hypothesis:          True mean is greater than 30
```

```
#
#Test Name:                          One Sample t-test
#
#Estimated Parameter(s):             mean of x = 34.56667
#
#Data:                               EPA.02d.Ex.9.mg.per.L.vec
#
#Test Statistic:                     t = 1.294273
#
#Test Statistic Parameter:           df = 59
#
#P-value:                            0.1003072
#
#95% Confidence Interval:            LCL = 28.67044
#                                    UCL =      Inf

#==========

# Clean up
#---------
rm(htestEnvStats.obj, htest.obj)
```

`print.htestCensored` *Print Output of Hypothesis Tests Based on Censored Data*

Description

Formats and prints the results of performing a hypothesis test based on censored data. This method is automatically called by `print` when given an object of class `"htestCensored"`. The names of the **EnvStats** functions that perform hypothesis tests based on censored data and that produce objects of class `"htestCensored"` are listed in the section **Hypothesis Tests** in the help file EnvStats Functions for Censored Data. Currently, the only function listed is `twoSampleLinearRankTestCensored`.

Usage

```
## S3 method for class 'htestCensored'
print(x, show.cen.levels = TRUE, ...)
```

Arguments

`x`	an object of class `"htestCensored"`. See `htestCensored.object` for details.
`show.cen.levels`	logical scalar indicating whether to print the censoring levels. The default value is `show.cen.levels=TRUE`.
`...`	arguments that can be supplied to the `format` function.

Details

This is the "htestCensored" method for the generic function print. Prints null and alternative hypotheses, name of the test, censoring side, estimated population parameter(s) involved in the null hypothesis, estimation method (if present), data name, censoring variable, sample size (if present), percent of observations that are censored, number of missing observations removed prior to performing the test (if present), value of the test statistic, parameters associated with the null distribution of the test statistic, p-value associated with the test statistic, and confidence interval for the population parameter (if present).

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

Censored Data, htestCensored.object, print.

---

print.permutationTest *Print Output of Permutation Tests*

Description

Formats and prints the results of performing a permutation test. This method is automatically called by print when given an object of class "permutationTest". Currently, the **EnvStats** functions that perform permutation tests and produce objects of class "permutationTest" are: oneSamplePermutationTest, twoSamplePermutationTestLocation, and twoSamplePermutationTestProportion.

Usage

```
## S3 method for class 'permutationTest'
print(x, ...)
```

Arguments

x	an object of class "permutationTest". See permutationTest.object for details.
...	arguments that can be supplied to the format function.

Details

This is the "permutationTest" method for the generic function print. Prints null and alternative hypotheses, name of the test, estimated population parameter(s) involved in the null hypothesis, estimation method (if present), data name, sample size (if present), number of missing observations removed prior to performing the test (if present), value of the test statistic, parameters associated with the null distribution of the test statistic, and p-value associated with the test statistic.

Value

Invisibly returns the input x.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

permutationTest.object, oneSamplePermutationTest, twoSamplePermutationTestLocation, twoSamplePermutationTestProportion, Hypothesis Tests, print.

---

print.summaryStats *Print Summary Statistics*

Description

Formats and prints the results of calling summaryStats or summaryFull. This method is automatically called by print when given an object of class "summaryStats".

Usage

```
## S3 method for class 'summaryStats'
print(x, ...)
```

Arguments

x	an object of class "summaryStats". See summaryStats.object for details.
...	arguments that can be supplied to the format function.

Details

This is the "summaryStats" method for the generic function print. Prints summary statistics.

Value

Invisibly returns the input `x`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`summaryStats`, `summaryFull`, `summaryStats.object`, `print`.

---

propTestMdd *Minimal Detectable Difference Associated with a One- or Two-Sample Proportion Test*

Description

Compute the minimal detectable difference associated with a one- or two-sample proportion test, given the sample size, power, and significance level.

Usage

```
propTestMdd(n.or.n1, n2 = n.or.n1, p0.or.p2 = 0.5, alpha = 0.05, power = 0.95,
  sample.type = "one.sample", alternative = "two.sided",
  two.sided.direction = "greater", approx = TRUE,
  correct = sample.type == "two.sample", warn = TRUE,
  return.exact.list = TRUE, tol = 1e-07, maxiter = 1000)
```

Arguments

`n.or.n1`	numeric vector of sample sizes. When `sample.type="one.sample"`, this argument denotes n, the number of observations in the single sample. When `sample.type="two.sample"`, this argument denotes n_1, the number of observations from group 1. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`n2`	numeric vector of sample sizes for group 2. The default value is `n2=n.or.n1`. This argument is ignored when `sample.type="one.sample"`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`p0.or.p2`	numeric vector of proportions. When `sample.type="one.sample"`, this argument denotes the hypothesized value of p, the probability of "success". When `sample.type="two.sample"`, this argument denotes the value of p_2, the probability of "success" in group 2. The default value is `p0.or.p2=0.5`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.

`alpha` numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.

`power` numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.

`sample.type` character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type="one.sample"`, the computed power is based on a hypothesis test for a single proportion. When `sample.type="two.sample"`, the computed power is based on a hypothesis test for the difference between two proportions. The default value is `sample.type="one.sample"`.

`alternative` character string indicating the kind of alternative hypothesis. The possible values are `"two.sided"` (the default), `"less"`, and `"greater"`.

`two.sided.direction`
character string indicating the direction (positive or negative) for the minimal detectable difference when `alternative="two.sided"`. When `two.sided.direction="greater"` (the default), the minimal detectable difference is positive. When `two.sided.direction="less"`, the minimal detectable difference is negative. This argument is ignored if `alternative="less"` or `alternative="greater"`.

`approx` logical scalar indicating whether to compute the power based on the normal approximation to the binomial distribution. The default value is `approx=TRUE`. Currently, the exact method (`approx=FALSE`) is only available for the one-sample case (i.e., `sample.type="one.sample"`).

`correct` logical scalar indicating whether to use the continuity correction when `approx=TRUE`. The default value is `approx=TRUE` when `sample.type="two.sample"` and `approx=FALSE` when `sample.type="one.sample"`. This argument is ignored when `approx=FALSE`.

`warn` logical scalar indicating whether to issue a warning. The default value is `warn=TRUE`. When `approx=TRUE` (power based on the normal approximation) and `warn=TRUE`, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate. When `approx=FALSE` (power based on the exact test) and `warn=TRUE`, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of `alpha`.

`return.exact.list`
logical scalar relevant to the case when `approx=FALSE` (i.e., when the power is based on the exact test). This argument indicates whether to return a list containing extra information about the exact test in addition to the power of the exact test. By default, `propTestMdd` returns only a vector containing the computed minimal detectable difference(s) (see the VALUE section below). When `return.exact.list=TRUE` (the default) and `approx=FALSE`, `propTestMdd` returns a list with components indicating the minimal detectable difference(s), power of the exact test, the true significance level associated with the exact test, and the critical values associated with the exact test (see the DETAILS section for more information).

`tol` numeric scalar passed to the `uniroot` function that indicates the tolerance to use in the search algorithm. The default value is `tol=1e-7`.

`maxiter` integer passed to the `uniroot` function that indicates the maximum number of iterations to use in the search algorithm. The default value is `maxiter=1000`.

Details

If the arguments `n.or.n1`, `n2`, `p0.or.p2`, `alpha`, and `power` are not all the same length, they are replicated to be the same length as the length of the longest argument.

One-Sample Case (`sample.type="one.sample"`)
The help file for `propTestPower` gives references that explain how the power of the one-sample proportion test is computed based on the values of p_0 (the hypothesized value for p, the probability of "success"), p (the true value of p), the sample size n, and the Type I error level α. The function `propTestMdd` computes the value of the minimal detectable difference $p - p_0$ for specified values of sample size, power, and Type I error level by calling the `uniroot` function to perform a search.

Two-Sample Case (`sample.type="two.sample"`)
The help file for `propTestPower` gives references that explain how the power of the two-sample proportion test is computed based on the values of p_1 (the value of the probability of "success" for group 1), p_2 (the value of the probability of "success" for group 2), the sample sizes for groups 1 and 2 (n_1 and n_2), and the Type I error level α. The function `propTestMdd` computes the value of the minimal detectable difference $p_1 - p_2$ for specified values of sample size, power, and Type I error level by calling the `uniroot` function to perform a search.

Value

Approximate Test (`approx=TRUE`). numeric vector of minimal detectable differences.

Exact Test (`approx=FALSE`). If `return.exact.list=FALSE`, `propTestMdd` returns a numeric vector of minimal detectable differences.

If `return.exact.list=TRUE`, `propTestMdd` returns a list with the following components:

`delta` numeric vector of minimal detectable differences.

`power` numeric vector of powers.

`alpha` numeric vector containing the true significance levels. Because of the discrete nature of the binomial distribution, the true significance levels usually do not equal the significance level supplied by the user in the argument `alpha`.

`q.critical.lower`
numeric vector of lower critical values for rejecting the null hypothesis. If the observed number of "successes" is *less than or equal to* these values, the null hypothesis is rejected. (Not present if `alternative="greater"`.)

`q.critical.upper`
numeric vector of upper critical values for rejecting the null hypothesis. If the observed number of "successes" is *greater than* these values, the null hypothesis is rejected. (Not present if `alternative="less"`.)

Note

See the help file for `propTestPower`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for propTestPower.

See Also

propTestPower, propTestN, plotPropTestDesign, prop.test, binom.test.

Examples

```
# Look at how the minimal detectable difference of the one-sample
# proportion test increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

mdd <- propTestMdd(n.or.n1 = 50, power = seq(0.5, 0.9, by=0.1))

round(mdd, 2)
#[1] 0.14 0.16 0.17 0.19 0.22

#----------

# Repeat the last example, but compute the minimal detectable difference
# based on the exact test instead of the approximation.  Note that with a
# sample size of 50, the largest significance level less than or equal to
# 0.05 for the two-sided alternative is 0.03.

mdd.list <- propTestMdd(n.or.n1 = 50, power = seq(0.5, 0.9, by = 0.1),
  approx = FALSE)

lapply(mdd.list, round, 2)
#$delta
#[1] 0.15 0.17 0.18 0.20 0.23
#
#$power
#[1] 0.5 0.6 0.7 0.8 0.9
#
#$alpha
#[1] 0.03 0.03 0.03 0.03 0.03
#
#$q.critical.lower
#[1] 17 17 17 17 17
#
#$q.critical.upper
#[1] 32 32 32 32 32

#==========
```

```
# Look at how the minimal detectable difference for the two-sample
# proportion test decreases with increasing sample sizes.  Note that for
# the specified significance level, power, and true proportion in group 2,
# no minimal detectable difference is attainable for a sample size of 10 in
# each group.

seq(10, 50, by=10)
#[1] 10 20 30 40 50

propTestMdd(n.or.n1 = seq(10, 50, by = 10), p0.or.p2 = 0.5,
  sample.type = "two", alternative="greater")
#[1]        NA 0.4726348 0.4023564 0.3557916 0.3221412
#Warning messages:
#1: In propTestMdd(n.or.n1 = seq(10, 50, by = 10), p0.or.p2 = 0.5,
#     sample.type = "two",  :
#  Elements with a missing value (NA) indicate no attainable minimal detectable
#    difference for the given values of 'n1', 'n2', 'p2', 'alpha', and 'power'
#2: In propTestMdd(n.or.n1 = seq(10, 50, by = 10), p0.or.p2 = 0.5,
#      sample.type = "two",  :
#  The sample sizes 'n1' and 'n2' are too small, relative to the computed value
#    of 'p1' and the given value of 'p2', for the normal approximation to work
#    well for the following element indices:
#         2 3

#----------

# Look at how the minimal detectable difference for the two-sample proportion
# test decreases with increasing values of Type I error:

mdd <- propTestMdd(n.or.n1 = 100, n2 = 120, p0.or.p2 = 0.4, sample.type = "two",
   alpha = c(0.01, 0.05, 0.1, 0.2))

round(mdd, 2)
#[1] 0.29 0.25 0.23 0.20

#----------

# Clean up
#---------
rm(mdd, mdd.list)

#==========

# Modifying the example on pages 8-5 to 8-7 of USEPA (1989b), determine the
# minimal detectable difference to detect a difference in the proportion of
# detects of cadmium between the background and compliance wells.  Set the
# compliance well to "group 1" and the background well to "group 2".  Assume
# the true probability of a "detect" at the background well is 1/3, use a
# 5% significance level, use 80%, 90%, and 95% power, use the given sample
# sizes of 64 observations at the compliance well and 24 observations at the
# background well, and use the upper one-sided alternative (probability of a
# "detect" at the compliance well is greater than the probability of a "detect"
# at the background well).
```

```
# (The data are stored in EPA.89b.cadmium.df.)
#
# Note that the minimal detectable difference increases from 0.32 to 0.37 to 0.40 as
# the required power increases from 80% to 90% to 95%.  Thus, in order to detect a
# difference in probability of detection between the compliance and background
# wells, the probability of detection at the compliance well must be 0.65, 0.70,
# or 0.74 (depending on the required power).

EPA.89b.cadmium.df
#   Cadmium.orig Cadmium Censored  Well.type
#1           0.1   0.100    FALSE Background
#2          0.12   0.120    FALSE Background
#3           BDL   0.000     TRUE Background
# ......................................
#86          BDL   0.000     TRUE Compliance
#87          BDL   0.000     TRUE Compliance
#88          BDL   0.000     TRUE Compliance

p.hat.back <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Background"]))

p.hat.back
#[1] 0.3333333

p.hat.comp <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Compliance"]))

p.hat.comp
#[1] 0.375

n.back <- with(EPA.89b.cadmium.df,
  sum(Well.type == "Background"))

n.back
#[1] 24

n.comp <- with(EPA.89b.cadmium.df,
  sum(Well.type == "Compliance"))

n.comp
#[1] 64

mdd <- propTestMdd(n.or.n1 = n.comp, n2 = n.back,
  p0.or.p2 = p.hat.back, power = c(.80, .90, .95),
  sample.type = "two", alternative = "greater")

round(mdd, 2)
#[1] 0.32 0.37 0.40

round(mdd + p.hat.back, 2)
#[1] 0.65 0.70 0.73

#----------
```

```
# Clean up
#---------
rm(p.hat.back, p.hat.comp, n.back, n.comp, mdd)
```

`propTestN` *Compute Sample Size Necessary to Achieve a Specified Power for a One- or Two-Sample Proportion Test*

Description

Compute the sample size necessary to achieve a specified power for a one- or two-sample proportion test, given the true proportion(s) and significance level.

Usage

```
propTestN(p.or.p1, p0.or.p2, alpha = 0.05, power = 0.95,
  sample.type = "one.sample", alternative = "two.sided",
  ratio = 1, approx = TRUE,
  correct = sample.type == "two.sample",
  round.up = TRUE, warn = TRUE, return.exact.list = TRUE,
  n.min = 2, n.max = 10000, tol.alpha = 0.1 * alpha,
  tol = 1e-7, maxiter = 1000)
```

Arguments

`p.or.p1`	numeric vector of proportions. When `sample.type="one.sample"`, this argument denotes the true value of p, the probability of "success". When `sample.type="two.sample"`, this argument denotes the value of p_1, the probability of "success" in group 1. The default value is `p.or.p1=0.5`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`p0.or.p2`	numeric vector of proportions. When `sample.type="one.sample"`, this argument denotes the hypothesized value of p, the probability of "success". When `sample.type="two.sample"`, this argument denotes the value of p_2, the probability of "success" in group 2. The default value is `p0.or.p2=0.5`. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`alpha`	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.
`power`	numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.
`sample.type`	character string indicating whether to compute sample size based on a one-sample or two-sample hypothesis test. When `sample.type="one.sample"`, the computed sample size is based on a hypothesis test for a single proportion.

When sample.type="two.sample", the computed sample size is based on a hypothesis test for the difference between two proportions.
The default value is sample.type="one.sample".

alternative — character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

ratio — numeric vector indicating the ratio of sample size in group 2 to sample size in group 1 (n_2/n_1). The default value is ratio=1. All values of ratio must be greater than or equal to 1. This argument is ignored if sample.type="one.sample".

approx — logical scalar indicating whether to compute the sample size based on the normal approximation to the binomial distribution. The default value is approx=TRUE. Currently, the exact method (approx=FALSE) is only available for the one-sample case (i.e., sample.type="one.sample").

correct — logical scalar indicating whether to use the continuity correction when approx=TRUE. The default value is approx=TRUE when sample.type="two.sample" and approx=FALSE when sample.type="one.sample". This argument is ignored when approx=FALSE.

round.up — logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is round.up=TRUE.

warn — logical scalar indicating whether to issue a warning. The default value is warn=TRUE. When approx=TRUE (sample size based on the normal approximation) and warn=T, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate.
When approx=FALSE (sample size based on the exact test) and warn=TRUE, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of alpha.

return.exact.list — logical scalar relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates whether to return a list containing extra information about the exact test in addition to the power of the exact test. By default, propTestN returns only a vector containing the computed sample size(s) (see the VALUE section below). When return.exact.list=TRUE (the default) and approx=FALSE, propTestN returns a list with components indicating the required sample size, power of the exact test, the true significance level associated with the exact test, and the critical values associated with the exact test (see the DETAILS section for more information).

n.min — integer relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the minimum allowed value for n to use in the search algorithm. The default value is n.min=2.

n.max — integer relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the maximum allowed value for n to use in the search algorithm. The default value is n.max=10000.

tol.alpha — numeric vector relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates the tolerance on alpha to

use in the search algorithm (i.e., how close the actual Type I error level is to the value prescribed by alpha). The default value is tol.alpha=0.1*alpha.

tol — numeric scalar relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument is passed to the uniroot function and indicates the tolerance to use in the search algorithm. The default value is tol=1e-7.

maxiter — integer relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument is passed to the uniroot function and indicates the maximum number of iterations to use in the search algorithm. The default value is maxiter=1000.

Details

If the arguments p.or.p1, p0.or.p2, alpha, power, ratio, and tol.alpha are not all the same length, they are replicated to be the same length as the length of the longest argument.

The computed sample size is based on the difference p.or.p1 - p0.or.p2.

One-Sample Case (sample.type="one.sample").

approx=TRUE**.** When sample.type="one.sample" and approx=TRUE, sample size is computed based on the test that uses the normal approximation to the binomial distribution; see the help file for prop.test. The formula for this test and the associated power is presented in standard statistics texts, including Zar (2010, pp. 534-537, 539-541). These equations can be inverted to solve for the sample size, given a specified power, significance level, hypothesized proportion, and true proportion.

approx=FALSE**.** When sample.type="one.sample" and approx=FALSE, sample size is computed based on the exact binomial test; see the help file for binom.test. The formula for this test and its associated power is presented in standard statistics texts, including Zar (2010, pp. 532-534, 539) and Millard and Neerchal (2001, pp. 385-386, 504-506). The formula for the power involves five quantities: the hypothesized proportion (p_0), the true proportion (p), the significance level ($alpha$), the power, and the sample size (n). In this case the function propTestN uses a search algorithm to determine the required sample size to attain a specified power, given the values of the hypothesized and true proportions and the significance level.

Two-Sample Case (sample.type="two.sample").

When sample.type="two.sample", sample size is computed based on the test that uses the normal approximation to the binomial distribution; see the help file for prop.test. The formula for this test and its associated power is presented in standard statistics texts, including Zar (2010, pp. 549-550, 552-553) and Millard and Neerchal (2001, pp. 443-445, 508-510). These equations can be inverted to solve for the sample size, given a specified power, significance level, true proportions, and ratio of sample size in group 2 to sample size in group 1.

Value

Approximate Test (approx=TRUE).

When sample.type="one.sample", or sample.type="two.sample" and ratio=1 (i.e., equal sample sizes for each group), propTestN returns a numeric vector of sample sizes. When sample.type="two.sample" and at least one element of ratio is greater than 1, propTestN returns a list with two components called n1 and n2, specifying the sample sizes for each group.

Exact Test (`approx=FALSE`).

If `return.exact.list=FALSE`, `propTestN` returns a numeric vector of sample sizes.

If `return.exact.list=TRUE`, `propTestN` returns a list with the following components:

`n` numeric vector of sample sizes.

`power` numeric vector of powers.

`alpha` numeric vector containing the true significance levels. Because of the discrete nature of the binomial distribution, the true significance levels usually do not equal the significance level supplied by the user in the argument `alpha`.

`q.critical.lower` numeric vector of lower critical values for rejecting the null hypothesis. If the observed number of "successes" is *less than or equal to* these values, the null hypothesis is rejected. (Not present if `alternative="greater"`.)

`q.critical.upper` numeric vector of upper critical values for rejecting the null hypothesis. If the observed number of "successes" is *greater than* these values, the null hypothesis is rejected. (Not present if `alternative="less"`.)

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", p, is the same on each trial. A binomial discrete random variable X is the number of "successes" in n independent trials. A special case of the binomial distribution occurs when $n = 1$, in which case X is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7).

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, power, significance level, and the difference between the hypothesized and true proportions if one of the objectives of the sampling program is to determine whether a proprtion differs from a specified level or two proportions differ from each other. The functions `propTestPower`, `propTestN`, `propTestMdd`, and `plotPropTestDesign` can be used to investigate these relationships for the case of binomial proportions.

Studying the two-sample proportion test, Haseman (1978) found that the formulas used to estimate the power that do not incorporate the continuity correction tend to underestimate the power. Casagrande, Pike, and Smith (1978) found that the formulas that do incorporate the continuity correction provide an excellent approximation.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapter 15.

Casagrande, J.T., M.C. Pike, and P.G. Smith. (1978). An Improved Approximation Formula for Calculating Sample Sizes for Comparing Two Binomial Distributions. *Biometrics* **34**, 483-486.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. Second Edition. John Wiley and Sons, New York, Chapters 1-2.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Haseman, J.K. (1978). Exact Sample Sizes for Use with the Fisher-Irwin Test for 2x2 Tables. *Biometrics* **34**, 106-109.

Millard, S.P., and N. Neerchal. (2001). *Environmental Statistics with S-Plus*. CRC Press, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

propTestPower, propTestMdd, plotPropTestDesign, prop.test, binom.test.

Examples

```
# Look at how the required sample size of the one-sample
# proportion test with a two-sided alternative and Type I error
# set to 5% increases with increasing power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

propTestN(p.or.p1 = 0.7, p0.or.p2 = 0.5,
  power = seq(0.5, 0.9, by = 0.1))
#[1] 25 31 38 47 62

#----------

# Repeat the last example, but compute the sample size based on
# the exact test instead of the approximation.  Note that because
# we require the actual Type I error (alpha) to be within
# 10% of the supplied value of alpha (which is 0.05 by default),
# due to the discrete nature of the exact binomial test
# we end up with more power then we specified.

n.list <- propTestN(p.or.p1 = 0.7,  p0.or.p2 = 0.5,
  power = seq(0.5, 0.9, by = 0.1), approx = FALSE)

lapply(n.list, round, 3)
#$n
#[1] 37 37 44 51 65
#
#$power
```

```
#[1] 0.698 0.698 0.778 0.836 0.910
#
#$alpha
#[1] 0.047 0.047 0.049 0.049 0.046
#
#$q.critical.lower
#[1] 12 12 15 18 24
#
#$q.critical.upper
#[1] 24 24 28 32 40

#----------

# Using the example above, see how the sample size changes
# if we allow the Type I error to deviate by more than 10 percent
# of the value of alpha (i.e., by more than 0.005).

n.list <- propTestN(p.or.p1 = 0.7,  p0.or.p2 = 0.5,
  power = seq(0.5, 0.9, by = 0.1), approx = FALSE, tol.alpha = 0.01)

lapply(n.list, round, 3)
#$n
#[1] 25 35 42 49 65
#
#$power
#[1] 0.512 0.652 0.743 0.810 0.910
#
#$alpha
#[1] 0.043 0.041 0.044 0.044 0.046
#
#$q.critical.lower
#[1]  7 11 14 17 24
#
#$q.critical.upper
#[1] 17 23 27 31 40

#----------

# Clean up
#---------
rm(n.list)

#==========

# Look at how the required sample size for the two-sample
# proportion test decreases with increasing difference between
# the two population proportions:

seq(0.4, 0.1, by = -0.1)
#[1] 0.4 0.3 0.2 0.1

propTestN(p.or.p1 = seq(0.4, 0.1, by = -0.1),
  p0.or.p2 = 0.5, sample.type = "two")
```

```
#[1] 661 163 70 36
#Warning message:
#In propTestN(p.or.p1 = seq(0.4, 0.1, by = -0.1), p0.or.p2 = 0.5,  :
#  The computed sample sizes 'n1' and 'n2' are too small,
#  relative to the given values of 'p1' and 'p2', for the normal
#  approximation to work well for the following element indices:
#         4

#----------

# Look at how the required sample size for the two-sample
# proportion test decreases with increasing values of Type I error:

propTestN(p.or.p1 = 0.7, p0.or.p2 = 0.5,
  sample.type = "two",
  alpha = c(0.001, 0.01, 0.05, 0.1))
#[1] 299 221 163 137

#==========

# Modifying the example on pages 8-5 to 8-7 of USEPA (1989b),
# determine the required sample size to detect a difference in the
# proportion of detects of cadmium between the background and
# compliance wells. Set the complicance well to "group 1" and
# the backgound well to "group 2".  Assume the true probability
# of a "detect" at the background well is 1/3, set the probability
# of a "detect" at the compliance well to 0.4 and 0.5, use a 5%
# significance level and 95% power, and use the upper
# one-sided alternative (probability of a "detect" at the compliance
# well is greater than the probability of a "detect" at the background
# well).  (The original data are stored in EPA.89b.cadmium.df.)
#
# Note that the required sample size decreases from about
# 1160 at each well to about 200 at each well as the difference in
# proportions changes from (0.4 - 1/3) to (0.5 - 1/3), but both of
# these sample sizes are enormous compared to the number of samples
# usually collected in the field.

EPA.89b.cadmium.df
#   Cadmium.orig Cadmium Censored  Well.type
#1           0.1   0.100    FALSE Background
#2          0.12   0.120    FALSE Background
#3           BDL   0.000     TRUE Background
# ......................................
#86          BDL   0.000     TRUE Compliance
#87          BDL   0.000     TRUE Compliance
#88          BDL   0.000     TRUE Compliance

p.hat.back <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Background"]))

p.hat.back
#[1] 0.3333333
```

```
p.hat.comp <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Compliance"]))

p.hat.comp
#[1] 0.375

n.back <- with(EPA.89b.cadmium.df,
  sum(Well.type == "Background"))

n.back
#[1] 24

n.comp <- with(EPA.89b.cadmium.df,
  sum(Well.type == "Compliance"))

n.comp
#[1] 64

propTestN(p.or.p1 = c(0.4, 0.50), p0.or.p2 = p.hat.back,
  alt="greater", sample.type="two")
#[1] 1159 199

#----------

# Clean up
#---------
rm(p.hat.back, p.hat.comp, n.back, n.comp)
```

`propTestPower` *Compute the Power of a One- or Two-Sample Proportion Test*

Description

Compute the power of a one- or two-sample proportion test, given the sample size(s), true proportion(s), and significance level.

Usage

```
propTestPower(n.or.n1, p.or.p1 = 0.5, n2 = n.or.n1,
  p0.or.p2 = 0.5, alpha = 0.05, sample.type = "one.sample",
  alternative = "two.sided", approx = TRUE,
  correct = sample.type == "two.sample", warn = TRUE,
  return.exact.list = TRUE)
```

Arguments

`n.or.n1` numeric vector of sample sizes. When `sample.type="one.sample"`, this argument denotes n, the number of observations in the single sample. When

sample.type="two.sample", this argument denotes n_1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

p.or.p1 numeric vector of proportions. When sample.type="one.sample", this argument denotes the true value of p, the probability of "success". When sample.type="two.sample", this argument denotes the value of p_1, the probability of "success" in group 1. The default value is p.or.p1=0.5. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

n2 numeric vector of sample sizes for group 2. The default value is n2=n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

p0.or.p2 numeric vector of proportions. When sample.type="one.sample", this argument denotes the hypothesized value of p, the probability of "success". When sample.type="two.sample", this argument denotes the value of p_2, the probability of "success" in group 2. The default value is p0.or.p2=0.5. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are not allowed.

alpha numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.

sample.type character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single proportion. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two proportions. The default value is sample.type="one.sample".

alternative character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".

approx logical scalar indicating whether to compute the power based on the normal approximation to the binomial distribution. The default value is approx=TRUE. Currently, the exact method (approx=FALSE) is only available for the one-sample case (i.e., sample.type="one.sample").

correct logical scalar indicating whether to use the continuity correction when approx=TRUE. The default value is approx=TRUE when sample.type="two.sample" and approx=FALSE when sample.type="one.sample". This argument is ignored when approx=FALSE.

warn logical scalar indicating whether to issue a warning. The default value is warn=TRUE. When approx=TRUE (power based on the normal approximation) and warn=TRUE, a warning is issued for cases when the normal approximation to the binomial distribution probably is not accurate. When approx=FALSE (power based on the exact test) and warn=TRUE, a warning is issued when the user-supplied sample size is too small to yield a significance level less than or equal to the user-supplied value of alpha.

return.exact.list logical scalar relevant to the case when approx=FALSE (i.e., when the power is based on the exact test). This argument indicates whether to return a list containing extra information about the exact test in addition to the power of the exact

test. By default, propTestPower returns only a vector containing the computed power(s) (see the VALUE section below). When return.exact.list=TRUE (the default) and approx=FALSE, propTestPower returns a list with components indicating the power of the exact test, the true significance level associated with the exact test, and the critical values associated with the exact test (see the DETAILS section for more information).

Details

If the arguments n.or.n1, p.or.p1, n2, p0.or.p2, and alpha are not all the same length, they are replicated to be the same length as the length of the longest argument.

The power is based on the difference p.or.p1 - p0.or.p2.

One-Sample Case (sample.type="one.sample").

approx=TRUE When sample.type="one.sample" and approx=TRUE, power is computed based on the test that uses the normal approximation to the binomial distribution; see the help file for prop.test. The formula for this test and its associated power is presented in most standard statistics texts, including Zar (2010, pp. 534-537, 539-541).

approx=FALSE When sample.type="one.sample" and approx=FALSE, power is computed based on the exact binomial test; see the help file for binom.test. The formula for this test and its associated power is presented in most standard statistics texts, including Zar (2010, pp. 532-534, 539) and Millard and Neerchal (2001, pp. 385-386, 504-506).

Two-Sample Case (sample.type="two.sample").

When sample.type="two.sample", power is computed based on the test that uses the normal approximation to the binomial distribution; see the help file for prop.test. The formula for this test and its associated power is presented in standard statistics texts, including Zar (2010, pp. 549-550, 552-553) and Millard and Neerchal (2001, pp. 443-445, 508-510).

Value

By default, propTestPower returns a numeric vector of powers. For the one-sample proportion test (sample.type="one.sample"), when approx=FALSE and return.exact.list=TRUE, propTestPower returns a list with the following components:

power numeric vector of powers.

alpha numeric vector containing the true significance levels. Because of the discrete nature of the binomial distribution, the true significance levels usually do not equal the significance level supplied by the user in the argument alpha.

q.critical.lower
numeric vector of lower critical values for rejecting the null hypothesis. If the observed number of "successes" is *less than or equal to* these values, the null hypothesis is rejected. (Not present if alternative="greater".)

q.critical.upper
numeric vector of upper critical values for rejecting the null hypothesis. If the observed number of "successes" is *greater than* these values, the null hypothesis is rejected. (Not present if alternative="less".)

Note

The binomial distribution is used to model processes with binary (Yes-No, Success-Failure, Heads-Tails, etc.) outcomes. It is assumed that the outcome of any one trial is independent of any other trial, and that the probability of "success", p, is the same on each trial. A binomial discrete random variable X is the number of "successes" in n independent trials. A special case of the binomial distribution occurs when $n = 1$, in which case X is also called a Bernoulli random variable.

In the context of environmental statistics, the binomial distribution is sometimes used to model the proportion of times a chemical concentration exceeds a set standard in a given period of time (e.g., Gilbert, 1987, p.143), or to compare the proportion of detects in a compliance well vs. a background well (e.g., USEPA, 1989b, Chapter 8, p.3-7).

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, power, significance level, and the difference between the hypothesized and true proportions if one of the objectives of the sampling program is to determine whether a proprtion differs from a specified level or two proportions differ from each other. The functions `propTestPower`, `propTestN`, `propTestMdd`, and `plotPropTestDesign` can be used to investigate these relationships for the case of binomial proportions.

Studying the two-sample proportion test, Haseman (1978) found that the formulas used to estimate the power that do not incorporate the continuity correction tend to underestimate the power. Casagrande, Pike, and Smith (1978) found that the formulas that do incorporate the continuity correction provide an excellent approximation.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (1994). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton, FL, Chapter 15.

Casagrande, J.T., M.C. Pike, and P.G. Smith. (1978). An Improved Approximation Formula for Calculating Sample Sizes for Comparing Two Binomial Distributions. *Biometrics* **34**, 483-486.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. Second Edition. John Wiley and Sons, New York, Chapters 1-2.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York, NY.

Haseman, J.K. (1978). Exact Sample Sizes for Use with the Fisher-Irwin Test for 2x2 Tables. *Biometrics* **34**, 106-109.

Millard, S.P., and N. Neerchal. (2001). *Environmental Statistics with S-Plus*. CRC Press, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`propTestN`, `propTestMdd`, `plotPropTestDesign`, `prop.test`, `binom.test`.

Examples

```
# Look at how the power of the one-sample proportion test
# increases with increasing sample size:

seq(20, 50, by=10)
#[1] 20 30 40 50

power <- propTestPower(n.or.n1 = seq(20, 50, by=10),
  p.or.p1 = 0.7, p0 = 0.5)

round(power, 2)
#[1] 0.43 0.60 0.73 0.83

#----------

# Repeat the last example, but compute the power based on
# the exact test instead of the approximation.
# Note that the significance level varies with sample size and
# never attains the requested level of 0.05.

prop.test.list <- propTestPower(n.or.n1 = seq(20, 50, by=10),
  p.or.p1 = 0.7, p0 = 0.5, approx=FALSE)

lapply(prop.test.list, round, 2)
#$power:
#[1] 0.42 0.59 0.70 0.78
#
#$alpha:
#[1] 0.04 0.04 0.04 0.03
#
#$q.critical.lower:
#[1] 5 9 13 17
#
#$q.critical.upper:
#[1] 14 20 26 32

#==========

# Look at how the power of the two-sample proportion test
# increases with increasing difference between the two
# population proportions:

seq(0.5, 0.1, by=-0.1)
#[1] 0.5 0.4 0.3 0.2 0.1

power <- propTestPower(30, sample.type = "two",
  p.or.p1 = seq(0.5, 0.1, by=-0.1))
#Warning message:
#In propTestPower(30, sample.type = "two", p.or.p1 = seq(0.5, 0.1,  :
#The sample sizes 'n1' and 'n2' are too small, relative to the given
# values of 'p1' and 'p2', for the normal approximation to work well
# for the following element indices:
```

```
#         5

round(power, 2)
#[1] 0.05 0.08 0.26 0.59 0.90

#----------

# Look at how the power of the two-sample proportion test
# increases with increasing values of Type I error:

power <- propTestPower(30, sample.type = "two",
  p.or.p1 = 0.7,
  alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
#[1] 0.02 0.10 0.26 0.37

#==========

# Clean up
#---------
rm(power, prop.test.list)

#==========

# Modifying the example on pages 8-5 to 8-7 of USEPA (1989b),
# determine how adding another 20 observations to the background
# well to increase the sample size from 24 to 44 will affect the
# power of detecting a difference in the proportion of detects of
# cadmium between the background and compliance wells.  Set the
# compliance well to "group 1" and set the background well to
# "group 2".  Assume the true probability of a "detect" at the
# background well is 1/3, set the probability of a "detect" at the
# compliance well to 0.4, use a 5% significance level, and use the
# upper one-sided alternative (probability of a "detect" at the
# compliance well is greater than the probability of a "detect" at
# the background well).
# (The original data are stored in EPA.89b.cadmium.df.)
#
# Note that the power does increase (from 9% to 12%), but is relatively
# very small.

EPA.89b.cadmium.df
#   Cadmium.orig Cadmium Censored  Well.type
#1           0.1   0.100    FALSE Background
#2          0.12   0.120    FALSE Background
#3           BDL   0.000     TRUE Background
# ....................................
#86          BDL   0.000     TRUE Compliance
#87          BDL   0.000     TRUE Compliance
#88          BDL   0.000     TRUE Compliance
```

```
p.hat.back <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Background"]))

p.hat.back
#[1] 0.3333333

p.hat.comp <- with(EPA.89b.cadmium.df,
  mean(!Censored[Well.type=="Compliance"]))

p.hat.comp
#[1] 0.375

n.back <- with(EPA.89b.cadmium.df,
  sum(Well.type == "Background"))

n.back
#[1] 24

n.comp <- with(EPA.89b.cadmium.df,
  sum(Well.type == "Compliance"))

n.comp
#[1] 64

propTestPower(n.or.n1 = n.comp,
  p.or.p1 = 0.4,
  n2 = c(n.back, 44), p0.or.p2 = p.hat.back,
  alt="greater", sample.type="two")
#[1] 0.08953013 0.12421135

#----------

# Clean up
#---------
rm(p.hat.back, p.hat.comp, n.back, n.comp)
```

`ProUCL.5.2.TRS.df` *Real dataset from ProUCL 5.2.0.*

Description

A real data set of size n=55 with 18.8% Nondetects (=10). The name of the Excel file that comes with ProUCL 5.2.0 and contains these data is **TRS-Real-data-with-NDs.xls**.

Usage

```
ProUCL.5.2.TRS.df
data(ProUCL.5.2.TRS.df)
```

Format

A data frame with 55 observations on the following 3 variables.

`Value` numeric vector indicating the concentration.

`Detect` numeric vector of 0s (nondetects) and 1s (detects) indicating censoring status.

`Censored` logical vector indicating censoring status.

Source

USEPA. (2022a). *ProUCL Version 5.2.0 Technical Guide: Statistical Software for Environmental Applications for Data Sets with and without Nondetect Observations*. Prepared by: Neptune and Company, Inc., 1435 Garrison Street, Suite 201, Lakewood, CO 80215. p. 143. https://www.epa.gov/land-research/proucl-software.

USEPA. (2022b). *ProUCL Version 5.2.0 User Guide: Statistical Software for Environmental Applications for Data Sets with and without Nondetect Observations*. Prepared by: Neptune and Company, Inc., 1435 Garrison Street, Suite 201, Lakewood, CO 80215. p. 6-115. https://www.epa.gov/land-research/proucl-software.

ProUCL.Crit.Vals.for.AD.Test.for.Gamma.array

ProUCL Critical Values for Anderson-Darling Goodness-of-Fit Test for Gamma Distribution

---

Description

Critical Values for the Anderson-Darling Goodness-of-Fit Test for a Gamma Distribution, as presented in Tables A-1, A-3, and A-5 on pages 283, 285, and 287, respectively, of USEPA (2015).

Usage

```
data("ProUCL.Crit.Vals.for.AD.Test.for.Gamma.array")
```

Format

An array of dimensions 32 by 11 by 3, with the first dimension indicating the sample size (between 5 and 1000), the second dimension indicating the value of the maximum likelihood estimate of the shape parameter (between 0.025 and 50), and the third dimension indicating the assumed significance level (0.01, 0.05, and 0.10).

Details

See USEPA (2015, pp.281-282) and the help file for `gofTest` for more information. The data in this array are used when the function `gofTest` is called with `test="proucl.ad.gamma"`. The letter k is used to indicate the value of the estimated shape parameter.

Source

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C., pp. 283, 285, and 287.

References

USEPA. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C.

`ProUCL.Crit.Vals.for.KS.Test.for.Gamma.array`

ProUCL Critical Values for Kolmogorov-Smirnov Goodness-of-Fit Test for Gamma Distribution

---

Description

Critical Values for the Kolmogorov-Smirnov Goodness-of-Fit Test for a Gamma Distribution, as presented in Tables A-2, A-4, and A-6 on pages 284, 286, and 288, respectively, of USEPA (2015).

Usage

```
data("ProUCL.Crit.Vals.for.KS.Test.for.Gamma.array")
```

Format

An array of dimensions 32 by 11 by 3, with the first dimension indicating the sample size (between 5 and 1000), the second dimension indicating the value of the maximum likelihood estimate of the shape parameter (between 0.025 and 50), and the third dimension indicating the assumed significance level (0.01, 0.05, and 0.10).

Details

See USEPA (2015, pp.281-282) for more information. The data in this array are used when the function `gofTest` is called with `test="proucl.ks.gamma"`.

Source

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C., pp. 284, 286, and 288.

References

USEPA. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2015). *ProUCL Version 5.1.002 Technical Guide*. EPA/600/R-07/041, October 2015. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C.

`pwMoment` *Estimate Probability-Weighted Moments*

Description

Estimate the $1jk$'th probability-weighted moment from a random sample, where either $j = 0$, $k = 0$, or both.

Usage

```
pwMoment(x, j = 0, k = 0, method = "unbiased",
  plot.pos.cons = c(a = 0.35, b = 0), na.rm = FALSE)
```

Arguments

`x`	numeric vector of observations.
`j`, `k`	non-negative integers specifying the order of the moment.
`method`	character string specifying what method to use to compute the probability-weighted moment. The possible values are `"unbiased"` (method based on the U-statistic; the default), or `"plotting.position"` (method based on the plotting position formula). See the DETAILS section for more information.
`plot.pos.cons`	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when `method="plotting.position"`. The default value is `plot.pos.cons=c(a=0.35, b=0)`. If this vector has a names attribute with the value `c("a","b")` or `c("b","a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name `"a"` and the second element to the name `"b"`. See the DETAILS section for more information. This argument is ignored if `method="ubiased"`.
`na.rm`	logical scalar indicating whether to remove missing values from `x`. If `na.rm=FALSE` (the default) and `x` contains missing values, then a missing value (`NA`) is returned. If `na.rm=TRUE`, missing values are removed from `x` prior to computing the probability-weighted moment.

Details

The definition of a probability-weighted moment, introduced by Greenwood et al. (1979), is as follows. Let X denote a random variable with cdf F, and let $x(p)$ denote the p'th quantile of the distribution. Then the ijk'th probability-weighted moment is given by:

$$M(i,j,k) = E[X^i F^j (1-F)^k] = \int_0^1 [x(F)]^i F^j (1-F)^k \, dF$$

where i, j, and k are real numbers. Note that if i is a nonnegative integer, then $M(i,0,0)$ is the conventional i'th moment about the origin.

Greenwood et al. (1979) state that in the special case where i, j, and k are nonnegative integers:

$$M(i,j,k) = B(j+1, k+1) E[X^i_{j+1,j+k+1}]$$

where $B(a,b)$ denotes the beta function evaluated at a and b, and

$$E[X^i_{j+1,j+k+1}]$$

denotes the i'th moment about the origin of the $(j+1)$'th order statistic for a sample of size $(j+k+1)$. In particular,

$$M(1,0,k) = \frac{1}{k+1} E[X_{1,k+1}]$$

$$M(1,j,0) = \frac{1}{j+1} E[X_{j+1,j+1}]$$

where

$$E[X_{1,k+1}]$$

denotes the expected value of the first order statistic (i.e., the minimum) in a sample of size $(k+1)$, and

$$E[X_{j+1,j+1}]$$

denotes the expected value of the $(j+1)$'th order statistic (i.e., the maximum) in a sample of size $(j+1)$.

Unbiased Estimators (`method="unbiased"`)
Landwehr et al. (1979) show that, given a random sample of n values from some arbitrary distribution, an unbiased, distribution-free, and parameter-free estimator of $M(1,0,k)$ is given by:

$$\hat{M}(1,0,k) = \frac{1}{n} \sum_{i=1}^{n-k} x_{i,n} \frac{\binom{n-i}{k}}{\binom{n-1}{k}}$$

where the quantity $x_{i,n}$ denotes the i'th order statistic in the random sample of size n. Hosking et al. (1985) note that this estimator is closely related to U-statistics (Hoeffding, 1948; Lehmann, 1975, pp. 362-371). Hosking et al. (1985) note that an unbiased, distribution-free, and parameter-free estimator of $M(1,j,0)$ is given by:

$$\hat{M}(1,j,0) = \frac{1}{n} \sum_{i=j+1}^{n} x_{i,n} \frac{\binom{i-1}{j}}{\binom{n-1}{j}}$$

Plotting-Position Estimators (`method="plotting.position"`)
Hosking et al. (1985) propose alternative estimators of $M(1, 0, k)$ and $M(1, j, 0)$ based on plotting positions:

$$\hat{M}(1, 0, k) = \frac{1}{n}\sum_{i=1}^{n}(1 - p_{i,n})^k x_{i,n}$$

$$\hat{M}(1, j, 0) = \frac{1}{n}\sum_{i=1}^{n} p_{i,n}^j x_{i,n}$$

where

$$p_{i,n} = \hat{F}(x_{i,n})$$

denotes the plotting position of the i'th order statistic in the random sample of size n, that is, a distribution-free estimate of the cdf of X evaluated at the i'th order statistic. Typically, plotting positions have the form:

$$p_{i,n} = \frac{i - a}{n + b}$$

where $b > -a > -1$. For this form of plotting position, the plotting-position estimators are asymptotically equivalent to the U-statistic estimators.

Value

A numeric scalar–the value of the $1jk$'th probability-weighted moment as defined by Greenwood et al. (1979).

Note

Greenwood et al. (1979) introduced the concept of probability-weighted moments as a tool to derive estimates of distribution parameters for distributions that can be (perhaps only be) expressed in inverse form. The term "inverse form" simply means that instead of characterizing the distribution by the formula for its cumulative distribution function (cdf), the distribution is characterized by the formula for the p'th quantile ($0 \leq p \leq 1$).

For distributions that can only be expressed in inverse form, moment estimates of their parameters are not available, and maximum likelihood estimates are not easy to compute. Greenwood et al. (1979) show that in these cases, it is often possible to derive expressions for the distribution parameters in terms of probability-weighted moments. Thus, for these cases the distribution parameters can be estimated based on the sample probability-weighted moments, which are fairly easy to compute. Furthermore, for distributions whose parameters can be expressed as functions of conventional moments, the method of probability-weighted moments provides an alternative to method of moments and maximum likelihood estimators.

Landwehr et al. (1979) use the method of probability-weighted moments to estimate the parameters of the Type I Extreme Value (Gumbel) distribution.

Hosking et al. (1985) use the method of probability-weighted moments to estimate the parameters of the generalized extreme value distribution.

Hosking (1990) and Hosking and Wallis (1995) show the relationship between probabiity-weighted moments and L-moments.

Hosking and Wallis (1995) recommend using the unbiased estimators of probability-weighted moments for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Greenwood, J.A., J.M. Landwehr, N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments: Definition and Relation to Parameters of Several Distributions Expressible in Inverse Form. *Water Resources Research* **15**(5), 1049–1054.

Hoeffding, W. (1948). A Class of Statistics with Asymptotically Normal Distribution. *Annals of Mathematical Statistics* **19**, 293–325.

Hosking, J.R.M. (1990). L-Moments: Analysis and Estimation of Distributions Using Linear Combinations of Order Statistics. *Journal of the Royal Statistical Society, Series B* **52**(1), 105–124.

Hosking, J.R.M., and J.R. Wallis (1995). A Comparison of Unbiased and Plotting-Position Estimators of L Moments. *Water Resources Research* **31**(8), 2019–2025.

Hosking, J.R.M., J.R. Wallis, and E.F. Wood. (1985). Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments. *Technometrics* **27**(3), 251–261.

Landwehr, J.M., N.C. Matalas, and J.R. Wallis. (1979). Probability Weighted Moments Compared With Some Traditional Techniques in Estimating Gumbel Parameters and Quantiles. *Water Resources Research* **15**(5), 1055–1064.

Lehmann, E.L. (1975). *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, Oakland, CA, pp.362-371.

See Also

eevd, egevd, lMoment.

Examples

```
# Generate 20 observations from a generalized extreme value distribution
# with parameters location=10, scale=2, and shape=.25, then compute the
# 0'th, 1'st and 2'nd probability-weighted moments.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rgevd(20, location = 10, scale = 2, shape = 0.25)

pwMoment(dat)
#[1] 10.59556

pwMoment(dat, 1)
#[1] 5.798481

pwMoment(dat, 2)
#[1] 4.060574

pwMoment(dat, k = 1)
#[1] 4.797081
```

```
pwMoment(dat, k = 2)
#[1] 3.059173

pwMoment(dat, 1, method = "plotting.position")
# [1] 5.852913

pwMoment(dat, 1, method = "plotting.position",
  plot.pos = c(.325, 1))
#[1] 5.586817

#----------

# Clean Up
#---------
rm(dat)
```

---

qqPlot *Quantile-Quantile (Q-Q) Plot*

Description

Produces a quantile-quantile (Q-Q) plot, also called a probability plot. The qqPlot function is a modified version of the R functions qqnorm and qqplot. The **EnvStats** function qqPlot allows the user to specify a number of different distributions in addition to the normal distribution, and to optionally estimate the distribution parameters of the fitted distribution.

Usage

```
qqPlot(x, y = NULL, distribution = "norm", param.list = list(mean = 0, sd = 1),
  estimate.params = plot.type == "Tukey Mean-Difference Q-Q",
  est.arg.list = NULL, plot.type = "Q-Q", plot.pos.con = NULL, plot.it = TRUE,
  equal.axes = qq.line.type == "0-1" || estimate.params, add.line = FALSE,
  qq.line.type = "least squares", duplicate.points.method = "standard",
  points.col = 1, line.col = 1, line.lwd = par("cex"), line.lty = 1,
  digits = .Options$digits, ..., main = NULL, xlab = NULL, ylab = NULL,
  xlim = NULL, ylim = NULL)
```

Arguments

x	numeric vector of observations. When y is not supplied, x represents a sample from the hypothesized distribution specifed by distribution. When y is supplied, the distribution of x is compared with the distribuiton of y. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
y	optional numeric vector of observations (not necessarily the same lenght as x). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

distribution — when y is not supplied, a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations. This argument is ignored if y is supplied.

param.list — when y is not supplied, a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. This argument is ignored if y is supplied or estimate.params=TRUE.

estimate.params — when y is not supplied, a logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (estimate.params=TRUE) or using the known distribution parameters specified in param.list (estimate.params=FALSE). The default value of estimate.params is FALSE if plot.type="Q-Q" because the default configuration is a standard normal (mean=0, sd=1) Q-Q plot, which will yield roughly a straight line if the observations in x are from any normal distribution. The default value of estimate.params is TRUE if plot.type="Tukey Mean-Difference Q-Q". The argument estimate.params is ignored if y is supplied.

est.arg.list — when y is not supplied and estimate.params=TRUE, a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the help file Estimating Distribution Parameters). For example, all functions used to estimate distribution parameters have an optional argument called method that specifies the method to use to estimate the parameters. (See the help file for Distribution.df for a list of available estimation methods for each distribution.) To override the default estimation method, supply the argument est.arg.list with a component called method; for example est.arg.list=list(method="mle"). The default value is est.arg.list=NULL so that all default values for the estimating function are used. This argument is ignored if estimate.params=FALSE or y is supplied.

plot.type — a character string denoting the kind of plot. Possible values are "Q-Q" (Quantile-Quantile plot, the default) and "Tukey Mean-Difference Q-Q" (Tukey mean-difference Q-Q plot). This argument may be abbreviated (e.g., plot.type="T" to indicate a Tukey mean-difference Q-Q plot).

plot.pos.con — numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value of plot.pos.con depends on whether the argument y is supplied, and if not the value of the argument distribution. When y is supplied, the default value is plot.pos.con=0.5, corresponding to Hazen plotting positions. When y is not supplied, for the normal, lognormal, three-parameter lognormal, zero-modified normal, and zero-modified lognormal distributions, the default value is plot.pos.con=0.375. For the Type I extreme value (Gumbel) distribution (distribution="evd"), the default value is plot.pos.con=0.44. For all other distributions, the default value is plot.pos.con=0.4.

plot.it — a logical scalar indicating whether to create a plot on the current graphics device. The default value is plot.it=TRUE.

`equal.axes` a logical scalar indicating whether to use the same range on the x- and y-axes when `plot.type="Q-Q"`. The default value is `TRUE` if `qq.line.type="0-1"` or `estimate.params=TRUE`, otherwise it is `FALSE`. This argument is ignored if `plot.type="Tukey Mean-Difference Q-Q"`.

`add.line` a logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey Mean-Difference Q-Q"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=FALSE`.

`qq.line.type` character string determining what kind of line to add to the Q-Q plot. Possible values are `"least squares"` (the default), `"0-1"` and `"robust"`. For the value `"least squares"`, a least squares line is fit and added. For the value `"0-1"`, a line with intercept 0 and slope 1 is added. For the value `"robust"`, a line is fit through the first and third quartiles of the `x` and `y` data. This argument is ignored if `add.line=FALSE` or `plot.type="Tukey Mean-Difference Q-Q"`.

`duplicate.points.method`
a character string denoting how to plot points with duplicate (x, y) values. Possible values are `"standard"` (the default), `"jitter"`, and `"number"`. For the value `"standard"`, a single plotting symbol is plotted (this is the default behavior of R). For the value `"jitter"`, a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y. For the value `"number"`, a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate.

`points.col` a numeric scalar or character string determining the color of the points in the plot. The default value is `points.col=1`. See the entry for `col` in the help file for `par` for more information.

`line.col` a numeric scalar or character string determining the color of the line in the plot. The default value is `points.col=1`. See the entry for `col` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`line.lwd` a numeric scalar determining the width of the line in the plot. The default value is `line.lwd=par("cex")`. See the entry for `lwd` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`line.lty` a numeric scalar determining the line type of the line in the plot. The default value is `line.lty=1`. See the entry for `lty` in the help file for `par` for more information. This argument is ignored if `add.line=FALSE`.

`digits` a scalar indicating how many significant digits to print for the distribution parameters. The default value is `digits=.Options$digits`.

`main`, `xlab`, `ylab`, `xlim`, `ylim`, `...`
additional graphical parameters (see `par`).

Details

If `y` is not supplied, the vector `x` is assumed to be a sample from the probability distribution specified by the argument `distribution` (and `param.list` if `estimate.params=FALSE`). When `plot.type="Q-Q"`, the quantiles of `x` are plotted on the y-axis against the quantiles of the assumed distribution on the x-axis.

If y is supplied and plot.type="Q-Q", the empirical quantiles of y are plotted against the empirical quantiles of x.

When plot.type="Tukey Mean-Difference Q-Q", the difference of the quantiles is plotted on the y-axis against the mean of the quantiles on the x-axis.

Special Distributions

When y is not supplied and the argument distribution specifies one of the following distributions, the function qqPlot behaves in the manner described below.

"lnorm" *Lognormal Distribution.* The log-transformed quantiles are plotted against quantiles from a Normal (Gaussian) distribution.

"lnormAlt" *Lognormal Distribution (alternative parameterization).* The untransformed quantiles are plotted against quantiles from a Lognormal distribution.

"lnorm3" *Three-Parameter Lognormal Distribution.* The quantiles of log(x-threshold) are plotted against quantiles from a Normal (Gaussian) distribution. The value of threshold is either specified in the argument param.list, or, if estimate.params=TRUE, then it is estimated.

"zmnorm" *Zero-Modified Normal Distribution.* The quantiles of the non-zero values (i.e., x[x!=0]) are plotted against quantiles from a Normal (Gaussian) distribution.

"zmlnorm" *Zero-Modified Lognormal Distribution.* The quantiles of the log-transformed positive values (i.e., log(x[x>0])) are plotted against quantiles from a Normal (Gaussian) distribution.

"zmlnormAlt" *Lognormal Distribution (alternative parameterization).* The quantiles of the untransformed positive values (i.e., x[x>0]) are plotted against quantiles from a Lognormal distribution.

Explanation of Q-Q Plots

A ***probability plot*** or ***quantile-quantile (Q-Q) plot*** is a graphical display invented by Wilk and Gnanadesikan (1968) to compare a data set to a particular probability distribution or to compare it to another data set. The idea is that if two population distributions are exactly the same, then they have the same quantiles (percentiles), so a plot of the quantiles for the first distribution vs. the quantiles for the second distribution will fall on the 0-1 line (i.e., the straight line $y = x$ with intercept 0 and slope 1). If the two distributions have the same shape and spread but different locations, then the plot of the quantiles will fall on the line $y = x + b$ (parallel to the 0-1 line) where b denotes the difference in locations. If the distributions have different locations and differ by a multiplicative constant m, then the plot of the quantiles will fall on the line $y = mx + b$ (D'Agostino, 1986a, p. 25; Helsel and Hirsch, 1986, p. 42). Various kinds of differences between distributions will yield various kinds of deviations from a straight line.

Comparing Observations to a Hypothesized Distribution

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote the observations in a random sample of size n from some unknown distribution with cumulative distribution function $F()$, and let $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ denote the ordered observations. Depending on the particular formula used for the empirical cdf (see ecdfPlot), the i'th order statistic is an estimate of the $i/(n+1)$'th, $(i-0.5)/n$'th, etc., quantile. For the moment, assume the i'th order statistic is an estimate of the $i/(n+1)$'th quantile, that is:

$$\hat{F}[x_{(i)}] = \hat{p}_i = \frac{i}{n+1} \qquad (1)$$

so

$$x_{(i)} \approx F^{-1}(\hat{p}_i) \qquad (2)$$

If we knew the form of the true cdf F, then the plot of $x_{(i)}$ vs. $F^{-1}(\hat{p}_i)$ would form approximately a straight line based on Equation (2) above. A probability plot is a plot of $x_{(i)}$ vs. $F_0^{-1}(\hat{p}_i)$, where F_0 denotes the cdf associated with the hypothesized distribution. The probability plot should fall roughly on the line $y = x$ if $F = F_0$. If F and F_0 merely differ by a shift in location and scale, that is, if $F[(x - \mu)/\sigma] = F_0(x)$, then the plot should fall roughly on the line $y = \sigma x + \mu$.

The quantity $\hat{p}_i = i/(n+1)$ in Equation (1) above is called the ***plotting position*** for the probability plot. This particular formula for the plotting position is appealing because it can be shown that for any continuous distribution

$$E\{F[x_{(i)}]\} = \frac{i}{n+1} \qquad (3)$$

(Nelson, 1982, pp. 299-300; Stedinger et al., 1993). That is, the i'th plotting position defined as in Equation (1) is the expected value of the true cdf evaluated at the i'th order statistic. Many authors and practitioners, however, prefer to use a plotting position that satisfies:

$$F^{-1}(\hat{p}_i) = E[x_{(i)}] \qquad (4)$$

or one that satisfies

$$F^{-1}(\hat{p}_i) = M[x_{(i)}] = F^{-1}\{M[u_{(i)}]\} \qquad (5)$$

where $M[x_{(i)}]$ denotes the median of the distribution of the i'th order statistic, and $u_{(i)}$ denotes the i'th order statistic in a random sample of n uniform (0,1) random variates.

The plotting positions in Equation (4) are often approximated since the expected value of the i'th order statistic is often difficult and time-consuming to compute. Note that these plotting positions will differ for different distributions.

The plotting positions in Equation (5) were recommended by Filliben (1975) because they require computing or approximating only the medians of uniform (0,1) order statistics, no matter what the form of the assumed cdf F_0. Also, the median may be preferred as a measure of central tendency because the distributions of most order statistics are skewed.

Most plotting positions can be written as:

$$\hat{p}_i = \frac{i - a}{n - 2a + 1} \qquad (6)$$

where $0 \leq a \leq 1$ (D'Agostino, 1986a, p.25; Stedinger et al., 1993). The quantity a is sometimes called the "plotting position constant", and is determined by the argument `plot.pos.con` in the function `qqPlot`. The table below, adapted from Stedinger et al. (1993), displays commonly used plotting positions based on equation (6) for several distributions.

Name	***a***	**Distribution Often Used With**	**References**
Weibull	0	Weibull, Uniform	Weibull (1939), Stedinger et al. (1993)
Median	0.3175	Several	Filliben (1975), Vogel (1986)
Blom	0.375	Normal	Blom (1958),

		and Others	Looney and Gulledge (1985)
Cunnane	0.4	Several	Cunnane (1978), Chowdhury et al. (1991)
Gringorten	0.44	Gumbel	Gringorton (1963), Vogel (1986)
Hazen	0.5	Several	Hazen (1914), Chambers et al. (1983), Cleveland (1993)

For moderate and large sample sizes, there is very little difference in visual appearance of the Q-Q plot for different choices of plotting positions.

Comparing Two Data Sets

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote the observations in a random sample of size n from some unknown distribution with cumulative distribution function $F()$, and let $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ denote the ordered observations. Similarly, let $\underline{y} = y_1, y_2, \ldots, y_m$ denote the observations in a random sample of size m from some unknown distribution with cumulative distribution function $G()$, and let $y_{(1)}, y_{(2)}, \ldots, y_{(m)}$ denote the ordered observations. Suppose we are interested in investigating whether the shape of the distribution with cdf F is the same as the shape of the distribution with cdf G (e.g., F and G may both be normal distributions but differ in mean and standard deviation).

When $n = m$, we can visually explore this question by plotting $y_{(i)}$ vs. $x_{(i)}$, for $i = 1, 2, \ldots, n$. The values in $\underline{y}$ are spread out in a certain way depending on the true distribution: they may be more or less symmetric about some value (the population mean or median) or they may be skewed to the right or left; they may be concentrated close to the mean or median (platykurtic) or there may be several observations "far away" from the mean or median on either side (leptokurtic). Similarly, the values in $\underline{x}$ are spread out in a certain way. If the values in $\underline{x}$ and $\underline{y}$ are spread out in the same way, then the plot of $y_{(i)}$ vs. $x_{(i)}$ will be approximately a straight line. If the cdf F is exactly the same as the cdf G, then the plot of $y_{(i)}$ vs. $x_{(i)}$ will fall roughly on the straight line $y = x$. If F and G differ by a shift in location and scale, that is, if $F[(x - \mu)/\sigma] = G(x)$, then the plot will fall roughly on the line $y = \sigma x + \mu$.

When $n > m$, a slight adjustment has to be made to produce the plot. Let $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_m$ denote the plotting positions corresponding to the m empirical quantiles for the y's and let $\hat{p}_1^*, \hat{p}_2^*, \ldots, \hat{p}_n^*$ denote the plotting positions corresponding the n empirical quantiles for the x's. Then we plot $y_{(j)}$ vs. $x_{(j)}^*$ for $j = 1, 2, \ldots, m$ where

$$x_{(j)}^* = (1 - r)x_{(i)} + r x_{(i+1)} \qquad (7)$$

$$r = \frac{\hat{p}_j - \hat{p}_i^*}{\hat{p}_{i+1}^* - \hat{p}_i^*} \qquad (8)$$

$$\hat{p}_i^* \le \hat{p}_j \le \hat{p}_{i+1}^* \qquad (9)$$

That is, the values for the $x_{(j)}^*$'s are determined by linear interpolation based on the values of the plotting positions for $\underline{x}$ and $\underline{y}$.

A similar adjustment is made when $n < m$.

Note that the R function `qqplot` uses a different method than the one in Equation (7) above; it uses linear interpolation based on `1:n` and `m` by calling the `approx` function.

Value

qqPlot returns a list with components x and y, giving the (x, y) coordinates of the points that have been or would have been plotted. There are four cases to consider:

1. The argument y is not supplied and plot.type="Q-Q".

x	the quantiles from the theoretical distribution.
y	the observed quantiles (order statistics) based on the data in the argument x.

2. The argument y is not supplied and plot.type="Tukey Mean-Difference Q-Q".

x	the averages of the observed and theoretical quantiles.
y	the differences between the observed quantiles (order statistics) and the theoretical quantiles.

3. The argument y is supplied and plot.type="Q-Q".

x	the observed quantiles based on the data in the argument x. Note that these are adjusted quantiles if the number of observations in the argument x is greater then the number of observations in the argument y.
y	the observed quantiles based on the data in the argument y. Note that these are adjusted quantiles if the number of observations in the argument y is greater then the number of observations in the argument x.

4. The argument y is supplied and plot.type="Tukey Mean-Difference Q-Q".

x	the averages of the quantiles based on the argument x and the quantiles based on the argument y.
y	the differences between the quantiles based on the argument x and the quantiles based on the argument y.

Note

A *quantile-quantile (Q-Q) plot*, also called a *probability plot*, is a plot of the observed order statistics from a random sample (the empirical quantiles) against their (estimated) mean or median values based on an assumed distribution, or against the empirical quantiles of another set of data (Wilk and Gnanadesikan, 1968). Q-Q plots are used to assess whether data come from a particular distribution, or whether two datasets have the same parent distribution. If the distributions have the same shape (but not necessarily the same location or scale parameters), then the plot will fall roughly on a straight line. If the distributions are exactly the same, then the plot will fall roughly on the straight line $y = x$.

A *Tukey mean-difference Q-Q plot*, also called an *m-d plot*, is a modification of a Q-Q plot. Rather than plotting observed quantiles vs. theoretical quantiles or observed y-quantiles vs. observed x-quantiles, a Tukey mean-difference Q-Q plot plots the difference between the quantiles on the y-axis vs. the average of the quantiles on the x-axis (Cleveland, 1993, pp.22-23). If the two sets of quantiles come from the same parent distribution, then the points in this plot should fall roughly

along the horizontal line $y = 0$. If one set of quantiles come from the same distribution with a shift in median, then the points in this plot should fall along a horizontal line above or below the line $y = 0$. A Tukey mean-difference Q-Q plot enhances our perception of how the points in the Q-Q plot deviate from a straight line, because it is easier to judge deviations from a horizontal line than from a line with a non-zero slope.

In a Q-Q plot, the extreme points have more variability than points toward the center. A U-shaped Q-Q plot indicates that the underlying distribution for the observations on the y-axis is skewed to the right relative to the underlying distribution for the observations on the x-axis. An upside-down-U-shaped Q-Q plot indicates the y-axis distribution is skewed left relative to the x-axis distribution. An S-shaped Q-Q plot indicates the y-axis distribution has shorter tails than the x-axis distribution. Conversely, a plot that is bent down on the left and bent up on the right indicates that the y-axis distribution has longer tails than the x-axis distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11-16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7-62.

See Also

ppoints, ecdfPlot, Distribution.df, qqPlotGestalt, qqPlotCensored, qqnorm.

Examples

```
# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area.  These data are strored
# in the data frame EPA.94b.tccb.df.
#
# Create an Q-Q plot for the reference area data first assuming a
# normal distribution, then a lognormal distribution, then a
# gamma distribution.

# Assume a normal distribution
#-----------------------------

dev.new()
with(EPA.94b.tccb.df, qqPlot(TcCB[Area == "Reference"]))

dev.new()
with(EPA.94b.tccb.df, qqPlot(TcCB[Area == "Reference"], add.line = TRUE))
```

```
dev.new()
with(EPA.94b.tccb.df, qqPlot(TcCB[Area == "Reference"],
  plot.type = "Tukey", add.line = TRUE))


# The Q-Q plot based on assuming a normal distribution shows a U-shape,
# indicating the Reference area TcCB data are skewed to the right
# compared to a normal distribuiton.

# Assume a lognormal distribution
#-------------------------------

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "lnorm",
    digits = 2, points.col = "blue", add.line = TRUE))

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "lnorm",
    digits = 2, plot.type = "Tukey", points.col = "blue",
    add.line = TRUE))

# Alternative parameterization

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "lnormAlt",
    estimate.params = TRUE, digits = 2, points.col = "blue",
    add.line = TRUE))

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "lnormAlt",
    digits = 2, plot.type = "Tukey", points.col = "blue",
    add.line = TRUE))


# The lognormal distribution appears to be an adequate fit.
# Now look at a Q-Q plot assuming a gamma distribution.
#-----------------------------------------------------------

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "gamma",
    estimate.params = TRUE, digits = 2, points.col = "blue",
    add.line = TRUE))

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "gamma",
    digits = 2, plot.type = "Tukey", points.col = "blue",
    add.line = TRUE))
```

```
# Alternative Parameterization

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "gammaAlt",
    estimate.params = TRUE, digits = 2, points.col = "blue",
    add.line = TRUE))

dev.new()
with(EPA.94b.tccb.df,
  qqPlot(TcCB[Area == "Reference"], dist = "gammaAlt",
    digits = 2, plot.type = "Tukey", points.col = "blue",
    add.line = TRUE))

#----------------------------------------------------------------------------

# Generate 20 observations from a gamma distribution with parameters
# shape=2 and scale=2, then create a normal (Gaussian) Q-Q plot for these data.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(357)
dat <- rgamma(20, shape=2, scale=2)
dev.new()
qqPlot(dat, add.line = TRUE)

# Now assume a gamma distribution and estimate the parameters
#------------------------------------------------------------

dev.new()
qqPlot(dat, dist = "gamma", estimate.params = TRUE, add.line = TRUE)

# Clean up
#---------
rm(dat)
graphics.off()
```

---

qqPlotCensored *Quantile-Quantile (Q-Q) Plot for Type I Censored Data*

Description

Produces a quantile-quantile (Q-Q) plot, also called a probability plot, for Type I censored data.

Usage

```
qqPlotCensored(x, censored, censoring.side = "left",
  prob.method = "michael-schucany", plot.pos.con = NULL,
  distribution = "norm", param.list = list(mean = 0, sd = 1),
  estimate.params = plot.type == "Tukey Mean-Difference Q-Q",
```

```
    est.arg.list = NULL, plot.type = "Q-Q", plot.it = TRUE,
    equal.axes = qq.line.type == "0-1" || estimate.params,
    add.line = FALSE, qq.line.type = "least squares",
    duplicate.points.method = "standard", points.col = 1, line.col = 1,
    line.lwd = par("cex"), line.lty = 1, digits = .Options$digits,
    include.cen = FALSE, cen.pch = ifelse(censoring.side == "left", 6, 2),
    cen.cex = par("cex"), cen.col = 4, ..., main = NULL, xlab = NULL,
    ylab = NULL, xlim = NULL, ylim = NULL)
```

Arguments

x	numeric vector of observations that is assumed to represent a sample from the hypothesized distribution specifed by distribution. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
prob.method	character string indicating what method to use to compute the plotting positions (empirical probabilities). Possible values are: "kaplan-meier" (product-limit method of Kaplan and Meier (1958)), "modified kaplan-meier" (same as "kaplan-meier" except the maximum value is plotted too), "nelson" (hazard plotting method of Nelson (1972)), "michael-schucany" (generalization of the product-limit method due to Michael and Schucany (1986)), and "hirsch-stedinger" (generalization of the product-limit method due to Hirsch and Stedinger (1987)). The default value is prob.method="michael-schucany". The "nelson" method is only available for censoring.side="right", and the "modified kaplan-meier" method is only available for censoring.side="left". See the DETAILS section for more explanation.
plot.pos.con	numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value is plot.pos.con=0.375. See the DETAILS section for more information. This argument is used only if prob.method is equal to "michael-schucany" or "hirsch-stedinger".
distribution	a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations.
param.list	a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for

the names and possible values of the parameters associated with each distribution. This argument is ignored if `estimate.params=TRUE`.

`estimate.params`

a logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (`estimate.params=TRUE`) or using the known distribution parameters specified in param.list (`estimate.params=FALSE`, the default). The default value of `estimate.params` is `FALSE` if `plot.type="Q-Q"` because the default configuration is a standard normal (mean=0, sd=1) Q-Q plot, which will yield roughly a straight line if the observations in `x` are from any normal distribution. The default value of `estimate.params` is `TRUE` if `plot.type="Tukey Mean-Difference Q-Q"`.

You can set `estimate.params=TRUE` only when the argument `distribution` specifies a distribution that has an associated function for estimating distribution parameters in the case of Type I censored data. Currently this includes the normal (`dist="norm"`), lognormal (`dist="lnorm"` or `dist="lnormAlt"`), and Poisson (`dist="pois"`) distributions (see the section *Estimating Distribution Parameters* in the help file EnvStats Functions for Censored Data).

`est.arg.list` a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the section *Estimating Distribution Parameters* in the help file EnvStats Functions for Censored Data). For example, the function `enormCensored` has an optional argument called `method` that specifies the method to use to estimate the parameters. To override the default estimation method, supply the argument `est.arg.list` with a component called `method`; for example `est.arg.list=list(method="impute.w.qq.reg")`. The default value is `est.arg.list=NULL` so that all default values for the estimating function are used. This argument is ignored if `estimate.params=FALSE`.

`plot.type` a character string denoting the kind of plot. Possible values are `"Q-Q"` (Quantile-Quantile plot, the default) and `"Tukey Mean-Difference Q-Q"` (Tukey mean-difference Q-Q plot). This argument may be abbreviated (e.g., `plot.type="T"` to indicate a Tukey mean-difference Q-Q plot).

`plot.it` a logical scalar indicating whether to create a plot on the current graphics device. The default value is `plot.it=TRUE`.

`equal.axes` a logical scalar indicating whether to use the same range on the x- and y-axes when `plot.type="Q-Q"`. The default value is `TRUE` if `qq.line.type="0-1"` or `estimate.params=TRUE`, otherwise it is `FALSE`. This argument is ignored if `plot.type="Tukey Mean-Difference Q-Q"`.

`add.line` a logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey Mean-Difference Q-Q"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=FALSE`.

`qq.line.type` character string determining what kind of line to add to the Q-Q plot. Possible values are `"least squares"` (the default), `"0-1"` and `"robust"`. For the value `"least squares"`, a least squares line is fit and added. For the value `"0-1"`, a line with intercept 0 and slope 1 is added. For the value `"robust"`, a line is fit

through the first and third quartiles of the x and y data. This argument is ignored if add.line=FALSE or plot.type="Tukey Mean-Difference Q-Q".

duplicate.points.method
: a character string denoting how to plot points with duplicate (x, y) values. Possible values are "standard" (the default), "jitter", and "number". For the value "standard", a single plotting symbol is plotted (this is the default behavior of R). For the value "jitter", a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y. For the value "number", a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate.

points.col
: a numeric scalar or character string determining the color of the points in the plot. The default value is points.col=1. See the entry for col in the help file for par for more information.

line.col
: a numeric scalar or character string determining the color of the line in the plot. The default value is points.col=1. See the entry for col in the help file for par for more information. This argument is ignored if add.line=FALSE.

line.lwd
: a numeric scalar determining the width of the line in the plot. The default value is line.lwd=par("cex"). See the entry for lwd in the help file for par for more information. This argument is ignored if add.line=FALSE.

line.lty
: a numeric scalar determining the line type of the line in the plot. The default value is line.lty=1. See the entry for lty in the help file for par for more information. This argument is ignored if add.line=FALSE.

digits
: a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

include.cen
: logical scalar indicating whether to include censored values in the plot. The default value is include.cen=FALSE. If include.cen=TRUE, censored values are plotted using the plotting character indicated by the argument cen.pch (see below).

cen.pch
: numeric scalar or character string indicating the plotting character to use to plot censored values. The default value is cen.pch=2 (hollow triangle pointing up) when censoring.side="right", and cen.pch=6 (hollow triangle pointing down) when censoring.side="left". See the help file for points for a list of other possible plotting characters. This argument is ignored if include.cen=FALSE.

cen.cex
: numeric scalar that determines the size of the plotting character used to plot censored values. The default value is the current value of the cex graphics parameter. See the entry for cex in the help file for par for more information. This argument is ignored if include.cen=FALSE.

cen.col
: numeric scalar or character string that determines the color of the plotting character used to plot censored values. The default value is cen.col=4. See the entry for col in the help file for par for more information. This argument is ignored if include.cen=FALSE.

main, xlab, ylab, xlim, ylim, ...
: additional graphical parameters (see par).

Details

The function qqPlotCensored does exactly the same thing as qqPlot (when the argument y is not supplied to qqPlot), except qqPlotCensored calls the function ppointsCensored to compute the plotting positions (estimated cumulative probabilities).

The vector x is assumed to be a sample from the probability distribution specified by the argument distribution (and param.list if estimate.params=FALSE). When plot.type="Q-Q", the quantiles of x are plotted on the y-axis against the quantiles of the assumed distribution on the x-axis.

When plot.type="Tukey Mean-Difference Q-Q", the difference of the quantiles is plotted on the y-axis against the mean of the quantiles on the x-axis.

When prob.method="kaplan-meier" and censoring.side="left" and the assumed distribution has a maximum support of infinity (Inf; e.g., the normal or lognormal distribution), the point invovling the largest value of x is not plotted because it corresponds to an estimated cumulative probability of 1 which corresponds to an infinite plotting position.

When prob.method="modified kaplan-meier" and censoring.side="left", the estimated cumulative probability associated with the maximum value is modified from 1 to be $(N - .375)/(N + .25)$ where N denotes the sample size (i.e., the Blom plotting position) so that the point associated with the maximum value can be displayed.

Value

qqPlotCensored returns a list with the following components:

x	numeric vector of x-coordinates for the plot. When plot.type="Q-Q" these are the quantiles from the theoretical distribution. When plot.type="Tukey Mean-Difference Q-Q" these are the averages of the observed and theoretical quantiles.
y	numeric vector of y-coordinates for the plot. When plot.type="Q-Q" these are the observed quantiles (order statistics). When plot.type="Tukey Mean-Difference Q-Q" these are the differences between the observed quantiles (order statistics) and the theoretical quantiles.
Order.Statistics	numeric vector of the "ordered" observations. When plot.type="Q-Q" this component is exactly the same as the component y.
Cumulative.Probabilities	numeric vector of the plotting positions associated with the order statistics.
Censored	logical vector indicating which of the ordered observations are censored.
Censoring.Side	character string indicating whether the data are left- or right-censored. This is same value as the argument censoring.side.
Prob.Method	character string indicating what method was used to compute the plotting positions. This is the same value as the argument prob.method.

Optional Component (only present when prob.method="michael-schucany" or prob.method="hirsch-stedinger"):

Plot.Pos.Con	numeric scalar containing the value of the plotting position constant that was used. This is the same as the argument plot.pos.con.

Note

A *quantile-quantile (Q-Q) plot*, also called a *probability plot*, is a plot of the observed order statistics from a random sample (the empirical quantiles) against their (estimated) mean or median values based on an assumed distribution, or against the empirical quantiles of another set of data (Wilk and Gnanadesikan, 1968). Q-Q plots are used to assess whether data come from a particular distribution, or whether two datasets have the same parent distribution. If the distributions have the same shape (but not necessarily the same location or scale parameters), then the plot will fall roughly on a straight line. If the distributions are exactly the same, then the plot will fall roughly on the straight line $y = x$.

A *Tukey mean-difference Q-Q plot*, also called an *m-d plot*, is a modification of a Q-Q plot. Rather than plotting observed quantiles vs. theoretical quantiles or observed y-quantiles vs. observed x-quantiles, a Tukey mean-difference Q-Q plot plots the difference between the quantiles on the y-axis vs. the average of the quantiles on the x-axis (Cleveland, 1993, pp.22-23). If the two sets of quantiles come from the same parent distribution, then the points in this plot should fall roughly along the horizontal line $y = 0$. If one set of quantiles come from the same distribution with a shift in median, then the points in this plot should fall along a horizontal line above or below the line $y = 0$. A Tukey mean-difference Q-Q plot enhances our perception of how the points in the Q-Q plot deviate from a straight line, because it is easier to judge deviations from a horizontal line than from a line with a non-zero slope.

In a Q-Q plot, the extreme points have more variability than points toward the center. A U-shaped Q-Q plot indicates that the underlying distribution for the observations on the y-axis is skewed to the right relative to the underlying distribution for the observations on the x-axis. An upside-down-U-shaped Q-Q plot indicates the y-axis distribution is skewed left relative to the x-axis distribution. An S-shaped Q-Q plot indicates the y-axis distribution has shorter tails than the x-axis distribution. Conversely, a plot that is bent down on the left and bent up on the right indicates that the y-axis distribution has longer tails than the x-axis distribution.

Censored observations complicate the procedures used to graphically explore data. Techniques from survival analysis and life testing have been developed to generalize the procedures for constructing plotting positions, empirical cdf plots, and Q-Q plots to data sets with censored observations (see `ppointsCensored`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Tukey. (1983). *Graphical Methods for Data Analysis*. Duxbury Press, Boston, MA, pp.11-16.

Cleveland, W.S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, 360pp.

D'Agostino, R.B. (1986a). Graphical Analysis. In: D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, Chapter 2, pp.7-62.

Gillespie, B.W., Q. Chen, H. Reichert, A. Franzblau, E. Hedgeman, J. Lepkowski, P. Adriaens, A. Demond, W. Luksemburg, and D.H. Garabrant. (2010). Estimating Population Distributions When Some Data Are Below a Limit of Detection by Using a Reverse Kaplan-Meier Estimator. *Epidemiology* **21**(4), S64–S70.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R, Second Edition*. John Wiley & Sons, Hoboken, New Jersey.

Helsel, D.R., and T.A. Cohn. (1988). Estimation of Descriptive Statistics for Multiply Censored Water Quality Data. *Water Resources Research* **24**(12), 1997-2004.

Hirsch, R.M., and J.R. Stedinger. (1987). Plotting Positions for Historical Floods and Their Precision. *Water Resources Research* **23**(4), 715-727.

Kaplan, E.L., and P. Meier. (1958). Nonparametric Estimation From Incomplete Observations. *Journal of the American Statistical Association* **53**, 457-481.

Lee, E.T., and J. Wang. (2003). *Statistical Methods for Survival Data Analysis, Third Edition*. John Wiley and Sons, New York.

Michael, J.R., and W.R. Schucany. (1986). Analysis of Data from Censored Samples. In D'Agostino, R.B., and M.A. Stephens, eds. *Goodness-of Fit Techniques*. Marcel Dekker, New York, 560pp, Chapter 11, 461-496.

Nelson, W. (1972). Theory and Applications of Hazard Plotting for Censored Failure Data. *Technometrics* **14**, 945-966.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C. Chapter 15.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

ppointsCensored, EnvStats Functions for Censored Data, qqPlot, ecdfPlotCensored, qqPlotGestalt.

Examples

```
# Generate 20 observations from a normal distribution with mean=20 and sd=5,
# censor all observations less than 18, then generate a Q-Q plot assuming
# a normal distribution for the complete data set and the censored data set.
# Note that the Q-Q plot for the censored data set starts at the first ordered
# uncensored observation, and that for values of x > 18 the two Q-Q plots are
# exactly the same.  This is because there is only one censoring level and
# no uncensored observations fall below the censored observations.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(333)
x <- rnorm(20, mean=20, sd=5)
censored <- x < 18

sum(censored)
#[1] 7

new.x <- x
new.x[censored] <- 18
```

```
dev.new()
qqPlot(x, ylim = range(pretty(x)),
  main = "Q-Q Plot for\nComplete Data Set")

dev.new()
qqPlotCensored(new.x, censored, ylim = range(pretty(x)),
  main="Q-Q Plot for\nCensored Data Set")

# Clean up
#---------
rm(x, censored, new.x)

#--------------------------------------------------------------------------------

# Example 15-1 of USEPA (2009, page 15-10) gives an example of
# computing plotting positions based on censored manganese
# concentrations (ppb) in groundwater collected at 5 monitoring
# wells.  The data for this example are stored in
# EPA.09.Ex.15.1.manganese.df.  Here we will create a Q-Q
# plot based on the Kaplan-Meier method.  First we'll assume
# a normal distribution, then a lognormal distribution, then a
# gamma distribution.

EPA.09.Ex.15.1.manganese.df
#   Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
#1       1 Well.1                 <5           5.0     TRUE
#2       2 Well.1               12.1          12.1    FALSE
#3       3 Well.1               16.9          16.9    FALSE
#4       4 Well.1               21.6          21.6    FALSE
#5       5 Well.1                 <2           2.0     TRUE
#...
#21      1 Well.5               17.9          17.9    FALSE
#22      2 Well.5               22.7          22.7    FALSE
#23      3 Well.5                3.3           3.3    FALSE
#24      4 Well.5                8.4           8.4    FALSE
#25      5 Well.5                 <2           2.0     TRUE

# Assume normal distribution
#---------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  qqPlotCensored(Manganese.ppb, Censored,
    prob.method = "kaplan-meier", points.col = "blue", add.line = TRUE,
    main = paste("Normal Q-Q Plot of Manganese Data",
        "Based on Kaplan-Meier Plotting Positions", sep = "\n")))

# Include max value in the plot
#------------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
```

```
  qqPlotCensored(Manganese.ppb, Censored,
    prob.method = "modified kaplan-meier", points.col = "blue",
    add.line = TRUE,
    main = paste("Normal Q-Q Plot of Manganese Data",
        "Based on Kaplan-Meier Plotting Positions",
        "(Max Included)", sep = "\n")))


# Assume lognormal distribution
#------------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  qqPlotCensored(Manganese.ppb, Censored, dist = "lnorm",
    prob.method = "kaplan-meier", points.col = "blue", add.line = TRUE,
    main = paste("Lognormal Q-Q Plot of Manganese Data",
        "Based on Kaplan-Meier Plotting Positions", sep = "\n")))

# Include max value in the plot
#------------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  qqPlotCensored(Manganese.ppb, Censored, dist = "lnorm",
    prob.method = "modified kaplan-meier", points.col = "blue",
    add.line = TRUE,
    main = paste("Lognormal Q-Q Plot of Manganese Data",
        "Based on Kaplan-Meier Plotting Positions",
        "(Max Included)", sep = "\n")))


# The lognormal distribution appears to be a better fit.
# Now create a Q-Q plot assuming a gamma distribution.  Here we'll
# need to set estimate.params=TRUE.

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  qqPlotCensored(Manganese.ppb, Censored, dist = "gamma",
    estimate.params = TRUE, prob.method = "kaplan-meier",
    points.col = "blue", add.line = TRUE,
    main = paste("Gamma Q-Q Plot of Manganese Data",
        "Based on Kaplan-Meier Plotting Positions", sep = "\n")))

# Include max value in the plot
#------------------------------

dev.new()
with(EPA.09.Ex.15.1.manganese.df,
  qqPlotCensored(Manganese.ppb, Censored, dist = "gamma",
    estimate.params = TRUE, prob.method = "modified kaplan-meier",
    points.col = "blue", add.line = TRUE,
    main = paste("Gamma Q-Q Plot of Manganese Data",
        "Based on Kaplan-Meier Plotting Positions",
```

```
        "(Max Included)", sep = "\n")))

#==========

# Clean up
#---------
graphics.off()
```

---

qqPlotGestalt *Develop Gestalt of Q-Q Plots for Specific Distributions*

Description

Produce a series of quantile-quantile (Q-Q) plots (also called probability plots) or Tukey mean-difference Q-Q plots for a user-specified distribution.

Usage

```
qqPlotGestalt(distribution = "norm", param.list = list(mean = 0, sd = 1),
 estimate.params = FALSE, est.arg.list = NULL, sample.size = 10, num.pages = 2,
num.plots.per.page = 4, nrow = ceiling(num.plots.per.page/2), plot.type = "Q-Q",
 plot.pos.con = switch(dist.abb, norm = , lnorm = , lnormAlt = , lnorm3 = 0.375,
    evd = 0.44, 0.4), equal.axes = (qq.line.type == "0-1" || estimate.params),
  margin.title = NULL, add.line = FALSE, qq.line.type = "least squares",
  duplicate.points.method = "standard", points.col = 1, line.col = 1,
  line.lwd = par("cex"), line.lty = 1, digits = .Options$digits,
  same.window = TRUE, ask = same.window & num.pages > 1,
  mfrow = c(nrow, num.plots.per.page/nrow),
  mar = c(4, 4, 1, 1) + 0.1, oma = c(0, 0, 7, 0), mgp = c(2, 0.5, 0), ...,
  main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL)
```

Arguments

`distribution` a character string denoting the distribution abbreviation. The default value is `distribution="norm"`. See the help file for `Distribution.df` for a list of possible distribution abbreviations. This argument is ignored if `y` is supplied.

`param.list` a list with values for the parameters of the distribution. The default value is `param.list=list(mean=0, sd=1)`. See the help file for `Distribution.df` for the names and possible values of the parameters associated with each distribution. This argument is ignored if `estimate.params=TRUE`.

`estimate.params`

a logical scalar indicating whether to compute quantiles based on estimating the distribution parameters (`estimate.params=TRUE`) or using the known distribution parameters specified in `param.list` (`estimate.params=FALSE`, the default). The default value of `estimate.params` is `FALSE` because the default configuration is to generate random numbers from a standard normal (mean=0, sd=1) distribution and produce a standard normal Q-Q plot.

`est.arg.list` a list whose components are optional arguments associated with the function used to estimate the parameters of the assumed distribution (see the help file Estimating Distribution Parameters). For example, all functions used to estimate distribution parameters have an optional argument called `method` that specifies the method to use to estimate the parameters. (See the help file for `Distribution.df` for a list of available estimation methods for each distribution.) To override the default estimation method, supply the argument `est.arg.list` with a component called `method`; for example `est.arg.list=list(method="mle")`. The default value is `est.arg.list=NULL` so that all default values for the estimating function are used. This argument is ignored if `estimate.params=FALSE`.

`sample.size` numeric scalar indicating the number of observations to generate for each Q-Q plot. The default value is `sample.size=10`.

`num.pages` numeric scalar indicating the number of pages of plots to generate. The default value is `num.pages=2`.

`num.plots.per.page` numeric scalar indicating the number of plots per page. The default value is `num.pages=4`.

`nrow` numeric scalar indicating the number of rows of plots on each page. The default value is the smallest integer greater than or equal to `num.plots.per.page/2`.

`plot.type` a character string denoting the kind of plot. Possible values are `"Q-Q"` (Quantile-Quantile plot, the default) and `"Tukey Mean-Difference Q-Q"` (Tukey mean-difference Q-Q plot). This argument may be abbreviated (e.g., `plot.type="T"` to indicate a Tukey mean-difference Q-Q plot).

`plot.pos.con` numeric scalar between 0 and 1 containing the value of the plotting position constant. The default value of `plot.pos.con` depends on the value of the argument `distribution`. For the normal, lognormal, three-parameter lognormal, zero-modified normal, and zero-modified lognormal distributions, the default value is `plot.pos.con=0.375`. For the Type I extreme value (Gumbel) distribution (`distribution="evd"`), the default value is `plot.pos.con=0.44`. For all other distributions, the default value is `plot.pos.con=0.4`. See the help file for `qqPlot` for the motivation behind these values for plotting positions.

`equal.axes` logical scalar indicating whether to use the same range on the x- and y-axes when `plot.type="Q-Q"`. The default value is `TRUE` if `qq.line.type="0-1"` or `estimate.params=TRUE`, otherwise it is `FALSE`. This argument is ignored if `plot.type="Tukey Mean-Difference Q-Q"`.

`margin.title` character string indicating the title printed in the top margin on each page of plots. The default value indicates the kind of Q-Q plot, the probability distribution, the sample size, and the estimation method used (if any).

`add.line` logical scalar indicating whether to add a line to the plot. If `add.line=TRUE` and `plot.type="Q-Q"`, a line determined by the value of `qq.line.type` is added to the plot. If `add.line=TRUE` and `plot.type="Tukey Mean-Difference Q-Q"`, a horizontal line at $y = 0$ is added to the plot. The default value is `add.line=FALSE`.

`qq.line.type` character string determining what kind of line to add to the Q-Q plot. Possible values are `"least squares"` (the default), `"0-1"` and `"robust"`. For the value

"least squares", a least squares line is fit and added. For the value "0-1", a line with intercept 0 and slope 1 is added. For the value "robust", a line is fit through the first and third quartiles of the x and y data. This argument is ignored if add.line=FALSE or plot.type="Tukey Mean-Difference Q-Q".

duplicate.points.method
: character string denoting how to plot points with duplicate (x, y) values. Possible values are "standard" (the default), "jitter", and "number". For the value "standard", a single plotting symbol is plotted (this is the default behavior of R). For the value "jitter", a separate plotting symbol is plotted for each duplicate point, where the plotting symbols cluster around the true value of x and y. For the value "number", a single number is plotted at (x, y) that represents how many duplicate points are at that (x, y) coordinate.

points.col
: numeric scalar or character string determining the color of the points in the plot. The default value is points.col=1. See the entry for col in the help file for par for more information.

line.col
: numeric scalar or character string determining the color of the line in the plot. The default value is points.col=1. See the entry for col in the help file for par for more information. This argument is ignored if add.line=FALSE.

line.lwd
: numeric scalar determining the width of the line in the plot. The default value is line.lwd=par("cex"). See the entry for lwd in the help file for par for more information. This argument is ignored if add.line=FALSE.

line.lty
: a numeric scalar determining the line type of the line in the plot. The default value is line.lty=1. See the entry for lty in the help file for par for more information. This argument is ignored if add.line=FALSE.

digits
: a scalar indicating how many significant digits to print for the distribution parameters. The default value is digits=.Options$digits.

same.window
: logical scalar indicating whether to produce all plots in the same graphics window (same.window=TRUE; the default), or to create a new graphics window for each separate plot (same.window=FALSE).

ask
: logical scalar supplied to the function devAskNewPage, indicating whether to prompt the user before creating a new plot within a single graphics window. The default value is TRUE if same.window=TRUE and num.pages > 1, otherwise it is FALSE.

mfrow, mar, oma, mgp, main, xlab, ylab, xlim, ylim, ...
: additional graphical parameters (see par).

Details

The function qqPlotGestalt allows the user to display several Q-Q plots or Tukey mean-difference Q-Q plots for a specified probability distribution. The distribution is specified with the arguments distribution and param.list. By default, normal (Gaussian) Q-Q plots are produced.

If estimate.params=FALSE (the default), the theoretical quantiles on the x-axis are computed using the known distribution parameters specified in param.list. If estimate.params=TRUE, the distribution parameters are estimated based on the sample, and these estimated parameters are then used to compute the theoretical quantiles. For distributions that can be specified by a location and scale parameter (e.g., Normal, Logistic, extreme value, etc.), the value of estimate.params will

not affect the general shape of the plot, only the values recorded on the x-axis. For distributions that cannot be specified by a location and scale parameter (e.g., exponential, gamma, etc.), it is recommended that `estimate.params` be set to `TRUE` since in pracitice the values of the distribution parameters are not known but must be estimated from the sample.

The purpose of `qqPlotGestalt` is to allow the user to build-up a visual memory of "typical" Q-Q plots. A Q-Q plot is a graphical tool that allows you to assess how well a particular set of observations fit a particular probability distribution. The value of this tool depends on the user having an internal reference set of Q-Q plots with which to compare the current Q-Q plot.

See the help file for `qqPlot` for more information.

Value

The `NULL` value is returned.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the REFERENCES section for `qqPlot`.

See Also

`qqPlot`.

Examples

```
# Look at eight typical normal (Gaussian) Q-Q plots for random samples
# of size 10 from a N(0,1) distribution
# Are you surprised by the variability in the plots?
#
# (Note:  you must use set.seed if you want to reproduce the exact
#         same plots more than once.)

set.seed(298)
qqPlotGestalt(same.window = FALSE)

# Add lines to these same Q-Q plots
#----------------------------------
set.seed(298)
qqPlotGestalt(same.window = FALSE, add.line = TRUE)

# Add lines to different Q-Q plots
#---------------------------------
qqPlotGestalt(same.window = FALSE, add.line = TRUE)

## Not run:
# Look at 4 sets of plots all in the same graphics window
#--------------------------------------------------------
qqPlotGestalt(add.line = TRUE, num.pages = 4)
```

```
## End(Not run)

  #==========

  # Look at Q-Q plots for a gamma distribution
  #-------------------------------------------

  qqPlotGestalt(dist = "gammaAlt",
    param.list = list(mean = 10, cv = 1),
    estimate.params = TRUE, num.pages = 3,
    same.window = FALSE, add.line = TRUE)


  # Look at Tukey Mean Difference Q-Q plots
  # for a gamma distribution
  #----------------------------------------

  qqPlotGestalt(dist = "gammaAlt",
    param.list = list(mean = 10, cv = 1),
    estimate.params = TRUE, num.pages = 3,
    plot.type = "Tukey", same.window = FALSE, add.line = TRUE)

  #==========

  # Clean up
  #---------
  graphics.off()
```

---

quantileTest *Two-Sample Rank Test to Detect a Shift in a Proportion of the "Treated" Population*

Description

Two-sample rank test to detect a positive shift in a proportion of one population (here called the "treated" population) compared to another (here called the "reference" population). This test is usually called the quantile test (Johnson et al., 1987).

Usage

```
quantileTest(x, y, alternative = "greater", target.quantile = 0.5,
  target.r = NULL, exact.p = TRUE)
```

Arguments

x	numeric vector of observations from the "treatment" group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
y	numeric vector of observations from the "reference" group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.

`alternative` character string indicating the kind of alternative hypothesis. The possible values are `"greater"` (right tail of treatment group shifted to the right of the right tail of the reference group) and `"less"` (left tail of treatment group shifted to the left of the left tail of the reference group). The default value is `alternative="greater"`.

`target.quantile` numeric scalar between 0 and 1 indicating the desired quantile to use as the lower cut off point for the test. Because of the discrete nature of empirical quantiles, the upper bound for the possible empirical quantiles will often differ from the value of `target.quantile`. The default value is `target.quantile=0.5` (i.e., the median). This argument is ignored if the argument `target.r` is supplied.

`target.r` integer indicating the rank of the observation to use as the lower cut off point for the test. The value of `target.r` must be greater than or equal to 2 and less than or equal to N (the total number of valid observations contained in the arguments `x` and `y`). The actual rank of the cut off point may differ from `target.r` in the case of tied observations in `x` and/or `y`. The default value of this argument is `NULL`, in which case the argument `target.quantile` is used to determine the lower cut off for the test.

`exact.p` logical scalar indicating whether to compute the p-value based on the exact distribution of the test statistic (`exact.p=TRUE`; the default) or based on the normal approximation (`exact.p=FALSE`).

Details

Let X denote a random variable representing measurements from a "treatment" group with cumulative distribution function (cdf)

$$F_X(t) = Pr(X \leq t) \qquad (1)$$

and let $x_1, x_2, \ldots, x_m$ denote m observations from this treatment group. Let Y denote a random variable from a "reference" group with cdf

$$F_Y(t) = Pr(Y \leq t) \qquad (2)$$

and let $y_1, y_2, \ldots, y_n$ denote n observations from this reference group. Consider the null hypothesis:

$$H_0 : F_X(t) = F_Y(t), \;\; -\infty < t < \infty \qquad (3)$$

versus the alternative hypothesis

$$H_a : F_X(t) = (1-\epsilon)F_Y(t) + \epsilon F_Z(t) \qquad (4)$$

where Z denotes some random variable with cdf

$$F_Z(t) = Pr(Z \leq t) \qquad (5)$$

and $0 < \epsilon \leq 1$, $F_Z(t) \leq F_Y(t)$ for all values of t, and $F_Z(t) \neq F_Y(t)$ for at least one value of t.

In English, the alternative hypothesis (4) says that a portion ϵ of the distribution for the treatment group (the distribution of X) is shifted to the right of the distribution for the reference group (the

distribution of Y). The alternative hypothesis (4) with $\epsilon = 1$ is the alternative hypothesis associated with testing a location shift, for which the the Wilcoxon rank sum test can be used.

Johnson et al. (1987) investigated locally most powerful rank tests for the test of the null hypothesis (3) against the alternative hypothesis (4). They considered the case when Y and Z were normal random variables and the case when the densities of Y and Z assumed only two positive values. For the latter case, the locally most powerful rank test reduces to the following procedure, which Johnson et al. (1987) call the quantile test.

1. Combine the n observations from the reference group and the m observations from the treatment group and rank them from smallest to largest. Tied observations receive the average rank of all observations tied at that value.
2. Choose a quantile q and determine the smallest rank r such that

$$\frac{r}{m+n+1} > q \qquad (6)$$

   Note that because of the discrete nature of ranks, any quantile q' such that

$$\frac{r}{m+n+1} > q' \geq \frac{r-1}{m+n+1} \qquad (7)$$

   will yield the same value for r as the quantile q does. Alternatively, choose a value of r. The bounds on an associated quantile are then given in Equation (7). Note: the component called `parameters` in the list returned by `quantileTest` contains an element named `quantile.ub`. The value of this element is the left-hand side of Equation (7).
3. Set k equal to the number of observations from the treatment group (the number of X observations) with ranks bigger than or equal to r.
4. Under the null hypothesis (3), the probability that at least k out of the r largest observations come from the treatment group is given by:

$$p = \sum_{i=k}^{r} \frac{\binom{m+n-r}{m-i}\binom{r}{i}}{\binom{m+n}{n}} \qquad (8)$$

   This probability may be approximated by:

$$p = 1 - \Phi\left(\frac{k - \mu_k - 1/2}{\sigma_k}\right) \qquad (9)$$

   where

$$\mu_k = \frac{mr}{m+n} \qquad (10)$$

$$\sigma_k^2 = \frac{mnr(m+n-r)}{(m+n)^2(m+n-1)} \qquad (11)$$

   and Φ denotes the cumulative distribution function of the standard normal distribution (USEPA, 1994, pp.7.16-7.17). (See `quantileTestPValue`.)
5. Reject the null hypothesis (3) in favor of the alternative hypothesis (4) at significance level α if $p \leq \alpha$.

Johnson et al. (1987) note that their quantile test is asymptotically equivalent to one proposed by Carrano and Moore (1982) in the context of a two-sided test. Also, when $q = 0.5$, the quantile test reduces to Mood's median test for two groups (see Zar, 2010, p.172; Conover, 1980, pp.171-178).

The optimal choice of q or r in Step 2 above (i.e., the choice that yields the largest power) depends on the true underlying distributions of Y and Z and the mixing proportion ϵ. Johnson et al. (1987) performed a simulation study and showed that the quantile test performs better than the Wilcoxon rank sum test and the normal scores test under the alternative of a mixed normal distribution with a shift of at least 2 standard deviations in the Z distribution. USEPA (1994, pp.7.17-7.21) shows that when the mixing proportion ϵ is small and the shift is large, the quantile test is more powerful than the Wilcoxon rank sum test, and when ϵ is large and the shift is small the Wilcoxon rank sum test is more powerful than the quantile test.

Value

A list of class `"htestEnvStats"` containing the results of the hypothesis test. See the help file for `htestEnvStats.object` for details.

Note

The EPA guidance document *Statistical Methods for Evaluating the Attainment of Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid Media* (USEPA, 1994, pp.4.7-4.9) recommends three different statistical tests for determining whether a remediated Superfund site has attained compliance: the Wilcoxon rank sum test, the quantile test, and the "hot measurement" comparison test. The Wilcoxon rank sum test and quantile test are nonparametric tests that compare chemical concentrations in the cleanup area with those in the reference area. The hot-measurement comparison test compares concentrations in the cleanup area with a pre-specified upper limit value *Hm* (the value of *Hm* must be negotiated between the EPA and the Superfund-site owner or operator). The Wilcoxon rank sum test is appropriate for detecting uniform failure of remedial action throughout the cleanup area. The quantile test is appropriate for detecting failure in only a few areas within the cleanup area. The hot-measurement comparison test is appropriate for detecting hot spots that need to be remediated regardless of the outcomes of the other two tests.

USEPA (1994, pp.4.7-4.9) recommends applying all three tests to all cleanup units within a cleanup area. This leads to the usual multiple comparisons problem: the probability of at least one of the tests indicating non-compliance, when in fact the cleanup area is in compliance, is greater than the pre-set Type I error level for any of the individual tests. USEPA (1994, p.3.3) recommends against using multiple comparison procedures to control the overall Type I error and suggests instead a resampling scheme where additional samples are taken in cases where non-compliance is indicated.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Carrano, A., and D. Moore. (1982). The Rationale and Methodology for Quantifying Sister Chromatid Exchange in Humans. In Heddle, J.A., ed., *Mutagenicity: New Horizons in Genetic Toxocology*. Academic Press, New York, pp.268-304.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York, Chapter 4.

Johnson, R.A., S. Verrill, and D.H. Moore. (1987). Two-Sample Rank Tests for Detecting Changes That Occur in a Small Proportion of the Treated Population. *Biometrics* **43**, 641-655.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.435-439.

USEPA. (1994). *Statistical Methods for Evaluating the Attainment of Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid Media*. EPA/230-R-94-004. Office of Policy, Planning, and Evaluation, U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

quantileTestPValue, wilcox.test, htestEnvStats.object, Hypothesis Tests.

Examples

```
# Following Example 7.5 on pages 7.23-7.24 of USEPA (1994b), perform the
# quantile test for the TcCB data (the data are stored in EPA.94b.tccb.df).
# There are n=47 observations from the reference area and m=77 observations
# from the cleanup unit.  The target rank is set to 9, resulting in a value
# of quantile.ub=0.928.  Note that the p-value is 0.0114, not 0.0117.

EPA.94b.tccb.df
#    TcCB.orig   TcCB Censored      Area
#1         0.22   0.22    FALSE Reference
#2         0.23   0.23    FALSE Reference
#...
#46        1.20   1.20    FALSE Reference
#47        1.33   1.33    FALSE Reference
#48       <0.09   0.09     TRUE   Cleanup
#49        0.09   0.09    FALSE   Cleanup
#...
#123      51.97  51.97    FALSE   Cleanup
#124     168.64 168.64    FALSE   Cleanup

# Determine the values to use for r and k for
# a desired significance level of 0.01
#--------------------------------------------

p.vals <- quantileTestPValue(m = 77, n = 47,
  r = c(rep(8, 3), rep(9, 3), rep(10, 3)),
  k = c(6, 7, 8, 7, 8, 9, 8, 9, 10))

round(p.vals, 3)
#[1] 0.355 0.122 0.019 0.264 0.081 0.011 0.193 0.053 0.007

# Choose r=9, k=9 to get a significance level of 0.011
#-----------------------------------------------------

with(EPA.94b.tccb.df,
  quantileTest(TcCB[Area=="Cleanup"], TcCB[Area=="Reference"],
  target.r = 9))
```

```
#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 e = 0
#
#Alternative Hypothesis:          Tail of Fx Shifted to Right of
#                                 Tail of Fy.
#                                 0 < e <= 1, where
#                                 Fx(t) = (1-e)*Fy(t) + e*Fz(t),
#                                 Fz(t) <= Fy(t) for all t,
#                                 and Fy != Fz
#
#Test Name:                       Quantile Test
#
#Data:                            x = TcCB[Area == "Cleanup"]
#                                 y = TcCB[Area == "Reference"]
#
#Sample Sizes:                    nx = 77
#                                 ny = 47
#
#Test Statistics:                 k (# x obs of r largest) = 9
#                                 r                        = 9
#
#Test Statistic Parameters:       m           = 77.000
#                                 n           = 47.000
#                                 quantile.ub =  0.928
#
#P-value:                         0.01136926

#==========

# Clean up
#---------

rm(p.vals)
```

---

quantileTestPValue *Compute p-Value for the Quantile Test*

Description

Compute the p-value associated with a specified combination of m, n, r, and k for the quantile test (useful for determining r and k for a given significance level α).

Usage

```
quantileTestPValue(m, n, r, k, exact.p = TRUE)
```

Arguments

m	numeric vector of integers indicating the number of observations from the "treatment" group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
n	numeric vector of integers indicating the number of observations from the "reference" group. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
r	numeric vector of integers indicating the ranks of the observations to use as the lower cut off for the quantile test. All values of r must be greater than or equal to 2 and less than or equal to the corresponding elements of m+n (the total number of observations from both groups). Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
k	numeric vector of integers indicating the number of observations from the "treatment" group contained in the r largest observations. This is the critical value used to decide whether to reject the null hypothesis. All values of k must be greater than or equal to 0 and less than or equal to the corresponding elements of r. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
exact.p	logical scalar indicating whether to compute the p-value based on the exact distribution of the test statistic (exact.p=TRUE; the default) or based on the normal approximation (exact.p=FALSE).

Details

If the arguments m, n, r, and k are not all the same length, they are replicated to be the same length as the length of the longest argument.

For details on how the p-value is computed, see the help file for quantileTest.

The function quantileTestPValue is useful for determining what values to use for r and k, given the values of m, n, and a specified significance level α. The function quantileTestPValue can be used to reproduce Tables A.6-A.9 in USEPA (1994, pp.A.22-A.25).

Value

numeric vector of p-values.

Note

See the help file for quantileTest.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for quantileTest.

See Also

quantileTest, wilcox.test, htest.object, Hypothesis Tests.

Examples

```
# Reproduce the first column of Table A.9 in USEPA (1994, p.A.25):
#-----------------------------------------------------------------

p.vals <- quantileTestPValue(m = 5, n = seq(15, 45, by = 5),
  r = c(9, 3, 4, 4, 5, 5, 6), k = c(4, 2, 2, 2, 2, 2, 2))

round(p.vals, 3)
#[1] 0.098 0.091 0.119 0.089 0.109 0.087 0.103

#==========

# Clean up
#---------

rm(p.vals)
```

Refinery.CO.df *Carbon Monoxide Emissions from Oil Refinery.*

Description

Carbon monoxide (CO) emissions (ppm) from an oil refinery near San Francisco. The refinery submitted 31 daily measurements from its stack for the period April 16, 1993 through May 16, 1993 to the Bay Area Air Quality Management District (BAAQMD). The BAAQMD made nine of its own indepent measurements for the period September 11, 1990 through March 30, 1993.

Usage

```
data(Refinery.CO.df)
```

Format

A data frame with 40 observations on the following 3 variables.

`CO.ppm` a numeric vector of CO emissions (ppm)

`Source` a factor indicating the source of the measurment (`BAAQMD` or `refinery`

`Date` a Date object indicating the date the measurement was taken

Source

Data and Story Library, http://lib.stat.cmu.edu/DASL/Datafiles/Refinery.html.

References

Zou, G.Y., C.Y. Huo, and J. Taleban. (2009). Simple Confidence Intervals for Lognormal Means and their Differences with Environmental Applications. *Environmetrics*, **20**, 172–180.

`rosnerTest` *Rosner's Test for Outliers*

Description

Perform Rosner's generalized extreme Studentized deviate test for up to k potential outliers in a dataset, assuming the data without any outliers come from a normal (Gaussian) distribution.

Usage

```
rosnerTest(x, k = 3, alpha = 0.05, warn = TRUE)
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed. There must be at least 10 non-missing, finite observations in `x`.
`k`	positive integer indicating the number of suspected outliers. The argument `k` must be between 1 and $n-2$ where n denotes the number of non-missing, finite values in the arguemnt `x`. The default value is `k=3`.
`alpha`	numeric scalar between 0 and 1 indicating the Type I error associated with the test of hypothesis. The default value is `alpha=0.05`.
`warn`	logical scalar indicating whether to issue a warning (`warn=TRUE`; the default) when the number of non-missing, finite values in `x` and the value of `k` are such that the assumed Type I error level might not be maintained. See the DETAILS section below.

Details

Let $x_1, x_2, \ldots, x_n$ denote the n observations. We assume that $n-k$ of these observations come from the same normal (Gaussian) distribution, and that the k most "extreme" observations may or may not represent observations from a different distribution. Let $x_1^*, x_2^*, \ldots, x_{n-i}^*$ denote the $n-i$ observations left after omiting the i most extreme observations, where $i = 0, 1, \ldots, k-1$. Let $\bar{x}^{(i)}$ and $s^{(i)}$ denote the mean and standard deviation, respectively, of the $n-i$ observations in the data that remain after removing the i most extreme observations. Thus, $\bar{x}^{(0)}$ and $s^{(0)}$ denote the mean and standard deviation for the full sample, and in general

$$\bar{x}^{(i)} = \frac{1}{n-i}\sum_{j=1}^{n-i} x_j^* \qquad (1)$$

$$s^{(i)} = \sqrt{\frac{1}{n-i-1}\sum_{j=1}^{n-i}(x_j^* - \bar{x}^{(i)})^2} \qquad (2)$$

For a specified value of i, the most extreme observation $x^{(i)}$ is the one that is the greatest distance from the mean for that data set, i.e.,

$$x^{(i)} = \max_{j=1,2,\ldots,n-i} |x_j^* - \bar{x}^{(i)}| \qquad (3)$$

Thus, an extreme observation may be the smallest or the largest one in that data set.

Rosner's test is based on the k statistics $R_1, R_2, \ldots, R_k$, which represent the extreme Studentized deviates computed from successively reduced samples of size $n, n-1, \ldots, n-k+1$:

$$R_{i+1} = \frac{|x^{(i)} - \bar{x}^{(i)}|}{s^{(i)}} \qquad (4)$$

Critical values for R_{i+1} are denoted λ_{i+1} and are computed as:

$$\lambda_{i+1} = \frac{t_{p,n-i-2}(n-i-1)}{\sqrt{(n-i-2+t^2_{p,n-i-2})(n-i)}} \qquad (5)$$

where $t_{p,\nu}$ denotes the p'th quantile of Student's t-distribution with ν degrees of freedom, and in this case

$$p = 1 - \frac{\alpha/2}{n-i} \qquad (6)$$

where α denotes the Type I error level.

The algorithm for determining the number of outliers is as follows:

1. Compare R_k with λ_k. If $R_k > \lambda_k$ then conclude the k most extreme values are outliers.
2. If $R_k \leq \lambda_k$ then compare R_{k-1} with λ_{k-1}. If $R_{k-1} > \lambda_{k-1}$ then conclude the $k-1$ most extreme values are outliers.
3. Continue in this fashion until a certain number of outliers have been identified or Rosner's test finds no outliers at all.

Based on a study using N=1,000 simulations, Rosner's (1983) Table 1 shows the estimated true Type I error of declaring at least one outlier when none exists for various sample sizes n ranging from 10 to 100, and the declared maximum number of outliers k ranging from 1 to 10. Based on that table, Roser (1983) declared that for an assumed Type I error level of 0.05, as long as $n \geq 25$, the estimated α levels are quite close to 0.05, and that similar results were obtained assuming a Type I error level of 0.01. However, the table below is an expanded version of Rosner's (1983) Table 1 and shows results based on N=10,000 simulations. You can see that for an assumed Type I error of 0.05, the test maintains the Type I error fairly well for sample sizes as small as $n = 3$ as long as $k = 1$, and for $n \geq 15$, as long as $k \leq 2$. Also, for an assumed Type I error of 0.01, the test maintains the Type I error fairly well for sample sizes as small as $n = 15$ as long as $k \leq 7$.

Based on these results, when `warn=TRUE`, a warning is issued for the following cases indicating that the assumed Type I error may not be correct:

- `alpha` is greater than `0.01`, the sample size is less than 15, and `k` is greater than `1`.
- `alpha` is greater than `0.01`, the sample size is at least 15 and less than 25, and `k` is greater than `2`.
- `alpha` is less than or equal to `0.01`, the sample size is less than 15, and `k` is greater than `1`.

- k is greater than 10, or greater than the floor of half of the sample size (i.e., greater than the greatest integer less than or equal to half of the sample size). A warning is given for this case because simulations have not been done for this case.

Table 1a. Observed Type I Error Levels based on 10,000 Simulations, $n = 3$ to 5.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
3	1	0.047	0.043	0.051	0.009	0.007	0.01
4	1	0.049	0.045	0.053	0.010	0.008	0.012
	2	0.107	0.101	0.113	0.021	0.018	0.024
5	1	0.048	0.044	0.053	0.008	0.006	0.009
	2	0.095	0.090	0.101	0.020	0.018	0.023

Table 1b. Observed Type I Error Levels based on 10,000 Simulations, $n = 6$ to 10.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
6	1	0.048	0.044	0.053	0.010	0.009	0.012
	2	0.085	0.080	0.091	0.017	0.015	0.020
	3	0.141	0.134	0.148	0.028	0.025	0.031
7	1	0.048	0.044	0.053	0.013	0.011	0.015
	2	0.080	0.075	0.086	0.017	0.015	0.020
	3	0.112	0.106	0.118	0.022	0.019	0.025
8	1	0.048	0.044	0.053	0.011	0.009	0.013
	2	0.080	0.074	0.085	0.017	0.014	0.019
	3	0.102	0.096	0.108	0.020	0.017	0.023
	4	0.143	0.136	0.150	0.028	0.025	0.031
9	1	0.052	0.048	0.057	0.010	0.008	0.012
	2	0.069	0.064	0.074	0.014	0.012	0.016
	3	0.097	0.091	0.103	0.018	0.015	0.021
	4	0.120	0.114	0.126	0.024	0.021	0.027
10	1	0.051	0.047	0.056	0.010	0.008	0.012
	2	0.068	0.063	0.073	0.012	0.010	0.014
	3	0.085	0.080	0.091	0.015	0.013	0.017
	4	0.106	0.100	0.112	0.021	0.018	0.024
	5	0.135	0.128	0.142	0.025	0.022	0.028

Table 1c. Observed Type I Error Levels based on 10,000 Simulations, $n = 11$ to 15.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
11	1	0.052	0.048	0.056	0.012	0.010	0.014
	2	0.070	0.065	0.075	0.014	0.012	0.017
	3	0.082	0.077	0.088	0.014	0.011	0.016
	4	0.101	0.095	0.107	0.019	0.016	0.021
	5	0.116	0.110	0.123	0.022	0.019	0.024

12	1	0.052	0.047	0.056	0.011	0.009	0.013
	2	0.067	0.062	0.072	0.011	0.009	0.013
	3	0.074	0.069	0.080	0.016	0.013	0.018
	4	0.088	0.082	0.093	0.016	0.014	0.019
	5	0.099	0.093	0.105	0.016	0.013	0.018
	6	0.117	0.111	0.123	0.021	0.018	0.023
13	1	0.048	0.044	0.052	0.010	0.008	0.012
	2	0.064	0.059	0.069	0.014	0.012	0.016
	3	0.070	0.065	0.075	0.013	0.011	0.015
	4	0.079	0.074	0.084	0.014	0.012	0.017
	5	0.088	0.083	0.094	0.015	0.013	0.018
	6	0.109	0.103	0.115	0.020	0.017	0.022
14	1	0.046	0.042	0.051	0.009	0.007	0.011
	2	0.062	0.057	0.066	0.012	0.010	0.014
	3	0.069	0.064	0.074	0.012	0.010	0.014
	4	0.077	0.072	0.082	0.015	0.013	0.018
	5	0.084	0.079	0.090	0.016	0.013	0.018
	6	0.091	0.085	0.097	0.017	0.014	0.019
	7	0.107	0.101	0.113	0.018	0.016	0.021
15	1	0.054	0.050	0.059	0.010	0.008	0.012
	2	0.057	0.053	0.062	0.010	0.008	0.012
	3	0.065	0.060	0.069	0.013	0.011	0.016
	4	0.073	0.068	0.078	0.014	0.011	0.016
	5	0.074	0.069	0.079	0.012	0.010	0.014
	6	0.086	0.081	0.092	0.015	0.013	0.017
	7	0.099	0.094	0.105	0.018	0.015	0.020

Table 1d. Observed Type I Error Levels based on 10,000 Simulations, $n = 16$ to 20.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
16	1	0.052	0.048	0.057	0.010	0.008	0.012
	2	0.055	0.051	0.059	0.011	0.009	0.013
	3	0.068	0.063	0.073	0.011	0.009	0.013
	4	0.074	0.069	0.079	0.015	0.013	0.017
	5	0.077	0.072	0.082	0.015	0.013	0.018
	6	0.075	0.070	0.080	0.013	0.011	0.016
	7	0.087	0.082	0.093	0.017	0.014	0.020
	8	0.096	0.090	0.101	0.016	0.014	0.019
17	1	0.047	0.043	0.051	0.008	0.007	0.010
	2	0.059	0.054	0.063	0.011	0.009	0.013
	3	0.062	0.057	0.067	0.012	0.010	0.014
	4	0.070	0.065	0.075	0.012	0.009	0.014
	5	0.069	0.064	0.074	0.012	0.010	0.015
	6	0.071	0.066	0.076	0.015	0.012	0.017
	7	0.081	0.076	0.087	0.014	0.012	0.016
	8	0.083	0.078	0.088	0.015	0.013	0.017
18	1	0.051	0.047	0.055	0.010	0.008	0.012

	2	0.056	0.052	0.061	0.012	0.010	0.014
	3	0.065	0.060	0.070	0.012	0.010	0.015
	4	0.065	0.060	0.070	0.013	0.011	0.015
	5	0.069	0.064	0.074	0.012	0.010	0.014
	6	0.068	0.063	0.073	0.014	0.011	0.016
	7	0.072	0.067	0.077	0.014	0.011	0.016
	8	0.076	0.071	0.081	0.012	0.010	0.014
	9	0.081	0.076	0.086	0.012	0.010	0.014
19	1	0.051	0.046	0.055	0.008	0.006	0.010
	2	0.059	0.055	0.064	0.012	0.010	0.014
	3	0.059	0.054	0.064	0.011	0.009	0.013
	4	0.061	0.057	0.066	0.012	0.010	0.014
	5	0.067	0.062	0.072	0.013	0.010	0.015
	6	0.066	0.061	0.071	0.011	0.009	0.013
	7	0.069	0.064	0.074	0.013	0.011	0.015
	8	0.074	0.069	0.079	0.012	0.010	0.014
	9	0.082	0.077	0.087	0.015	0.013	0.018
20	1	0.053	0.048	0.057	0.011	0.009	0.013
	2	0.056	0.052	0.061	0.010	0.008	0.012
	3	0.060	0.056	0.065	0.009	0.007	0.011
	4	0.063	0.058	0.068	0.012	0.010	0.014
	5	0.063	0.059	0.068	0.014	0.011	0.016
	6	0.063	0.058	0.067	0.011	0.009	0.013
	7	0.065	0.061	0.070	0.011	0.009	0.013
	8	0.070	0.065	0.076	0.012	0.010	0.014
	9	0.076	0.070	0.081	0.013	0.011	0.015
	10	0.081	0.076	0.087	0.012	0.010	0.014

Table 1e. Observed Type I Error Levels based on 10,000 Simulations, $n = 21$ to 25.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
21	1	0.054	0.049	0.058	0.013	0.011	0.015
	2	0.054	0.049	0.058	0.012	0.010	0.014
	3	0.058	0.054	0.063	0.012	0.010	0.014
	4	0.058	0.054	0.063	0.011	0.009	0.013
	5	0.064	0.059	0.069	0.013	0.011	0.016
	6	0.066	0.061	0.071	0.012	0.010	0.015
	7	0.063	0.058	0.068	0.013	0.011	0.015
	8	0.066	0.061	0.071	0.010	0.008	0.012
	9	0.073	0.068	0.078	0.013	0.011	0.015
	10	0.071	0.066	0.076	0.012	0.010	0.014
22	1	0.047	0.042	0.051	0.010	0.008	0.012
	2	0.058	0.053	0.062	0.012	0.010	0.015
	3	0.056	0.052	0.061	0.010	0.008	0.012
	4	0.059	0.055	0.064	0.012	0.010	0.014
	5	0.061	0.057	0.066	0.009	0.008	0.011
	6	0.063	0.058	0.068	0.013	0.010	0.015

	7	0.065	0.060	0.070	0.013	0.010	0.015
	8	0.065	0.060	0.070	0.014	0.012	0.016
	9	0.065	0.060	0.070	0.012	0.010	0.014
	10	0.067	0.062	0.072	0.012	0.009	0.014
23	1	0.051	0.047	0.056	0.008	0.007	0.010
	2	0.056	0.052	0.061	0.010	0.009	0.012
	3	0.056	0.052	0.061	0.011	0.009	0.013
	4	0.062	0.057	0.066	0.011	0.009	0.013
	5	0.061	0.056	0.065	0.010	0.009	0.012
	6	0.060	0.055	0.064	0.012	0.010	0.014
	7	0.062	0.057	0.066	0.011	0.009	0.013
	8	0.063	0.058	0.068	0.012	0.010	0.014
	9	0.066	0.061	0.071	0.012	0.010	0.014
	10	0.068	0.063	0.073	0.014	0.012	0.017
24	1	0.051	0.046	0.055	0.010	0.008	0.012
	2	0.056	0.051	0.060	0.011	0.009	0.013
	3	0.058	0.053	0.062	0.010	0.008	0.012
	4	0.060	0.056	0.065	0.013	0.011	0.015
	5	0.057	0.053	0.062	0.012	0.010	0.014
	6	0.065	0.060	0.069	0.011	0.009	0.013
	7	0.062	0.057	0.066	0.012	0.010	0.014
	8	0.060	0.055	0.065	0.012	0.010	0.014
	9	0.066	0.061	0.071	0.012	0.010	0.014
	10	0.064	0.059	0.068	0.012	0.010	0.015
25	1	0.054	0.050	0.059	0.012	0.009	0.014
	2	0.055	0.051	0.060	0.010	0.008	0.012
	3	0.057	0.052	0.062	0.011	0.009	0.013
	4	0.055	0.051	0.060	0.011	0.009	0.013
	5	0.060	0.055	0.065	0.012	0.010	0.014
	6	0.060	0.055	0.064	0.011	0.009	0.013
	7	0.057	0.052	0.061	0.011	0.009	0.013
	8	0.062	0.058	0.067	0.011	0.009	0.013
	9	0.058	0.053	0.062	0.012	0.010	0.014
	10	0.061	0.057	0.066	0.010	0.008	0.012

Table 1f. Observed Type I Error Levels based on 10,000 Simulations, $n = 26$ to 30.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
26	1	0.051	0.047	0.055	0.012	0.010	0.014
	2	0.057	0.053	0.062	0.013	0.011	0.015
	3	0.055	0.050	0.059	0.012	0.010	0.014
	4	0.055	0.051	0.060	0.010	0.008	0.012
	5	0.058	0.054	0.063	0.011	0.009	0.013
	6	0.061	0.056	0.066	0.012	0.010	0.014
	7	0.059	0.054	0.064	0.011	0.009	0.013
	8	0.060	0.056	0.065	0.010	0.008	0.012
	9	0.060	0.056	0.065	0.011	0.009	0.013

	10	0.061	0.056	0.065	0.011	0.009	0.013
27	1	0.050	0.046	0.054	0.009	0.007	0.011
	2	0.054	0.050	0.059	0.011	0.009	0.013
	3	0.062	0.057	0.066	0.012	0.010	0.014
	4	0.063	0.058	0.068	0.011	0.009	0.013
	5	0.051	0.047	0.055	0.010	0.008	0.012
	6	0.058	0.053	0.062	0.011	0.009	0.013
	7	0.060	0.056	0.065	0.010	0.008	0.012
	8	0.056	0.052	0.061	0.010	0.008	0.012
	9	0.061	0.056	0.066	0.012	0.010	0.014
	10	0.055	0.051	0.060	0.008	0.006	0.010
28	1	0.049	0.045	0.053	0.010	0.008	0.011
	2	0.057	0.052	0.061	0.011	0.009	0.013
	3	0.056	0.052	0.061	0.012	0.009	0.014
	4	0.057	0.053	0.062	0.011	0.009	0.013
	5	0.057	0.053	0.062	0.010	0.008	0.012
	6	0.056	0.051	0.060	0.010	0.008	0.012
	7	0.057	0.052	0.061	0.010	0.008	0.012
	8	0.058	0.054	0.063	0.011	0.009	0.013
	9	0.054	0.050	0.058	0.011	0.009	0.013
	10	0.062	0.057	0.067	0.011	0.009	0.013
29	1	0.049	0.045	0.053	0.011	0.009	0.013
	2	0.053	0.048	0.057	0.010	0.008	0.012
	3	0.056	0.051	0.060	0.010	0.009	0.012
	4	0.055	0.050	0.059	0.010	0.008	0.012
	5	0.056	0.051	0.060	0.010	0.008	0.012
	6	0.057	0.053	0.062	0.012	0.010	0.014
	7	0.055	0.050	0.059	0.010	0.008	0.012
	8	0.057	0.052	0.061	0.011	0.009	0.013
	9	0.056	0.051	0.061	0.011	0.009	0.013
	10	0.057	0.052	0.061	0.011	0.009	0.013
30	1	0.050	0.046	0.054	0.009	0.007	0.011
	2	0.054	0.049	0.058	0.011	0.009	0.013
	3	0.056	0.052	0.061	0.012	0.010	0.015
	4	0.054	0.049	0.058	0.010	0.008	0.012
	5	0.058	0.053	0.063	0.012	0.010	0.014
	6	0.062	0.058	0.067	0.012	0.010	0.014
	7	0.056	0.052	0.061	0.012	0.010	0.014
	8	0.059	0.054	0.064	0.011	0.009	0.013
	9	0.056	0.052	0.061	0.010	0.009	0.012
	10	0.058	0.053	0.062	0.012	0.010	0.015

Table 1g. Observed Type I Error Levels based on 10,000 Simulations, n = 31 to 35.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
31	1	0.051	0.047	0.056	0.009	0.007	0.011
	2	0.054	0.050	0.059	0.010	0.009	0.012

	3	0.053	0.049	0.058	0.010	0.008	0.012
	4	0.055	0.050	0.059	0.010	0.008	0.012
	5	0.053	0.049	0.057	0.011	0.009	0.013
	6	0.055	0.050	0.059	0.010	0.008	0.012
	7	0.055	0.050	0.059	0.012	0.010	0.014
	8	0.056	0.051	0.060	0.010	0.008	0.012
	9	0.057	0.053	0.062	0.011	0.009	0.013
	10	0.058	0.053	0.062	0.011	0.009	0.013
32	1	0.054	0.049	0.058	0.010	0.008	0.012
	2	0.054	0.050	0.059	0.010	0.008	0.012
	3	0.052	0.047	0.056	0.009	0.007	0.011
	4	0.056	0.051	0.060	0.011	0.009	0.013
	5	0.056	0.052	0.061	0.011	0.009	0.013
	6	0.055	0.051	0.060	0.011	0.009	0.013
	7	0.055	0.051	0.060	0.010	0.008	0.012
	8	0.055	0.051	0.060	0.010	0.008	0.012
	9	0.057	0.053	0.062	0.012	0.010	0.014
	10	0.054	0.050	0.059	0.010	0.008	0.012
33	1	0.051	0.046	0.055	0.011	0.009	0.013
	2	0.055	0.051	0.060	0.011	0.009	0.013
	3	0.056	0.052	0.061	0.010	0.008	0.012
	4	0.052	0.048	0.057	0.010	0.008	0.012
	5	0.055	0.050	0.059	0.010	0.008	0.012
	6	0.058	0.053	0.062	0.011	0.009	0.013
	7	0.057	0.052	0.061	0.010	0.008	0.012
	8	0.058	0.054	0.063	0.011	0.009	0.013
	9	0.057	0.053	0.062	0.012	0.010	0.014
	10	0.055	0.051	0.060	0.011	0.009	0.013
34	1	0.052	0.048	0.056	0.009	0.007	0.011
	2	0.053	0.049	0.058	0.011	0.009	0.013
	3	0.055	0.050	0.059	0.012	0.010	0.014
	4	0.056	0.052	0.061	0.010	0.008	0.012
	5	0.053	0.048	0.057	0.009	0.007	0.011
	6	0.055	0.050	0.059	0.010	0.008	0.012
	7	0.052	0.048	0.057	0.012	0.010	0.014
	8	0.055	0.050	0.059	0.009	0.008	0.011
	9	0.055	0.051	0.060	0.011	0.009	0.013
	10	0.054	0.049	0.058	0.010	0.008	0.012
35	1	0.051	0.046	0.055	0.010	0.009	0.012
	2	0.054	0.049	0.058	0.010	0.009	0.012
	3	0.055	0.050	0.059	0.010	0.009	0.012
	4	0.053	0.048	0.057	0.011	0.009	0.013
	5	0.056	0.051	0.061	0.011	0.009	0.013
	6	0.055	0.051	0.059	0.012	0.010	0.014
	7	0.054	0.050	0.059	0.011	0.009	0.013
	8	0.054	0.049	0.058	0.011	0.009	0.013
	9	0.061	0.056	0.066	0.012	0.010	0.014
	10	0.053	0.048	0.057	0.011	0.009	0.013

Table 1h. Observed Type I Error Levels based on 10,000 Simulations, n = 36 to 40.

		Assumed	$\alpha = 0.05$		**Assumed**	$\alpha = 0.01$	
n	k	$\hat{\alpha}$	*95% LCL*	*95% UCL*	$\hat{\alpha}$	*95% LCL*	*95% UCL*
36	1	0.047	0.043	0.051	0.010	0.008	0.012
	2	0.058	0.053	0.062	0.012	0.010	0.015
	3	0.052	0.047	0.056	0.009	0.007	0.011
	4	0.052	0.048	0.056	0.012	0.010	0.014
	5	0.052	0.048	0.057	0.010	0.008	0.012
	6	0.055	0.051	0.059	0.012	0.010	0.014
	7	0.053	0.048	0.057	0.011	0.009	0.013
	8	0.056	0.051	0.060	0.012	0.010	0.014
	9	0.056	0.051	0.060	0.011	0.009	0.013
	10	0.056	0.051	0.060	0.011	0.009	0.013
37	1	0.050	0.046	0.055	0.010	0.008	0.012
	2	0.054	0.049	0.058	0.011	0.009	0.013
	3	0.054	0.049	0.058	0.011	0.009	0.013
	4	0.054	0.050	0.058	0.010	0.008	0.012
	5	0.054	0.049	0.058	0.010	0.008	0.012
	6	0.054	0.050	0.058	0.011	0.009	0.013
	7	0.055	0.051	0.060	0.010	0.008	0.012
	8	0.055	0.050	0.059	0.011	0.009	0.013
	9	0.053	0.049	0.058	0.011	0.009	0.013
	10	0.049	0.045	0.054	0.009	0.007	0.011
38	1	0.049	0.045	0.053	0.009	0.007	0.011
	2	0.052	0.047	0.056	0.008	0.007	0.010
	3	0.054	0.050	0.059	0.011	0.009	0.013
	4	0.055	0.050	0.059	0.011	0.009	0.013
	5	0.056	0.052	0.061	0.012	0.010	0.014
	6	0.055	0.050	0.059	0.011	0.009	0.013
	7	0.049	0.045	0.053	0.009	0.007	0.011
	8	0.052	0.048	0.057	0.010	0.008	0.012
	9	0.054	0.050	0.059	0.010	0.009	0.012
	10	0.055	0.050	0.059	0.011	0.009	0.013
39	1	0.047	0.043	0.051	0.010	0.008	0.012
	2	0.055	0.051	0.059	0.010	0.008	0.012
	3	0.053	0.049	0.057	0.010	0.008	0.012
	4	0.053	0.049	0.058	0.010	0.009	0.012
	5	0.052	0.048	0.057	0.010	0.008	0.012
	6	0.053	0.049	0.058	0.010	0.008	0.012
	7	0.057	0.052	0.061	0.011	0.009	0.013
	8	0.057	0.053	0.062	0.012	0.010	0.014
	9	0.050	0.046	0.055	0.010	0.008	0.012
	10	0.056	0.051	0.060	0.011	0.009	0.013
40	1	0.049	0.045	0.054	0.010	0.008	0.012
	2	0.052	0.048	0.057	0.010	0.009	0.012
	3	0.055	0.050	0.059	0.011	0.009	0.013
	4	0.054	0.050	0.059	0.011	0.009	0.013

5	0.054	0.050	0.059	0.010	0.008	0.012
6	0.049	0.045	0.053	0.010	0.008	0.012
7	0.056	0.051	0.060	0.011	0.009	0.013
8	0.054	0.050	0.059	0.011	0.009	0.013
9	0.047	0.043	0.052	0.010	0.008	0.011
10	0.058	0.054	0.063	0.010	0.008	0.012

Value

A list of class `"gofOutlier"` containing the results of the hypothesis test. See the help file for `gofOutlier.object` for details.

Note

Rosner's test is a commonly used test for "outliers" when you are willing to assume that the data without outliers follows a normal (Gaussian) distribution. It is designed to avoid *masking*, which occurs when an outlier goes undetected because it is close in value to another outlier.

Rosner's test is a kind of discordancy test (Barnett and Lewis, 1995). The test statistic of a discordancy test is usually a ratio: the numerator is the difference between the suspected outlier and some summary statistic of the data set (e.g., mean, next largest observation, etc.), while the denominator is always a measure of spread within the data (e.g., standard deviation, range, etc.). Both USEPA (2009) and USEPA (2013a,b) discuss two commonly used discordancy tests: Dixon's test and Rosner's test. Both of these tests assume that all of the data that are not outliers come from a normal (Gaussian) distribution.

There are many forms of Dixon's test (Barnett and Lewis, 1995). The one presented in USEPA (2009) and USEPA (20013a,b) assumes just one outlier (Dixon, 1953). This test is vulnerable to "masking" in which the presence of several outliers masks the fact that even one outlier is present. There are also other forms of Dixon's test that allow for more than one outlier based on a sequence of sub-tests, but these tests are also vulnerable to masking.

Rosner's test allows you to test for several possible outliers and avoids the problem of masking. Rosner's test requires you to set the number of suspected outliers, k, in advance. As in the case of Dixon's test, there are several forms of Rosner's test, so you need to be aware of which one you are using. The form of Rosner's test presented in USEPA (2009) is based on the extreme Studentized deviate (ESD) (Rosner, 1975), whereas the form of Rosner's test performed by the **EnvStats** function `rosnerTest` and presented in USEPA (2013a,b) is based on the **generalized** ESD (Rosner, 1983; Gilbert, 1987). USEPA (2013a, p. 190) cites both Rosner (1975) and Rosner (1983), but presents only the test given in Rosner (1983). Rosner's test based on the ESD has the appropriate Type I error level if there are no outliers in the dataset, but if there are actually say m outliers, where $m < k$, then the ESD version of Rosner's test tends to declare more than m outliers with a probability that is greater than the stated Type I error level (referred to as "swamping"). Rosner's test based on the generalized ESD fixes this problem. USEPA (2013a, pp. 17, 191) incorrectly states that the generalized ESD version of Rosner's test is vulnerable to masking. Surprisingly, the well-known book on statistical outliers by Barnett and Lewis (1995) does not discuss Rosner's generalized ESD test.

As noted, using Rosner's test requires specifying the number of suspected outliers, k, in advance. USEPA (2013a, pp.190-191) states: "A graphical display (Q-Q plot) can be used to identify suspected outliers needed to perform the Rosner test", and USEPA (2009, p. 12-11) notes: "A potential drawback of Rosner's test is that the user must first identify the maximum number of potential

outliers (k) prior to running the test. Therefore, this requirement makes the test ill-advised as an automatic outlier screening tool, and somewhat reliant on the user to identify candidate outliers."

When observations contain non-detect values (NDs), USEPA (2013a, p. 191) states: "one may replace the NDs by their respective detection limits (DLs), DL/2, or may just ignore them ...." This is bad advice, as this method of dealing with non-detects will produce Type I error rates that are not correct.

OUTLIERS ARE NOT NECESSARILY INCORRECT VALUES

Whether an observation is an "outlier" depends on the underlying assumed statistical model. McBean and Rovers (1992) state:

"It may be possible to ignore the outlier if a physical rationale is available but, failing that, the value must be included .... Note that the use of statistics does not interpret the facts, it simply makes the facts easier to see. Therefore, it is incumbent on the analyst to identify whether or not the high value ... is truly representative of the chemical being monitored or, instead, is an outlier for reasons such as a result of sampling or laboratory error."

USEPA (2006, p.51) states:

"If scientific reasoning does not explain the outlier, it should not be discarded from the data set."

Finally, an editorial by the Editor-in-Chief of the journal *Science* deals with this topic (McNutt, 2014).

You can use the functions `qqPlot` and `gofTest` to explore other possible statistical models for the data, or you can use nonparametric statistics if you do not want to assume a particular distribution.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Barnett, V., and T. Lewis. (1995). *Outliers in Statistical Data*. Third Edition. John Wiley & Sons, Chichester, UK, pp. 235–236.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY, pp.188–191.

McBean, E.A, and F.A. Rovers. (1992). Estimation of the Probability of Exceedance of Contaminant Concentrations. *Ground Water Monitoring Review* **Winter**, pp. 115–119.

McNutt, M. (2014). Raising the Bar. *Science* **345**(6192), p. 9.

Rosner, B. (1975). On the Detection of Many Outliers. *Technometrics* **17**, 221–227.

Rosner, B. (1983). Percentage Points for a Generalized ESD Many-Outlier Procedure. *Technometrics* **25**, 165–172.

USEPA. (2006). *Data Quality Assessment: A Reviewer's Guide*. EPA QA/G-9R. EPA/240/B-06/002, February 2006. Office of Environmental Information, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C., pp. 12-10 to 12-14.

USEPA. (2013a). *ProUCL Version 5.0.00 Technical Guide*. EPA/600/R-07/041, September 2013. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C., pp. 190–195.

USEPA. (2013b). *ProUCL Version 5.0.00 User Guide*. EPA/600/R-07/041, September 2013. Office of Research and Development. U.S. Environmental Protection Agency, Washington, D.C., pp. 190–195.

See Also

gofTest, gofOutlier.object, print.gofOutlier, Normal, qqPlot.

Examples

```
# Combine 30 observations from a normal distribution with mean 3 and
# standard deviation 2, with 3 observations from a normal distribution
# with mean 10 and standard deviation 1, then run Rosner's Test on these
# data, specifying k=4 potential outliers based on looking at the
# normal Q-Q plot.
# (Note: the call to set.seed simply allows you to reproduce
# this example.)

set.seed(250)

dat <- c(rnorm(30, mean = 3, sd = 2), rnorm(3, mean = 10, sd = 1))

dev.new()
qqPlot(dat)

rosnerTest(dat, k = 4)

#Results of Outlier Test
#-------------------------
#
#Test Method:                     Rosner's Test for Outliers
#
#Hypothesized Distribution:       Normal
#
#Data:                            dat
#
#Sample Size:                     33
#
#Test Statistics:                 R.1 = 2.848514
#                                 R.2 = 3.086875
#                                 R.3 = 3.033044
#                                 R.4 = 2.380235
#
#Test Statistic Parameter:        k = 4
#
#Alternative Hypothesis:          Up to 4 observations are not
#                                 from the same Distribution.
#
#Type I Error:                    5%
```

```
#
#Number of Outliers Detected:     3
#
#  i   Mean.i     SD.i      Value Obs.Num    R.i+1 lambda.i+1 Outlier
#1 0 3.549744 2.531011 10.7593656      33 2.848514   2.951949    TRUE
#2 1 3.324444 2.209872 10.1460427      31 3.086875   2.938048    TRUE
#3 2 3.104392 1.856109  8.7340527      32 3.033044   2.923571    TRUE
#4 3 2.916737 1.560335 -0.7972275      25 2.380235   2.908473   FALSE

#----------
# Clean up

rm(dat)
graphics.off()

#--------------------------------------------------------------------

# Example 12-4 of USEPA (2009, page 12-12) gives an example of
# using Rosner's test to test for outliers in napthalene measurements (ppb)
# taken at 5 background wells over 5 quarters.  The data for this example
# are stored in EPA.09.Ex.12.4.naphthalene.df.

EPA.09.Ex.12.4.naphthalene.df
#   Quarter Well Naphthalene.ppb
#1        1 BW.1            3.34
#2        2 BW.1            5.39
#3        3 BW.1            5.74
# ...
#23       3 BW.5            5.53
#24       4 BW.5            4.42
#25       5 BW.5           35.45

longToWide(EPA.09.Ex.12.4.naphthalene.df, "Naphthalene.ppb", "Quarter", "Well",
  paste.row.name = TRUE)
#          BW.1 BW.2  BW.3 BW.4  BW.5
#Quarter.1 3.34 5.59  1.91 6.12  8.64
#Quarter.2 5.39 5.96  1.74 6.05  5.34
#Quarter.3 5.74 1.47 23.23 5.18  5.53
#Quarter.4 6.88 2.57  1.82 4.43  4.42
#Quarter.5 5.85 5.39  2.02 1.00 35.45


# Look at Q-Q plots for both the raw and log-transformed data
#-----------------------------------------------------------

dev.new()
with(EPA.09.Ex.12.4.naphthalene.df,
  qqPlot(Naphthalene.ppb, add.line = TRUE,
    main = "Figure 12-6.  Naphthalene Probability Plot"))

dev.new()
with(EPA.09.Ex.12.4.naphthalene.df,
  qqPlot(Naphthalene.ppb, dist = "lnorm", add.line = TRUE,
```

```
    main = "Figure 12-7.  Log Naphthalene Probability Plot"))


# Test for 2 potential outliers on the original scale:
#-----------------------------------------------------

with(EPA.09.Ex.12.4.naphthalene.df, rosnerTest(Naphthalene.ppb, k = 2))

#Results of Outlier Test
#-------------------------
#
#Test Method:                     Rosner's Test for Outliers
#
#Hypothesized Distribution:       Normal
#
#Data:                            Naphthalene.ppb
#
#Sample Size:                     25
#
#Test Statistics:                 R.1 = 3.930957
#                                 R.2 = 4.160223
#
#Test Statistic Parameter:        k = 2
#
#Alternative Hypothesis:          Up to 2 observations are not
#                                 from the same Distribution.
#
#Type I Error:                    5%
#
#Number of Outliers Detected:     2
#
#  i  Mean.i     SD.i Value Obs.Num    R.i+1 lambda.i+1 Outlier
#1 0 6.44240 7.379271 35.45      25 3.930957   2.821681    TRUE
#2 1 5.23375 4.325790 23.23      13 4.160223   2.801551    TRUE

#----------
# Clean up

graphics.off()
```

`serialCorrelationTest` *Test for the Presence of Serial Correlation*

Description

`serialCorrelationTest` is a generic function used to test for the presence of lag-one serial correlation using either the rank von Neumann ratio test, the normal approximation based on the Yule-Walker estimate of lag-one correlation, or the normal approximation based on the MLE of lag-one correlation. The function invokes particular `methods` which depend on the `class` of the first argument.

Currently, there is a default method and a method for objects of class `"lm"`.

Usage

```
  serialCorrelationTest(x, ...)

  ## Default S3 method:
serialCorrelationTest(x, test = "rank.von.Neumann",
    alternative = "two.sided", conf.level = 0.95, ...)

  ## S3 method for class 'lm'
serialCorrelationTest(x, test = "rank.von.Neumann",
    alternative = "two.sided", conf.level = 0.95, ...)
```

Arguments

x	numeric vector of observations, a numeric univariate time series of class "ts", or an object of class "lm". Undefined (NaN) and infinite (Inf, -Inf) values are not allowed for x when x is a numeric vector or time series, nor for the residuals associated with x when x is an object of class "lm". When test="AR1.mle", missing (NA) values are allowed, otherwise they are not allowed. When x is a numeric vector of observations or a numeric univariate time series of class "ts", it must contain at least 3 non-missing values. When x is an object of class "lm", the residuals must contain at least 3 non-missing values. Note: when x is an object of class "lm", the linear model should have been fit using the argument na.action=na.exclude in the call to lm in order to correctly deal with missing values.
test	character string indicating which test to use. The possible values are: "rank.von.Neumann" (rank von Neumann ratio test; the default), "AR1.yw" (z-test based on Yule-Walker lag-one estimate of correlation), and "AR1.mle" (z-test based on MLE of lag-one correlation).
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
conf.level	numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the population lag-one autocorrelation. The default value is conf.level=0.95.
...	optional arguments for possible future methods. Currently not used.

Details

Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote n observations from a stationary time series sampled at equispaced points in time with normal (Gaussian) errors. The function serialCorrelationTest tests the null hypothesis:

$$H_0 : \rho_1 = 0 \qquad (1)$$

where ρ_1 denotes the true lag-1 autocorrelation (also called the lag-1 serial correlation coefficient). Actually, the null hypothesis is that the lag-k autocorrelation is 0 for all values of k greater than 0 (i.e., the time series is purely random).

In the case when the argument `x` is a linear model, the function `serialCorrelationTest` tests the null hypothesis (1) for the residuals.

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`):

$$H_a : \rho_1 > 0 \qquad (2)$$

the lower one-sided alternative (`alternative="less"`):

$$H_a : \rho_1 < 0 \qquad (3)$$

and the two-sided alternative:

$$H_a : \rho_1 \neq 0 \qquad (4)$$

Testing the Null Hypothesis of No Lag-1 Autocorrelation
There are several possible methods for testing the null hypothesis (1) versus any of the three alternatives (2)-(4). The function `serialCorrelationTest` allows you to use one of three possible tests:

- The rank von Neuman ratio test.
- The test based on the normal approximation for the distribution of the Yule-Walker estimate of lag-one correlation.
- The test based on the normal approximation for the distribution of the maximum likelihood estimate (MLE) of lag-one correlation.

Each of these tests is described below.

Test Based on Yule-Walker Estimate (`test="AR1.yw"`)
The Yule-Walker estimate of the lag-1 autocorrelation is given by:

$$\hat{\rho}_1 = \frac{\hat{\gamma}_1}{\hat{\gamma}_0} \qquad (5)$$

where

$$\hat{\gamma}_k = \frac{1}{n}\sum_{t=1}^{n-k}(x_t - \bar{x})(x_{t+k} - \bar{x}) \qquad (6)$$

is the estimate of the lag-k autocovariance. (This estimator does not allow for missing values.)

Under the null hypothesis (1), the estimator of lag-1 correlation in Equation (5) is approximately distributed as a normal (Gaussian) random variable with mean 0 and variance given by:

$$Var(\hat{\rho}_1) \approx \frac{1}{n} \qquad (7)$$

(Box and Jenkins, 1976, pp.34-35). Thus, the null hypothesis (1) can be tested with the statistic

$$z = \sqrt{n}\hat{\rho}_1 \qquad (8)$$

which is distributed approximately as a standard normal random variable under the null hypothesis that the lag-1 autocorrelation is 0.

Test Based on the MLE (`test="AR1.mle"`)
The function `serialCorrelationTest` the R function `arima` to compute the MLE of the lag-one autocorrelation and the estimated variance of this estimator. As for the test based on the Yule-Walker estimate, the z-statistic is computed as the estimated lag-one autocorrelation divided by the square root of the estimated variance.

Test Based on Rank von Neumann Ratio (`test="rank.von.Neumann"`)
The null distribution of the serial correlation coefficient may be badly affected by departures from normality in the underlying process (Cox, 1966; Bartels, 1977). It is therefore a good idea to consider using a nonparametric test for randomness if the normality of the underlying process is in doubt (Bartels, 1982).

Wald and Wolfowitz (1943) introduced the rank serial correlation coefficient, which for lag-1 autocorrelation is simply the Yule-Walker estimate (Equation (5) above) with the actual observations replaced with their ranks.

von Neumann et al. (1941) introduced a test for randomness in the context of testing for trend in the mean of a process. Their statistic is given by:

$$V = \frac{\sum_{i=1}^{n-1}(x_i - x_{i+1})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad (9)$$

which is the ratio of the square of successive differences to the usual sums of squared deviations from the mean. This statistic is bounded between 0 and 4, and for a purely random process is symmetric about 2. Small values of this statistic indicate possible positive autocorrelation, and large values of this statistics indicate possible negative autocorrelation. Durbin and Watson (1950, 1951, 1971) proposed using this statistic in the context of checking the independence of residuals from a linear regression model and provided tables for the distribution of this statistic. This statistic is therefore often called the "Durbin-Watson statistic" (Draper and Smith, 1998, p.181).

The rank version of the von Neumann ratio statistic is given by:

$$V_{rank} = \frac{\sum_{i=1}^{n-1}(R_i - R_{i+1})^2}{\sum_{i=1}^{n}(R_i - \bar{R})^2} \qquad (10)$$

where R_i denotes the rank of the i'th observation (Bartels, 1982). (This test statistic does not allow for missing values.) In the absence of ties, the denominator of this test statistic is equal to

$$\sum_{i=1}^{n}(R_i - \bar{R})^2 = \frac{n(n^2-1)}{12} \qquad (11)$$

The range of the V_{rank} test statistic is given by:

$$[\frac{12}{(n)(n+1)}, 4 - \frac{12}{(n)(n+1)}] \qquad (12)$$

if n is even, with a negligible adjustment if n is odd (Bartels, 1982), so asymptotically the range is from 0 to 4, just as for the V test statistic in Equation (9) above.

Bartels (1982) shows that asymptotically, the rank von Neumann ratio statistic is a linear transformation of the rank serial correlation coefficient, so any asymptotic results apply to both statistics.

For any fixed sample size n, the exact distribution of the V_{rank} statistic in Equation (10) above can be computed by simply computing the value of V_{rank} for all possible permutations of the serial

order of the ranks. Based on this exact distribution, Bartels (1982) presents a table of critical values for the numerator of the RVN statistic for sample sizes between 4 and 10.

Determining the exact distribution of V_{rank} becomes impractical as the sample size increases. For values of n between 10 and 100, Bartels (1982) approximated the distribution of V_{rank} by a beta distribution over the range 0 to 4 with shape parameters `shape1`=ν and `shape2`=ω and:

$$\nu = \omega = \frac{5n(n+1)(n-1)^2}{2(n-2)(5n^2-2n-9)} - \frac{1}{2} \qquad (13)$$

Bartels (1982) checked this approximation by simulating the distribution of V_{rank} for $n = 25$ and $n = 50$ and comparing the empirical quantiles at 0.005, 0.01, 0.025, 0.05, and 0.1 with the approximated quantiles based on the beta distribution. He found that the quantiles agreed to 2 decimal places for eight of the 10 values, and differed by 0.01 for the other two values.

Note: The definition of the beta distribution assumes the random variable ranges from 0 to 1. This definition can be generalized as follows. Suppose the random variable Y has a beta distribution over the range $a \leq y \leq b$, with shape parameters ν and ω. Then the random variable X defined as:

$$X = \frac{Y-a}{b-a} \qquad (14)$$

has the "standard beta distribution" as described in the help file for Beta (Johnson et al., 1995, p.210).

Bartels (1982) shows that asymptotically, V_{rank} has normal distribution with mean 2 and variance $4/n$, but notes that a slightly better approximation is given by using a variance of $20/(5n+7)$.

To test the null hypothesis (1) when `test="rank.von.Neumann"`, the function `serialCorrelationTest` does the following:

- When the sample size is between 3 and 10, the exact distribution of V_{rank} is used to compute the p-value.
- When the sample size is between 11 and 100, the beta approximation to the distribution of V_{rank} is used to compute the p-value.
- When the sample size is larger than 100, the normal approximation to the distribution of V_{rank} is used to compute the p-value. (This uses the variance $20/(5n+7)$.)

When ties are present in the observations and midranks are used for the tied observations, the distribution of the V_{rank} statistic based on the assumption of no ties is not applicable. If the number of ties is small, however, they may not grossly affect the assumed p-value.

When ties are present, the function `serialCorrelationTest` issues a warning. When the sample size is between 3 and 10, the p-value is computed based on rounding up the computed value of V_{rank} to the nearest possible value that could be observed in the case of no ties.

Computing a Confidence Interval for the Lag-1 Autocorrelation

The function `serialCorrelationTest` computes an approximate $100(1-\alpha)\%$ confidence interval for the lag-1 autocorrelation as follows:

$$[\hat{\rho}_1 - z_{1-\alpha/2}\hat{\sigma}_{\hat{\rho}_1}, \hat{\rho}_1 + z_{1-\alpha/2}\hat{\sigma}_{\hat{\rho}_1}] \qquad (15)$$

where $\hat{\sigma}_{\hat{\rho}_1}$ denotes the estimated standard deviation of the estimated of lag-1 autocorrelation and z_p denotes the p'th quantile of the standard normal distribution.

When `test="AR1.yw"` or `test="rank.von.Neumann"`, the Yule-Walker estimate of lag-1 autocorrelation is used and the variance of the estimated lag-1 autocorrelation is approximately:

$$Var(\hat{\rho}_1) \approx \frac{1}{n}(1 - \rho_1^2) \qquad (16)$$

(Box and Jenkins, 1976, p.34), so

$$\hat{\sigma}_{\hat{\rho}_1} = \sqrt{\frac{1 - \hat{\rho}_1^2}{n}} \qquad (17)$$

When `test="AR1.mle"`, the MLE of the lag-1 autocorrelation is used, and its standard deviation is estimated with the square root of the estimated variance returned by `arima`.

Value

A list of class `"htestEnvStats"` containing the results of the hypothesis test. See the help file for `htestEnvStats.object` for details.

Note

Data collected over time on the same phenomenon are called a time series. A time series is usually modeled as a single realization of a stochastic process; that is, if we could go back in time and repeat the experiment, we would get different results that would vary according to some probabilistic law. The simplest kind of time series is a stationary time series, in which the mean value is constant over time, the variability of the observations is constant over time, etc. That is, the probability distribution associated with each future observation is the same.

A common concern in applying standard statistical tests to time series data is the assumption of independence. Most conventional statistical hypothesis tests assume the observations are independent, but data collected sequentially in time may not satisfy this assumption. For example, high observations may tend to follow high observations (positive serial correlation), or low observations may tend to follow high observations (negative serial correlation). One way to investigate the assumption of independence is to estimate the lag-one serial correlation and test whether it is significantly different from 0.

The null distribution of the serial correlation coefficient may be badly affected by departures from normality in the underlying process (Cox, 1966; Bartels, 1977). It is therefore a good idea to consider using a nonparametric test for randomness if the normality of the underlying process is in doubt (Bartels, 1982). Knoke (1977) showed that under normality, the test based on the rank serial correlation coefficient (and hence the test based on the rank von Neumann ratio statistic) has asymptotic relative efficiency of 0.91 with respect to using the test based on the ordinary serial correlation coefficient against the alternative of first-order autocorrelation.

Bartels (1982) performed an extensive simulation study of the power of the rank von Neumann ratio test relative to the standard von Neumann ratio test (based on the statistic in Equation (9) above) and the runs test (Lehmann, 1975, 313-315). He generated a first-order autoregressive process for sample sizes of 10, 25, and 50, using 6 different parent distributions: normal, Cauchy, contaminated normal, Johnson, Stable, and exponential. Values of lag-1 autocorrelation ranged from -0.8 to 0.8. Bartels (1982) found three important results:

- The rank von Neumann ratio test is far more powerful than the runs test.

- For the normal process, the power of the rank von Neumann ratio test was never less than 89% of the power of the standard von Neumann ratio test.
- For non-normal processes, the rank von Neumann ratio test was often much more powerful than of the standard von Neumann ratio test.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Bartels, R. (1982). The Rank Version of von Neumann's Ratio Test for Randomness. *Journal of the American Statistical Association* **77**(377), 40–46.

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Box, G.E.P., and G.M. Jenkins. (1976). *Time Series Analysis: Forecasting and Control*. Prentice Hall, Englewood Cliffs, NJ, Chapter 2.

Cox, D.R. (1966). The Null Distribution of the First Serial Correlation Coefficient. *Biometrika* **53**, 623–626.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York, pp.69-70;181-192.

Durbin, J., and G.S. Watson. (1950). Testing for Serial Correlation in Least Squares Regression I. *Biometrika* **37**, 409–428.

Durbin, J., and G.S. Watson. (1951). Testing for Serial Correlation in Least Squares Regression II. *Biometrika* **38**, 159–178.

Durbin, J., and G.S. Watson. (1971). Testing for Serial Correlation in Least Squares Regression III. *Biometrika* **58**, 1–19.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, pp.250–253.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York, Chapter 25.

Knoke, J.D. (1975). Testing for Randomness Against Autocorrelation Alternatives: The Parametric Case. *Biometrika* **62**, 571–575.

Knoke, J.D. (1977). Testing for Randomness Against Autocorrelation Alternatives: Alternative Tests. *Biometrika* **64**, 523–529.

Lehmann, E.L. (1975). *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, Oakland, CA, 457pp.

von Neumann, J., R.H. Kent, H.R. Bellinson, and B.I. Hart. (1941). The Mean Square Successive Difference. *Annals of Mathematical Statistics* **12**(2), 153–162.

Wald, A., and J. Wolfowitz. (1943). An Exact Test for Randomness in the Non-Parametric Case Based on Serial Correlation. *Annals of Mathematical Statistics* **14**, 378–388.

See Also

htestEnvStats.object, acf, ar, arima, arima.sim, ts.plot, plot.ts, lag.plot, Hypothesis Tests.

Examples

```
# Generate a purely random normal process, then use serialCorrelationTest
# to test for the presence of correlation.
# (Note: the call to set.seed allows you to reproduce this example.)

set.seed(345)
x <- rnorm(100)

# Look at the data
#-----------------
dev.new()
ts.plot(x)

dev.new()
acf(x)

# Test for serial correlation
#----------------------------
serialCorrelationTest(x)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 rho = 0
#
#Alternative Hypothesis:          True rho is not equal to 0
#
#Test Name:                       Rank von Neumann Test for
#                                 Lag-1 Autocorrelation
#                                 (Beta Approximation)
#
#Estimated Parameter(s):          rho = 0.02773737
#
#Estimation Method:               Yule-Walker
#
#Data:                            x
#
#Sample Size:                     100
#
#Test Statistic:                  RVN = 1.929733
#
#P-value:                         0.7253405
#
#Confidence Interval for:         rho
#
#Confidence Interval Method:      Normal Approximation
#
```

```
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = -0.1681836
#                                 UCL =  0.2236584

# Clean up
#---------
rm(x)
graphics.off()

#==========

# Now use the R function arima.sim to generate an AR(1) process with a
# lag-1 autocorrelation of 0.8, then test for autocorrelation.

set.seed(432)
y <- arima.sim(model = list(ar = 0.8), n = 100)

# Look at the data
#-----------------
dev.new()
ts.plot(y)

dev.new()
acf(y)

# Test for serial correlation
#----------------------------
serialCorrelationTest(y)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 rho = 0
#
#Alternative Hypothesis:          True rho is not equal to 0
#
#Test Name:                       Rank von Neumann Test for
#                                 Lag-1 Autocorrelation
#                                 (Beta Approximation)
#
#Estimated Parameter(s):          rho = 0.835214
#
#Estimation Method:               Yule-Walker
#
#Data:                            y
#
#Sample Size:                     100
#
#Test Statistic:                  RVN = 0.3743174
#
```

```
#P-value:                          0
#
#Confidence Interval for:          rho
#
#Confidence Interval Method:       Normal Approximation
#
#Confidence Interval Type:         two-sided
#
#Confidence Level:                 95%
#
#Confidence Interval:              LCL = 0.7274307
#                                  UCL = 0.9429973

#----------

# Clean up
#---------
rm(y)
graphics.off()

#==========

# The data frame Air.df contains information on ozone (ppb^1/3),
# radiation (langleys), temperature (degrees F), and wind speed (mph)
# for 153 consecutive days between May 1 and September 30, 1973.
# First test for serial correlation in (the cube root of) ozone.
# Note that we must use the test based on the MLE because the time series
# contains missing values.  Serial correlation appears to be present.
# Next fit a linear model that includes the predictor variables temperature,
# radiation, and wind speed, and test for the presence of serial correlation
# in the residuals.  There is no evidence of serial correlation.

# Look at the data
#-----------------

Air.df
#              ozone radiation temperature wind
#05/01/1973 3.448217       190          67  7.4
#05/02/1973 3.301927       118          72  8.0
#05/03/1973 2.289428       149          74 12.6
#05/04/1973 2.620741       313          62 11.5
#05/05/1973       NA        NA          56 14.3
#...
#09/27/1973       NA       145          77 13.2
#09/28/1973 2.410142       191          75 14.3
#09/29/1973 2.620741       131          76  8.0
#09/30/1973 2.714418       223          68 11.5

#----------

# Test for serial correlation
#----------------------------
```

```
with(Air.df,
  serialCorrelationTest(ozone, test = "AR1.mle"))

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 rho = 0
#
#Alternative Hypothesis:          True rho is not equal to 0
#
#Test Name:                       z-Test for
#                                 Lag-1 Autocorrelation
#                                 (Wald Test Based on MLE)
#
#Estimated Parameter(s):          rho = 0.5641616
#
#Estimation Method:               Maximum Likelihood
#
#Data:                            ozone
#
#Sample Size:                     153
#
#Number NA/NaN/Inf's:             37
#
#Test Statistic:                  z = 7.586952
#
#P-value:                         3.28626e-14
#
#Confidence Interval for:         rho
#
#Confidence Interval Method:      Normal Approximation
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = 0.4184197
#                                 UCL = 0.7099034

#----------

# Next fit a linear model that includes the predictor variables temperature,
# radiation, and wind speed, and test for the presence of serial correlation
# in the residuals.  Note setting the argument na.action = na.exclude in the
# call to lm to correctly deal with missing values.
#-------------------------------------------------------------------------

lm.ozone <- lm(ozone ~ radiation + temperature + wind +
  I(temperature^2) + I(wind^2),
  data = Air.df, na.action = na.exclude)


# Now test for serial correlation in the residuals.
```

```
#-----------------------------------------------

serialCorrelationTest(lm.ozone, test = "AR1.mle")

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 rho = 0
#
#Alternative Hypothesis:          True rho is not equal to 0
#
#Test Name:                       z-Test for
#                                 Lag-1 Autocorrelation
#                                 (Wald Test Based on MLE)
#
#Estimated Parameter(s):          rho = 0.1298024
#
#Estimation Method:               Maximum Likelihood
#
#Data:                            Residuals
#
#Data Source:                     lm.ozone
#
#Sample Size:                     153
#
#Number NA/NaN/Inf's:             42
#
#Test Statistic:                  z = 1.285963
#
#P-value:                         0.1984559
#
#Confidence Interval for:         rho
#
#Confidence Interval Method:      Normal Approximation
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Interval:             LCL = -0.06803223
#                                 UCL =  0.32763704

# Clean up
#---------
rm(lm.ozone)
```

---

signTest *One-Sample or Paired-Sample Sign Test on a Median*

Description

Estimate the median, test the null hypothesis that the median is equal to a user-specified value based on the sign test, and create a confidence interval for the median.

Usage

```
signTest(x, y = NULL, alternative = "two.sided", mu = 0, paired = FALSE,
  ci.method = "interpolate", approx.conf.level = 0.95,
  min.coverage = TRUE, lb = -Inf, ub = Inf)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
y	optional numeric vector of observations that are paired with the observations in x. The length of y must be the same as the length of x. This argument is ignored if paired=FALSE, and must be supplied if paired=TRUE. The default value is y=NULL. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
mu	numeric scalar indicating the hypothesized value of the median. The default value is mu=0.
paired	logical scalar indicating whether to perform a paired or one-sample sign test. The possible values are paired=FALSE (the default; indicates a one-sample sign test) and paired=TRUE.
ci.method	character string indicating the method to use to construct the confidence interval. The possible values are "interpolate" (the default), "exact", and "normal.approx". See the help file for eqnpar for more information on these methods.
approx.conf.level	a scalar between 0 and 1 indicating the desired confidence level of the confidence interval for the population median. The default value is appox.conf.level=0.95. The true confidence level usually will not be exactly equal to approx.conf.level. See the help file for eqnpar for more information
min.coverage	for the case when ci.method="exact", a logical scalar indicating whether the confidence interval for the median should have a minimum coverage at least as great as the value of the argument approx.conf.level. The default value is min.coverage=TRUE. This argument is ignored if ci.method is not equal to "exact".
lb, ub	scalars indicating lower and upper bounds on the distribution. By default, lb=-Inf and ub=Inf. If you are constructing a confidence interval for the median from a distribution that you know has a lower bound other than -Inf (e.g., 0), set lb to this value. Similarly, if you know the distribution has an upper bound other than Inf, set ub to this value.

Details

One-Sample Case (`paired=FALSE`)
Let $\underline{x} = x_1, x_2, \ldots, x_n$ be a vector of n independent observations from one or more distributions that all have the same median μ.

Consider the test of the null hypothesis:

$$H_0 : \mu = \mu_0 \qquad (1)$$

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`)

$$H_a : \mu > \mu_0 \qquad (2)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : \mu < \mu_0 \qquad (3)$$

and the two-sided alternative (`alternative="two.sided"`)

$$H_a : \mu \neq \mu_0 \qquad (4)$$

To perform the test of the null hypothesis (1) versus any of the three alternatives (2)-(4), the sign test uses the test statistic T which is simply the number of observations that are greater than μ_0 (Conover, 1980, p. 122; van Belle et al., 2004, p. 256; Hollander and Wolfe, 1999, p. 60; Lehmann, 1975, p. 120; Sheskin, 2011; Zar, 2010, p. 537). Under the null hypothesis, the distribution of T is a binomial random variable with parameters `size=`n and `prob=0.5`. Usually, however, cases for which the observations are equal to μ_0 are discarded, so the distribution of T is taken to be binomial with parameters `size=`r and `prob=0.5`, where r denotes the number of observations not equal to μ_0. The sign test only requires that the observations are independent and that they all come from one or more distributions (not necessarily the same ones) that all have the same population median.

For a two-sided alternative hypothesis (Equation (4)), the p-value is computed as:

$$p = Pr(X_{r,0.5} \leq r - m) + Pr(X_{r,0.5} > m) \qquad (5)$$

where $X_{r,p}$ denotes a binomial random variable with parameters `size=`r and `prob=`p, and m is defined by:

$$m = max(T, r - T) \qquad (6)$$

For a one-sided lower alternative hypothesis (Equation (3)), the p-value is computed as:

$$p = Pr(X_{m,0.5} \leq T) \qquad (7)$$

and for a one-sided upper alternative hypothesis (Equation (2)), the p-value is computed as:

$$p = Pr(X_{m,0.5} \geq T) \qquad (8)$$

It is obvious that the sign test is simply a special case of the binomial test with `p=0.5`.

Computing Confidence Intervals
Based on the relationship between hypothesis tests and confidence intervals, we can construct a confidence interval for the population median based on the sign test (e.g., Hollander and Wolfe, 1999, p. 72; Lehmann, 1975, p. 182). It turns out that this is equivalent to using the formulas for a

nonparametric confidence intervals for the 0.5 quantile (see eqnpar).

Paired-Sample Case (paired=TRUE)
When the argument paired=TRUE, the arguments x and y are assumed to have the same length, and the n differences $d_i = x_i - y_i, \quad i = 1, 2, \ldots, n$ are assumed to be independent observations from distributions with the same median μ. The sign test can then be applied to the differences.

Value

A list of class "htestEnvStats" containing the results of the hypothesis test. See the help file for htestEnvStats.object for details.

Note

A frequent question in environmental statistics is "Is the concentration of chemical X greater than Y units?". For example, in groundwater assessment (compliance) monitoring at hazardous and solid waste sites, the concentration of a chemical in the groundwater at a downgradient well must be compared to a groundwater protection standard (GWPS). If the concentration is "above" the GWPS, then the site enters corrective action monitoring. As another example, soil screening at a Superfund site involves comparing the concentration of a chemical in the soil with a pre-determined soil screening level (SSL). If the concentration is "above" the SSL, then further investigation and possible remedial action is required. Determining what it means for the chemical concentration to be "above" a GWPS or an SSL is a policy decision: the average of the distribution of the chemical concentration must be above the GWPS or SSL, or the median must be above the GWPS or SSL, or the 95th percentile must be above the GWPS or SSL, or something else. Often, the first interpretation is used.

Hypothesis tests you can use to perform tests of location include: Student's t-test, Fisher's randomization test, the Wilcoxon signed rank test, Chen's modified t-test, the sign test, and a test based on a bootstrap confidence interval. For a discussion comparing the performance of these tests, see Millard and Neerchal (2001, pp.408-409).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York, p.122

Hollander, M., and D.A. Wolfe. (1999). *Nonparametric Statistical Methods*. Second Edition. John Wiley and Sons, New York, p.60.

Lehmann, E.L. (1975). *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, Oakland, CA, p.120.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.404–406.

Sheskin, D.J. (2011). *Handbook of Parametric and Nonparametric Statistical Procedures* Fifth Edition. CRC Press, Boca Raton, FL.

van Belle, G., L.D. Fisher, Heagerty, P.J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences* 2nd Edition. John Wiley & Sons, New York.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ,

See Also

wilcox.test, Hypothesis Tests, eqnpar, htestEnvStats.object.

Examples

```
# Generate 10 observations from a lognormal distribution with parameters
# meanlog=2 and sdlog=1.  The median of this distribution is e^2 (about 7.4).
# Test the null hypothesis that the true median is equal to 5 against the
# alternative that the true mean is greater than 5.
# (Note: the call to set.seed allows you to reproduce this example).

set.seed(23)
dat <- rlnorm(10, meanlog = 2, sdlog = 1)
signTest(dat, mu = 5, lb = 0)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 median = 5
#
#Alternative Hypothesis:          True median is not equal to 5
#
#Test Name:                       Sign test
#
#Estimated Parameter(s):          median = 19.21717
#
#Data:                            dat
#
#Test Statistic:                  # Obs > median = 9
#
#P-value:                         0.02148438
#
#Confidence Interval for:         median
#
#Confidence Interval Method:      interpolate (Nyblom, 1992)
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                95%
#
#Confidence Limit Rank(s):        2 3 9 8
#
#Confidence Interval:             LCL =  7.000846
#                                 UCL = 26.937725

#----------
```

```
# Redo the above example using an exact confidence interval
# and specifying min.coverage=FALSE
#-----------------------------------------------------------

signTest(dat, mu = 5, ci.method = "exact", min.coverage = FALSE, lb = 0)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 median = 5
#
#Alternative Hypothesis:          True median is not equal to 5
#
#Test Name:                       Sign test
#
#Estimated Parameter(s):          median = 19.21717
#
#Data:                            dat
#
#Test Statistic:                  # Obs > median = 9
#
#P-value:                         0.02148438
#
#Confidence Interval for:         median
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        two-sided
#
#Confidence Level:                94.43359%
#
#Confidence Limit Rank(s):        1 8
#
#Confidence Interval:             LCL =  4.784196
#                                 UCL = 22.364849

#----------

# Clean up
#---------
rm(dat)

#==========

# The guidance document "Supplemental Guidance to RAGS: Calculating the
# Concentration Term" (USEPA, 1992d) contains an example of 15 observations
# of chromium concentrations (mg/kg) which are assumed to come from a
# lognormal distribution.  These data are stored in the vector
# EPA.92d.chromium.vec.  Here, we will use the sign test to test the null
# hypothesis that the median chromium concentration is less than or equal to
# 100 mg/kg vs. the alternative that it is greater than 100 mg/kg.  The
# estimated median is 110 mg/kg.  There are 8 out of 15 observations greater
# than 100 mg/kg, the p-value is equal to 0.5, and the lower 95% confidence
```

```
# limit is 40.53 mg/kg.

summaryStats(EPA.92d.chromium.vec)

#                       N     Mean      SD Median Min  Max
#EPA.92d.chromium.vec 15 175.4667 318.544    110  10 1300

#----------

signTest(EPA.92d.chromium.vec, mu = 100, alternative = "greater")

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 median = 100
#
#Alternative Hypothesis:          True median is greater than 100
#
#Test Name:                       Sign test
#
#Estimated Parameter(s):          median = 110
#
#Data:                            EPA.92d.chromium.vec
#
#Test Statistic:                  # Obs > median = 8
#
#P-value:                         0.5
#
#Confidence Interval for:         median
#
#Confidence Interval Method:      interpolate (Nyblom, 1992)
#
#Confidence Interval Type:        lower
#
#Confidence Level:                95%
#
#Confidence Limit Rank(s):        4 5 NA NA
#
#Confidence Interval:             LCL =  40.53074
#                                 UCL = Inf

#----------

# Redo the above example using the exact confidence interval and
# setting min.coverage=FALSE
#---------------------------------------------------------------

signTest(EPA.92d.chromium.vec, mu = 100, alternative = "greater",
  ci.method = "exact", min.coverage = FALSE)

#Results of Hypothesis Test
#--------------------------
#
```

```
#Null Hypothesis:                 median = 100
#
#Alternative Hypothesis:          True median is greater than 100
#
#Test Name:                       Sign test
#
#Estimated Parameter(s):          median = 110
#
#Data:                            EPA.92d.chromium.vec
#
#Test Statistic:                  # Obs > median = 8
#
#P-value:                         0.5
#
#Confidence Interval for:         median
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        lower
#
#Confidence Level:                94.07654%
#
#Confidence Limit Rank(s):        5
#
#Confidence Interval:             LCL =  41
#                                 UCL = Inf

#----------

# Redo the above example using the exact confidence interval and
# setting min.coverage=TRUE
#---------------------------------------------------------------

signTest(EPA.92d.chromium.vec, mu = 100, alternative = "greater",
  ci.method = "exact")

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 median = 100
#
#Alternative Hypothesis:          True median is greater than 100
#
#Test Name:                       Sign test
#
#Estimated Parameter(s):          median = 110
#
#Data:                            EPA.92d.chromium.vec
#
#Test Statistic:                  # Obs > median = 8
#
#P-value:                         0.5
#
```

```
#Confidence Interval for:         median
#
#Confidence Interval Method:      exact
#
#Confidence Interval Type:        lower
#
#Confidence Level:                98.24219%
#
#Confidence Limit Rank(s):        4
#
#Confidence Interval:             LCL =  36
#                                 UCL = Inf
```

`simulateMvMatrix` *Simulate a Multivariate Matrix Based on a Specified Rank Correlation Mat*

Description

Simulate a multivariate matrix of random numbers from specified theoretical probability distributions and/or empirical probability distributions based on a specified rank correlation matrix, using either Latin Hypercube sampling or simple random sampling.

Usage

```
simulateMvMatrix(n, distributions = c(Var.1 = "norm", Var.2 = "norm"),
 param.list = list(Var.1 = list(mean = 0, sd = 1), Var.2 = list(mean = 0, sd = 1)),
  cor.mat = diag(length(distributions)), sample.method = "SRS", seed = NULL,
  left.tail.cutoff = ifelse(is.finite(supp.min), 0, .Machine$double.eps),
  right.tail.cutoff = ifelse(is.finite(supp.max), 0, .Machine$double.eps),
  tol.1 = .Machine$double.eps, tol.symmetry = .Machine$double.eps,
  tol.recip.cond.num = .Machine$double.eps, max.iter = 10)
```

Arguments

`n`	a positive integer indicating the number of random vectors (i.e., the number of rows of the matrix) to generate.
`distributions`	a character vector of length k denoting the distribution abbreviations for each of the k distributions. If there is a `names` attribute associated with this character vector, these names will be the column names of the resulting matrix. The default value of `distributions` is `c(Var.1="norm", Var.2="norm")`, indicating that $k = 2$, both distributions are the normal distribution, and the column names of the resulting $n \times k$ matrix will be `"Var.1"` and `"Var.2"`. See the help file for `Distribution.df` for a list of possible distribution abbreviations. Alternatively, the character string `"emp"` may be used to denote sampling from an empirical distribution based on a set of observations. The vector containing the observations is specified in the argument `param.list`.

`param.list` a list containing k lists that specify the values for the parameters of the k distributions. If `param.list` has a `names` attribute (not necessary), the `names` attribute should be exactly the same as the `names` attribute of the argument `distributions`. The default value of `param.list` is `list(Var.1=list(mean=0, sd=1), Var.2=list(mean=0, sd=1))`. See the help file for `Distribution.df` for the names and possible values of the parameters associated with each distribution.

Alternatively, if you specify an empirical distribution for the j'th distribution by setting the j'th element of `distribution` to `"emp"`, then the j'th component of `param.list` must be a list of the form `list(obs=`*name*`)`, where *name* denotes the name of the vector containing the observations to use for the empirical distribution. In this case, you may also supply arguments to the `qemp` function through the j'th component of `param.list`. For example, you may set this component to `list(obs=`*name*`, discrete=T)` to specify an empirical distribution based on a discrete random variable.

`cor.mat` a $k \times k$ matrix specifying the rank correlations between the k distributions. This argument must be a positive definite symmetric matrix, with all 1's on the diagonal. All elements on the off-diagonal must be between -1 and 1. The default value is the $k \times k$ identity matrix, specifying no rank correlation between any of the variables.

`sample.method` a character vector of length 1 or k indicating, for each distribution, whether to use Latin Hypercube sampling or simple random sampling. If `sample.method` is of length 1, it is replicated to length k. Each element of `sample.method` must be the character string `"LHS"` (Latin Hypercube sampling) or `"SRS"` (simple random sampling), or an abbreviation of one of these strings. The default value is `"SRS"`, indicating simple random sampling for each distribution. Note that by specifying `sample.method` as a vector of length k, you may use different sampling methods for different distributions.

`seed` integer to supply to the R function `set.seed`. The default value is `seed=NULL`, in which case the random seed is not set but instead based on the current value of `.Random.seed`.

`left.tail.cutoff`

a numeric vector of length k indicating, for each distribution, what proportion of the left-tail of the probability distribution to omit for Latin Hypercube sampling. All elements of `left.tail.cutoff` must be between 0 and 1. For densities with a finite support minimum (e.g., Lognormal or Empirical) the default value is `left.tail.cutoff=0`; for densities with a support minimum of $-\infty$, the default value is `left.tail.cutoff=.Machine$double.eps`. The j'th element of this argument is ignored if the j'th element of `sample.method` is equal to `"SRS"`.

`right.tail.cutoff`

a numeric vector of length k indicating, for each distribution, what proportion of the right-tail of the probability distribution to omit for Latin Hypercube sampling. All elements of `right.tail.cutoff` must be between 0 and 1. For densities with a finite support maximum (e.g., Beta or Empirical) the default value is `right.tail.cutoff=0`; for densities with a support maximum of ∞, the default value is `right.tail.cutoff=.Machine$double.eps`. The j'th ele-

ment of this argument is ignored if the j'th element of `sample.method` is equal to `"SRS"`.

`tol.1` a positive numeric scalar indicating the allowable absolute deviation from 1 for the diagonal elements of `cor.mat`. The default value is `.Machine$double.eps`.

`tol.symmetry` a positive numeric scalar indicating the allowable absolute deviation from 0 for the difference between symmetric elements of `cor.mat` (e.g., `abs(cor.mat[3,2]-cor.mat[2,3])`. The default value is `.Machine$double.eps`.

`tol.recip.cond.num`
a positive numeric scalar indicating the allowable minimum value of the reciprocal of the condition number for `cor.mat`. The condition number is defined to be the largest eigen value divided by the smallest eigen value. The reciprocal of the condition number is some number between 0 and 1. This value must be sufficiently large for `cor.mat` to be of full rank (i.e., to not be singular). The default value of `tol.recip.cond.num` is `.Machine$double.eps`.

`max.iter` a positive integer indicating the maximum number of iterations to use to produce the R matrix in the algorithm to create the output matrix. The sample correlation matrix of R must be positive definite. The number of iterations will rarely be more than 2 for moderate to large sample sizes (e.g., $n > 2k$). The default value is `max.iter=10`. See the DETAILS section below for more information on the R matrix.

Details

Motivation

In risk assessment and Monte Carlo simulation, the outcome variable of interest, say Y, is usually some function of one or more other random variables:

$$Y = h(\underline{X}) = h(X_1, X_2, \ldots, X_k) \qquad (1)$$

For example, Y may be the incremental lifetime cancer risk due to ingestion of soil contaminated with benzene (Thompson et al., 1992; Hamed and Bedient, 1997). In this case the random vector $\underline{X}$ may represent observations from several kinds of distributions that characterize exposure and dose-response, such as benzene concentration in the soil, soil ingestion rate, average body weight, the cancer potency factor for benzene, etc. These distributions may or may not be assumed to be independent of one another (Smith et al., 1992; Bukowski et al., 1995). Often, input variables in a Monte Carlo simulation are in fact known to be correlated, such as body weight and dermal area.

Characterizing the joint distribution of a random vector $\underline{X}$, where different elements of $\underline{X}$ come from different distributions, is usually mathematically complex or impossible unless the elements (random variables) of $\underline{X}$ are independent. Iman and Conover (1982) present an algorithm for creating a set of n multivariate observations with a rank correlation matrix that is approximately equal to a specified rank correlation matrix. This method allows for different probability distributions for each element of the multivariate vector. The details of this algorithm are as follows.

Algorithm

1. Specify n, the desired number of random vectors (i.e., number of rows of the $n \times k$ output matrix). This is specified by the argument `n` for the function `simulateMvMatrix`.

2. Create C, the desired $k \times k$ correlation matrix. This is specified by the argument `cor.mat`.
3. Compute P, where P is a lower triangular $k \times k$ matrix and

$$PP^{'} = C \qquad (2)$$

where $P^{'}$ denotes the transpose of P. The function `simulateMvMatrix` uses the Cholesky decomposition to compute P (see the R help file for `chol`).

4. Create R, an $n \times k$ matrix, whose columns represent k independent permutations of van der Waerden scores. That is, each column of R is a random permutation of the scores

$$\Phi^{-1}(\frac{i}{n+1}),\ i = 1, 2, \ldots, n \qquad (3)$$

where Φ denotes the cumulative distribution function of the standard normal distribution.

5. Compute T, the $k \times k$ Pearson sample correlation matrix of R. Make sure T is positive definite; if it is not, then repeat step 4.
6. Compute Q, where Q is a lower triangular $k \times k$ matrix and

$$QQ^{'} = T \qquad (4)$$

The function `simulateMvMatrix` uses the Cholesky decomposition to compute Q (see the R help file for `chol`).

7. Compute the lower triangular $k \times k$ matrix S, where

$$S = PQ^{-1} \qquad (5)$$

8. Compute the matrix R^*, where

$$R^* = RS^{'} \qquad (6)$$

9. Generate an $n \times k$ matrix of random numbers $\underline{X}$, where each column of $\underline{X}$ comes from the distribution specified by the arguments `distributions` and `param.list`. Generate each column of random numbers independently of the other columns. If the j'th element of `sample.method` equals `"SRS"`, use simple random sampling to generate the random numbers for the j'th column of $\underline{X}$. If the j'th element of `sample.method` equals `"LHS"`, use Latin Hypercube sampling to generate the random numbers for the j'th column of $\underline{X}$. At this stage in the algorithm, the function `simulateMvMatrix` calls the function `simulateVector` to create each column of $\underline{X}$.
10. Order the observations within each column of $\underline{X}$ so that the order of the ranks within each column of $\underline{X}$ matches the order of the ranks within each column of R^*. This way, $\underline{X}$ and R^* have exactly the same sample rank correlation matrix.

Explanation

Iman and Conover (1982) present two algorithms for computing an $n \times k$ output matrix with a specified rank correlation. The algorithm presented above is the second, more complicated one. In order to explain the reasoning behind this algorithm, we need to explain the simple algorithm first.

Simple Algorithm

Let R_i denote the i'th row vector of the matrix R, the matrix of scores. This row vector has a

population correlation matrix of I, where I denotes the $k \times k$ identity matrix. Thus, the $1 \times k$ vector $R_i P^{'}$ has a population correlation matrix equal to C. Therefore, if we define R^* by

$$R^* = RP^{'} \qquad (7)$$

each row of R^* has the same multivariate distribution with population correlation matrix C. The rank correlation matrix of R^* should therefore be close to C. Ordering the columns of $\underline{X}$ as described in Step 10 above will yield a matrix of observations with the specified distributions and the exact same rank correlation matrix as the rank correlation matrix of R^*.

Iman and Conover (1982) use van der Waerden scores instead of raw ranks to create R because van der Waerden scores yield more "natural-looking" pairwise scatterplots.

If the Pearson sample correlation matrix of R, denoted T in Step 5 above, is exactly equal to the true population correlation matrix I, then the sample correlation matrix of R^* is exactly equal to C, and the rank correlation matrix of R^* is approximately equal to C. The Pearson sample correlation matrix of R, however, is an estimate of the true population correlation matrix I, and is therefore "bouncing around" I. Likewise, the Pearson sample correlation matrix of R^* is an estimate of the true population correlation matrix C, and is therefore bouncing around C. Using this simple algorithm, the Pearson sample correlation matrix of R^*, as R^* is defined in Equation (7) above, may not be "close" enough to the desired rank correlation matrix C, and thus the rank correlation of R^* will not be close enough to C. Iman and Conover (1982), therefore present a more complicated algorithm.

More Complicated Algorithm
To get around the problem mentioned above, Iman and Conover (1982) find a $k \times k$ lower triangular matrix S such that the matrix R^* as defined in Equation (6) above has a correlation matrix exactly equal to C. The formula for S is given in Steps 6 and 7 of the algorithm above.

Iman and Conover (1982, p.330) note that even if the desired rank correlation matrix C is in fact the identity matrix I, this method of generating the matrix will produce a matrix with an associated rank correlation that more closely resembles I than you would get by simply generating random numbers within each column of $\underline{X}$.

Value

A numeric matrix of dimension $n \times k$ of random numbers, where the j'th column of numbers comes from the distribution specified by the j'th elements of the arguments `distributions` and `param.list`, and the rank correlation of this matrix is approximately equal to the argument `cor.mat`. The value of n is determined by the argument `n`, and the value of k is determined by the length of the argument `distributions`.

Note

Monte Carlo simulation and risk assessment often involve looking at the distribution or characteristics of the distribution of some outcome variable that depends upon several input variables (see Equation (1) above). Usually these input variables can be considered random variables. An important part of both sensitivity analysis and uncertainty analysis involves looking at how the distribution of the outcome variable changes with changing assumptions on the input variables. One important assumption is the correlation between the input random variables.

Often, the input random variables are assumed to be independent when in fact they are know to be correlated (Smith et al., 1992; Bukowski et al., 1995). It is therefore important to assess the effect of the assumption of independence on the distribution of the outcome variable. One way to assess the effect of this assumption is to run the Monte Carlo simulation assuming independence and then also run it assuming certain forms of correlations among the input variables.

Iman and Davenport (1982) present a series of scatterplots showing "typical" scatterplots with various distributions on the x- and y-axes and various assumed rank correlations. These plots are meant to aid in developing reasonable estimates of rank correlation between input variables. These plots can easily be produced using the `simulateMvMatrix` and `plot` functions.

Author(s)

Steven P. Millard (`<EnvStats@ProbStatInfo.com>`)

References

Bukowski, J., L. Korn, and D. Wartenberg. (1995). Correlated Inputs in Quantitative Risk Assessment: The Effects of Distributional Shape. *Risk Analysis* **15**(2), 215–219.

Hamed, M., and P.B. Bedient. (1997). On the Effect of Probability Distributions of Input Variables in Public Health Risk Assessment. *Risk Analysis* **17**(1), 97–105.

Iman, R.L., and W.J. Conover. (1980). Small Sample Sensitivity Analysis Techniques for Computer Models, With an Application to Risk Assessment (with Comments). *Communications in Statistics–Volume A, Theory and Methods*, **9**(17), 1749–1874.

Iman, R.L., and W.J. Conover. (1982). A Distribution-Free Approach to Inducing Rank Correlation Among Input Variables. *Communications in Statistics–Volume B, Simulation and Computation*, **11**(3), 311–334.

Iman, R.L., and J.M. Davenport. (1982). Rank Correlation Plots For Use With Correlated Input Variables. *Communications in Statistics–Volume B, Simulation and Computation*, **11**(3), 335–360.

Iman, R.L., and J.C. Helton. (1988). An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models. *Risk Analysis* **8**(1), 71–90.

Iman, R.L. and J.C. Helton. (1991). The Repeatability of Uncertainty and Sensitivity Analyses for Complex Probabilistic Risk Assessments. *Risk Analysis* **11**(4), 591–606.

McKay, M.D., R.J. Beckman., and W.J. Conover. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code. *Technometrics* **21**(2), 239–245.

Millard, S.P. (2013). *EnvStats: an R Package for Environmental Statistics*. Springer, New York. https://link.springer.com/book/10.1007/978-1-4614-8456-1.

Smith, A.E., P.B. Ryan, and J.S. Evans. (1992). The Effect of Neglecting Correlations When Propagating Uncertainty and Estimating the Population Distribution of Risk. *Risk Analysis* **12**(4), 467–474.

Thompson, K.M., D.E. Burmaster, and E.A.C. Crouch. (1992). Monte Carlo Techniques for Quantitative Uncertainty Analysis in Public Health Risk Assessments. *Risk Analysis* **12**(1), 53–63.

Vose, D. (2008). *Risk Analysis: A Quantitative Guide*. Third Edition. John Wiley & Sons, West Sussex, UK, 752 pp.

See Also

Probability Distributions and Random Numbers, Empirical, `simulateVector`, `cor`, `set.seed`.

Examples

```
# Generate 5 observations from a standard bivariate normal distribution
# with a rank correlation matrix (approximately) equal to the 2 x 2
# identity matrix, using simple random sampling for each
# marginal distribution.

simulateMvMatrix(5, seed = 47)
#           Var.1       Var.2
#[1,]  0.01513086  0.03960243
#[2,] -1.08573747  0.09147291
#[3,] -0.98548216  0.49382018
#[4,] -0.25204590 -0.92245624
#[5,] -1.46575030 -1.82822917

#==========

# Look at the observed rank correlation matrix for 100 observations
# from a standard bivariate normal distribution with a rank correlation matrix
# (approximately) equal to the 2 x 2 identity matrix. Compare this observed
# rank correlation matrix with the observed rank correlation matrix based on
# generating two independent sets of standard normal random numbers.
# Note that the cross-correlation is closer to 0 for the matrix created with
# simulateMvMatrix.

cor(simulateMvMatrix(100, seed = 47), method = "spearman")
#             Var.1        Var.2
#Var.1  1.000000000 -0.005976598
#Var.2 -0.005976598  1.000000000

cor(matrix(simulateVector(200, seed = 47), 100 , 2), method = "spearman")
#            [,1]        [,2]
#[1,]  1.00000000 -0.05374137
#[2,] -0.05374137  1.00000000

#==========

# Generate 1000 observations from a bivariate distribution, where the first
# distribution is a normal distribution with parameters mean=10 and sd=2,
# the second distribution is a lognormal distribution with parameters
# mean=10 and cv=1, and the desired rank correlation between the two
# distributions is 0.8.  Look at the observed rank correlation matrix, and
# plot the results.

mat <- simulateMvMatrix(1000,
  distributions = c(N.10.2 = "norm", LN.10.1 = "lnormAlt"),
  param.list = list(N.10.2  = list(mean=10, sd=2),
                    LN.10.1 = list(mean=10, cv=1)),
  cor.mat = matrix(c(1, .8, .8, 1), 2, 2), seed = 47)
```

```
round(cor(mat, method = "spearman"), 2)
#        N.10.2 LN.10.1
#N.10.2    1.00    0.78
#LN.10.1   0.78    1.00

dev.new()
plot(mat, xlab = "Observations from N(10, 2)",
  ylab = "Observations from LN(mean=10, cv=1)",
  main = "Lognormal vs. Normal Deviates with Rank Correlation 0.8")

#----------

# Repeat the last example, but use Latin Hypercube sampling for both
# distributions. Note the wider range on the y-axis.

mat.LHS <- simulateMvMatrix(1000,
  distributions = c(N.10.2 = "norm", LN.10.1 = "lnormAlt"),
  param.list = list(N.10.2  = list(mean=10, sd=2),
                    LN.10.1 = list(mean=10, cv=1)),
  cor.mat = matrix(c(1, .8, .8, 1), 2, 2),
  sample.method = "LHS", seed = 298)

round(cor(mat.LHS, method = "spearman"), 2)
#        N.10.2 LN.10.1
#N.10.2    1.00    0.79
#LN.10.1   0.79    1.00

dev.new()
plot(mat.LHS, xlab = "Observations from N(10, 2)",
  ylab = "Observations from LN(mean=10, cv=1)",
  main = paste("Lognormal vs. Normal Deviates with Rank Correlation 0.8",
    "(Latin Hypercube Sampling)", sep = "\n"))

#==========

# Generate 1000 observations from a multivariate distribution, where the
# first distribution is a normal distribution with parameters
# mean=10 and sd=2, the second distribution is a lognormal distribution
# with parameters mean=10 and cv=1, the third distribution is a beta
# distribution with parameters shape1=2 and shape2=3, and the fourth
# distribution is an empirical distribution of 100 observations that
# we'll generate from a Pareto distribution with parameters
# location=10 and shape=2. Set the desired rank correlation matrix to:

cor.mat <- matrix(c(1, .8, 0, .5, .8, 1, 0, .7,
  0, 0, 1, .2, .5, .7, .2, 1), 4, 4)

cor.mat
#     [,1] [,2] [,3] [,4]
#[1,]  1.0  0.8  0.0  0.5
#[2,]  0.8  1.0  0.0  0.7
#[3,]  0.0  0.0  1.0  0.2
```

```
#[4,]  0.5  0.7  0.2  1.0

# Use Latin Hypercube sampling for each variable, look at the observed
# rank correlation matrix, and plot the results.

pareto.rns <- simulateVector(100, "pareto",
  list(location = 10, shape = 2), sample.method = "LHS",
  seed = 56)

mat <- simulateMvMatrix(1000,
  distributions = c(Normal = "norm", Lognormal = "lnormAlt",
    Beta = "beta", Empirical = "emp"),
  param.list = list(Normal = list(mean=10, sd=2),
                    Lognormal = list(mean=10, cv=1),
                    Beta = list(shape1 = 2, shape2 = 3),
                    Empirical = list(obs = pareto.rns)),
  cor.mat = cor.mat, seed = 47, sample.method = "LHS")

round(cor(mat, method = "spearman"), 2)
#          Normal Lognormal  Beta Empirical
#Normal      1.00      0.78 -0.01      0.47
#Lognormal   0.78      1.00 -0.01      0.67
#Beta       -0.01     -0.01  1.00      0.19
#Empirical   0.47      0.67  0.19      1.00

dev.new()
pairs(mat)

#==========

# Clean up
#---------
rm(mat, mat.LHS, pareto.rns)
graphics.off()
```

`simulateVector` *Simulate a Vector of Random Numbers From a Specified Theoretical or Empirical Probability Distribution*

Description

Simulate a vector of random numbers from a specified theoretical probability distribution or empirical probability distribution, using either Latin Hypercube sampling or simple random sampling.

Usage

```
simulateVector(n, distribution = "norm", param.list = list(mean = 0, sd = 1),
  sample.method = "SRS", seed = NULL, sorted = FALSE,
  left.tail.cutoff = ifelse(is.finite(supp.min), 0, .Machine$double.eps),
  right.tail.cutoff = ifelse(is.finite(supp.max), 0, .Machine$double.eps))
```

Arguments

n	a positive integer indicating the number of random numbers to generate.
distribution	a character string denoting the distribution abbreviation. The default value is distribution="norm". See the help file for Distribution.df for a list of possible distribution abbreviations. Alternatively, the character string "emp" may be used to denote sampling from an empirical distribution based on a set of observations. The vector containing the observations is specified in the argument param.list.
param.list	a list with values for the parameters of the distribution. The default value is param.list=list(mean=0, sd=1). See the help file for Distribution.df for the names and possible values of the parameters associated with each distribution. Alternatively, if you specify an empirical distribution by setting distribution="emp", then param.list must be a list of the form list(obs=*name*), where *name* denotes the name of the vector containing the observations to use for the empirical distribution. In this case, you may also supply arguments to the qemp function through param.list. For example, you may set param.list=list(obs=*name*, discrete=T) to specify an empirical distribution based on a discrete random variable.
sample.method	a character string indicating whether to use simple random sampling (sample.method="SRS", the default) or Latin Hypercube sampling (sample.method="LHS").
seed	integer to supply to the R function set.seed. The default value is seed=NULL, in which case the random seed is not set but instead based on the current value of .Random.seed.
sorted	logical scalar indicating whether to return the random numbers in sorted (ascending) order. The default value is sorted=FALSE.
left.tail.cutoff	a scalar between 0 and 1 indicating what proportion of the left-tail of the probability distribution to omit for Latin Hypercube sampling. For densities with a finite support minimum (e.g., Lognormal or Empirical) the default value is left.tail.cutoff=0; for densities with a support minimum of $-\infty$, the default value is left.tail.cutoff=.Machine$double.eps. This argument is ignored if sample.method="SRS".
right.tail.cutoff	a scalar between 0 and 1 indicating what proportion of the right-tail of the probability distribution to omit for Latin Hypercube sampling. For densities with a finite support maximum (e.g., Beta or Empirical) the default value is right.tail.cutoff=0; for densities with a support maximum of ∞, the default value is right.tail.cutoff=.Machine$double.eps. This argument is ignored if sample.method="SRS".

Details

Simple Random Sampling (sample.method="SRS")
When sample.method="SRS", the function simulateVector simply calls the function r*abb*, where

abb denotes the abbreviation of the specified distribution (e.g., `rlnorm`, `remp`, etc.).

Latin Hypercube Sampling (`sample.method="LHS"`)
When `sample.method="LHS"`, the function `simulateVector` generates `n` random numbers using Latin Hypercube sampling. The distribution is divided into `n` intervals of equal probability $1/n$ and simple random sampling is performed once within each interval; i.e., Latin Hypercube sampling is simply stratified sampling without replacement, where the strata are defined by the 0'th, 100(1/n)'th, 100(2/n)'th, ..., and 100'th percentiles of the distribution.

Latin Hypercube sampling, sometimes abbreviated ***LHS***, is a method of sampling from a probability distribution that ensures all portions of the probability distribution are represented in the sample. It was introduced in the published literature by McKay et al. (1979) to overcome the following problem in Monte Carlo simulation based on simple random sampling (SRS). Suppose we want to generate random numbers from a specified distribution. If we use simple random sampling, there is a low probability of getting very many observations in an area of low probability of the distribution. For example, if we generate n observations from the distribution, the probability that none of these observations falls into the upper 98'th percentile of the distribution is 0.98^n. So, for example, there is a 13% chance that out of 100 random numbers, none will fall at or above the 98'th percentile. If we are interested in reproducing the shape of the distribution, we will need a very large number of observations to ensure that we can adequately characterize the tails of the distribution (Vose, 2008, pp. 59–62).

See Millard (2013) for a visual explanation of Latin Hypercube sampling.

Value

a numeric vector of random numbers from the specified distribution.

Note

Latin Hypercube sampling, sometimes abbreviated ***LHS***, is a method of sampling from a probability distribution that ensures all portions of the probability distribution are represented in the sample. It was introduced in the published literature by McKay et al. (1979). Latin Hypercube sampling is often used in probabilistic risk assessment, specifically for sensitivity and uncertainty analysis (e.g., Iman and Conover, 1980; Iman and Helton, 1988; Iman and Helton, 1991; Vose, 1996).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Iman, R.L., and W.J. Conover. (1980). Small Sample Sensitivity Analysis Techniques for Computer Models, With an Application to Risk Assessment (with Comments). *Communications in Statistics–Volume A, Theory and Methods*, **9**(17), 1749–1874.

Iman, R.L., and J.C. Helton. (1988). An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models. *Risk Analysis* **8**(1), 71–90.

Iman, R.L. and J.C. Helton. (1991). The Repeatability of Uncertainty and Sensitivity Analyses for Complex Probabilistic Risk Assessments. *Risk Analysis* **11**(4), 591–606.

McKay, M.D., R.J. Beckman., and W.J. Conover. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code. *Technometrics* **21**(2), 239–245.

Millard, S.P. (2013). *EnvStats: an R Package for Environmental Statistics*. Springer, New York. https://link.springer.com/book/10.1007/978-1-4614-8456-1.

Vose, D. (2008). *Risk Analysis: A Quantitative Guide*. Third Edition. John Wiley & Sons, West Sussex, UK, 752 pp.

See Also

Probability Distributions and Random Numbers, Empirical, `simulateMvMatrix`, `set.seed`.

Examples

```
# Generate 10 observations from a lognormal distribution with
# parameters mean=10 and cv=1 using simple random sampling:

simulateVector(10, distribution = "lnormAlt",
  param.list = list(mean = 10, cv = 1), seed = 47,
  sort = TRUE)
# [1]  2.086931  2.863589  3.112866  5.592502  5.732602  7.160707
# [7]  7.741327  8.251306 12.782493 37.214748

#----------

# Repeat the above example by calling rlnormAlt directly:

set.seed(47)
sort(rlnormAlt(10, mean = 10, cv = 1))
# [1]  2.086931  2.863589  3.112866  5.592502  5.732602  7.160707
# [7]  7.741327  8.251306 12.782493 37.214748

#----------

# Now generate 10 observations from the same lognormal distribution
# but use Latin Hypercube sampling.  Note that the largest value
# is larger than for simple random sampling:

simulateVector(10, distribution = "lnormAlt",
  param.list = list(mean = 10, cv = 1), seed = 47,
  sample.method = "LHS", sort = TRUE)
# [1]  2.406149  2.848428  4.311175  5.510171  6.467852  8.174608
# [7]  9.506874 12.298185 17.022151 53.552699

#==========

# Generate 50 observations from a Pareto distribution with parameters
# location=10 and shape=2, then use this resulting vector of
# observations as the basis for generating 3 observations from an
# empirical distribution using Latin Hypercube sampling:
```

```
set.seed(321)
pareto.rns <- rpareto(50, location = 10, shape = 2)

simulateVector(3, distribution = "emp",
  param.list = list(obs = pareto.rns), sample.method = "LHS")
#[1] 11.50685 13.50962 17.47335

#==========

# Clean up
#---------
rm(pareto.rns)
```

`Skagit.NH3_N.df` *Ammonia Nitrogen Concentrations in the Skagit River, Marblemount, Washington*

Description

Ammonia nitrogen (NH_3—N) concentration (mg/L) in the Skagit River measured monthly from January 1978 through December 2010 at the Marblemount, Washington monitoring station.

Usage

```
Skagit.NH3_N.df
```

Format

A data frame with 396 observations on the following 6 variables.

`Date` Date of collection.

`NH3_N.Orig.mg.per.L` a character vector of the ammonia nitrogen concentrations where values for non-detects are preceeded with the less-than sign (<).

`NH3_N.mg.per.L` a numeric vector of ammonia nitrogen concentrations; non-detects have been coded to their detection limit.

`DQ1` factor of data qualifier values.

- `U` = The analyte was not detected at or above the reported result.
- `J` = The analyte was positively identified. The associated numerical result is an estimate.
- `UJ` = The analyte was not detected at or above the reported estimated result.

`DQ2` factor of data qualifier values. An asterisk (`*`) indicates a possible quality problem for the result.

`Censored` a logical vector indicating which observations are censored.

Details

Station 04A100 - Skagit R \@ Marblemount. Located at the bridge on the Casdace River Road where Highway 20 (North Cascades Highway) turns 90 degrees in Marblemount.

Source

Washington State Deparment of Ecology.
https://ecology.wa.gov/Research-Data/Monitoring-assessment/River-stream-monitoring/Water-quality-monitoring/Using-river-stream-water-quality-data

---

`skewness` *Coefficient of Skewness*

Description

Compute the sample coefficient of skewness.

Usage

```
skewness(x, na.rm = FALSE, method = "fisher", l.moment.method = "unbiased",
  plot.pos.cons = c(a = 0.35, b = 0))
```

Arguments

`x`	numeric vector of observations.
`na.rm`	logical scalar indicating whether to remove missing values from `x`. If `na.rm=FALSE` (the default) and `x` contains missing values, then a missing value (`NA`) is returned. If `na.rm=TRUE`, missing values are removed from `x` prior to computing the coefficient of variation.
`method`	character string specifying what method to use to compute the sample coefficient of skewness. The possible values are `"fisher"` (ratio of unbiased moment estimators; the default), `"moments"` (ratio of product moment estimators), or `"l.moments"` (ratio of L-moment estimators).
`l.moment.method`	character string specifying what method to use to compute the L-moments when `method="l.moments"`. The possible values are `"ubiased"` (method based on the U-statistic; the default), or `"plotting.position"` (method based on the plotting position formula).
`plot.pos.cons`	numeric vector of length 2 specifying the constants used in the formula for the plotting positions when `method="l.moments"` and `l.moment.method="plotting.position"`. The default value is `plot.pos.cons=c(a=0.35, b=0)`. If this vector has a names attribute with the value `c("a","b")` or `c("b","a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name `"a"` and the second element to the name `"b"`.

Details

Let $\underline{x}$ denote a random sample of n observations from some distribution with mean μ and standard deviation σ.

Product Moment Coefficient of Skewness (`method="moment"` or `method="fisher"`)
The ***coefficient of skewness*** of a distribution is the third standardized moment about the mean:

$$\eta_3 = \sqrt{\beta_1} = \frac{\mu_3}{\sigma^3} \qquad (1)$$

where

$$\eta_r = E[(\frac{X-\mu}{\sigma})^r] = \frac{1}{\sigma^r}E[(X-\mu)^r] = \frac{\mu_r}{\sigma^r} \qquad (2)$$

and

$$\mu_r = E[(X-\mu)^r] \qquad (3)$$

denotes the r'th moment about the mean (central moment). That is, the coefficient of skewness is the third central moment divided by the cube of the standard deviation. The coefficient of skewness is 0 for a symmetric distribution. Distributions with positive skew have heavy right-hand tails, and distributions with negative skew have heavy left-hand tails.

When `method="moment"`, the coefficient of skewness is estimated using the method of moments estimator for the third central moment and and the method of moments estimator for the variance:

$$\hat{\eta}_3 = \frac{\hat{\mu}_3}{\sigma^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{[\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2]^{3/2}} \qquad (5)$$

where

$$\hat{\sigma}_m^2 = s_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2 \qquad (6)$$

This form of estimation should be used when resampling (bootstrap or jackknife).

When `method="fisher"`, the coefficient of skewness is estimated using the unbiased estimator for the third central moment (Serfling, 1980, p.73; Chen, 1995, p.769) and the unbiased estimator for the variance.

$$\hat{\eta}_3 = \frac{\frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}(x_i-\bar{x})^3}{s^3} \qquad (7)$$

where

$$\hat{\sigma}^2 = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2 \qquad (8)$$

(Note that Serfling, 1980, p.73 contains a typographical error in the numerator for the unbiased estimator of the third central moment.)

L-Moment Coefficient of skewness (`method="l.moments"`)
Hosking (1990) defines the L-moment analog of the coefficient of skewness as:

$$\tau_3 = \frac{\lambda_3}{\lambda_2} \qquad (9)$$

that is, the third L-moment divided by the second L-moment. He shows that this quantity lies in the interval (-1, 1).

When `l.moment.method="unbiased"`, the L-skewness is estimated by:

$$t_3 = \frac{l_3}{l_2} \qquad (10)$$

that is, the unbiased estimator of the third L-moment divided by the unbiased estimator of the second L-moment.

When `l.moment.method="plotting.position"`, the L-skewness is estimated by:

$$\tilde{\tau}_3 = \frac{\tilde{\lambda}_3}{\tilde{\lambda}_2} \qquad (11)$$

that is, the plotting-position estimator of the third L-moment divided by the plotting-position estimator of the second L-moment.

See the help file for `lMoment` for more information on estimating L-moments.

Value

A numeric scalar – the sample coefficient of skewness.

Note

Traditionally, the coefficient of skewness has been estimated using product moment estimators. Sometimes an estimate of skewness is used in a goodness-of-fit test for normality (e.g., set `test="skew"` in the call to `gofTest`).

Hosking (1990) introduced the idea of L-moments and L-skewness.

Vogel and Fennessey (1993) argue that L-moment ratios should replace product moment ratios because of their superior performance (they are nearly unbiased and better for discriminating between distributions). They compare product moment diagrams with L-moment diagrams.

Hosking and Wallis (1995) recommend using unbiased estimators of L-moments (vs. plotting-position estimators) for almost all applications.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Chen, L. (1995). Testing the Mean of Skewed Distributions. *Journal of the American Statistical Association* **90**(430), 767–772.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Serfling, R.J. (1980). *Approximation Theorems of Mathematical Statistics*. John Wiley and Sons, New York, p.73.

Taylor, J.K. (1990). *Statistical Techniques for Data Analysis*. Lewis Publishers, Boca Raton, FL.

Vogel, R.M., and N.M. Fennessey. (1993). L Moment Diagrams Should Replace Product Moment Diagrams. *Water Resources Research* **29**(6), 1745–1752.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

var, sd, cv, kurtosis, summaryFull, Summary Statistics.

Examples

```
# Generate 20 observations from a lognormal distribution with parameters
# mean=10 and cv=1, and estimate the coefficient of skewness.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

dat <- rlnormAlt(20, mean = 10, cv = 1)

skewness(dat)
#[1] 0.9876632

skewness(dat, method = "moment")
#[1] 0.9119889

skewness(dat, meth = "l.moment")
#[1] 0.2656674

#----------
# Clean up
rm(dat)
```

---

stat_mean_sd_text *Add Text Indicating the Mean and Standard Deviation to a ggplot2 Plot*

Description

For a strip plot or scatterplot produced using the package ggplot2 (e.g., with geom_point), for each value on the x-axis, add text indicating the mean and standard deviation of the y-values for that particular x-value.

Usage

```
stat_mean_sd_text(mapping = NULL, data = NULL,
  geom = ifelse(text.box, "label", "text"),
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, y.pos = NULL, y.expand.factor = 0.2,
  digits = 1, digit.type = "round",
```

```
    nsmall = ifelse(digit.type == "round", digits, 0), text.box = FALSE,
    alpha = 1, angle = 0, color = "black", family = "", fontface = "plain",
    hjust = 0.5, label.padding = ggplot2::unit(0.25, "lines"),
    label.r = ggplot2::unit(0.15, "lines"), label.size = 0.25,
    lineheight = 1.2, size = 4, vjust = 0.5, ...)
```

Arguments

`mapping`, `data`, `position`, `na.rm`, `show.legend`, `inherit.aes`
: See the help file for `geom_text`.

`geom`
: Character string indicating which geom to use to display the text. Setting `geom="text"` will use `geom_text` to display the text, and setting `geom="label"` will use `geom_label` to display the text. The default value is `geom="text"` unless the user sets `text.box=TRUE`.

`y.pos`
: Numeric scalar indicating the y-position of the text (i.e., the value of the argument `y` that will be used in the call to `geom_text` or `geom_label`). The default value is `y.pos=NULL`, in which case `y.pos` is set to the maximum value of all y-values plus a proportion of the range of all y-values, where the proportion is determined by the argument `y.expand.factor` (see below).

`y.expand.factor`
: For the case when `y.pos=NULL`, a numeric scalar indicating the proportion by which the range of all y-values should be multiplied by before adding this value to the maximum value of all y-values in order to compute the value of the argument `y.pos` (see above). The default value is `y.expand.factor=0.2`.

`digits`
: Integer indicating the number of digits to use for displaying the mean and standard deviation. When `digit.type="round"` (see below) the argument `digits` indicates the number of digits to round to, and when `digit.type="signif"` the argument `digits` indicates the number of significant digits to display. The default value is `digits=1`.

`digit.type`
: Character string indicating whether the `digits` argument (see above) refers to significant digits (`digit.type="signif"`), or how many decimal places to round to (`digit.type="round"`, the default).

`nsmall`
: Integer passed to the function `format` indicating the the minimum number of digits to the right of the decimal point for the computed mean and standard deviation. The default value is `nsmall=digits` when `digit.type="round"` and `nsmall=0` when `digit.type="signif"`. When `nsmall` is greater than 0, all computed means and standard deviations will have the same number of digits to the right of the decimal point (including, possibly, trailing zeros). To omit trailing zeros, set `nsmall=0`.

`text.box`
: Logical scalar indicating whether to surround the text with a text box (i.e., whether to use `geom_label` instead of `geom_text`). This argument can be overridden by simply specifying the argument `geom`.

`alpha`, `angle`, `color`, `family`, `fontface`, `hjust`, `vjust`, `lineheight`, `size`
: See the help file for `geom_text` and the vignette **Aesthetic specifications** at https://cran.r-project.org/package=ggplot2/vignettes/ggplot2-specs.html.

`label.padding`, `label.r`, `label.size`
: See the help file for `geom_text`.

`...`
: Other arguments passed on to `layer`.

Details

See the help file for `geom_text` for details about how `geom_text` and `geom_label` work.

See the vignette **Extending ggplot2** at `https://cran.r-project.org/package=ggplot2/vignettes/extending-ggplot2.html` for information on how to create a new stat.

Note

The function `stat_mean_sd_text` is called by the function `geom_stripchart`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis (Use R!)*. Second Edition. Springer.

See Also

`geom_stripchart`, `stat_median_iqr_text`, `stat_n_text`, `stat_test_text`, `geom_text`, `geom_label`, `mean`, `sd`.

Examples

```
# First, load and attach the ggplot2 package.
#-------------------------------------------

library(ggplot2)

#====================

# Example 1:

# Using the built-in data frame mtcars,
# plot miles per gallon vs. number of cylinders
# using different colors for each level of the number of cylinders.
#-------------------------------------------------------------------

p <- ggplot(mtcars, aes(x = factor(cyl), y = mpg, color = factor(cyl))) +
  theme(legend.position = "none")

p + geom_point() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

# Now add text indicating the mean and standard deviation
```

```
# for each level of cylinder.
#-------------------------------------------------------

dev.new()
p + geom_point() +
  stat_mean_sd_text() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 2:

# Repeat Example 1, but:
# 1) facet by transmission type,
# 2) make the size of the text smaller.
#--------------------------------------

dev.new()
p + geom_point() +
  stat_mean_sd_text(size = 3) +
  facet_wrap(~ am, labeller = label_both) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 3:

# Repeat Example 1, but specify the y-position for the text.
#-----------------------------------------------------------

dev.new()
p + geom_point() +
  stat_mean_sd_text(y.pos = 36) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 4:

# Repeat Example 1, but show the
# mean and standard deviation in a text box.
#-------------------------------------------

dev.new()
p + geom_point() +
  stat_mean_sd_text(text.box = TRUE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 5:

# Repeat Example 1, but use the color brown for the text.
```

```
#---------------------------------------------------------

dev.new()
p + geom_point() +
  stat_mean_sd_text(color = "brown") +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 6:

# Repeat Example 1, but:
# 1) use the same colors for the text that are used for each group,
# 2) use the bold monospaced font.
#-----------------------------------------------------------------

mat <- ggplot_build(p)$data[[1]]
group <- mat[, "group"]
colors <- mat[match(1:max(group), group), "colour"]

dev.new()
p + geom_point() +
  stat_mean_sd_text(color = colors, size = 5,
    family = "mono", fontface = "bold") +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Clean up
#---------

graphics.off()
rm(p, mat, group, colors)
```

`stat_median_iqr_text` *Add Text Indicating the Median and Interquartile Range to a ggplot2 Plot*

Description

For a strip plot or scatterplot produced using the package ggplot2 (e.g., with `geom_point`), for each value on the x-axis, add text indicating the median and interquartile range (IQR) of the y-values for that particular x-value.

Usage

```
stat_median_iqr_text(mapping = NULL, data = NULL,
  geom = ifelse(text.box, "label", "text"),
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, y.pos = NULL, y.expand.factor = 0.2,
```

```
    digits = 1, digit.type = "round",
    nsmall = ifelse(digit.type == "round", digits, 0), text.box = FALSE,
    alpha = 1, angle = 0, color = "black", family = "", fontface = "plain",
    hjust = 0.5, label.padding = ggplot2::unit(0.25, "lines"),
    label.r = ggplot2::unit(0.15, "lines"), label.size = 0.25,
    lineheight = 1.2, size = 4, vjust = 0.5, ...)
```

Arguments

`mapping`, `data`, `position`, `na.rm`, `show.legend`, `inherit.aes`
: See the help file for `geom_text`.

`geom`
: Character string indicating which geom to use to display the text. Setting `geom="text"` will use `geom_text` to display the text, and setting `geom="label"` will use `geom_label` to display the text. The default value is `geom="text"` unless the user sets `text.box=TRUE`.

`y.pos`
: Numeric scalar indicating the y-position of the text (i.e., the value of the argument `y` that will be used in the call to `geom_text` or `geom_label`). The default value is `y.pos=NULL`, in which case `y.pos` is set to the maximum value of all y-values plus a proportion of the range of all y-values, where the proportion is determined by the argument `y.expand.factor` (see below).

`y.expand.factor`
: For the case when `y.pos=NULL`, a numeric scalar indicating the proportion by which the range of all y-values should be multiplied by before adding this value to the maximum value of all y-values in order to compute the value of the argument `y.pos` (see above). The default value is `y.expand.factor=0.2`.

`digits`
: Integer indicating the number of digits to use for displaying the median and interquartile range. When `digit.type="round"` (see below) the argument `digits` indicates the number of digits to round to, and when `digit.type="signif"` the argument `digits` indicates the number of significant digits to display. The default value is `digits=1`.

`digit.type`
: Character string indicating whether the `digits` argument (see above) refers to significant digits (`digit.type="signif"`), or how many decimal places to round to (`digit.type="round"`, the default).

`nsmall`
: Integer passed to the function `format` indicating the the minimum number of digits to the right of the decimal point for the computed median and interquartile range. The default value is `nsmall=digits` when `digit.type="round"` and `nsmall=0` when `digit.type="signif"`. When `nsmall` is greater than 0, all computed medians and interquartile ranges will have the same number of digits to the right of the decimal point (including, possibly, trailing zeros). To omit trailing zeros, set `nsmall=0`.

`text.box`
: Logical scalar indicating whether to surround the text with a text box (i.e., whether to use `geom_label` instead of `geom_text`). This argument can be overridden by simply specifying the argument `geom`.

`alpha`, `angle`, `color`, `family`, `fontface`, `hjust`, `vjust`, `lineheight`, `size`
: See the help file for `geom_text` and the vignette **Aesthetic specifications** at https://cran.r-project.org/package=ggplot2/vignettes/ggplot2-specs.html.

`label.padding`, `label.r`, `label.size`
: See the help file for `geom_text`.

`...`
: Other arguments passed on to `layer`.

Details

See the help file for `geom_text` for details about how `geom_text` and `geom_label` work.

See the vignette **Extending ggplot2** at `https://cran.r-project.org/package=ggplot2/vignettes/extending-ggplot2.html` for information on how to create a new stat.

Note

The function `stat_median_iqr_text` is called by the function `geom_stripchart`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis (Use R!)*. Second Edition. Springer.

See Also

`geom_stripchart`, `stat_mean_sd_text`, `stat_n_text`, `stat_test_text`, `geom_text`, `geom_label`, `median`, `iqr`.

Examples

```
# First, load and attach the ggplot2 package.
#-------------------------------------------

library(ggplot2)

#====================

# Example 1:

# Using the built-in data frame mtcars,
# plot miles per gallon vs. number of cylinders
# using different colors for each level of the number of cylinders.
#-------------------------------------------------------------------

p <- ggplot(mtcars, aes(x = factor(cyl), y = mpg, color = factor(cyl))) +
  theme(legend.position = "none")

p + geom_point() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")
```

```
# Now add text indicating the median and interquartile range
# for each level of cylinder.
#-----------------------------------------------------------

dev.new()
p + geom_point() +
  stat_median_iqr_text() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 2:

# Repeat Example 1, but:
# 1) facet by transmission type,
# 2) make the size of the text smaller.
#---------------------------------------

dev.new()
p + geom_point() +
  stat_median_iqr_text(size = 2.75) +
  facet_wrap(~ am, labeller = label_both) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 3:

# Repeat Example 1, but specify the y-position for the text.
#-----------------------------------------------------------

dev.new()
p + geom_point() +
  stat_median_iqr_text(y.pos = 36) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")


#====================

# Example 4:

# Repeat Example 1, but show the
# median and interquartile range in a text box.
#----------------------------------------------

dev.new()
p + geom_point() +
  stat_median_iqr_text(text.box = TRUE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 5:
```

```
# Repeat Example 1, but use the color brown for the text.
#--------------------------------------------------------

dev.new()
p + geom_point() +
  stat_median_iqr_text(color = "brown") +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Example 6:

# Repeat Example 1, but:
# 1) use the same colors for the text that are used for each group,
# 2) use the bold monospaced font.
#-----------------------------------------------------------------

mat <- ggplot_build(p)$data[[1]]
group <- mat[, "group"]
colors <- mat[match(1:max(group), group), "colour"]

dev.new()
p + geom_point() +
  stat_median_iqr_text(color = colors,
    family = "mono", fontface = "bold") +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#====================

# Clean up
#---------

graphics.off()
rm(p, mat, group, colors)
```

---

stat_n_text *Add Text Indicating the Sample Size to a ggplot2 Plot*

Description

For a strip plot or scatterplot produced using the package ggplot2 (e.g., with geom_point), for each value on the x-axis, add text indicating the number of y-values for that particular x-value.

Usage

```
stat_n_text(mapping = NULL, data = NULL,
  geom = ifelse(text.box, "label", "text"),
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, y.pos = NULL, y.expand.factor = 0.1,
```

```
text.box = FALSE, alpha = 1, angle = 0, color = "black",
family = "", fontface = "plain", hjust = 0.5,
label.padding = ggplot2::unit(0.25, "lines"),
label.r = ggplot2::unit(0.15, "lines"), label.size = 0.25,
lineheight = 1.2, size = 4, vjust = 0.5, ...)
```

Arguments

`mapping`, `data`, `position`, `na.rm`, `show.legend`, `inherit.aes`
: See the help file for `geom_text`.

`geom`
: Character string indicating which geom to use to display the text. Setting `geom="text"` will use `geom_text` to display the text, and setting `geom="label"` will use `geom_label` to display the text. The default value is `geom="text"` unless the user sets `text.box=TRUE`.

`y.pos`
: Numeric scalar indicating the y-position of the text (i.e., the value of the argument `y` that will be used in the call to `geom_text` or `geom_label`). The default value is `y.pos=NULL`, in which case `y.pos` is set to the minimum value of all y-values minus a proportion of the range of all y-values, where the proportion is determined by the argument `y.expand.factor` (see below).

`y.expand.factor`
: For the case when `y.pos=NULL`, a numeric scalar indicating the proportion by which the range of all y-values should be multiplied by before subtracting this value from the minimum value of all y-values in order to compute the value of the argument `y.pos` (see above). The default value is `y.expand.factor=0.1`.

`text.box`
: Logical scalar indicating whether to surround the text with a text box (i.e., whether to use `geom_label` instead of `geom_text`). This argument can be overridden by simply specifying the argument `geom`.

`alpha`, `angle`, `color`, `family`, `fontface`, `hjust`, `vjust`, `lineheight`, `size`
: See the help file for `geom_text` and the vignette **Aesthetic specifications** at `https://cran.r-project.org/package=ggplot2/vignettes/ggplot2-specs.html`.

`label.padding`, `label.r`, `label.size`
: See the help file for `geom_text`.

`...`
: Other arguments passed on to `layer`.

Details

See the help file for `geom_text` for details about how `geom_text` and `geom_label` work.

See the vignette **Extending ggplot2** at `https://cran.r-project.org/package=ggplot2/vignettes/extending-ggplot2.html` for information on how to create a new stat.

Note

The function `stat_n_text` is called by the function `geom_stripchart`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis (Use R!)*. Second Edition. Springer.

See Also

geom_stripchart, stat_mean_sd_text, stat_median_iqr_text, stat_test_text, geom_text, geom_label.

Examples

```
# First, load and attach the ggplot2 package.
#-------------------------------------------

library(ggplot2)

#====================

# Example 1:

# Using the built-in data frame mtcars,
# plot miles per gallon vs. number of cylinders
# using different colors for each level of the number of cylinders.
#-------------------------------------------------------------------

p <- ggplot(mtcars, aes(x = factor(cyl), y = mpg, color = factor(cyl))) +
  theme(legend.position = "none")

p + geom_point() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")


# Now add the sample size for each level of cylinder.
#----------------------------------------------------

dev.new()
p + geom_point() +
  stat_n_text() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 2:

# Repeat Example 1, but:
# 1) facet by transmission type,
# 2) make the size of the text smaller.
#--------------------------------------

dev.new()
p + geom_point() +
  stat_n_text(size = 3) +
```

```
  facet_wrap(~ am, labeller = label_both) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 3:

# Repeat Example 1, but specify the y-position for the text.
#-----------------------------------------------------------

dev.new()
p + geom_point() +
  stat_n_text(y.pos = 5) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 4:

# Repeat Example 1, but show the sample size in a text box.
#----------------------------------------------------------

dev.new()
p + geom_point() +
  stat_n_text(text.box = TRUE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 5:

# Repeat Example 1, but use the color brown for the text.
#--------------------------------------------------------

dev.new()
p + geom_point() +
  stat_n_text(color = "brown") +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 6:

# Repeat Example 1, but:
# 1) use the same colors for the text that are used for each group,
# 2) use the bold monospaced font.
#-------------------------------------------------------------------

mat <- ggplot_build(p)$data[[1]]
group <- mat[, "group"]
colors <- mat[match(1:max(group), group), "colour"]

dev.new()
```

```
p + geom_point() +
  stat_n_text(color = colors, size = 5,
    family = "mono", fontface = "bold") +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Clean up
#---------

graphics.off()
rm(p, mat, group, colors)
```

`stat_test_text` *Add Text to a ggplot2 Plot Indicating the Results of a Hypothesis Test*

Description

For a strip plot or scatterplot produced using the package ggplot2 (e.g., with `geom_point`), add text indicating the results of a hypothesis test comparing locations betweeen groups, where the groups are defined based on the unique x-values.

Usage

```
stat_test_text(mapping = NULL, data = NULL,
  geom = ifelse(text.box, "label", "text"), position = "identity",
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE,
  y.pos = NULL, y.expand.factor = 0.35, test = "parametric",
  paired = FALSE, test.arg.list = list(), two.lines = TRUE,
  p.value.digits = 3, p.value.digit.type = "round",
  location.digits = 1, location.digit.type = "round",
  nsmall = ifelse(location.digit.type == "round", location.digits, 0),
  text.box = FALSE, alpha = 1, angle = 0, color = "black",
  family = "", fontface = "plain", hjust = 0.5,
  label.padding = ggplot2::unit(0.25, "lines"),
  label.r = ggplot2::unit(0.15, "lines"), label.size = 0.25,
  lineheight = 1.2, size = 4, vjust = 0.5, ...)
```

Arguments

`mapping`, `data`, `position`, `na.rm`, `show.legend`, `inherit.aes`
: See the help file for `geom_text`.

`geom`
: Character string indicating which geom to use to display the text. Setting `geom="text"` will use `geom_text` to display the text, and setting `geom="label"` will use `geom_label` to display the text. The default value is `geom="text"` unless the user sets `text.box=TRUE`.

`y.pos` Numeric scalar indicating the y-position of the text (i.e., the value of the argument `y` that will be used in the call to `geom_text` or `geom_label`). The default value is `y.pos=NULL`, in which case `y.pos` is set to the maximum value of all y-values plus a proportion of the range of all y-values, where the proportion is determined by the argument `y.expand.factor` (see below).

`y.expand.factor`
For the case when `y.pos=NULL`, a numeric scalar indicating the proportion by which the range of all y-values should be multiplied by before adding this value to the maximum value of all y-values in order to compute the value of the argument `y.pos` (see above). The default value is `y.expand.factor=0.35`.

`test` A character string indicating whether to use a standard parametric test (`test="parametric"`, the default) or nonparametric test (`test="nonparametric"`) to compare groups.

`paired` For the case of two groups, a logical scalar indicating whether the data should be considered to be paired. The default value is `paired=FALSE`.

NOTE: if the argument `test.arg.list` is supplied and it includes a component named `paired`, the value of that component is overriden by the value of the argument `paired`.

`test.arg.list` An optional list of arguments to pass to the function used to test for group differences in location. The default value is an empty list: `test.arg.list=list()`. In particular, when there are two groups, `ci.and.test="parametric"`, and `ci.arg.list` does not contain a component specifying the value for `var.equal`, this argument is updated to include the component `var.equal=TRUE`, which is **not** the default behavior of `t.test`.

NOTE: If `test.arg.list` contains a component named `"paired"`, the value of that component is set to the value of the argument `paired` (see above).

`two.lines` For the case of one or two groups, a logical scalar indicating whether the associated confidence interval should be be displayed on a second line instead of on the same line as the p-value. The default is `two.lines=TRUE`.

`p.value.digits` An integer indicating the number of digits to use for displaying the p-value. When `p.value.digit.type="round"` (see below) the argument `p.value.digits` indicates the number of digits to round to, and when `p.value.digit.type="signif"` the argument `p.value.digits` indicates the number of significant digits to display. The default value is `p.value.digits=3`.

`p.value.digit.type`
A character string indicating whether the `p.value.digits` argument (see above) refers to significant digits (`p.value.digit.type="signif"`), or how many decimal places to round to (`p.value.digit.type="round"`, the default).

`location.digits`
For the case of one or two groups, an integer indicating the number of digits to use for displaying the associated confidence interval. When `location.digit.type="round"` (see below) the argument `location.digits` indicates the number of digits to round to, and when `location.digit.type="signif"` the argument `location.digits` indicates the number of significant digits to display. The default value is `location.digits=1`.

`location.digit.type`
For the case of one or two groups, a character string indicating whether the `location.digits` argument (see above) refers to significant digits

(location.digit.type="signif"), or how many decimal places to round to (location.digit.type="round"; the default).

nsmall — For the case of one or two groups, an integer passed to the function format indicating the the minimum number of digits to use to the right of the decimal point for the associated confidence interval. The default value is nsmall=digits when digit.type="round" and nsmall=0 when digit.type="signif". When nsmall is greater than 0, the two confidence limits will have the same number of digits to the right of the decimal point (including, possibly, trailing zeros). To omit trailing zeros, set nsmall=0.

text.box — Logical scalar indicating whether to surround the text with a text box (i.e., whether to use geom_label instead of geom_text). This argument can be overridden by simply specifying the argument geom.

alpha, angle, color, family, fontface, hjust, vjust, lineheight, size — See the help file for geom_text and the vignette **Aesthetic specifications** at https://cran.r-project.org/package=ggplot2/vignettes/ggplot2-specs.html.

label.padding, label.r, label.size — See the help file for geom_text.

... — Other arguments passed on to layer.

Details

The table below shows which hypothesis tests are performed based on the number of groups and the values of the arguments test and paired.

# Groups	test	paired	Name	Function Called
1	"parametric"		One-Sample t-test	t.test
	"nonparametric"		Wilcoxon Signed Rank Test	wilcox.test
2	"parametric"	FALSE	Two-Sample t-test	t.test
		TRUE	Paired t-test	t.test
	"nonparametric"	FALSE	Wilcoxon Rank Sum Test	wilcox.test
		TRUE	Wilcoxon Signed Rank Test on Paired Differences	wilcox.test
≥ 3	"parametric"		Analysis of Variance	aov summary.aov
	"nonparametric"		Kruskal-Wallis Test	kruskal.test

See the help file for geom_text for details about how geom_text and geom_label work.

See the vignette **Extending ggplot2** at https://cran.r-project.org/package=ggplot2/vignettes/extending-ggplot2.html for information on how to create a new stat.

Note

The function stat_test_text is called by the function geom_stripchart.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis (Use R!)*. Second Edition. Springer.

See Also

geom_stripchart, stat_mean_sd_text, stat_median_iqr_text, stat_n_text, geom_text, geom_label, t.test, wilcox.test, aov, summary.aov, kruskal.test.

Examples

```
# First, load and attach the ggplot2 package.
#-------------------------------------------

library(ggplot2)

#==========

# Example 1:

# Using the built-in data frame mtcars,
# plot miles per gallon vs. number of cylinders
# using different colors for each level of the number of cylinders.
#-------------------------------------------------------------------

p <- ggplot(mtcars, aes(x = factor(cyl), y = mpg, color = factor(cyl))) +
  theme(legend.position = "none")

p + geom_point(show.legend = FALSE) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")


# Now add text indicating the sample size and
# mean and standard deviation for each level of cylinder, and
# test for the difference in means between groups.
#-------------------------------------------------------------

dev.new()
p + geom_point() +
  stat_n_text() + stat_mean_sd_text() +
```

```
  stat_test_text() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 2:

# Repeat Example 1, but show text indicating the median and IQR,
# and use the nonparametric test.
#---------------------------------------------------------------

dev.new()
p + geom_point() +
  stat_n_text() + stat_median_iqr_text() +
  stat_test_text(test = "nonparametric") +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 3:

# Repeat Example 1, but use only the groups with
# 4 and 8 cylinders.
#-----------------------------------------------

p <- ggplot(subset(mtcars, cyl %in% c(4, 8)),
  aes(x = factor(cyl), y = mpg, color = cyl)) +
  theme(legend.position = "none")

dev.new()
p + geom_point() +
  stat_n_text() + stat_mean_sd_text() +
  stat_test_text() +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")

#==========

# Example 4:

# Repeat Example 3, but
# 1) facet by transmission type,
# 2) make the text smaller,
# 3) put the text for the test results in a text box
#    and make them blue.
#-------------------------------------------------

dev.new()
p + geom_point() +
  stat_n_text(size = 3) + stat_mean_sd_text(size = 3) +
  stat_test_text(size = 3, text.box = TRUE, color = "blue") +
  facet_wrap(~ am, labeller = label_both) +
  labs(x = "Number of Cylinders", y = "Miles per Gallon")
```

```
#==========

# Clean up
#---------

graphics.off()
rm(p)
```

---

stripChart *1-D Scatter Plots with Confidence Intervals*

Description

stripChart is a modification of the R function stripchart. It is a generic function used to produce one dimensional scatter plots (or dot plots) of the given data, along with text indicating sample size and estimates of location (mean or median) and scale (standard deviation or interquartile range), as well as confidence intervals for the population location parameter. One dimensional scatterplots are a good alternative to boxplots when sample sizes are small or moderate. The function invokes particular methods which depend on the class of the first argument.

Usage

```
stripChart(x, ...)

## S3 method for class 'formula'
stripChart(x, data = NULL, dlab = NULL,
    subset, na.action = NULL, ...)

## Default S3 method:
stripChart(x,
    method = ifelse(paired && paired.lines, "overplot", "stack"),
    seed = 47, jitter = 0.1 * cex, offset = 1/2, vertical = TRUE,
    group.names, group.names.cex = cex, drop.unused.levels = TRUE,
    add = FALSE, at = NULL, xlim = NULL, ylim = NULL, ylab = NULL,
    xlab = NULL, dlab = "", glab = "", log = "", pch = 1, col = par("fg"),
    cex = par("cex"), points.cex = cex, axes = TRUE, frame.plot = axes,
    show.ci = TRUE, location.pch = 16, location.cex = cex,
    conf.level = 0.95, min.n.for.ci = 2,
    ci.offset = 3/ifelse(n > 2, (n-1)^(1/3), 1), ci.bar.lwd = cex,
    ci.bar.ends = TRUE, ci.bar.ends.size = 0.5 * cex, ci.bar.gap = FALSE,
    n.text = "bottom", n.text.line = ifelse(n.text == "bottom", 2, 0),
    n.text.cex = cex, location.scale.text = "top",
    location.scale.digits = 1, nsmall = location.scale.digits,
    location.scale.text.line = ifelse(location.scale.text == "top", 0, 3.5),
    location.scale.text.cex =
      cex * 0.8 * ifelse(n > 6, max(0.4, 1 - (n-6) * 0.06), 1),
    p.value = FALSE, p.value.digits = 3, p.value.line = 2, p.value.cex = cex,
```

```
  group.difference.ci = p.value, group.difference.conf.level = 0.95,
  group.difference.digits = location.scale.digits,
  ci.and.test = "parametric", ci.arg.list = NULL, test.arg.list = NULL,
  alternative = "two.sided", plot.diff = FALSE, diff.col = col[1],
 diff.method = "stack", diff.pch = pch[1], paired = FALSE, paired.lines = paired,
  paired.lty = 1:6, paired.lwd = 1, paired.pch = 1:14, paired.col = NULL,
  diff.name = NULL, diff.name.cex = group.names.cex, sep.line = TRUE,
  sep.lty = 2, sep.lwd = cex, sep.col = "gray", diff.lim = NULL,
  diff.at = NULL, diff.axis.label = NULL,
  plot.diff.mar = c(5, 4, 4, 4) + 0.1, ...)
```

Arguments

x	the data from which the plots are to be produced. In the default method the data can be specified as a list or data frame where each component is numeric, a numeric matrix, or a numeric vector. In the formula method, a symbolic specification of the form y ~ g can be given, indicating the observations in the vector y are to be grouped according to the levels of the factor g (the form y ~ 1 indicates no grouping). NAs are allowed in the data. NOTE: When the formula method is used and the argument paired=TRUE (see below), the data in the vector y must have the same number of observations for each level of the factor g and for each level sorted in the same way according to the pairing variable.
data	for the formula method, a data.frame (or list) from which the variables in x should be taken.
subset	for the formula method, an optional vector specifying a subset of observations to be used for plotting.
na.action	for the formula method, a function which indicates what should happen when the data contain NAs. The default is to ignore missing values in either the response or the group.
...	additional parameters passed to the default method, or by it to plot, points, axis, and title to control the appearance of the plot.
method	the method to be used to separate coincident points. When method="stack" coincident points are stacked, when method="jitter" coincident points are jittered, and when method="overplot" coincident points are overplotted. When there are 2 groups and paired=TRUE and paired.lines=TRUE the default value is method="overplot", otherwise the default method is method="stack" (which differs from the default value for the R function stripchart, which uses method="overplot" by default).
seed	when method="jitter" is used, the argument seed is passed to the R function set.seed. Since jittering depends on the R random number generator, using the same value of seed each time the same data are plotted with stripChart ensures that the resulting plot is the same.
jitter	when method="jitter" is used, jitter gives the amount of jittering applied.
offset	when stacking is used, points are stacked this many line-heights (symbol widths) apart.

`vertical` when `vertical=TRUE` (the default), the plots are drawn vertically rather than horizontally.

`group.names` Optional argument (forced to be a character string) explicitly providing the group labels that will be printed alongside (or underneath) each plot. When `group.names` is provided, it must have the same number of elements as the number of groups. When `group.names` is not provided, if the groups created based on the argument `x` have a `names` attribute, then that is used for the group names. Otherwise, the group names are set to the integers 1 to the number of groups (e.g., `1:3` for three groups).

`group.names.cex`
numeric scalar indicating the amount by which the group labels should be scaled relative to the default (see the help file for `plot.default`). The default is the current value of the graphics parameter `cex`.

`drop.unused.levels`
when `drop.unused.levels=TRUE`, groups with no observations are dropped.

`add` logical, if true *add* the chart to the current plot.

`at` numeric vector giving the locations where the charts should be drawn, particularly when `add=TRUE`; defaults to `1:n` where `n` is the number of groups.

`xlim`, `ylim` plot limits: see `plot.window`.

`ylab`, `xlab` labels: see `title`.

`dlab`, `glab` alternate way to specify axis labels. The `dlab` and `glab` labels may be used instead of `xlab` and `ylab` if those are not specified. `dlab` applies to the continuous data axis (the y-axis unless `vertical=FALSE`), and `glab` to the group axis.

`log` on which axes to use a log scale: see `plot.default`.

`pch`, `col`, `cex` Graphical parameters: see `par`.

`points.cex` Sets the `cex` value for the points plotted.

`axes`, `frame.plot`
Axis control: see `plot.default`.

`show.ci` logical scalar indicating whether to plot the confidence interval. The default is `show.ci=TRUE`.

`location.pch` integer indicating which plotting character to use to indicate the estimate of location (mean or median) for each group (see the help file for `plot.default`). The default is `location.pch=16`, a filled circle.

`location.cex` numeric scalar giving the amount by which the plotting characters indicating the estimate of location for each group should be scaled relative to the default (see the help file for `plot.default`). The default is the current value of the graphics parameter `cex`.

`conf.level` numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the group location (population mean or median). The default value is `conf.level=0.95`.

`min.n.for.ci` integer indicating the minimum sample size required in order to plot a confidence interval for the group location. The default value is `min.n.for.ci=2`.

`ci.offset` numeric scalar or vector of length equal to the number of groups (`n`) in units of `cex` indicating the amount of space between the line showing the confidence interval and tick mark associated with a particular group. The default value depends on the number of groups and is given by `3/ifelse(n > 2, (n-1)^(1/3), 1)`.

`ci.bar.lwd` numeric scalar indicating the line width for the confidence interval bars. The default is the current value of the graphics parameter `cex`.

`ci.bar.ends` logical scalar indicating whether to add flat ends to the confidence interval bars. The default value is `ci.bar.ends=TRUE`.

`ci.bar.ends.size`
numeric scalar in units of `cxy` indicating the size of confidence interval bar ends. The default value is half of the current value of `cex`.

`ci.bar.gap` logical scalar indicating with to add a gap between the estimate of group location and the confidence interval bar. The default value is `ci.bar.gap=FALSE`.

`n.text` character string indicating whether and where to indicate the sample size for each group. Possible values are `"bottom"` (the default), `"top"`, and `"none"`.

`n.text.line` integer indicating on which plot margin line to show the sample sizes for each group. The default value is `n.text.line=2` when `n.text="bottom"` and `0` otherwise.

`n.text.cex` numeric scalar giving the amount by which the text indicating the sample size for each group should be scaled relative to the default (see the help file for `plot.default`). The default is the current value of the graphics parameter `cex`.

`location.scale.text`
character string indicating whether and where to indicate the estimates of location (mean or median) and scale (standard deviation or interquartile range) for each group. Possible values are `"top"` (the default), `"bottom"`, and `"none"`.

`location.scale.digits`
integer indicating the number of digits to round the estimates of location and scale. The default value is `location.scale.digits=1`.

`nsmall` integer passed to the function `format` indicating the the minimum number of digits to the right of the decimal point for the estimates of location and scale. The default value is the value of `location.scale.digits`, which forces all estimates of location and scale have the same number of digits to the right of the decimal point (including, possibly, trailing zeros). To omit trailing zeros, set `nsmall=0`.

`location.scale.text.line`
integer indicating on which plot margin line to show the estimates of location and scale for each group. The default value is `location.scale.text.line=0` when `n.text="top"` and `3.5` otherwise.

`location.scale.text.cex`
numeric scalar giving the amount by which the text indicating the estimates of location and scale for each group should be scaled relative to the default (see the help file for `plot.default`). The default depends on the number of groups and is given by `cex * 0.8 * ifelse(n > 6, max(0.4, 1 - (n-6) * 0.06), 1)`, where `cex` denotes the current value of the graphics parameter `cex`.

`p.value` logical scalar indicating whether to show the p-value associated with testing whether all groups have the same population location. The default value is `p.value=FALSE`. The p-value is displayed at the top of the graph.

`p.value.digits` integer indicating the number of digits to round to when displaying the p-value associated with the test of equal group locations. The default value is `p.value.digits=3`.

`p.value.line` integer indicating on which plot margin line to show the p-value associated with the test of equal group locations. The default value is `p.value.line=2`.

`p.value.cex` numeric scalar giving the amount by which the text indicating the p-value associated with the test of equal group locations should be scaled relative to the default (see the help file for `plot.default`). The default is the current value of the graphics parameter `cex`.

`group.difference.ci`
for the case when there are just 2 groups, a logical scalar indicating whether to display the confidence interval for the difference between group locations. The default is the value of the `p.value` argument. The confidence interval is displayed at the top of the graph in the format [Lower CI, Upper CI].

`group.difference.conf.level`
for the case when there are just 2 groups, a numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the difference between group locations. The default is `conf.level=0.95`.

`group.difference.digits`
for the case when there are just 2 groups, an integer indicating the number of digits to round to when displaying the confidence interval for the difference between group locations. The default value is `group.difference.digits=location.scale.digits`.

`ci.and.test` character string indicating whether confidence intervals and tests should be based on parametric or nonparametric (`ci.and.test="nonparametric"`) methods. When `ci.and.test="parametric"` (the default), confidence intervals for the population mean are based on the one-sample t-test (see `t.test`), and the test of group differences is based on the two-sample t-test if there are two groups and the F-test (i.e., one-way analysis of variance, see `aov`) if there are three or more groups. When `ci.and.test="nonparametric"`, confidence intervals for the population pseudo-median are based on the Wilcoxon signed rank test (see `wilcox.test` and page 56 of Hollander and Wolfe, 1999), and the test of group differences is based on the Wilcoxon rank sum test if there are two groups (see `wilcox.test`) and the Kruskal-Wallis test (see `kruskal.test`) if there are three or more groups.

`ci.arg.list` an optional list of arguments to pass to the function used to compute confidence intervals. The default value is `ci.arg.list=NULL`.

`test.arg.list` an optional list of arguments to pass to the function used to test for group differences in location. The default value is `test.arg.list=NULL`. In particular, in the case when there are two groups, `ci.and.test="parametric"`, and `ci.arg.list` is `NULL` or does not contain a component specifying the value for `var.equal`, this argument is updated to include the component `var.equal=TRUE`, which is not the default behavior of `t.test`.

NOTE: If test.arg.list contains a component named "paired", the value of that component is set to the value of the argument paired (see below).

alternative — character string describing the alternative hypothesis for the test of group differences in the case when there are two groups. Possible values are "two.sided" (the default), "less", and "greater".

plot.diff — applicable only to the case when there are two groups:
logical scalar indicating whether to plot the confidence interval for the difference between the groups. The default is plot.diff=FALSE.

When plot.diff=TRUE and paired=FALSE, the confidence interval for the difference between the two locations is displayed and the right-hand axis (when vertical=TRUE) or top axis (when vertical=FALSE) is labeled in units of the confidence interval for the difference between the two locations. If ci.and.test="parametric", the confidence interval for the difference between the two means is displayed. If ci.and.test="nonparametric", the confidence interval for the median of the difference between a sample from the first group and a sample from the second group is displayed (see the help file for wilcox.test.

When plot.diff=TRUE and paired=TRUE, the paired differences are displayed and the right-hand axis (when vertical=TRUE) or top axis (when vertical=FALSE) is labeled in units of the paired differences. In addition, if show.ci=TRUE, the confidence interval based on the paired differences is displayed. In this case, if ci.and.test="parametric" the confidence interval for the mean of the paired differences is displayed, and if ci.and.test="nonparametric" the confidence interval for the pseudomedian is displayed (see the help file for wilcox.test.

diff.col — applicable only to the case when there are two groups and plot.diff=TRUE:
numeric or character scalar indicating what color to use for the confidence interval for the difference in locations between the two groups. When paired=TRUE, this argument also controls the color of the paired differences. The default is diff.col=col[1].

diff.method — applicable only to the case when there are two groups, plot.diff=TRUE, and paired=TRUE:
the method to be used to separate coincident points for the paired differences. The default value is diff.method="stack". Other options are diff.method="jitter" and diff.method="overplot". See the explanation for the argument method above.

diff.pch — applicable only to the case when there are two groups, plot.diff=TRUE, and paired=TRUE:
numeric or character scalar indicating what plotting symbol to use for the paired differences. The default is diff.pch=pch[1].

paired — applicable only to the case when there are two groups:
logical scalar indicating whether the observations in the first group are paired with those in the second group. The default is paired=FALSE.
NOTE 1: When the formula method for the argument x is used (see above) and the argument paired=TRUE, the data in the vector y must have the same number of observations for each level of the factor g and for each level sorted in the same way according to the pairing variable.

NOTE 2: If the argument `test.arg.list` (see above) contains a component named `"paired"`, the value of that component is set to the value of the argument `paired`.

`paired.lines` applicable only to the case when there are two groups and `paired=TRUE`:
logical scalar indicating whether to join the paired observations with lines. The default value is the value of the argument `paired`.

`paired.lty` applicable only to the case when there are two groups, `paired=TRUE`, and `paired.lines=TRUE`:
numeric vector indicating the line types to use to join the paired observations with lines. The default value is `paired.lty=1:6`.

`paired.lwd` applicable only to the case when there are two groups, `paired=TRUE`, and `paired.lines=TRUE`:
numeric vector indicating the widths of the lines used to join the paired observations with lines. The default value is `paired.lwd=1`.

`paired.pch` applicable only to the case when there are two groups, `paired=TRUE`, and `paired.lines=TRUE`:
numeric vector indicating the plotting characters to use at each end of the lines used to join the paired observations with lines. The default value is `paired.pch=1:14`.

`paired.col` applicable only to the case when there are two groups, `paired=TRUE`, and `paired.lines=TRUE`:
character or numeric vector indicating the colors for the lines (and plotting characters) used to join the paired observations with lines. The default value is `paired.col=NULL`, in which case the vector becomes `c("black", "red", "green3", "blue", "magenta", "darkgreen", "purple", "orange", "darkolivegreen", "steelblue", "darkgray")`.

`diff.name` applicable only to the case when there are two groups and `plot.diff=TRUE`:
character scalar indicating the label to use for the confidence interval for the difference between groups. For the case when `paired=TRUE`, this label also describes the paired differences. The default value is `diff.name=NULL`, in which case the label is "group 2 - group 1", where group 1 and group 2 denote the names for the each group. For example, if group 1 is labeled "A" and group 2 is labeled "B", then the default value is `diff.name="B-A"`.

`diff.name.cex` applicable only to the case when there are two groups and `plot.diff=TRUE`:
numeric scalar indicating the amount by which the label for group differences should be scaled relative to the default (see the help file for `plot.default`). The default value is `diff.name.cex=group.names.cex`.

`sep.line` applicable only to the case when there are two groups and `plot.diff=TRUE`:
logical scalar indicating whether to draw a line between the strip charts for the two groups and the confidence interval for the difference between the two groups (and paired differences when `paired=TRUE`). The default value is `sep.line=TRUE`.

`sep.lty` applicable only to the case when there are two groups, `plot.diff=TRUE`, and `sep.line=TRUE`:
numeric scalar indicating the line type to use for the line drawn between the strip charts for the two groups and the confidence interval for the difference between the two groups. The default value is `sep.lty=2`.

`sep.lwd` applicable only to the case when there are two groups, `plot.diff=TRUE`, and `sep.line=TRUE`:
numeric scalar indicating the line width to use for the line drawn between the strip charts for the two groups and the confidence interval for the difference between the two groups. The default value is the current value of the graphics parameter `cex`.

`sep.col` applicable only to the case when there are two groups, `plot.diff=TRUE`, and `sep.line=TRUE`:
numeric or character scalar indicating the color of the line drawn between the strip charts for the two groups and the confidence interval for the difference between the two groups. The default value is `sep.col="gray"`.

`diff.lim` applicable only to the case when there are two groups and `plot.diff=TRUE`:
numeric vector of length 2 indicating the limits to use for the axis associated with the confidence interval for the difference between the two groups. When `paired=FALSE`, the default value is the range of the y-axis, but centered at the mean of the confidence interval for the difference in locations. When `paired=TRUE`, the default value is `range(pretty(c(X, range(CI))))` where `X` denotes the vector containing the paired differences.

`diff.at` applicable only to the case when there are two groups and `plot.diff=TRUE`:
numeric vector indicating the locations of the tick marks for the axis associated with the confidence interval for the difference between groups (see the explanation for the argument `at` in the help file for `axis`). The default value is `diff.at=NULL`, in which case default values are used for the locations of the tick marks.

`diff.axis.label`
applicable only to the case when there are two groups and `plot.diff=TRUE`:
character string indicating the label to use for the axis associated with the confidence interval for the difference between groups. When `paired=FALSE` the default value is `"Difference Between Groups"`, and when `paired=TRUE` the default value is `"Paired Difference"`.

`plot.diff.mar` applicable only to the case when there are two groups, `plot.diff=TRUE`, and `add=FALSE`:
numeric vector of length 4 indicating the number of lines in the plotting margins (see the explanation for the argument `mar` in the help file for `par`). The default value is `plot.diff.mar = c(5, 4, 4, 4) + 0.1`.

Value

`stripChart` invisibly returns a list with the following components:

`group.centers` numeric vector of values on the group axis (the x-axis unless `vertical=FALSE`) indicating the centers of the groups.

`group.stats` a matrix with the number of rows equal to the number of groups and six columns indicating the sample size of the group (N), the estimate of the group location parameter (Mean or Median), the estimate of the group scale (SD or IQR), the lower confidence limit for the group location parameter (LCL), the upper confidence limit for the group location parameter (UCL), and the confidence level associated with the confidence interval (Conf.Level)

In addition, if the argument `p.value=TRUE` and/or 1) there are two groups and 2) `plot.diff=TRUE`, the list also includes these components:

`group.difference.p.value`
: numeric scalar indicating the p-value associated with the test of equal group locations.

`group.difference.conf.int`
: numeric vector of two elements indicating the confidence interval for the difference between the group locations. Only present when there are two groups.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Hollander, M., and D.A. Wolfe. (1999). *Nonparametric Statistical Methods*. Second Edition. John Wiley and Sons, New York.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`stripchart`, `t.test`, `wilcox.test`, `aov`, `kruskal.test`, `t.test`.

Examples

```
#------------------------
# Two Independent Samples
#------------------------

# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area.  These data are strored
# in the data frame EPA.94b.tccb.df.
#
# First create one-dimensional scatterplots to compare the
# TcCB concentrations between the areas and use a nonparametric
# test to test for a difference between areas.

dev.new()
stripChart(TcCB ~ Area, data = EPA.94b.tccb.df, col = c("red", "blue"),
  p.value = TRUE, ci.and.test = "nonparametric",
  ylab = "TcCB (ppb)")

#----------

# Now log-transform the TcCB data and use a parametric test
# to compare the areas.
```

```
dev.new()
stripChart(log10(TcCB) ~ Area, data = EPA.94b.tccb.df, col = c("red", "blue"),
  p.value = TRUE, ylab = "log10 [ TcCB (ppb) ]")

#----------

# Repeat the above procedure, but also plot the confidence interval
# for the difference between the means.

dev.new()
stripChart(log10(TcCB) ~ Area, data = EPA.94b.tccb.df, col = c("red", "blue"),
  p.value = TRUE, plot.diff = TRUE, diff.col = "black",
  ylab = "log10 [ TcCB (ppb) ]")

#----------

# Repeat the above procedure, but allow the variances to differ.

dev.new()
stripChart(log10(TcCB) ~ Area, data = EPA.94b.tccb.df, col = c("red", "blue"),
  p.value = TRUE, plot.diff = TRUE, diff.col = "black",
  ylab = "log10 [ TcCB (ppb) ]", test.arg.list = list(var.equal = FALSE))

#----------

# Repeat the above procedure, but jitter the points instead of
# stacking them.

dev.new()
stripChart(log10(TcCB) ~ Area, data = EPA.94b.tccb.df, col = c("red", "blue"),
  p.value = TRUE, plot.diff = TRUE, diff.col = "black",
  ylab = "log10 [ TcCB (ppb) ]", test.arg.list = list(var.equal = FALSE),
  method = "jitter", ci.offset = 4)

#----------

# Clean up
#---------
graphics.off()

#====================

#--------------------
# Paired Observations
#--------------------

# The data frame ACE.13.TCE.df contians paired observations of
# trichloroethylene (TCE; mg/L) at 10 groundwater monitoring wells
# before and after remediation.
#
# Create one-dimensional scatterplots to compare TCE concentrations
# before and after remediation and use a paired t-test to
```

```
# test for a difference between periods.

ACE.13.TCE.df
#   TCE.mg.per.L Well Period
#1        20.900    1 Before
#2         9.170    2 Before
#3         5.960    3 Before
#...      ......   .. ......
#18        0.520    8  After
#19        3.060    9  After
#20        1.900   10  After

dev.new()
stripChart(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  col = c("brown", "green"), p.value = TRUE, paired = TRUE,
  ylab = "TCE (mg/L)")

#----------

# Repeat the above procedure, but also plot the confidence interval
# for the mean of the paired differences.

dev.new()
stripChart(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  col = c("brown", "green"), p.value = TRUE, paired = TRUE,
  ylab = "TCE (mg/L)", plot.diff = TRUE, diff.col = "blue")


#==========

# Repeat the last two examples, but use a one-sided alternative since
# remediation should decrease TCE concentration.

dev.new()
stripChart(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  col = c("brown", "green"), p.value = TRUE, paired = TRUE,
  ylab = "TCE (mg/L)", alternative = "less",
  group.difference.digits = 2)

#----------

# Repeat the above procedure, but also plot the confidence interval
# for the mean of the paired differences.
#
# NOTE: Although stripChart can *report* one-sided confidence intervals
#       for the difference between two groups (see above example),
#       when *plotting* the confidence interval for the difference,
#       only two-sided CIs are allowed.
#       Here, we will set the confidence level of the confidence
#       interval for the mean of the paired differences to 90%,
#       so that the upper bound of the CI corresponds to the upper
#       bound of a 95% one-sided CI.
```

```
dev.new()
stripChart(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  col = c("brown", "green"), p.value = TRUE, paired = TRUE,
  ylab = "TCE (mg/L)", group.difference.digits = 2,
  plot.diff = TRUE, diff.col = "blue", group.difference.conf.level = 0.9)

#----------

# Clean up
#---------
graphics.off()

#==========

# The data frame Helsel.Hirsch.02.Mayfly.df contains paired counts
# of mayfly nymphs above and below industrial outfalls in 12 streams.
#
# Create one-dimensional scatterplots to compare the
# counts between locations and use a nonparametric test
# to compare counts above and below the outfalls.

Helsel.Hirsch.02.Mayfly.df
#   Mayfly.Count Stream Location
#1            12      1    Above
#2            15      2    Above
#3            11      3    Above
#...         ...     ..    .....
#22           60     10    Below
#23           53     11    Below
#24          124     12    Below

dev.new()
stripChart(Mayfly.Count ~ Location, data = Helsel.Hirsch.02.Mayfly.df,
  col = c("green", "brown"), p.value = TRUE, paired = TRUE,
  ci.and.test = "nonparametric", ylab = "Number of Mayfly Nymphs")

#----------

# Repeat the above procedure, but also plot the confidence interval
# for the pseudomedian of the paired differences.

dev.new()
stripChart(Mayfly.Count ~ Location, data = Helsel.Hirsch.02.Mayfly.df,
  col = c("green", "brown"), p.value = TRUE, paired = TRUE,
  ci.and.test = "nonparametric", ylab = "Number of Mayfly Nymphs",
  plot.diff = TRUE, diff.col = "blue")

#----------

# Clean up
#---------
graphics.off()
```

summaryFull *Full Complement of Summary Statistics*

Description

summaryFull is a generic function used to produce a full complement of summary statistics. The function invokes particular methods which depend on the class of the first argument. The summary statistics include: sample size, number of missing values, mean, median, trimmed mean, geometric mean, skew, kurtosis, min, max, range, 1st quartile, 3rd quartile, standard deviation, geometric standard deviation, interquartile range, median absolute deviation, and coefficient of variation.

Usage

```
summaryFull(object, ...)

## S3 method for class 'formula'
summaryFull(object, data = NULL, subset,
  na.action = na.pass, ...)

## Default S3 method:
summaryFull(object, group = NULL,
    combine.groups = FALSE, drop.unused.levels = TRUE,
    rm.group.na = TRUE, stats = NULL, trim = 0.1,
    sd.method = "sqrt.unbiased", geo.sd.method = "sqrt.unbiased",
    skew.list = list(), kurtosis.list = list(),
    cv.list = list(), digits = max(3, getOption("digits") - 3),
    digit.type = "signif", stats.in.rows = TRUE,
    drop0trailing = TRUE, data.name = deparse(substitute(object)),
    ...)

## S3 method for class 'data.frame'
summaryFull(object, ...)

## S3 method for class 'matrix'
summaryFull(object, ...)

## S3 method for class 'list'
summaryFull(object, ...)
```

Arguments

object	an object for which summary statistics are desired. In the default method, the argument object must be a numeric vector, a data frame, a matrix, or a list. When object is a data frame, all columns must be numeric. When object is a matrix, it must be a numeric matrix. When object is a list, all components must be numeric vectors. In the formula method, a symbolic specification of the form y ~ g can be given, indicating the observations in the vector y are to be grouped

according to the levels of the factor g (the form y ~ 1 indicates no grouping). NAs are allowed in the data.

data — when object is a formula, data specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which summaryFull is called.

subset — when object is a formula, subset specifies an optional vector specifying a subset of observations to be used.

na.action — when object is a formula, na.action specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.

group — when object is a numeric vector, group is a factor or character vector indicating which group each observation belongs to. When object is a matrix or data frame this argument is ignored and the columns define the groups. When object is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.

combine.groups — logical scalar indicating whether to show summary statistics for all groups combined. The default value is FALSE.

drop.unused.levels — when drop.unused.levels=TRUE, groups with no observations are dropped.

rm.group.na — logical scalar indicating whether to remove missing values from the group argument. By default rm.group.na=TRUE.

stats — character vector indicating which statistics to compute. Possible elements of the character vector include: "all" (indicating to include all summary statistics), "for.non.pos" (only compute statistics that are meaningful for datasets with non-positive values), "n" (number of non-missing values), "n.miss" (number of missing values), "mean", "median", "trimmed.mean", "geo.mean", "skew", "kurtosis", "min", "max", "range", "1st.quart", "3rd.quart", "sd", "geo.sd", "iqr", "mad", "cv". The default value is stats="for.non.pos" when object contains non-positive values (i.e., values ≤ 0), and stats="all" when object contains only positive values.

trim — fraction (between 0 and 0.5 inclusive) of values to be trimmed from each end of the ordered data to compute the trimmed mean. The default value is trim=0.1. If trim=0.5, this yields the median.

sd.method — character string specifying what method to use to compute the sample standard deviation. The possible values are "sqrt.ubiased" (the square root of the unbiased estimate of variance; the default), or "moments" (the method of moments estimator).

geo.sd.method — character string specifying what method to use to compute the sample standard deviation of the log-transformed observations prior to exponentiating this quantity. The possible values are "sqrt.ubiased" (the square root of the unbiased estimate of variance; the default), or "moments" (the method of moments estimator). See the help file for geoSD for more information.

skew.list	list of arguments to supply to the skewness function. See the help file for skewness for more information. The default value is skew.list=list(), which results in using the default arguments to skewness.
kurtosis.list	list of arguments to supply to the kurtosis function. See the help file for kurtosis for more information. The default value is kurtosis.list=list(), which results in using the default arguments to kurtosis.
cv.list	list of arguments to supply to the cv function. See the help file for cv for more information. The default value is cv.list=list(), which results in using the default arguments to cv.
digits	integer indicating the number of digits to use for the summary statistics. When digit.type="signif", digits indicates the number of significant digits. When digit.type="round", digits indicates the number of decimal places to round to. The default value is max(3, getOption("digits") - 3), that is, the maximum of 3 versus the current setting of the "digits" component of .Options minus 3.
digit.type	character string indicating whether the digits argument refers to significant digits (digit.type="signif", the default), or how many decimal places to round to (digit.type="round").
stats.in.rows	logical scalar indicating whether to show the summary statistics in the rows or columns of the output. The default is stats.in.rows=TRUE.
drop0trailing	logical scalar indicating whether to drop trailing 0's when printing the summary statistics. The value of this argument is added as an attribute to the returned list and is used by the print.summaryStats function. The default value is TRUE.
data.name	character string indicating the name of the data used for the summary statistics.
...	additional arguments affecting the summary statistics produced.

Details

The function summaryFull returns summary statistics that are useful to describe various characteristics of one or more variables. It is an extended version of the built-in R function summary specifically for non-factor numeric data. The table below shows what statistics are computed and what functions are called by summaryFull to compute these statistics.

The object returned by summaryFull is useful for printing or report purposes. You may also use the functions that summaryFull calls (see table below) to compute summary statistics to be used by other functions.

See the help files for the functions listed in the table below for more information on these summary statistics.

Summary Statistic	Function Used
Mean	mean
Median	median
Trimmed Mean	mean with trim argument
Geometric Mean	geoMean
Skew	skewness
Kurtosis	kurtosis

Min	min
Max	max
Range	range and diff
1st Quartile	quantile
3rd Quartile	quantile
Standard Deviation	sd
Geometric Standard Deviation	geoSD
Interquartile Range	iqr
Median Absolute Deviation	mad
Coefficient of Variation	cv

Value

an object of class "summaryStats" (see summaryStats.object. Objects of class "summaryStats" are numeric matrices that contain the summary statisics produced by a call to summaryStats or summaryFull. These objects have a special printing method that by default removes trailing zeros for sample size entries and prints blanks for statistics that are normally displayed as NA (see print.summaryStats).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, NY.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY.

Leidel, N.A., K.A. Busch, and J.R. Lynch. (1977). *Occupational Exposure Sampling Strategy Manual*. U.S. Department of Health, Education, and Welfare, Public Health Service, Center for Disease Control, National Institute for Occupational Safety and Health, Cincinnati, Ohio 45226, January, 1977, pp.102-103.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Ott, W.R. (1995). *Environmental Statistics and Data Analysis*. Lewis Publishers, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis, Fifth Edition*. Prentice-Hall, Upper Saddle River, NJ.

See Also

summary, summaryStats.

Examples

```
# Generate 20 observations from a lognormal distribution with
# parameters mean=10 and cv=1, and compute the summary statistics.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)

dat <- rlnormAlt(20, mean=10, cv=1)

summary(dat)
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#2.608   4.995   6.235   7.490   9.295  15.440

summaryFull(dat)
#                             dat
#N                            20
#Mean                          7.49
#Median                        6.235
#10% Trimmed Mean              7.125
#Geometric Mean                6.674
#Skew                          0.9877
#Kurtosis                     -0.03539
#Min                           2.608
#Max                          15.44
#Range                        12.83
#1st Quartile                  4.995
#3rd Quartile                  9.295
#Standard Deviation            3.803
#Geometric Standard Deviation  1.634
#Interquartile Range           4.3
#Median Absolute Deviation     2.607
#Coefficient of Variation      0.5078

#----------

# Compare summary statistics for normal and lognormal data:
log.dat <- log(dat)

summaryFull(list(dat = dat, log.dat = log.dat))
#                             dat      log.dat
#N                            20       20
#Mean                          7.49     1.898
#Median                        6.235    1.83
#10% Trimmed Mean              7.125    1.902
#Geometric Mean                6.674    1.835
#Skew                          0.9877   0.1319
#Kurtosis                     -0.03539 -0.4288
#Min                           2.608    0.9587
#Max                          15.44     2.737
#Range                        12.83     1.778
#1st Quartile                  4.995    1.607
```

```
#3rd Quartile                   9.295    2.227
#Standard Deviation             3.803    0.4913
#Geometric Standard Deviation   1.634    1.315
#Interquartile Range            4.3      0.62
#Median Absolute Deviation      2.607    0.4915
#Coefficient of Variation       0.5078   0.2588

# Clean up
rm(dat, log.dat)

#--------------------------------------------------------------------

# Compute summary statistics for 10 observations from a normal
# distribution with parameters mean=0 and sd=1.  Note that the
# geometric mean and geometric standard deviation are not computed
# since some of the observations are non-positive.

set.seed(287)

dat <- rnorm(10)

summaryFull(dat)
#                           dat
#N                          10
#Mean                        0.07406
#Median                      0.1095
#10% Trimmed Mean            0.1051
#Skew                       -0.1646
#Kurtosis                   -0.7135
#Min                        -1.549
#Max                         1.449
#Range                       2.998
#1st Quartile               -0.5834
#3rd Quartile                0.6966
#Standard Deviation          0.9412
#Interquartile Range         1.28
#Median Absolute Deviation   1.05

# Clean up
rm(dat)

#--------------------------------------------------------------------

# Compute summary statistics for the TcCB data given in USEPA (1994b)
# (the data are stored in EPA.94b.tccb.df).  Arbitrarily set the one
# censored observation to the censoring level. Group by the variable
# Area.

summaryFull(TcCB ~ Area, data = EPA.94b.tccb.df)
#                              Cleanup  Reference
#N                              77       47
#Mean                            3.915    0.5985
#Median                          0.43     0.54
```

```
#10% Trimmed Mean              0.6846   0.5728
#Geometric Mean                0.5784   0.5382
#Skew                          7.717    0.9019
#Kurtosis                     62.67     0.132
#Min                           0.09     0.22
#Max                         168.6      1.33
#Range                       168.5      1.11
#1st Quartile                  0.23     0.39
#3rd Quartile                  1.1      0.75
#Standard Deviation           20.02     0.2836
#Geometric Standard Deviation  3.898    1.597
#Interquartile Range           0.87     0.36
#Median Absolute Deviation     0.3558   0.2669
#Coefficient of Variation      5.112    0.4739
```

---

summaryStats *Summary Statistics*

Description

summaryStats is a generic function used to produce summary statistics, confidence intervals, and results of hypothesis tests. The function invokes particular methods which depend on the class of the first argument.

The summary statistics include: sample size, number of missing values, mean, standard deviation, median, min, and max. Optional additional summary statistics include 1st quartile, 3rd quartile, and stadard error.

Usage

```
summaryStats(object, ...)

## S3 method for class 'formula'
summaryStats(object, data = NULL, subset,
  na.action = na.pass, ...)

## Default S3 method:
summaryStats(object, group = NULL,
    drop.unused.levels = TRUE, se = FALSE, quartiles = FALSE,
    digits = max(3, getOption("digits") - 3),
    digit.type = "round", drop0trailing = TRUE,
    show.na = TRUE, show.0.na = FALSE, p.value = FALSE,
    p.value.digits = 2, p.value.digit.type = "signif",
    test = "parametric", paired = FALSE, test.arg.list = NULL,
    combine.groups = p.value, rm.group.na = TRUE,
    group.p.value.type = NULL, alternative = "two.sided",
    ci = NULL, ci.between = NULL, conf.level = 0.95,
    stats.in.rows = FALSE,
    data.name = deparse(substitute(object)), ...)
```

```
## S3 method for class 'factor'
summaryStats(object, group = NULL,
    drop.unused.levels = TRUE,
    digits = max(3, getOption("digits") - 3),
    digit.type = "round", drop0trailing = TRUE,
    show.na = TRUE, show.0.na = FALSE, p.value = FALSE,
    p.value.digits = 2, p.value.digit.type = "signif",
    test = "chisq", test.arg.list = NULL, combine.levels = TRUE,
    combine.groups = FALSE, rm.group.na = TRUE,
    ci = p.value & test != "chisq", conf.level = 0.95,
    stats.in.rows = FALSE, ...)

## S3 method for class 'character'
summaryStats(object, ...)

## S3 method for class 'logical'
summaryStats(object, ...)

## S3 method for class 'data.frame'
summaryStats(object, ...)

## S3 method for class 'matrix'
summaryStats(object, ...)

## S3 method for class 'list'
summaryStats(object, ...)
```

Arguments

object	an object for which summary statistics are desired. In the default method, the argument object can be a numeric vector, factor, character vector, logical vector, data frame, matrix, or list. When object is a character or logical vector, it is coerced to be a factor. When object is a data frame, all columns must be numeric or all columns must be factors. When object is a matrix, it must be a numeric or character matrix. When object is a list, all components must be numeric vectors or all components must be factors. In the formula method, a symbolic specification of the form y ~ g can be given, indicating the observations in the vector y are to be grouped according to the levels of the factor g (the form y ~ 1 indicates no grouping). NAs are allowed in the data.
data	when object is a formula, data specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which summaryStats is called.
subset	when object is a formula, subset specifies an optional vector specifying a subset of observations to be used.

`na.action` when `object` is a formula, `na.action` specifies a function which indicates what should happen when the data contain NAs. The default is `na.pass`.

`group` when `object` is a numeric vector or factor, `group` is a factor or character vector indicating which group each observation belongs to. When `object` is a matrix or data frame this argument is ignored and the columns define the groups. When `object` is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.

`drop.unused.levels` when `drop.unused.levels=TRUE`, groups with no observations are dropped.

`se` for numeric data, logical scalar indicating whether to include the standard error of the mean in the summary statistics. The default value is `se=FALSE`.

`quartiles` for numeric data, logical scalar indicating whether to include the estimated 25th and 75th percentiles in the summary statistics. The default value is `quartiles=FALSE`.

`digits` integer indicating the number of digits to use for the summary statistics. When `digit.type="signif"`, `digits` indicates the number of significant digits. When `digit.type="round"`, `digits` indicates the number of decimal places to round to. The default value is `max(3, getOption("digits") - 3)`, that is, the maximum of 3 versus the current setting of the `"digits"` component of `.Options` minus 3.

`digit.type` character string indicating whether the `digits` argument refers to significant digits (`digit.type="signif"`), or how many decimal places to round to (`digit.type="round"`, the default).

`drop0trailing` logical scalar indicating whether to drop trailing 0's when printing the summary statistics. The value of this argument is added as an attribute to the returned list and is used by the `print.summaryStats` function. The default value is `TRUE`.

`show.na` logical scalar indicating whether to return the number of missing values. The default value is `show.na=TRUE`.

`show.0.na` logical scalar indicating whether to diplay the number of missing values in the case when there are no missing values. The default value is `show.0.na=FALSE`.

`p.value` logical scalar indicating whether to return the p-value associated with a test of hypothesis. The default value is `p.value=FALSE`. **Numeric data:** if there are no groups the p-value is associated with the t-test to test whether the mean is different from 0; if there are groups see the explanation for the argument `group.p.value.type` below. **Factors:** the p-value is associated with the test specified by the argument `test` (see below).

`p.value.digits` integer indicating the number of digits to use for the p-value. When `p.value.digit.type="signif"`, `p.value.digits` indicates the number of significant digits. When `p.value.digit.type="round"`, `p.value.digits` indicates the number of decimal places to round to. The default value is `p.value.digits=2`.

`p.value.digit.type` character string indicating whether the `p.value.digits` argument refers to significant digits (`p.value.digit.type="signif"`, the default), or how many decimal places to round to (`p.value.digit.type="round"`).

`test` **Numeric data:** character string indicating whether to compute p-values and confidence intervals based on parametric (`test="parametric"`; the default) or nonparametric (`test="nonparametric"`) tests when `p.value=TRUE` and/or `ci=TRUE`. When `test="parametric"`, confidence intervals are based on the t-test (see `t.test`) and p-values are based on the t-test or F-test (see `anova.lm`). When `test="nonparametric"`, confidence intervals are based on the Wilcoxon rank sum test (see `wilcox.test`) and p-values are based on the Wilcoxon rank sum test or the Kruskal-Wallis rank sum test (see `kruskal.test`).

Factors: character string indicating which test to perform when `p.value=TRUE`. Possible values are `test="chisq"` for the chi-squared test as performed by the function `chisq.test` (the default), `test="prop"` for the chi-squared test as performed by the function `prop.test`, `test="fisher"` for Fisher's exact test as performed by the function `fisher.test`, and `test="binom"` for the one-sample exact binomial test as performed by `binom.test`. The chi-squared test as performed by `prop.test` is only available when the number of levels in `object` is 2 and either `group` is not supplied or the number of levels in `group` is 2. Fisher's exact test is only available when the number of levels in `group` is ≥ 2. The exact binomial test is only available when `group` is not supplied and the number of levels in `object` is 2.

`paired` applicable only to the case when there are two groups:
logical scalar indicating whether the observations in the first group are paired with those in the second group. The default is `paired=FALSE`. NOTE: If the argument `test.arg.list` (see below) contains a component named `paired`, the value of that component is set to the value of the argument `paired`.

`test.arg.list` a list with additional arguments to pass to the test used to compute p-values and confidence intervals. For numeric data, when `test="parametric"`, `p.value=TRUE`, `group.p.value.type="between"` and there are two groups, if this argument is `NULL` or does not contain a component named `var.equal`, it will be modified to contain the component `var.equal=TRUE`. Note that this overrides the default behavior of `t.test` when there are two groups.
NOTE: If `test.arg.list` contains a component named `paired`, the value of that component is set to the value of the argument `paired` (see above).

`combine.groups` logical scalar indicating whether to show summary statistics for all groups combined. **Numeric data:** the default value is `TRUE` if `p.value=TRUE`, otherwise `FALSE`. **Factors:** the default value is `FALSE`.

`rm.group.na` logical scalar indicating whether to remove missing values from the `group` argument. If `rm.group.na=FALSE` and `group` contains missing values then an error is returned. If `rm.group.na=TRUE` and `group` contains missing values then a warning is issued. By default `rm.group.na=TRUE`.

`group.p.value.type`
for numeric data, character string indicating which p-value(s) to compute when there is more than one group. When `group.p.value.type="between"` (the default when `combine.groups=TRUE`), the p-value is associated with the two-sample t-test (or the Wilcoxon rank sum test) in the case of two groups, and the one-way analysis of variance F-test (or Krukal-Wallis test) in the case of

three or more groups to test whether the group means (locations) are different from each other. When `group.p.value.type="within"` (the default when `combine.groups=FALSE`), the computed p-values for each group are associated with the one-sample t-test (or Wilcox signed rank test) to test whether the group mean (location) is different from 0.

`alternative` for numeric data, character string indicating which alternative to assume for p-values and confidence intervals. Possible values are `"two.sided"` (the default), `"less"`, and `"greater"`. This argument is ignored for p-values in the case of three or more groups when `group.p.value.type="between"`, and is ignored for confidence intervals in the case of three or more groups when `ci.between=TRUE`.

`ci` **Numeric data:** logical scalar indicating whether to compute a confidence interval for the mean or each group mean. The default value is `FALSE` unless `p.value=TRUE` and there are no groups, or when `p.value=TRUE` and there are groups and `group.p.value.type="within"`.

Factors: logical scalar indicating whether to compute a confidence interval. A confidence interval is computed only if the number of levels in `object` is 2. When `group` is not supplied, if `ci=TRUE` and `test="prop"` or `test="binom"`, a confidence interval for the *percent* (not probability) of the first level of `object` is computed. When `group` is supplied and the number of levels in `group` is 2, if `ci=TRUE` and `test="prop"`, a confidence interval for the difference between *percents* (not proportions) is computed, and if `test="fisher"` a confidence interval for the odds ratio is computed.

`ci.between` for numeric data, logical scalar indicating whether to compute a confidence interval for the difference between group means when there are two groups. The default value is `ci.between=TRUE` when `p.value=TRUE` and `group.p.value.type="between"`, otherwise this argument is ignored.

`conf.level` numeric scalar between 0 and 1 indicating the confidence level associated with the confidence intervals. The default value is `conf.level=0.95`.

`stats.in.rows` logical scalar indicating whether to show the summary statistics in the rows or columns of the output. The default is `stats.in.rows=FALSE`.

`data.name` character string indicating the name of the data used for the summary statistics.

`combine.levels` for factors, a logical scalar indicating whether to compute summary statistics based on combining all levels of a factor.

`...` additional arguments affecting the summary statistics produced.

Value

an object of class `"summaryStats"` (see `summaryStats.object`. Objects of class `"summaryStats"` are numeric matrices that contain the summary statisics produced by a call to `summaryStats` or `summaryFull`. These objects have a special printing method that by default removes trailing zeros for sample size entries and prints blanks for statistics that are normally displayed as `NA` (see `print.summaryStats`).

Summary statistics for numeric data include sample size, mean, standard deviation, median, min, and max. Options include the standard error of the mean (when `se=TRUE`), the estimated quartiles (when `quartiles=TRUE`), p-values (when `p.value=TRUE`), and/or confidence intervals (when `ci=TRUE` and/or `ci.between=TRUE`).

Summary statistics for factors include the sample size for each level of the factor and the percent of the total for that level. Options include a p-value (when `p.value=TRUE`).

Note that unlike the R function `summary` and the **EnvStats** function `summaryFull`, by default the `digits` argument for the **EnvStats** function `summaryStats` refers to how many decimal places to round to, not how many significant digits to use (see the explanation of the argument `digit.type` above).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers, Second Edition*. Lewis Publishers, Boca Raton, FL.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, Chapter 24.

See Also

`summary`, `summaryFull`, `t.test`, `anova.lm`, `wilcox.test`, `kruskal.test`, `chisq.test`, `fisher.test`, `binom.test`.

Examples

```
# The guidance document USEPA (1994b, pp. 6.22--6.25)
# contains measures of 1,2,3,4-Tetrachlorobenzene (TcCB)
# concentrations (in parts per billion) from soil samples
# at a Reference area and a Cleanup area. These data are strored
# in the data frame EPA.94b.tccb.df.

#----------
# First, create summary statistics by area based on the log-transformed data.

summaryStats(log10(TcCB) ~ Area, data = EPA.94b.tccb.df)
#           N    Mean     SD  Median     Min    Max
#Cleanup   77 -0.2377 0.5908 -0.3665 -1.0458 2.2270
#Reference 47 -0.2691 0.2032 -0.2676 -0.6576 0.1239

#----------
# Now create summary statistics by area based on the log-transformed data
# and use the t-test to compare the areas.

summaryStats(log10(TcCB) ~ Area, data = EPA.94b.tccb.df, p.value = TRUE)

summaryStats(log10(TcCB) ~ Area, data = EPA.94b.tccb.df,
  p.value = TRUE, stats.in.rows = TRUE)
#                Cleanup  Reference Combined
```

```
#N                77       47       124
#Mean             -0.2377  -0.2691   -0.2496
#SD                0.5908   0.2032    0.481
#Median           -0.3665  -0.2676   -0.3143
#Min              -1.0458  -0.6576   -1.0458
#Max               2.227    0.1239    2.227
#Diff                                -0.0313
#p.value.between                      0.73
#95%.LCL.between                     -0.2082
#95%.UCL.between                      0.1456

#==========================================================================

# Page 9-3 of USEPA (2009) lists trichloroethene
# concentrations (TCE; mg/L) collected from groundwater at two wells.
# Here, the seven non-detects have been set to their detection limit.

#----------
# First, compute summary statistics for all TCE observations.

summaryStats(TCE.mg.per.L ~ 1, data = EPA.09.Table.9.1.TCE.df,
  digits = 3, data.name = "TCE")
#     N Mean    SD Median   Min  Max NA's N.Total
#TCE 27 0.09 0.064    0.1 0.004 0.25    3      30

summaryStats(TCE.mg.per.L ~ 1, data = EPA.09.Table.9.1.TCE.df,
  se = TRUE, quartiles = TRUE, digits = 3, data.name = "TCE")
#     N Mean    SD    SE Median   Min  Max 1st Qu. 3rd Qu. NA's N.Total
#TCE 27 0.09 0.064 0.012    0.1 0.004 0.25   0.031    0.12    3      30

#----------
# Now compute summary statistics by well.

summaryStats(TCE.mg.per.L ~ Well, data = EPA.09.Table.9.1.TCE.df,
  digits = 3)
#        N  Mean    SD Median   Min  Max NA's N.Total
#Well.1 14 0.063 0.079  0.031 0.004 0.25    1      15
#Well.2 13 0.118 0.020  0.110 0.099 0.17    2      15

summaryStats(TCE.mg.per.L ~ Well, data = EPA.09.Table.9.1.TCE.df,
  digits = 3, stats.in.rows = TRUE)
#        Well.1 Well.2
#N       14     13
#Mean     0.063  0.118
#SD       0.079  0.02
#Median   0.031  0.11
#Min      0.004  0.099
#Max      0.25   0.17
#NA's     1      2
#N.Total 15     15

# If you want to keep trailing 0's, use the drop0trailing argument:
summaryStats(TCE.mg.per.L ~ Well, data = EPA.09.Table.9.1.TCE.df,
```

```
  digits = 3, stats.in.rows = TRUE, drop0trailing = FALSE)
#        Well.1 Well.2
#N       14.000 13.000
#Mean     0.063  0.118
#SD       0.079  0.020
#Median   0.031  0.110
#Min      0.004  0.099
#Max      0.250  0.170
#NA's     1.000  2.000
#N.Total 15.000 15.000

#====================================================================

# Page 13-3 of USEPA (2009) lists iron concentrations (ppm) in
# groundwater collected from 6 wells.

#----------
# First, compute summary statistics for each well.

summaryStats(Iron.ppm ~ Well, data = EPA.09.Ex.13.1.iron.df,
  combine.groups = FALSE, digits = 2, stats.in.rows = TRUE)
#       Well.1 Well.2 Well.3 Well.4 Well.5 Well.6
#N        4      4      4      4      4      4
#Mean    47.01  55.73  90.86  70.43 145.24 156.32
#SD      12.4   20.34  59.35  25.95  92.16  51.2
#Median  50.05  57.05  76.73  76.95 137.66 171.93
#Min     29.96  32.14  39.25  34.12  60.95  83.1
#Max     57.97  76.71 170.72  93.69 244.69 198.34

#----------
# Note the large differences in standard deviations between wells.
# Compute summary statistics for log(Iron), by Well.

summaryStats(log(Iron.ppm) ~ Well, data = EPA.09.Ex.13.1.iron.df,
  combine.groups = FALSE, digits = 2, stats.in.rows = TRUE)
#       Well.1 Well.2 Well.3 Well.4 Well.5 Well.6
#N      4      4      4      4      4      4
#Mean   3.82   3.97   4.35   4.19   4.8    5
#SD     0.3    0.4    0.66   0.45   0.7    0.4
#Median 3.91   4.02   4.29   4.34   4.8    5.14
#Min    3.4    3.47   3.67   3.53   4.11   4.42
#Max    4.06   4.34   5.14   4.54   5.5    5.29

#----------
# Include confidence intervals for the mean log(Fe) concentration
# at each well, and also the p-value from the one-way
# analysis of variance to test for a difference in well means.

summaryStats(log(Iron.ppm) ~ Well, data = EPA.09.Ex.13.1.iron.df,
  digits = 1, ci = TRUE, p.value = TRUE, stats.in.rows = TRUE)
#                Well.1 Well.2 Well.3 Well.4 Well.5 Well.6 Combined
#N                4      4      4      4      4      4     24
#Mean             3.8    4      4.3    4.2    4.8    5      4.4
```

```
#SD               0.3    0.4    0.7    0.5    0.7    0.4    0.6
#Median           3.9    4      4.3    4.3    4.8    5.1    4.3
#Min              3.4    3.5    3.7    3.5    4.1    4.4    3.4
#Max              4.1    4.3    5.1    4.5    5.5    5.3    5.5
#95%.LCL          3.3    3.3    3.3    3.5    3.7    4.4    4.1
#95%.UCL          4.3    4.6    5.4    4.9    5.9    5.6    4.6
#p.value.between                                            0.025

#=====================================================================

# Using the built-in dataset HairEyeColor, summarize the frequencies
# of hair color and test whether there is a difference in proportions.
# NOTE:  The data that was originally factor data has already been
#        collapsed into frequency counts by catetory in the object
#        HairEyeColor.  In the examples in this section, we recreate
#        the factor objects in order to show how summaryStats works
#        for factor objects.

Hair <- apply(HairEyeColor, 1, sum)
Hair
#Black Brown   Red Blond
#  108   286    71   127

Hair.color <- names(Hair)
Hair.fac <- factor(rep(Hair.color, times = Hair),
  levels = Hair.color)

#----------

# Compute summary statistics and perform the chi-square test
# for equal proportions of hair color

summaryStats(Hair.fac, digits = 1, p.value = TRUE)
#           N   Pct ChiSq_p
#Black    108  18.2
#Brown    286  48.3
#Red       71  12.0
#Blond    127  21.5
#Combined 592 100.0 2.5e-39

#----------
# Now test the hypothesis that 10% of the population from which
# this sample was drawn has Red hair, and compute a 95% confidence
# interval for the percent of subjects with red hair.

Red.Hair.fac <- factor(Hair.fac == "Red", levels = c(TRUE, FALSE),
  labels = c("Red", "Not Red"))

summaryStats(Red.Hair.fac, digits = 1, p.value = TRUE,
  ci = TRUE, test = "binom", test.arg.list = list(p = 0.1))
#           N Pct Exact_p 95%.LCL 95%.UCL
#Red       71  12             9.5    14.9
#Not Red  521  88
```

```
#Combined 592 100    0.11

#----------
# Now test whether the percent of people with Green eyes is the
# same for people with and without Red hair.

HairEye <- apply(HairEyeColor, 1:2, sum)
Hair.color <- rownames(HairEye)
Eye.color  <- colnames(HairEye)

n11 <-     HairEye[Hair.color == "Red", Eye.color == "Green"]
n12 <- sum(HairEye[Hair.color == "Red", Eye.color != "Green"])
n21 <- sum(HairEye[Hair.color != "Red", Eye.color == "Green"])
n22 <- sum(HairEye[Hair.color != "Red", Eye.color != "Green"])

Hair.fac <- factor(rep(c("Red", "Not Red"), c(n11+n12, n21+n22)),
  levels = c("Red", "Not Red"))
Eye.fac  <- factor(c(rep("Green", n11), rep("Not Green", n12),
  rep("Green", n21), rep("Not Green", n22)),
  levels = c("Green", "Not Green"))


#----------
# Here are the results using the chi-square test and computing
# confidence limits for the difference between the two percentages

summaryStats(Eye.fac, group = Hair.fac, digits = 1,
  p.value = TRUE, ci = TRUE, test = "prop",
  stats.in.rows = TRUE, test.arg.list = list(correct = FALSE))
#                Green Not Green Combined
#Red(N)           14    57        71
#Red(Pct)         19.7  80.3     100
#Not Red(N)       50   471       521
#Not Red(Pct)      9.6  90.4     100
#ChiSq_p                           0.01
#95%.LCL.between                   0.5
#95%.UCL.between                  19.7

#----------
# Here are the results using Fisher's exact test and computing
# confidence limits for the odds ratio

summaryStats(Eye.fac, group = Hair.fac, digits = 1,
  p.value = TRUE, ci = TRUE, test = "fisher",
  stats.in.rows = TRUE)
#             Green Not Green Combined
#Red(N)        14    57        71
#Red(Pct)      19.7  80.3     100
#Not Red(N)    50   471       521
#Not Red(Pct)   9.6  90.4     100
#Fisher_p                       0.015
#95%.LCL.OR                     1.1
#95%.UCL.OR                     4.6
```

```
rm(Hair, Hair.color, Hair.fac, Red.Hair.fac, HairEye, Eye.color,
  n11, n12, n21, n22, Eye.fac)

#====================================================================

# The data set EPA.89b.cadmium.df contains information on
# cadmium concentrations in groundwater collected from a
# background and compliance well.  Compare detection frequencies
# between the well types and test for a difference using
# Fisher's exact test.

summaryStats(factor(Censored) ~ Well.type, data = EPA.89b.cadmium.df,
  digits = 1, p.value = TRUE, test = "fisher")


summaryStats(factor(Censored) ~ Well.type, data = EPA.89b.cadmium.df,
  digits = 1, p.value = TRUE, test = "fisher", stats.in.rows = TRUE)
#                FALSE TRUE  Combined
#Background(N)     8    16    24
#Background(Pct)  33.3  66.7 100
#Compliance(N)    24    40    64
#Compliance(Pct)  37.5  62.5 100
#Fisher_p                    0.81
#95%.LCL.OR                  0.3
#95%.UCL.OR                  2.5

#====================================================================

#--------------------
# Paired Observations
#--------------------

# The data frame ACE.13.TCE.df contians paired observations of
# trichloroethylene (TCE; mg/L) at 10 groundwater monitoring wells
# before and after remediation.
#
# Compare TCE concentrations before and after remediation and
# use a paired t-test to test for a difference between periods.

summaryStats(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  p.value = TRUE, paired = TRUE)


summaryStats(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  p.value = TRUE, paired = TRUE, stats.in.rows = TRUE)
#                      Before   After    Combined
#N                      10       10       20
#Mean                   21.624    3.6329  12.6284
#SD                     13.5113   3.5544  13.3281
#Median                 20.3      2.48     8.475
#Min                     5.96     0.272    0.272
#Max                    41.5     10.7     41.5
```

```
#Diff                                    -17.9911
#paired.p.value.between                    0.0027
#95%.LCL.between                         -27.9097
#95%.UCL.between                          -8.0725

#==========

# Repeat the last example, but use a one-sided alternative since
# remediation should decrease TCE concentration.

summaryStats(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  p.value = TRUE, paired = TRUE, alternative = "less")


summaryStats(TCE.mg.per.L ~ Period, data = ACE.13.TCE.df,
  p.value = TRUE, paired = TRUE, alternative = "less",
  stats.in.rows = TRUE)
#                             Before   After    Combined
#N                            10       10       20
#Mean                         21.624    3.6329  12.6284
#SD                           13.5113   3.5544  13.3281
#Median                       20.3      2.48     8.475
#Min                           5.96     0.272    0.272
#Max                          41.5     10.7     41.5
#Diff                                           -17.9911
#paired.p.value.between.less                      0.0013
#95%.LCL.between                                    -Inf
#95%.UCL.between                                  -9.9537
```

`summaryStats.object` *S3 Class "summaryStats"*

Description

Objects of S3 class `"summaryStats"` are returned by the functions `summaryStats` and `summaryFull`.

Details

Objects of S3 class `"summaryStats"` are matrices that contain information about the summary statistics.

Value

Required Attributes

The following attributes must be included in a legitimate matrix of class `"summaryStats"`.

`stats.in.rows`	logical scalar indicating whether the statistics are stored by row (`stats.in.rows=TRUE`) or by column (`stats.in.rows=FALSE`).
`drop0trailing`	logical scalar indicating whether to drop trailing 0's when printing the summary statistics.

Methods

Generic functions that have methods for objects of class "summaryStats" include: print.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

See Also

summaryStats, summaryFull.

Examples

```
# Create an object of class "summaryStats", then print it out.
#-------------------------------------------------------------

summaryStats.obj <- summaryStats(TCE.mg.per.L ~ Well,
  data = EPA.09.Table.9.1.TCE.df, digits = 3)

is.matrix(summaryStats.obj)
#[1] TRUE

class(summaryStats.obj)
#[1] "summaryStats"

attributes(summaryStats.obj)
#$dim
#[1] 2 8
#
#$dimnames
#$dimnames[[1]]
#[1] "Well.1" "Well.2"
#
#$dimnames[[2]]
#[1] "N"       "Mean"    "SD"      "Median"  "Min"     "Max"
#[7] "NA's"    "N.Total"
#
#
#$class
#[1] "summaryStats"
#
#$stats.in.rows
#[1] FALSE
#
#$drop0trailing
#[1] TRUE

summaryStats.obj
#        N  Mean    SD Median   Min  Max NA's N.Total
#Well.1 14 0.063 0.079  0.031 0.004 0.25    1      15
#Well.2 13 0.118 0.020  0.110 0.099 0.17    2      15
```

```
#----------

# Clean up
#---------
rm(summaryStats.obj)
```

`tolIntGamma` *Tolerance Interval for a Gamma Distribution*

Description

Construct a β-content or β-expectation tolerance interval for a gamma distribution.

Usage

```
tolIntGamma(x, coverage = 0.95, cov.type = "content",
  ti.type = "two-sided", conf.level = 0.95, method = "exact",
  est.method = "mle", normal.approx.transform = "kulkarni.powar")

tolIntGammaAlt(x, coverage = 0.95, cov.type = "content",
  ti.type = "two-sided", conf.level = 0.95, method = "exact",
  est.method = "mle", normal.approx.transform = "kulkarni.powar")
```

Arguments

`x`	numeric vector of non-negative observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`coverage`	a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`. If `cov.type="expectation"`, this argument is ignored.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation). See the DETAILS section for more information.
`ti.type`	character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.
`conf.level`	a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level=0.95`.
`method`	for the case of a two-sided tolerance interval, a character string specifying the method for constructing the two-sided normal distribution tolerance interval using the transformed data. This argument is ignored if `ti.type="lower"` or `ti.type="upper"`. The possible values are `"exact"` (the default) and `"wald.wolfowitz"` (the Wald-Wolfowitz approximation). See the DETAILS section of the help file for `tolIntNorm` for more information.

`est.method` character string specifying the method of estimation for the shape and scale distribution parameters. The possible values are `"mle"` (maximum likelihood; the default), `"bcmle"` (bias-corrected mle), `"mme"` (method of moments), and `"mmue"` (method of moments based on the unbiased estimator of variance). See the DETAILS section of the help file for `egamma` for more information.

`normal.approx.transform` character string indicating which power transformation to use. Possible values are `"kulkarni.powar"` (the default), `"cube.root"`, and `"fourth.root"`. See the DETAILS section for more informaiton.

Details

The function `tolIntGamma` returns a tolerance interval as well as estimates of the shape and scale parameters. The function `tolIntGammaAlt` returns a tolerance interval as well as estimates of the mean and coefficient of variation.

The tolerance interval is computed by 1) using a power transformation on the original data to induce approximate normality, 2) using `tolIntNorm` to compute the tolerance interval, and then 3) back-transforming the interval to create a tolerance interval on the original scale. (Krishnamoorthy et al., 2008). The value `normal.approx.transform="cube.root"` uses the cube root transformation suggested by Wilson and Hilferty (1931) and used by Krishnamoorthy et al. (2008) and Singh et al. (2010b), and the value `normal.approx.transform="fourth.root"` uses the fourth root transformation suggested by Hawkins and Wixley (1986) and used by Singh et al. (2010b). The default value `normal.approx.transform="kulkarni.powar"` uses the "Optimum Power Normal Approximation Method" of Kulkarni and Powar (2010). The "optimum" power p is determined by:

$$\begin{array}{ll} p = -0.0705 - 0.178\, shape + 0.475\, \sqrt{shape} & \text{if } shape \leq 1.5 \\ p = 0.246 & \text{if } shape > 1.5 \end{array}$$

where $shape$ denotes the estimate of the shape parameter. Although Kulkarni and Powar (2010) use the maximum likelihood estimate of shape to determine the power p, for the functions `tolIntGamma` and `tolIntGammaAlt` the power p is based on whatever estimate of shape is used (e.g., `est.method="mle"`, `est.method="bcmle"`, etc.).

Value

A list of class `"estimate"` containing the estimated parameters, the tolerance interval, and other information. See `estimate.object` for details.

In addition to the usual components contained in an object of class `"estimate"`, the returned value also includes an additional component within the `"interval"` component:

`normal.transform.power` the value of the power used to transform the original data to approximate normality.

Warning

It is possible for the lower tolerance limit based on the transformed data to be less than 0. In this case, the lower tolerance limit on the original scale is set to 0 and a warning is issued stating that the normal approximation is not accurate in this case.

Note

The gamma distribution takes values on the positive real line. Special cases of the gamma are the exponential distribution and the chi-square distributions. Applications of the gamma include life testing, statistical ecology, queuing theory, inventory control, and precipitation processes. A gamma distribution starts to resemble a normal distribution as the shape parameter a tends to infinity.

Some EPA guidance documents (e.g., Singh et al., 2002; Singh et al., 2010a,b) strongly recommend against using a lognormal model for environmental data and recommend trying a gamma distribuiton instead.

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Evans, M., N. Hastings, and B. Peacock. (1993). *Statistical Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 18.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Hawkins, D. M., and R.A.J. Wixley. (1986). A Note on the Transformation of Chi-Squared Variables to Normality. *The American Statistician*, **40**, 296–298.

Johnson, N.L., S. Kotz, and N. Balakrishnan. (1994). *Continuous Univariate Distributions, Volume 1*. Second Edition. John Wiley and Sons, New York, Chapter 17.

Krishnamoorthy K., T. Mathew, and S. Mukherjee. (2008). Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability. *Technometrics*, **50**(1), 69–78.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Kulkarni, H.V., and S.K. Powar. (2010). A New Method for Interval Estimation of the Mean of the Gamma Distribution. *Lifetime Data Analysis*, **16**, 431–447.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Singh, A., A.K. Singh, and R.J. Iaci. (2002). *Estimation of the Exposure Point Concentration Term Using a Gamma Distribution*. EPA/600/R-02/084. October 2002. Technology Support Center for Monitoring and Site Characterization, Office of Research and Development, Office of Solid Waste and Emergency Response, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Wilson, E.B., and M.M. Hilferty. (1931). The Distribution of Chi-Squares. *Proceedings of the National Academy of Sciences*, **17**, 684–688.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

GammaDist, estimate.object, egamma, tolIntNorm, predIntGamma.

Examples

```
# Generate 20 observations from a gamma distribution with parameters
# shape=3 and scale=2, then create a tolerance interval.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rgamma(20, shape = 3, scale = 2)
tolIntGamma(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 2.203862
#                                 scale = 2.174928
```

```
#
#Estimation Method:               mle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Tolerance Interval Coverage:     95%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact using
#                                 Kulkarni & Powar (2010)
#                                 transformation to Normality
#                                 based on mle of 'shape'
#
#Tolerance Interval Type:         two-sided
#
#Confidence Level:                95%
#
#Number of Future Observations:   1
#
#Tolerance Interval:              LTL =  0.2340438
#                                 UTL = 21.2996464

#----------------------------------------------------------------------

# Using the same data as in the previous example, create an upper
# one-sided tolerance interval and use the bias-corrected estimate of
# shape.

tolIntGamma(dat, ti.type = "upper", est.method = "bcmle")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 1.906616
#                                 scale = 2.514005
#
#Estimation Method:               bcmle
#
#Data:                            dat
#
#Sample Size:                     20
#
#Tolerance Interval Coverage:     95%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact using
#                                 Kulkarni & Powar (2010)
```

```
#                                 transformation to Normality
#                                 based on bcmle of 'shape'
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =  0.00000
#                                 UTL = 17.72107

#----------

# Clean up
rm(dat)

#--------------------------------------------------------------------

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence  using chrysene data and assuming a lognormal
# distribution.  Here we will use the same chrysene data but assume a
# gamma distribution.

attach(EPA.09.Ex.17.3.chrysene.df)
Chrysene <- Chrysene.ppb[Well.type == "Background"]

#----------
# First perform a goodness-of-fit test for a gamma distribution

gofTest(Chrysene, dist = "gamma")

#Results of Goodness-of-Fit Test
#-------------------------------
#
#Test Method:                     Shapiro-Wilk GOF Based on
#                                 Chen & Balakrisnan (1995)
#
#Hypothesized Distribution:       Gamma
#
#Estimated Parameter(s):          shape = 2.806929
#                                 scale = 5.286026
#
#Estimation Method:               mle
#
#Data:                            Chrysene
#
#Sample Size:                     8
#
#Test Statistic:                  W = 0.9156306
#
#Test Statistic Parameter:        n = 8
#
#P-value:                         0.3954223
```

```
#
#Alternative Hypothesis:          True cdf does not equal the
#                                 Gamma Distribution.

#----------
# Now compute the upper tolerance limit

tolIntGamma(Chrysene, ti.type = "upper", coverage = 0.95,
  conf.level = 0.95)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 2.806929
#                                 scale = 5.286026
#
#Estimation Method:               mle
#
#Data:                            Chrysene
#
#Sample Size:                     8
#
#Tolerance Interval Coverage:     95%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact using
#                                 Kulkarni & Powar (2010)
#                                 transformation to Normality
#                                 based on mle of 'shape'
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =  0.00000
#                                 UTL = 69.32425

#----------
# Compare this upper tolerance limit of 69 ppb to the upper tolerance limit
# assuming a lognormal distribution.

tolIntLnorm(Chrysene, ti.type = "upper", coverage = 0.95,
  conf.level = 0.95)$interval$limits["UTL"]

#    UTL
#90.9247

#----------
# Clean up
```

```
rm(Chrysene)
detach("EPA.09.Ex.17.3.chrysene.df")

#----------------------------------------------------------------------

# Reproduce some of the example on page 73 of
# Krishnamoorthy et al. (2008), which uses alkalinity concentrations
# reported in Gibbons (1994) and Gibbons et al. (2009) to construct
# two-sided and one-sided upper tolerance limits for various values
# of coverage using a 95% confidence level.

tolIntGamma(Gibbons.et.al.09.Alkilinity.vec, ti.type = "upper",
  coverage = 0.9, normal.approx.transform = "cube.root")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 9.375013
#                                 scale = 6.202461
#
#Estimation Method:               mle
#
#Data:                            Gibbons.et.al.09.Alkilinity.vec
#
#Sample Size:                     27
#
#Tolerance Interval Coverage:     90%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact using
#                                 Wilson & Hilferty (1931) cube-root
#                                 transformation to Normality
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =  0.00000
#                                 UTL = 97.70502

tolIntGamma(Gibbons.et.al.09.Alkilinity.vec,
  coverage = 0.99, normal.approx.transform = "cube.root")

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Gamma
#
#Estimated Parameter(s):          shape = 9.375013
#                                 scale = 6.202461
```

```
#
#Estimation Method:              mle
#
#Data:                           Gibbons.et.al.09.Alkilinity.vec
#
#Sample Size:                    27
#
#Tolerance Interval Coverage:    99%
#
#Coverage Type:                  content
#
#Tolerance Interval Method:      Exact using
#                                Wilson & Hilferty (1931) cube-root
#                                transformation to Normality
#
#Tolerance Interval Type:        two-sided
#
#Confidence Level:               95%
#
#Tolerance Interval:             LTL =  13.14318
#                                UTL = 148.43876
```

`tolIntLnorm` *Tolerance Interval for a Lognormal Distribution*

Description

Estimate the mean and standard deviation on the log-scale for a lognormal distribution, or estimate the mean and coefficient of variation for a lognormal distribution (alternative parameterization), and construct a β-content or β-expectation tolerance interval.

Usage

```
tolIntLnorm(x, coverage = 0.95, cov.type = "content", ti.type = "two-sided",
  conf.level = 0.95, method = "exact")

tolIntLnormAlt(x, coverage = 0.95, cov.type = "content", ti.type = "two-sided",
  conf.level = 0.95, method = "exact", est.method = "mvue")
```

Arguments

`x` For `tolIntLnorm`, `x` can be a numeric vector of positive observations, or an object resulting from a call to an estimating function that assumes a lognormal distribution (i.e., `elnorm` or `elnormCensored`). You *cannot* supply objects resulting from a call to estimating functions that use the alternative parameterization such as `elnormAlt` or `elnormAltCensored`.

For `tolIntLnormAlt`, a numeric vector of positive observations.

If `x` is a numeric vector, missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.

`coverage` a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`. If `cov.type="expectation"`, this argument is ignored.

`cov.type` character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation). See the DETAILS section for more information.

`ti.type` character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

`conf.level` a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level=0.95`.

`method` for the case of a two-sided tolerance interval, a character string specifying the method for constructing the tolerance interval. This argument is ignored if `ti.type="lower"` or `ti.type="upper"`. The possible values are `"exact"` (the default) and `"wald.wolfowitz"` (the Wald-Wolfowitz approximation). See the DETAILS section in this help file and the DETAILS section in the help file for `tolIntNorm` for more information.

`est.method` for `tolIntLnormAlt`, a character string specifying the method of estimating the mean and coefficient of variation. *This argument has no effect on the method of constructing the tolerance interval.* Possible values are `"mvue"` (minimum variance unbiased; the default), `"qmle"` (quasi maximum likelihood), `"mle"` (maximum likelihood), `"mme"` (method of moments), and `"mmue"` (method of moments based on the unbiased estimate of variance). See the DETAILS section of `elnormAlt` for more information on these estimation methods.

Details

The function `tolIntLnorm` returns a tolerance interval as well as estimates of the meanlog and sdlog parameters. The function `tolIntLnormAlt` returns a tolerance interval as well as estimates of the mean and coefficient of variation.

A tolerance interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a tolerance interval based on the normal (Gaussian) distribution by calling `tolIntNorm`. These tolerance limits are then exponentiated to produce a tolerance interval on the original scale of the data.

Value

If `x` is a numeric vector, a list of class `"estimate"` containing the estimated parameters, a component called `interval` containing the tolerance interval information, and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, a list whose class is the same as `x`. The list contains the same components as `x`. If `x` already has a component called `interval`, this component is replaced with the tolerance interval information.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002,

Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Odeh, R.E., and D.B. Owen. (1980). *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*. Marcel Dekker, New York.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wald, A., and J. Wolfowitz. (1946). Tolerance Limits for a Normal Distribution. *Annals of Mathematical Statistics* **17**, 208-215.

See Also

tolIntNorm, Lognormal, LognormalAlt, estimate.object, elnorm, elnormAlt, eqlnorm, predIntLnorm, Tolerance Intervals, Estimating Distribution Parameters, Estimating Distribution Quantiles.

Examples

```
# Generate 20 observations from a lognormal distribution with parameters
# meanlog=0 and sdlog=1.  Use tolIntLnorm to estimate
# the mean and standard deviation of the log of the true distribution, and
# construct a two-sided 90% beta-content tolerance interval with associated
# confidence level 95%.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
dat <- rlnorm(20)
tolIntLnorm(dat, coverage = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = -0.06941976
#                                 sdlog   =  0.59011300
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Tolerance Interval Coverage:     90%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         two-sided
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL = 0.237457
#                                 UTL = 3.665369


# The exact two-sided interval that contains 90% of this distribution
# is given by: [0.193, 5.18].
```

```
qlnorm(p = c(0.05, 0.95))
#[1] 0.1930408 5.1802516

# Clean up
rm(dat)

#==========

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence  using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5).  The tolerance limit
# is based on the data from the background wells.

head(EPA.09.Ex.17.3.chrysene.df)
#  Month   Well  Well.type Chrysene.ppb
#1     1 Well.1 Background         19.7
#2     2 Well.1 Background         39.2
#3     3 Well.1 Background          7.8
#4     4 Well.1 Background         12.8
#5     1 Well.2 Background         10.2
#6     2 Well.2 Background          7.2

longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene.ppb", "Month", "Well")
#  Well.1 Well.2 Well.3 Well.4 Well.5
#1   19.7   10.2   68.0   26.8   47.0
#2   39.2    7.2   48.9   17.7   30.5
#3    7.8   16.1   30.1   31.9   15.0
#4   12.8    5.7   38.1   22.2   23.4

with(EPA.09.Ex.17.3.chrysene.df,
  tolIntLnorm(Chrysene.ppb[Well.type == "Background"],
  ti.type = "upper", coverage = 0.95, conf.level = 0.95))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          meanlog = 2.5085773
#                                 sdlog   = 0.6279479
#
#Estimation Method:               mvue
#
#Data:                            Chrysene.ppb[Well.type == "Background"]
#
#Sample Size:                     8
#
#Tolerance Interval Coverage:     95%
```

```
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =  0.0000
#                                 UTL = 90.9247

#----------

# Repeat the above example, but estimate the mean and
# coefficient of variation on the original scale
#-----------------------------------------------

with(EPA.09.Ex.17.3.chrysene.df,
  tolIntLnormAlt(Chrysene.ppb[Well.type == "Background"],
  ti.type = "upper", coverage = 0.95, conf.level = 0.95))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Estimated Parameter(s):          mean = 14.5547353
#                                 cv   =  0.6390825
#
#Estimation Method:               mvue
#
#Data:                            Chrysene.ppb[Well.type == "Background"]
#
#Sample Size:                     8
#
#Tolerance Interval Coverage:     95%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =  0.0000
#                                 UTL = 90.9247
```

---

tolIntLnormCensored *Tolerance Interval for a Lognormal Distribution Based on Censored Data*

Description

Construct a β-content or β-expectation tolerance interval for a lognormal distribution based on Type I or Type II censored data.

Usage

```
tolIntLnormCensored(x, censored, censoring.side = "left", coverage = 0.95,
  cov.type = "content", ti.type = "two-sided", conf.level = 0.95,
  method = "mle", ti.method = "exact.for.complete", seed = NULL,
  nmc = 1000)
```

Arguments

`x`	numeric vector of positive observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`censored`	numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `censored` is `"logical"`, `TRUE` values correspond to elements of `x` that are censored, and `FALSE` values correspond to elements of `x` that are not censored. If the mode of `censored` is `"numeric"`, it must contain only `1`'s and `0`'s; `1` corresponds to `TRUE` and `0` corresponds to `FALSE`. Missing (`NA`) values are allowed but will be removed.
`censoring.side`	character string indicating on which side the censoring occurs. The possible values are `"left"` (the default) and `"right"`.
`coverage`	a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation). See the DETAILS section for more information.
`ti.type`	character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.
`conf.level`	a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level=0.95`.
`method`	character string indicating the method to use for parameter estimation on the log-scale. For singly censored data, possible values are `"mle"` (the default), `"bcmle"`, `"qq.reg"`, `"qq.reg.w.cen.level"`, `"impute.w.qq.reg"`, `"impute.w.qq.reg.w.cen.level"`, `"impute.w.mle"`, `"iterative.impute.w.qq.reg"`, `"m.est"`, and `"half.cen.level"`. See the help file for `enormCensored` for details. For multiply censored data, possible values are `"mle"` (the default), `"qq.reg"`, `"impute.w.qq.reg"`, and `"half.cen.level"`. See the help file for `enormCensored` for details.
`ti.method`	character string specifying the method for constructing the tolerance interval. Possible values are:

	`"exact.for.complete"` (the default), `"gpq"` (Generalized Pivotal Quantity), and `"wald.wolfowitz.for.complete"` (only available for a two-sided tolerance interval, i.e., when `ti.type="two-sided"`). See the DETAILS section for more information.
`seed`	for the case when `ti.method="gpq"`, a positive integer to pass to the function `gpqTolIntNormSinglyCensored` or `gpqTolIntNormMultiplyCensored`. This argument is ignored if `seed=NULL` (the default). Using the `seed` argument lets you reproduce the exact same result if all other arguments stay the same.
`nmc`	for the case when `ti.method="gpq"`, a positive integer ≥ 10 indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).

Details

A tolerance interval for a lognormal distribution is constructed by taking the natural logarithm of the observations and constructing a tolerance interval based on the normal (Gaussian) distribution by calling `tolIntNormCensored`. These tolerance limits are then exponentiated to produce a tolerance interval on the original scale of the data.

Value

A list of class `"estimateCensored"` containing the estimated parameters, the tolerance interval, and other information. See `estimateCensored.object` for details.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Odeh, R.E., and D.B. Owen. (1980). *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*. Marcel Dekker, New York.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wald, A., and J. Wolfowitz. (1946). Tolerance Limits for a Normal Distribution. *Annals of Mathematical Statistics* **17**, 208-215.

See Also

`tolIntNormCensored`, `gpqTolIntNormSinglyCensored`, `eqnormCensored`, `enormCensored`, `estimateCensored.object`.

Examples

```
# Generate 20 observations from a lognormal distribution with parameters
# mean=10 and cv=1, censor the observations less than 5,
# then create a one-sided upper tolerance interval with 90%
# coverage and 95% confidence based on these Type I left, singly
# censored data.
# (Note: the call to set.seed allows you to reproduce this example.)

set.seed(250)
dat <- rlnormAlt(20, mean = 10, cv = 1)
sort(dat)
```

```
# [1]  2.608298  3.185459  4.196216  4.383764  4.569752  5.136130
# [7]  5.209538  5.916284  6.199076  6.214755  6.255779  6.778361
#[13]  7.074972  7.100494  8.930845 10.388766 11.402769 14.247062
#[19] 14.559506 15.437340

censored <- dat < 5
dat[censored] <- 5

tolIntLnormCensored(dat, censored, coverage = 0.9, ti.type="upper")

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              5
#
#Estimated Parameter(s):          meanlog = 1.8993686
#                                 sdlog   = 0.4804343
#
#Estimation Method:               MLE
#
#Data:                            dat
#
#Censoring Variable:              censored
#
#Sample Size:                     20
#
#Percent Censored:                25%
#
#Assumed Sample Size:             20
#
#Tolerance Interval Coverage:     90%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact for
#                                 Complete Data
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =  0.00000
#                                 UTL = 16.85556

## Not run:
  # Note:  The true 90'th percentile is 20.55231
  #---------------------------------------------
  qlnormAlt(0.9, mean = 10, cv = 1)
```

```
#[1] 20.55231


# Compare the result using the method "gpq"
tolIntLnormCensored(dat, censored, coverage = 0.9, ti.type="upper",
  ti.method = "gpq", seed = 432)$interval$limits

#     LTL      UTL
# 0.00000 17.85474

# Clean Up
#---------
rm(dat, censored)

#--------------------------------------------------------------

# Example 15-1 of USEPA (2009, p. 15-10) shows how to estimate
# the mean and standard deviation using log-transformed multiply
# left-censored manganese concentration data.  Here we'll construct a
# 95

EPA.09.Ex.15.1.manganese.df
#    Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
# 1       1 Well.1                 <5           5.0     TRUE
# 2       2 Well.1               12.1          12.1    FALSE
# 3       3 Well.1               16.9          16.9    FALSE
# ...
# 23      3 Well.5                3.3           3.3    FALSE
# 24      4 Well.5                8.4           8.4    FALSE
# 25      5 Well.5                <2           2.0     TRUE

with(EPA.09.Ex.15.1.manganese.df,
  tolIntLnormCensored(Manganese.ppb, Censored, coverage = 0.9,
    ti.type = "upper"))

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#----------------------------------------
#
#Assumed Distribution:            Lognormal
#
#Censoring Side:                  left
#
#Censoring Level(s):              2 5
#
#Estimated Parameter(s):          meanlog = 2.215905
#                                 sdlog   = 1.356291
#
#Estimation Method:               MLE
#
#Data:                            Manganese.ppb
#
#Censoring Variable:              censored
```

```
#
#Sample Size:                     25
#
#Percent Censored:                24
#
#Assumed Sample Size:             25
#
#Tolerance Interval Coverage:     90
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact for
#                                 Complete Data
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95
#
#Tolerance Interval:              LTL =   0.0000
#                                 UTL = 110.9305

## End(Not run)
```

`tolIntNorm` *Tolerance Interval for a Normal Distribution*

Description

Construct a β-content or β-expectation tolerance interval for a normal distribution.

Usage

```
tolIntNorm(x, coverage = 0.95, cov.type = "content",
  ti.type = "two-sided", conf.level = 0.95, method = "exact")
```

Arguments

`x`	numeric vector of observations, or an object resulting from a call to an estimating function that assumes a normal (Gaussian) distribution (i.e., `enorm` or `enormCensored`). If `x` is a numeric vector, missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`coverage`	a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`. If `cov.type="expectation"`, this argument is ignored.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation). See the DETAILS section for more information.

`ti.type`	character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.
`conf.level`	a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level=0.95`.
`method`	for the case of a two-sided tolerance interval, a character string specifying the method for constructing the tolerance interval. This argument is ignored if `ti.type="lower"` or `ti.type="upper"`. The possible values are `"exact"` (the default) and `"wald.wolfowitz"` (the Wald-Wolfowitz approximation). See the DETAILS section for more information.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

A tolerance interval for some population is an interval on the real line constructed so as to contain $100\beta\%$ of the population (i.e., $100\beta\%$ of all future observations), where $0 < \beta < 1$. The quantity $100\beta\%$ is called the *coverage*.

There are two kinds of tolerance intervals (Guttman, 1970):

- A β-content tolerance interval with confidence level $100(1-\alpha)\%$ is constructed so that it contains at least $100\beta\%$ of the population (i.e., the coverage is at least $100\beta\%$) with probability $100(1-\alpha)\%$, where $0 < \alpha < 1$. The quantity $100(1-\alpha)\%$ is called the confidence level or confidence coefficient associated with the tolerance interval.
- A β-expectation tolerance interval is constructed so that the *average* coverage of the interval is $100\beta\%$.

Note: A β-expectation tolerance interval with coverage $100\beta\%$ is equivalent to a prediction interval for one future observation with associated confidence level $100\beta\%$. Note that there is no explicit confidence level associated with a β-expectation tolerance interval. If a β-expectation tolerance interval is treated as a β-content tolerance interval, the confidence level associated with this tolerance interval is usually around 50% (e.g., Guttman, 1970, Table 4.2, p.76).

For a normal distribution, the form of a two-sided $100(1-\alpha)\%$ tolerance interval is:

$$[\bar{x} - Ks,\ \bar{x} + Ks]$$

where $\bar{x}$ denotes the sample mean, s denotes the sample standard deviation, and K denotes a constant that depends on the sample size n, the coverage, and, for a β-content tolerance interval (but not a β-expectation tolerance interval), the confidence level.

Similarly, the form of a one-sided lower tolerance interval is:

$$[\bar{x} - Ks,\ \infty]$$

and the form of a one-sided upper tolerance interval is:

$$[-\infty,\ \bar{x} + Ks]$$

but K differs for one-sided versus two-sided tolerance intervals. The derivation of the constant K is explained in the help file for `tolIntNormK`.

Value

If `x` is a numeric vector, `tolIntNorm` returns a list of class `"estimate"` containing the estimated parameters, a component called `interval` containing the tolerance interval information, and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `tolIntNorm` returns a list whose class is the same as `x`. The list contains the same components as `x`. If `x` already has a component called `interval`, this component is replaced with the tolerance interval information.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Odeh, R.E., and D.B. Owen. (1980). *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*. Marcel Dekker, New York.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wald, A., and J. Wolfowitz. (1946). Tolerance Limits for a Normal Distribution. *Annals of Mathematical Statistics* **17**, 208-215.

See Also

tolIntNormK, tolIntLnorm, Normal, estimate.object, enorm, eqnorm, predIntNorm, Tolerance Intervals, Estimating Distribution Parameters, Estimating Distribution Quantiles.

Examples

```
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=2, then create a tolerance interval.
# (Note: the call to set.seed simply allows you to reproduce this
# example.)

set.seed(250)
dat <- rnorm(20, mean = 10, sd = 2)
tolIntNorm(dat)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 9.861160
#                                 sd   = 1.180226
#
#Estimation Method:               mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Tolerance Interval Coverage:     95%
#
#Coverage Type:                   content
#
```

```
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         two-sided
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =  6.603328
#                                 UTL = 13.118993

#----------

# Clean up
rm(dat)

#--------------------------------------------------------------------

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence  using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5).  The tolerance limit
# is based on the data from the background wells.

# Here we will first take the log of the data and
# then construct the tolerance interval; note however that it is
# easier to call the function tolIntLnorm instead using the
# original data.

head(EPA.09.Ex.17.3.chrysene.df)
#  Month   Well  Well.type Chrysene.ppb
#1     1 Well.1 Background         19.7
#2     2 Well.1 Background         39.2
#3     3 Well.1 Background          7.8
#4     4 Well.1 Background         12.8
#5     1 Well.2 Background         10.2
#6     2 Well.2 Background          7.2

longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene.ppb", "Month", "Well")
#  Well.1 Well.2 Well.3 Well.4 Well.5
#1   19.7   10.2   68.0   26.8   47.0
#2   39.2    7.2   48.9   17.7   30.5
#3    7.8   16.1   30.1   31.9   15.0
#4   12.8    5.7   38.1   22.2   23.4

tol.int.list <- with(EPA.09.Ex.17.3.chrysene.df,
  tolIntNorm(log(Chrysene.ppb[Well.type == "Background"]),
  ti.type = "upper", coverage = 0.95, conf.level = 0.95))

tol.int.list

#Results of Distribution Parameter Estimation
```

```
#----------------------------------------
#
#Assumed Distribution:            Normal
#
#Estimated Parameter(s):          mean = 2.5085773
#                                 sd   = 0.6279479
#
#Estimation Method:               mvue
#
#Data:                            log(Chrysene.ppb[Well.type == "Background"])
#
#Sample Size:                     8
#
#Tolerance Interval Coverage:     95%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =     -Inf
#                                 UTL = 4.510032

# Compute the upper tolerance interaval on the original scale
# by exponentiating the upper tolerance limit:

exp(tol.int.list$interval$limits["UTL"])
#    UTL
#90.9247

#----------

# Clean up

rm(tol.int.list)
```

`tolIntNormCensored` *Tolerance Interval for a Normal Distribution Based on Censored Data*

Description

Construct a β-content or β-expectation tolerance interval for a normal distribution based on Type I or Type II censored data.

Usage

```
tolIntNormCensored(x, censored, censoring.side = "left", coverage = 0.95,
```

```
cov.type = "content", ti.type = "two-sided", conf.level = 0.95,
method = "mle", ti.method = "exact.for.complete", seed = NULL,
nmc = 1000)
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
censored	numeric or logical vector indicating which values of x are censored. This must be the same length as x. If the mode of censored is "logical", TRUE values correspond to elements of x that are censored, and FALSE values correspond to elements of x that are not censored. If the mode of censored is "numeric", it must contain only 1's and 0's; 1 corresponds to TRUE and 0 corresponds to FALSE. Missing (NA) values are allowed but will be removed.
censoring.side	character string indicating on which side the censoring occurs. The possible values are "left" (the default) and "right".
coverage	a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.
cov.type	character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation). See the DETAILS section for more information.
ti.type	character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level	a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is conf.level=0.95.
method	character string indicating the method to use for parameter estimation. For singly censored data, possible values are "mle" (the default), "bcmle", "qq.reg", "qq.reg.w.cen.level", "impute.w.qq.reg", "impute.w.qq.reg.w.cen.level", "impute.w.mle", "iterative.impute.w.qq.reg", "m.est", and "half.cen.level". See the help file for enormCensored for details. For multiply censored data, possible values are "mle" (the default), "qq.reg", "impute.w.qq.reg", and "half.cen.level". See the help file for enormCensored for details.
ti.method	character string specifying the method for constructing the tolerance interval. Possible values are: "exact.for.complete" (the default), "gpq" (Generalized Pivotal Quantity), and "wald.wolfowitz.for.complete" (only available for a two-sided tolerance interval, i.e., when ti.type="two-sided"). See the DETAILS section for more information.
seed	for the case when ti.method="gpq", a positive integer to pass to the function gpqTolIntNormSinglyCensored or gpqTolIntNormMultiplyCensored. This argument is ignored if seed=NULL (the default). Using the seed argument lets you reproduce the exact same result if all other arguments stay the same.

nmc for the case when ti.method="gpq", a positive integer ≥ 10 indicating the number of Monte Carlo trials to run in order to compute the GPQ(s).

Details

See the help file for tolIntNorm for an explanation of tolerance intervals. When ti.method="gpq", the tolerance interval is constructed using the method of Generalized Pivotal Quantities as explained in Krishnamoorthy and Mathew (2009, p. 327). When ti.method="exact.for.complete" or ti.method="wald.wolfowitz.for.complete", the tolerance interval is constructed by first computing the maximum likelihood estimates of the mean and standard deviation by calling enormCensored, then passing these values to the function tolIntNorm to produce the tolerance interval as if the estimates were based on complete rather than censored data. These last two methods are purely ad-hoc and their properties need to be studied.

Value

A list of class "estimateCensored" containing the estimated parameters, the tolerance interval, and other information. See estimateCensored.object for details.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Odeh, R.E., and D.B. Owen. (1980). *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*. Marcel Dekker, New York.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wald, A., and J. Wolfowitz. (1946). Tolerance Limits for a Normal Distribution. *Annals of Mathematical Statistics* **17**, 208-215.

See Also

gpqTolIntNormSinglyCensored, eqnormCensored, enormCensored, estimateCensored.object.

Examples

```
# Generate 20 observations from a normal distribution with parameters
# mean=10 and sd=3, censor the observations less than 9,
# then create a one-sided upper tolerance interval with 90%
# coverage and 95% confidence based on these Type I left, singly
# censored data.
# (Note: the call to set.seed allows you to reproduce this example.

set.seed(250)
dat <- sort(rnorm(20, mean = 10, sd = 3))
dat
# [1]  6.406313  7.126621  8.119660  8.277216  8.426941  8.847961
# [7]  8.899098  9.357509  9.525756  9.534858  9.558567  9.847663
#[13] 10.001989 10.014964 10.841384 11.386264 11.721850 12.524300
#[19] 12.602469 12.813429

censored <- dat < 9
```

```
dat[censored] <- 9

tolIntNormCensored(dat, censored, coverage = 0.9, ti.type="upper")

#Results of Distribution Parameter Estimation
#Based on Type I Censored Data
#--------------------------------------------
#
#Assumed Distribution:            Normal
#
#Censoring Side:                  left
#
#Censoring Level(s):              9
#
#Estimated Parameter(s):          mean = 9.700962
#                                 sd   = 1.845067
#
#Estimation Method:               MLE
#
#Data:                            dat
#
#Censoring Variable:              censored
#
#Sample Size:                     20
#
#Percent Censored:                35%
#
#Assumed Sample Size:             20
#
#Tolerance Interval Coverage:     90%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact for
#                                 Complete Data
#
#Tolerance Interval Type:         upper
#
#Confidence Level:                95%
#
#Tolerance Interval:              LTL =     -Inf
#                                 UTL = 13.25454
## Not run:

# Note:  The true 90'th percentile is 13.84465
#--------------------------------------------
qnorm(0.9, mean = 10, sd = 3)
# [1] 13.84465

# Compare the result using the method "gpq"
tolIntNormCensored(dat, censored, coverage = 0.9, ti.type="upper",
  ti.method = "gpq", seed = 432)$interval$limits
#     LTL      UTL
```

```
#    -Inf 13.56826

# Clean Up
#---------
rm(dat, censored)

#==========

# Example 15-1 of USEPA (2009, p. 15-10) shows how to estimate
# the mean and standard deviation using log-transformed multiply
# left-censored manganese concentration data.  Here we'll construct a
# 95

EPA.09.Ex.15.1.manganese.df
#    Sample   Well Manganese.Orig.ppb Manganese.ppb Censored
# 1       1 Well.1                 <5           5.0     TRUE
# 2       2 Well.1               12.1          12.1    FALSE
# 3       3 Well.1               16.9          16.9    FALSE
# ...
# 23      3 Well.5                3.3           3.3    FALSE
# 24      4 Well.5                8.4           8.4    FALSE
# 25      5 Well.5                <2           2.0     TRUE

with(EPA.09.Ex.15.1.manganese.df,
  tolIntNormCensored(log(Manganese.ppb), Censored, coverage = 0.9,
    ti.type = "upper"))

# Results of Distribution Parameter Estimation
# Based on Type I Censored Data
# --------------------------------------------
#
# Assumed Distribution:            Normal
#
# Censoring Side:                  left
#
# Censoring Level(s):              0.6931472 1.6094379
#
# Estimated Parameter(s):          mean = 2.215905
#                                  sd   = 1.356291
#
# Estimation Method:               MLE
#
# Data:                            log(Manganese.ppb)
#
# Censoring Variable:              censored
#
# Sample Size:                     25
#
# Percent Censored:                24
#
# Assumed Sample Size:             25
#
# Tolerance Interval Coverage:     90
```

```
#
# Coverage Type:                   content
#
# Tolerance Interval Method:       Exact for
#                                  Complete Data
#
# Tolerance Interval Type:         upper
#
# Confidence Level:                95
#
# Tolerance Interval:              LTL =     -Inf
#                                  UTL = 4.708904

## End(Not run)
```

`tolIntNormHalfWidth` *Half-Width of a Tolerance Interval for a Normal Distribution*

Description

Compute the half-width of a tolerance interval for a normal distribution.

Usage

```
tolIntNormHalfWidth(n, sigma.hat = 1, coverage = 0.95, cov.type = "content",
  conf.level = 0.95, method = "wald.wolfowitz")
```

Arguments

`n`	numeric vector of positive integers greater than 1 indicating the sample size upon which the prediction interval is based. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`sigma.hat`	numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is `sigma.hat=1`.
`coverage`	numeric vector of values between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation).
`conf.level`	numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.
`method`	character string specifying the method for constructing the tolerance interval. The possible values are `"exact"` (the default) and `"wald.wolfowitz"` (the Wald-Wolfowitz approximation).

Details

If the arguments `n`, `sigma.hat`, `coverage`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help files for `tolIntNorm` and `tolIntNormK` give formulas for a two-sided tolerance interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level and coverage. Specifically, the two-sided tolerance interval is given by:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, and the coverage (see the help file for `tolIntNormK`). Thus, the half-width of the tolerance interval is given by:

$$HW = Ks \qquad (4)$$

Value

numeric vector of half-widths.

Note

See the help file for `tolIntNorm`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce tolerance intervals. The functions `tolIntNormHalfWidth`, `tolIntNormN`, and `plotTolIntNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `tolIntNorm`.

See Also

`tolIntNorm`, `tolIntNormK`, `tolIntNormN`, `plotTolIntNormDesign`, `Normal`.

Examples

```
# Look at how the half-width of a tolerance interval increases with
# increasing coverage:

seq(0.5, 0.9, by=0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

round(tolIntNormHalfWidth(n = 10, coverage = seq(0.5, 0.9, by = 0.1)), 2)
#[1] 1.17 1.45 1.79 2.21 2.84

#----------

# Look at how the half-width of a tolerance interval decreases with
# increasing sample size:

2:5
#[1] 2 3 4 5

round(tolIntNormHalfWidth(n = 2:5), 2)
#[1] 37.67 9.92 6.37 5.08

#----------

# Look at how the half-width of a tolerance interval increases with
# increasing estimated standard deviation for a fixed sample size:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

round(tolIntNormHalfWidth(n = 10, sigma.hat = seq(0.5, 2, by = 0.5)), 2)
#[1] 1.69 3.38 5.07 6.76

#----------

# Look at how the half-width of a tolerance interval increases with
# increasing confidence level for a fixed sample size:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

round(tolIntNormHalfWidth(n = 5, conf = seq(0.5, 0.9, by = 0.1)), 2)
#[1] 2.34 2.58 2.89 3.33 4.15

#==========

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence  using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5).  The tolerance limit
```

```
# is based on the data from the background wells.

# Here we will first take the log of the data and then estimate the
# standard deviation based on the two background wells.  We will use this
# estimate of standard deviation to compute the half-widths of
# future tolerance intervals on the log-scale for various sample sizes.

head(EPA.09.Ex.17.3.chrysene.df)
#  Month   Well  Well.type Chrysene.ppb
#1     1 Well.1 Background         19.7
#2     2 Well.1 Background         39.2
#3     3 Well.1 Background          7.8
#4     4 Well.1 Background         12.8
#5     1 Well.2 Background         10.2
#6     2 Well.2 Background          7.2

longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene.ppb", "Month", "Well")
#  Well.1 Well.2 Well.3 Well.4 Well.5
#1   19.7   10.2   68.0   26.8   47.0
#2   39.2    7.2   48.9   17.7   30.5
#3    7.8   16.1   30.1   31.9   15.0
#4   12.8    5.7   38.1   22.2   23.4

summary.stats <- summaryStats(log(Chrysene.ppb) ~ Well.type,
  data = EPA.09.Ex.17.3.chrysene.df)

summary.stats
#            N   Mean     SD Median    Min    Max
#Background  8 2.5086 0.6279 2.4359 1.7405 3.6687
#Compliance 12 3.4173 0.4361 3.4111 2.7081 4.2195

sigma.hat <- summary.stats["Background", "SD"]
sigma.hat
#[1] 0.6279

tolIntNormHalfWidth(n = c(4, 8, 16), sigma.hat = sigma.hat)
#[1] 3.999681 2.343160 1.822759

#==========

# Clean up
#---------
rm(summary.stats, sigma.hat)
```

`tolIntNormK` *Compute the Value of K for a Tolerance Interval for a Normal Distribution*

Description

Compute the value of K (the multiplier of estimated standard deviation) used to construct a tolerance interval based on data from a normal distribution.

Usage

```
tolIntNormK(n, df = n - 1, coverage = 0.95, cov.type = "content",
  ti.type = "two-sided", conf.level = 0.95, method = "exact",
  rel.tol = 1e-07, abs.tol = rel.tol)
```

Arguments

n	a positive integer greater than 2 indicating the sample size upon which the tolerance interval is based.
df	the degrees of freedom associated with the tolerance interval. The default is df=n-1.
coverage	a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is coverage=0.95.
cov.type	character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation). See the help file for tolIntNorm for more information on the difference between β-content and β-expectation tolerance intervals.
ti.type	character string indicating what kind of tolerance interval to compute. The possible values are "two-sided" (the default), "lower", and "upper".
conf.level	a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is conf.level=0.95.
method	for the case of a two-sided tolerance interval, a character string specifying the method for constructing the tolerance interval. This argument is ignored if ti.type="lower" or ti.type="upper". The possible values are "exact" (the default) and "wald.wolfowitz" (the Wald-Wolfowitz approximation). See the DETAILS section for more information.
rel.tol	in the case when ti.type="two-sided" and method="exact", the argument rel.tol is passed to the function integrate. The default value is rel.tol=1e-07.
abs.tol	in the case when ti.type="two-sided" and method="exact", the argument abs.tol is passed to the function integrate. The default value is the value of rel.tol.

Details

A tolerance interval for some population is an interval on the real line constructed so as to contain $100\beta\%$ of the population (i.e., $100\beta\%$ of all future observations), where $0 < \beta < 1$. The quantity $100\beta\%$ is called the coverage.

There are two kinds of tolerance intervals (Guttman, 1970):

- A β-content tolerance interval with confidence level $100(1-\alpha)\%$ is constructed so that it contains at least $100\beta\%$ of the population (i.e., the coverage is at least $100\beta\%$) with probability $100(1-\alpha)\%$, where $0 < \alpha < 1$. The quantity $100(1-\alpha)\%$ is called the confidence level or confidence coefficient associated with the tolerance interval.
- A β-expectation tolerance interval is constructed so that the *average* coverage of the interval is $100\beta\%$.

Note: A β-expectation tolerance interval with coverage $100\beta\%$ is equivalent to a prediction interval for one future observation with associated confidence level $100\beta\%$. Note that there is no explicit confidence level associated with a β-expectation tolerance interval. If a β-expectation tolerance interval is treated as a β-content tolerance interval, the confidence level associated with this tolerance interval is usually around 50% (e.g., Guttman, 1970, Table 4.2, p.76).

For a normal distribution, the form of a two-sided $100(1-\alpha)\%$ tolerance interval is:

$$[\bar{x} - Ks,\ \bar{x} + Ks]$$

where $\bar{x}$ denotes the sample mean, s denotes the sample standard deviation, and K denotes a constant that depends on the sample size n, the coverage, and, for a β-content tolerance interval (but not a β-expectation tolerance interval), the confidence level.

Similarly, the form of a one-sided lower tolerance interval is:

$$[\bar{x} - Ks,\ \infty]$$

and the form of a one-sided upper tolerance interval is:

$$[-\infty,\ \bar{x} + Ks]$$

but K differs for one-sided versus two-sided tolerance intervals.

The Derivation of K for a β-Content Tolerance Interval

One-Sided Case

When `ti.type="upper"` or `ti.type="lower"`, the constant K for a $100\beta\%$ β-content tolerance interval with associated confidence level $100(1-\alpha)\%$ is given by:

$$K = t(n-1, 1-\alpha, z_\beta\sqrt{n})/\sqrt{n}$$

where $t(\nu, p, \delta)$ denotes the p'th quantile of a non-central t-distribution with ν degrees of freedom and noncentrality parameter δ (see the help file for TDist), and z_p denotes the p'th quantile of a standard normal distribution.

Two-Sided Case

When `ti.type="two-sided"` and `method="exact"`, the exact formula for the constant K for a $100\beta\%$ β-content tolerance interval with associated confidence level $100(1-\alpha)\%$ requires numerical integration and has been derived by several different authors, including Odeh (1978), Eberhardt et al. (1989), Jilek (1988), Fujino (1989), and Janiga and Miklos (2001). Specifically, for given values of the sample size n, degrees of freedom ν, confidence level $(1-\alpha)$, and coverage β, the constant K is the solution to the equation:

$$\sqrt{\frac{n}{2\pi}} \int_{-\infty}^{\infty} F(x, K, \nu, R)\, e^{(-nx^2)/2}\, dx = 1 - \alpha$$

where $F(x, K, \nu, R)$ denotes the upper-tail area from $(\nu\, R^2)/K^2$ to ∞ of the chi-squared distribution with ν degrees of freedom, and R is the solution to the equation:

$$\Phi(x + R) - \Phi(x - R) = \beta$$

where $\Phi()$ denotes the standard normal cumulative distribuiton function.

When `ti.type="two-sided"` and `method="wald.wolfowitz"`, the approximate formula due to Wald and Wolfowitz (1946) for the constant K for a $100\beta\%$ β-content tolerance interval with associated confidence level $100(1-\alpha)\%$ is given by:

$$K \approx r\,u$$

where r is the solution to the equation:

$$\Phi(\frac{1}{\sqrt{n}} + r) - \Phi(\frac{1}{\sqrt{n}} - r) = \beta$$

$\Phi()$ denotes the standard normal cumulative distribuiton function, and u is given by:

$$u = \sqrt{\frac{n-1}{\chi^2(n-1, \alpha)}}$$

where $\chi^2(\nu, p)$ denotes the p'th quantile of the chi-squared distribution with ν degrees of freedom.

The Derivation of K for a β-Expectation Tolerance Interval

As stated above, a β-expectation tolerance interval with coverage $100\beta\%$ is equivalent to a prediction interval for one future observation with associated confidence level $100\beta\%$. This is because the probability that any single future observation will fall into this interval is $100\beta\%$, so the distribution of the number of N future observations that will fall into this interval is binomial with parameters `size` = N and `prob` = β (see the help file for Binomial). Hence the expected proportion of future observations that will fall into this interval is $100\beta\%$ and is independent of the value of N. See the help file for `predIntNormK` for information on how to derive K for these intervals.

Value

The value of K, a numeric scalar used to construct tolerance intervals for a normal (Gaussian) distribution.

Note

Tabled values of K are given in Gibbons et al. (2009), Gilbert (1987), Guttman (1970), Krishnamoorthy and Mathew (2009), Owen (1962), Odeh and Owen (1980), and USEPA (2009).

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Lewis Publishers, Boca Raton.

Draper, N., and H. Smith. (1998). *Applied Regression Analysis*. Third Edition. John Wiley and Sons, New York.

Eberhardt, K.R., R.W. Mee, and C.P. Reeve. (1989). Computing Factors for Exact Two-Sided Tolerance Limits for a Normal Distribution. *Communications in Statistics, Part B-Simulation and Computation* **18**, 397-413.

Ellison, B.E. (1964). On Two-Sided Tolerance Intervals for a Normal Distribution. *Annals of Mathematical Statistics* **35**, 762-772.

Fujino, T. (1989). Exact Two-Sided Tolerance Limits for a Normal Distribution. *Japanese Journal of Applied Statistics* **18**, 29-36.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Gilbert, R.O. (1987). *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold, New York.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J. (1970b). Statistical Intervals for a Normal Population, Part I: Tables, Examples and Applications. *Journal of Quality Technology* **2**(3), 115-125.

Hahn, G.J. (1970c). Statistical Intervals for a Normal Population, Part II: Formulas, Assumptions, Some Derivations. *Journal of Quality Technology* **2**(4), 195-206.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Jilek, M. (1988). *Statisticke Tolerancni Meze*. SNTL, Praha.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Janiga, I., and R. Miklos. (2001). Statistical Tolerance Intervals for a Normal Distribution. *Measurement Science Review* **1**1, 29-32.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Odeh, R.E. (1978). Tables of Two-Sided Tolerance Factors for a Normal Distribution. *Communications in Statistics, Part B-Simulation and Computation* **7**, 183-201.

Odeh, R.E., and D.B. Owen. (1980). *Tables for Normal Tolerance Limits, Sampling Plans, and Screening*. Marcel Dekker, New York.

Owen, D.B. (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, MA.

Singh, A., R. Maichle, and N. Armbya. (2010a). *ProUCL Version 4.1.00 User Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

Singh, A., N. Armbya, and A. Singh. (2010b). *ProUCL Version 4.1.00 Technical Guide (Draft)*. EPA/600/R-07/041, May 2010. Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Wald, A., and J. Wolfowitz. (1946). Tolerance Limits for a Normal Distribution. *Annals of Mathematical Statistics* **17**, 208-215.

See Also

tolIntNorm, predIntNorm, Normal, estimate.object, enorm, eqnorm, Tolerance Intervals, Prediction Intervals, Estimating Distribution Parameters, Estimating Distribution Quantiles.

Examples

```
# Compute the value of K for a two-sided 95% beta-content
# tolerance interval with associated confidence level 95%
# given a sample size of n=20.

#----------
# Exact method

tolIntNormK(n = 20)
#[1] 2.760346

#----------
# Approximate method due to Wald and Wolfowitz (1946)

tolIntNormK(n = 20, method = "wald")
# [1] 2.751789


#----------------------------------------------------------------------

# Compute the value of K for a one-sided upper tolerance limit
# with 99% coverage and associated confidence level 90%
# given a samle size of n=20.

tolIntNormK(n = 20, ti.type = "upper", coverage = 0.99,
  conf.level = 0.9)
#[1] 3.051543

#----------------------------------------------------------------------

# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence  using chrysene data and assuming a lognormal
# distribution.  The sample size is n = 8 observations from
# the two compliance wells.  Here we will compute the
```

```
# multiplier for the log-transformed data.

tolIntNormK(n = 8, ti.type = "upper")
#[1] 3.187294
```

`tolIntNormN` *Sample Size for a Specified Half-Width of a Tolerance Interval for a Normal Distribution*

Description

Compute the sample size necessary to achieve a specified half-width of a tolerance interval for a normal distribution, given the estimated standard deviation, coverage, and confidence level.

Usage

```
tolIntNormN(half.width, sigma.hat = 1, coverage = 0.95, cov.type = "content",
  conf.level = 0.95, method = "wald.wolfowitz", round.up = TRUE, n.max = 5000,
  tol = 1e-07, maxiter = 1000)
```

Arguments

`half.width`	numeric vector of (positive) half-widths. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`sigma.hat`	numeric vector specifying the value(s) of the estimated standard deviation(s). The default value is `sigma.hat=1`.
`coverage`	numeric vector of values between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation).
`conf.level`	numeric vector of values between 0 and 1 indicating the confidence level of the prediction interval. The default value is `conf.level=0.95`.
`method`	character string specifying the method for constructing the tolerance interval. The possible values are `"exact"` (the default) and `"wald.wolfowitz"` (the Wald-Wolfowitz approximation).
`round.up`	logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is `round.up=TRUE`.
`n.max`	positive integer greater than 1 specifying the maximum possible sample size. The default value is `n.max=5000`.
`tol`	numeric scalar indicating the tolerance to use in the `uniroot` search algorithm. The default value is `tol=1e-7`.
`maxiter`	positive integer indicating the maximum number of iterations to use in the `uniroot` search algorithm. The default value is `maxiter=1000`.

Details

If the arguments `half.width`, `sigma.hat`, `coverage`, and `conf.level` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help files for `tolIntNorm` and `tolIntNormK` give formulas for a two-sided tolerance interval based on the sample size, the observed sample mean and sample standard deviation, and specified confidence level and coverage. Specifically, the two-sided tolerance interval is given by:

$$[\bar{x} - Ks, \bar{x} + Ks] \qquad (1)$$

where $\bar{x}$ denotes the sample mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

s denotes the sample standard deviation:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (3)$$

and K denotes a constant that depends on the sample size n, the confidence level, and the coverage (see the help file for `tolIntNormK`). Thus, the half-width of the tolerance interval is given by:

$$HW = Ks \qquad (4)$$

The function `tolIntNormN` uses the `uniroot` search algorithm to determine the sample size for specified values of the half-width, sample standard deviation, coverage, and confidence level. **Note that unlike a confidence interval, the half-width of a tolerance interval does *not* approach 0 as the sample size increases.**

Value

numeric vector of sample sizes.

Note

See the help file for `tolIntNorm`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, confidence level, and half-width if one of the objectives of the sampling program is to produce tolerance intervals. The functions `tolIntNormHalfWidth`, `tolIntNormN`, and `plotTolIntNormDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `tolIntNorm`.

See Also

tolIntNorm, tolIntNormK, tolIntNormHalfWidth, plotTolIntNormDesign, Normal.

Examples

```
# Look at how the required sample size for a tolerance interval increases
# with increasing coverage:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

tolIntNormN(half.width = 3, coverage = seq(0.5, 0.9, by = 0.1))
#[1] 4 4 5 6 9

#----------

# Look at how the required sample size for a tolerance interval decreases
# with increasing half-width:

3:6
#[1] 3 4 5 6

tolIntNormN(half.width = 3:6)
#[1] 15 8 6 5

tolIntNormN(3:6, round = FALSE)
#[1] 14.199735  7.022572  5.092374  4.214371

#----------

# Look at how the required sample size for a tolerance interval increases
# with increasing estimated standard deviation for a fixed half-width:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

tolIntNormN(half.width = 4, sigma.hat = seq(0.5, 2, by = 0.5))
#[1]    4    8   24 3437

#----------

# Look at how the required sample size for a tolerance interval increases
# with increasing confidence level for a fixed half-width:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

tolIntNormN(half.width = 3, conf.level = seq(0.5, 0.9, by = 0.1))
#[1]  3  4  5  7 11

#==========
```

```
# Example 17-3 of USEPA (2009, p. 17-17) shows how to construct a
# beta-content upper tolerance limit with 95% coverage and 95%
# confidence  using chrysene data and assuming a lognormal distribution.
# The data for this example are stored in EPA.09.Ex.17.3.chrysene.df,
# which contains chrysene concentration data (ppb) found in water
# samples obtained from two background wells (Wells 1 and 2) and
# three compliance wells (Wells 3, 4, and 5).  The tolerance limit
# is based on the data from the background wells.

# Here we will first take the log of the data and then estimate the
# standard deviation based on the two background wells.  We will use this
# estimate of standard deviation to compute required sample sizes for
# various half-widths on the log-scale.

head(EPA.09.Ex.17.3.chrysene.df)
#  Month   Well  Well.type Chrysene.ppb
#1     1 Well.1 Background         19.7
#2     2 Well.1 Background         39.2
#3     3 Well.1 Background          7.8
#4     4 Well.1 Background         12.8
#5     1 Well.2 Background         10.2
#6     2 Well.2 Background          7.2

longToWide(EPA.09.Ex.17.3.chrysene.df, "Chrysene.ppb", "Month", "Well")
#  Well.1 Well.2 Well.3 Well.4 Well.5
#1   19.7   10.2   68.0   26.8   47.0
#2   39.2    7.2   48.9   17.7   30.5
#3    7.8   16.1   30.1   31.9   15.0
#4   12.8    5.7   38.1   22.2   23.4

summary.stats <- summaryStats(log(Chrysene.ppb) ~ Well.type,
  data = EPA.09.Ex.17.3.chrysene.df)

summary.stats
#            N   Mean     SD Median    Min    Max
#Background  8 2.5086 0.6279 2.4359 1.7405 3.6687
#Compliance 12 3.4173 0.4361 3.4111 2.7081 4.2195

sigma.hat <- summary.stats["Background", "SD"]
sigma.hat
#[1] 0.6279

tolIntNormN(half.width = c(4, 2, 1), sigma.hat = sigma.hat)
#[1]  4 12 NA
#Warning message:
#In tolIntNormN(half.width = c(4, 2, 1), sigma.hat = sigma.hat) :
#  Value of 'half.width' is too smallfor element3.
#  Try increasing the value of 'n.max'.

# NOTE:  We cannot achieve a half-width of 1 for the given value of
#        sigma.hat for a tolerance interval with 95% coverage and
#        95% confidence.  The default value of n.max is 5000, but in fact,
#        even with a million observations the half width is greater than 1.
```

```
tolIntNormHalfWidth(n = 1e6, sigma.hat = sigma.hat)
#[1] 1.232095

#==========

# Clean up
#---------
rm(summary.stats, sigma.hat)
```

---

tolIntNpar *Nonparametric Tolerance Interval for a Continuous Distribution*

Description

Construct a β-content or β-expectation tolerance interval nonparametrically without making any assumptions about the form of the distribution except that it is continuous.

Usage

```
tolIntNpar(x, coverage, conf.level, cov.type = "content",
  ltl.rank = ifelse(ti.type == "upper", 0, 1),
  n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1),
  lb = -Inf, ub = Inf, ti.type = "two-sided")
```

Arguments

x	numeric vector of observations. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
coverage	a scalar between 0 and 1 indicating the desired coverage of the β-content tolerance interval. The default value is coverage=0.95. If cov.type="content", you must supply a value for coverage or a value for conf.level, but not both. If cov.type="expectation", this argument is ignored.
conf.level	a scalar between 0 and 1 indicating the confidence level associated with the β-content tolerance interval. The default value is conf.level=0.95. If cov.type="content", you must supply a value for coverage or a value for conf.level, but not both. If cov.type="expectation", this argument is ignored.
cov.type	character string specifying the coverage type for the tolerance interval. The possible values are "content" (β-content; the default), and "expectation" (β-expectation). See the DETAILS section for more information.
ltl.rank	positive integer indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If ti.type="two-sided" or ti.type="lower", the default value is ltl.rank=1 (implying the minimum value of x is used as the lower bound of the tolerance interval). If ti.type="upper", this argument is set equal to 0 and the value of lb is used as the lower bound of the tolerance interval.

`n.plus.one.minus.utl.rank`
: positive integer related to the rank of the order statistic to use for the upper bound of the toleracne interval. A value of `n.plus.one.minus.utl.rank=1` (the default) means use the first largest value of `x`, and in general a value of `n.plus.one.minus.utl.rank=`i means use the i'th largest value. If `ti.type="lower"`, this argument is set equal to `0` and the value of `ub` is used as the upper bound of the tolerance interval.

`lb`, `ub`
: scalars indicating lower and upper bounds on the distribution. By default, `lb=-Inf` and `ub=Inf`. If you are constructing a tolerance interval for a distribution that you know has a lower bound other than `-Inf` (e.g., `0`), set `lb` to this value. Similarly, if you know the distribution has an upper bound other than `Inf`, set `ub` to this value. The argument `lb` is ignored if `ti.type="two-sided"` or `ti.type="lower"`. The argument `ub` is ignored if `ti.type="two-sided"` or `ti.type="upper"`.

`ti.type`
: character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

Details

A tolerance interval for some population is an interval on the real line constructed so as to contain $100\beta\%$ of the population (i.e., $100\beta\%$ of all future observations), where $0 < \beta < 1$. The quantity $100\beta\%$ is called the *coverage*.

There are two kinds of tolerance intervals (Guttman, 1970):

- A β-content tolerance interval with confidence level $100(1-\alpha)\%$ is constructed so that it contains at least $100\beta\%$ of the population (i.e., the coverage is at least $100\beta\%$) with probability $100(1-\alpha)\%$, where $0 < \alpha < 1$. The quantity $100(1-\alpha)\%$ is called the confidence level or confidence coefficient associated with the tolerance interval.
- A β-expectation tolerance interval is constructed so that the *average* coverage of the interval is $100\beta\%$.

Note: A β-expectation tolerance interval with coverage $100\beta\%$ is equivalent to a prediction interval for one future observation with associated confidence level $100\beta\%$. Note that there is no explicit confidence level associated with a β-expectation tolerance interval. If a β-expectation tolerance interval is treated as a β-content tolerance interval, the confidence level associated with this tolerance interval is usually around 50% (e.g., Guttman, 1970, Table 4.2, p.76).

The Form of a Nonparametric Tolerance Interval

Let $\underline{x}$ denote a random sample of n independent observations from some continuous distribution and let $x_{(i)}$ denote the i'th order statistic in $\underline{x}$. A two-sided nonparametric tolerance interval is constructed as:

$$[x_{(u)}, x_{(v)}] \qquad (1)$$

where u and v are positive integers between 1 and n, and $u < v$. That is, u denotes the rank of the lower tolerance limit, and v denotes the rank of the upper tolerance limit. To make it easier to write some equations later on, we can also write the tolerance interval (1) in a slightly different way as:

$$[x_{(u)}, x_{(n+1-w)}] \qquad (2)$$

where

$$w = n + 1 - v \qquad (3)$$

so that w is a positive integer between 1 and $n - 1$, and $u < n + 1 - w$. In terms of the arguments to the function `tolIntNpar`, the argument `ltl.rank` corresponds to u, and the argument `n.plus.one.minus.utl.rank` corresponds to w.

If we allow $u = 0$ and $w = 0$ and define lower and upper bounds as:

$$x_{(0)} = lb \qquad (4)$$

$$x_{(n+1)} = ub \qquad (5)$$

then equation (2) above can also represent a one-sided lower or one-sided upper tolerance interval as well. That is, a one-sided lower nonparametric tolerance interval is constructed as:

$$[x_{(u)}, x_{(n+1)}] = [x_{(u)}, ub] \qquad (6)$$

and a one-sided upper nonparametric tolerance interval is constructed as:

$$[x_{(0)}, x_{(v)}] = [lb, x_{(v)}] \qquad (7)$$

Usually, $lb = -\infty$ or $lb = 0$ and $ub = \infty$.

Let C be a random variable denoting the coverage of the above nonparametric tolerance intervals. Wilks (1941) showed that the distribution of C follows a beta distribution with parameters `shape1`$= v - u$ and `shape2`$= w + u$ when the unknown distribution is continuous.

Computations for a β-Content Tolerance Interval

For a β-content tolerance interval, if the coverage $C = \beta$ is specified, then the associated confidence level $(1 - \alpha)100\%$ is computed as:

$$1 - \alpha = 1 - F(\beta, v - u, w + u) \qquad (8)$$

where $F(y, \delta, \gamma)$ denotes the cumulative distribution function of a beta random variable with parameters `shape1`$=\delta$ and `shape2`$=\gamma$ evaluated at y.

Similarly, if the confidence level associated with the tolerance interval is specified as $(1 - \alpha)100\%$, then the coverage $C = \beta$ is computed as:

$$\beta = B(\alpha, v - u, w + u) \qquad (9)$$

where $B(p, \delta, \gamma)$ denotes the p'th quantile of a beta distribution with parameters `shape1`$=\delta$ and `shape2`$=\gamma$.

Computations for a β-Expectation Tolerance Interval

For a β-expectation tolerance interval, the expected coverage is simply the mean of a beta random variable with parameters `shape1`$= v - u$ and `shape2`$= w + u$, which is given by:

$$E(C) = \frac{v - u}{n + 1} \qquad (10)$$

As stated above, a β-expectation tolerance interval with coverage $\beta 100\%$ is equivalent to a prediction interval for one future observation with associated confidence level $\beta 100\%$. This is because the probability that any single future observation will fall into this interval is $\beta 100\%$, so the distribution of the number of N future observations that will fall into this interval is binomial with parameters `size`$=N$ and `prob`$=\beta$. Hence the expected proportion of future observations that fall into this interval is $\beta 100\%$ and is independent of the value of N. See the help file for `predIntNpar` for more information on constructing a nonparametric prediction interval.

Value

A list of class `"estimate"` containing the estimated parameters, the tolerance interval, and other information. See `estimate.object` for details.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York.

Danziger, L., and S. Davis. (1964). Tables of Distribution-Free Tolerance Limits. *Annals of Mathematical Statistics* **35**(5), 1361–1365.

Davis, C.B. (1994). Environmental Regulatory Statistics. In Patil, G.P., and C.R. Rao, eds., *Handbook of Statistics, Vol. 12: Environmental Statistics*. North-Holland, Amsterdam, a division of Elsevier, New York, NY, Chapter 26, 817–865.

Davis, C.B., and R.J. McNichols. (1994a). Ground Water Monitoring Statistics Update: Part I: Progress Since 1988. *Ground Water Monitoring and Remediation* **14**(4), 148–158.

Gibbons, R.D. (1991b). Statistical Tolerance Limits for Ground-Water Monitoring. *Ground Water* **29**, 563–570.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT, Chapter 2.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York, 392pp.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, pp.88-90.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

Wilks, S.S. (1941). Determination of Sample Sizes for Setting Tolerance Limits. *Annals of Mathematical Statistics* **12**, 91–96.

See Also

eqnpar, estimate.object, tolIntNparN, Tolerance Intervals, Estimating Distribution Parameters, Estimating Distribution Quantiles.

Examples

```
# Generate 20 observations from a lognormal mixture distribution
# with parameters mean1=1, cv1=0.5, mean2=5, cv2=1, and p.mix=0.1.
# The exact two-sided interval that contains 90% of this distribution is given by:
# [0.682312, 13.32052].  Use tolIntNpar to construct a two-sided 90%
# \eqn{\beta}-content tolerance interval.  Note that the associated confidence level
# is only 61%.  A larger sample size is required to obtain a larger confidence
# level (see the help file for tolIntNparN).
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(23)
dat <- rlnormMixAlt(20, 1, 0.5, 5, 1, 0.1)
tolIntNpar(dat, coverage = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            dat
#
#Sample Size:                     20
#
#Tolerance Interval Coverage:     90%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         two-sided
#
#Confidence Level:                60.8253%
#
#Tolerance Limit Rank(s):         1 20
#
#Tolerance Interval:              LTL = 0.5035035
#                                 UTL = 9.9504662

#----------

# Clean up
rm(dat)
```

```
#----------

# Reproduce Example 17-4 on page 17-21 of USEPA (2009).  This example uses
# copper concentrations (ppb) from 3 background wells to set an upper
# limit for 2 compliance wells.  The maximum value from the 3 wells is set
# to the 95% confidence upper tolerance limit, and we need to determine the
# coverage of this tolerance interval.  The data are stored in EPA.92c.copper2.df.
# Note that even though these data are Type I left singly censored, it is still
# possible to compute an upper tolerance interval using any of the uncensored
# observations as the upper limit.

EPA.92c.copper2.df
#   Copper.orig Copper Censored Month Well  Well.type
#1           <5    5.0     TRUE     1    1 Background
#2           <5    5.0     TRUE     2    1 Background
#3          7.5    7.5    FALSE     3    1 Background
#...
#9          9.2    9.2    FALSE     1    2 Background
#10          <5    5.0     TRUE     2    2 Background
#11          <5    5.0     TRUE     3    2 Background
#...
#17          <5    5.0     TRUE     1    3 Background
#18         5.4    5.4    FALSE     2    3 Background
#19         6.7    6.7    FALSE     3    3 Background
#...
#29         6.2    6.2    FALSE     5    4 Compliance
#30          <5    5.0     TRUE     6    4 Compliance
#31         7.8    7.8    FALSE     7    4 Compliance
#...
#38          <5    5.0     TRUE     6    5 Compliance
#39         5.6    5.6    FALSE     7    5 Compliance
#40          <5    5.0     TRUE     8    5 Compliance

with(EPA.92c.copper2.df,
  tolIntNpar(Copper[Well.type=="Background"],
    conf.level = 0.95, lb = 0, ti.type = "upper"))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            Copper[Well.type == "Background"]
#
#Sample Size:                     24
#
#Tolerance Interval Coverage:     88.26538%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         upper
```

```
#
#Confidence Level:                95%
#
#Tolerance Limit Rank(s):         24
#
#Tolerance Interval:              LTL = 0.0
#                                 UTL = 9.2

#----------

# Repeat the last example, except compute an upper
# \eqn{\beta}-expectation tolerance interval:

with(EPA.92c.copper2.df,
  tolIntNpar(Copper[Well.type=="Background"],
    cov.type = "expectation", lb = 0, ti.type = "upper"))

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            None
#
#Data:                            Copper[Well.type == "Background"]
#
#Sample Size:                     24
#
#Tolerance Interval Coverage:     96%
#
#Coverage Type:                   expectation
#
#Tolerance Interval Method:       Exact
#
#Tolerance Interval Type:         upper
#
#Tolerance Limit Rank(s):         24
#
#Tolerance Interval:              LTL = 0.0
#                                 UTL = 9.2
```

`tolIntNparConfLevel` *Confidence Level for Nonparametric Tolerance Interval for Continuous Distribution*

Description

Compute the confidence level associated with a nonparametric β-content tolerance interval for a continuous distribution given the sample size, coverage, and ranks of the order statistics used for the interval.

Usage

```
tolIntNparConfLevel(n, coverage = 0.95,
  ltl.rank = ifelse(ti.type == "upper", 0, 1),
  n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1),
  ti.type = "two.sided")
```

Arguments

`n`	vector of positive integers specifying the sample sizes. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`coverage`	numeric vector of values between 0 and 1 indicating the desired coverage of the β-content tolerance interval.
`ltl.rank`	vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If `ti.type="two-sided"` or `ti.type="lower"`, the default value is `ltl.rank=1` (implying the minimum value of `x` is used as the lower bound of the tolerance interval). If `ti.type="upper"`, this argument is set equal to `0`.
`n.plus.one.minus.utl.rank`	vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of `n.plus.one.minus.utl.rank=1` (the default) means use the first largest value, and in general a value of `n.plus.one.minus.utl.rank=`i means use the i'th largest value. If `ti.type="lower"`, this argument is set equal to `0`.
`ti.type`	character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

Details

If the arguments `n`, `coverage`, `ltl.rank`, and `n.plus.one.minus.utl.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for `tolIntNpar` explains how nonparametric β-content tolerance intervals are constructed and how the confidence level associated with the tolerance interval is computed based on specified values for the sample size, the coverage, and the ranks of the order statistics used for the bounds of the tolerance interval.

Value

vector of values between 0 and 1 indicating the confidence level associated with the specified nonparametric tolerance interval.

Note

See the help file for `tolIntNpar`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions `tolIntNparN`, `tolIntNparCoverage`, `tolIntNparConfLevel`, and `plotTolIntNparDesign` can be used to investigate these relationships for constructing nonparametric tolerance intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for tolIntNpar.

See Also

tolIntNpar, tolIntNparN, tolIntNparCoverage, plotTolIntNparDesign.

Examples

```
# Look at how the confidence level of a nonparametric tolerance interval increases with
# increasing sample size:

seq(10, 60, by=10)
#[1] 10 20 30 40 50 60

round(tolIntNparConfLevel(n = seq(10, 60, by = 10)), 2)
#[1] 0.09 0.26 0.45 0.60 0.72 0.81

#----------

# Look at how the confidence level of a nonparametric tolerance interval decreases with
# increasing coverage:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

round(tolIntNparConfLevel(n = 10, coverage = seq(0.5, 0.9, by = 0.1)), 2)
#[1] 0.99 0.95 0.85 0.62 0.26

#----------

# Look at how the confidence level of a nonparametric tolerance interval decreases with the
# rank of the lower tolerance limit:

round(tolIntNparConfLevel(n = 60, ltl.rank = 1:5), 2)
#[1] 0.81 0.58 0.35 0.18 0.08

#==========

# Example 17-4 on page 17-21 of USEPA (2009) uses copper concentrations (ppb) from 3
# background wells to set an upper limit for 2 compliance wells.  There are 6 observations
# per well, and the maximum value from the 3 wells is set to the 95% confidence upper
# tolerance limit, and we need to determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
#[1] 0.8826538

# Here we will modify the example and determine the confidence level of the tolerance
```

```
# interval when we set the coverage to 95%.

tolIntNparConfLevel(n = 24, coverage = 0.95, ti.type = "upper")
# [1] 0.708011
```

`tolIntNparCoverage` *Coverage for Nonparametric Tolerance Interval for Continuous Distribution*

Description

Compute the coverage associated with a nonparametric tolerance interval for a continuous distribution given the sample size, confidence level, coverage type (β-content versus β-expectation), and ranks of the order statistics used for the interval.

Usage

```
tolIntNparCoverage(n, conf.level = 0.95, cov.type = "content",
  ltl.rank = ifelse(ti.type == "upper", 0, 1),
  n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1),
  ti.type = "two.sided")
```

Arguments

`n`	vector of positive integers specifying the sample sizes. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are not allowed.
`conf.level`	numeric vector of values between 0 and 1 indicating the confidence level of the tolerance interval.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation).
`ltl.rank`	vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If `ti.type="two-sided"` or `ti.type="lower"`, the default value is `ltl.rank=1` (implying the minimum value of `x` is used as the lower bound of the tolerance interval). If `ti.type="upper"`, this argument is set equal to `0`.
`n.plus.one.minus.utl.rank`	vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of `n.plus.one.minus.utl.rank=1` (the default) means use the first largest value, and in general a value of `n.plus.one.minus.utl.rank=`i means use the i'th largest value. If `ti.type="lower"`, this argument is set equal to `0`.
`ti.type`	character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

Details

If the arguments `n`, `conf.level`, `ltl.rank`, and `n.plus.one.minus.utl.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for `tolIntNpar` explains how nonparametric β-content tolerance intervals are constructed and how the coverage associated with the tolerance interval is computed based on specified values for the sample size, the confidence level, and the ranks of the order statistics used for the bounds of the tolerance interval.

Value

vector of values between 0 and 1 indicating the coverage associated with the specified nonparametric tolerance interval.

Note

See the help file for `tolIntNpar`.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions `tolIntNparN`, `tolIntNparConfLevel`, `tolIntNparCoverage`, and `plotTolIntNparDesign` can be used to investigate these relationships for constructing nonparametric tolerance intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for `tolIntNpar`.

See Also

`tolIntNpar`, `tolIntNparN`, `tolIntNparConfLevel`, `plotTolIntNparDesign`.

Examples

```
# Look at how the coverage of a nonparametric tolerance interval increases with
# increasing sample size:

seq(10, 60, by=10)
#[1] 10 20 30 40 50 60

round(tolIntNparCoverage(n = seq(10, 60, by = 10)), 2)
#[1] 0.61 0.78 0.85 0.89 0.91 0.92

#---------

# Look at how the coverage of a nonparametric tolerance interval decreases with
# increasing confidence level:
```

```
seq(0.5, 0.9, by=0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

round(tolIntNparCoverage(n = 10, conf.level = seq(0.5, 0.9, by = 0.1)), 2)
#[1] 0.84 0.81 0.77 0.73 0.66

#----------

# Look at how the coverage of a nonparametric tolerance interval decreases with
# the rank of the lower tolerance limit:

round(tolIntNparCoverage(n = 60, ltl.rank = 1:5), 2)
#[1] 0.92 0.90 0.88 0.85 0.83

#==========

# Example 17-4 on page 17-21 of USEPA (2009) uses copper concentrations (ppb) from 3
# background wells to set an upper limit for 2 compliance wells.  The maximum value from
# the 3 wells is set to the 95% confidence upper tolerance limit, and we need to
# determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
#[1] 0.8826538
```

`tolIntNparN` *Sample Size for Nonparametric Tolerance Interval for Continuous Distribution*

Description

Compute the sample size necessary for a nonparametric tolerance interval (for a continuous distribution) with a specified coverage and, in the case of a β-content tolerance interval, a specified confidence level, given the ranks of the order statistics used for the interval.

Usage

```
tolIntNparN(coverage = 0.95, conf.level = 0.95, cov.type = "content",
  ltl.rank = ifelse(ti.type == "upper", 0, 1),
  n.plus.one.minus.utl.rank = ifelse(ti.type == "lower", 0, 1),
  ti.type = "two.sided")
```

Arguments

`coverage`	numeric vector of values between 0 and 1 indicating the desired coverage of the tolerance interval.
`conf.level`	numeric vector of values between 0 and 1 indicating the confidence level of the tolerance interval.

`cov.type` character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation).

`ltl.rank` vector of positive integers indicating the rank of the order statistic to use for the lower bound of the tolerance interval. If `ti.type="two-sided"` or `ti.type="lower"`, the default value is `ltl.rank=1` (implying the minimum value of `x` is used as the lower bound of the tolerance interval). If `ti.type="upper"`, this argument is set equal to `0`.

`n.plus.one.minus.utl.rank` vector of positive integers related to the rank of the order statistic to use for the upper bound of the tolerance interval. A value of `n.plus.one.minus.utl.rank=1` (the default) means use the first largest value, and in general a value of `n.plus.one.minus.utl.rank=`i means use the i'th largest value. If `ti.type="lower"`, this argument is set equal to `0`.

`ti.type` character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

Details

If the arguments `coverage`, `conf.level`, `ltl.rank`, and `n.plus.one.minus.utl.rank` are not all the same length, they are replicated to be the same length as the length of the longest argument.

The help file for `tolIntNpar` explains how nonparametric tolerance intervals are constructed.

Computing Required Sample Size for a β-Content Tolerance Interval (`cov.type="content"`)
For a β-content tolerance interval, if the coverage $C = \beta$ is specified, then the associated confidence level $(1 - \alpha)100\%$ is computed as:

$$1 - \alpha = 1 - F(\beta, v - u, w + u) \qquad (1)$$

where $F(y, \delta, \gamma)$ denotes the cumulative distribution function of a beta random variable with parameters `shape1=`δ and `shape2=`γ evaluated at y. The value of $1 - \alpha$ is determined by the argument `conf.level`. The value of β is determined by the argument `coverage`. The value of u is determined by the argument `ltl.rank`. The value of w is determined by the argument `n.plus.one.minus.utl.rank`. Once these values have been determined, the above equation can be solved implicitly for n, since

$$v = n + 1 - w \qquad (2)$$

Computing Required Sample Size for a β-Expectation Tolerance Interval (`cov.type="expectation"`)
For a β-expectation tolerance interval, the expected coverage is simply the mean of a beta random variable with parameters `shape1=`$v - u$ and `shape2=`$w + u$, which is given by:

$$E(C) = \frac{v - u}{n + 1} \qquad (3)$$

or, using Equation (2) above, we can re-write the formula for the expected coverage as:

$$E(C) = \frac{n + 1 - w - u}{n + 1} = 1 - \frac{u + w}{n + 1} \qquad (4)$$

Thus, for user-specified values of u (`ltl.rank`), w (`n.plus.one.minus.utl.rank`), and expected coverage, the required sample size is computed as:

$$n = Ceiling\{[\frac{u + w}{1 - E(C)}] - 1\} \qquad (5)$$

where $Ceiling(x)$ denotes the smallest integer greater than or equal to x. (See the R help file for ceiling).

Value

A vector of positive integers indicating the required sample size(s) for the specified nonparametric tolerance interval(s).

Note

See the help file for tolIntNpar.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, coverage, and confidence level if one of the objectives of the sampling program is to produce tolerance intervals. The functions tolIntNparN, tolIntNparCoverage, tolIntNparConfLevel, and plotTolIntNparDesign can be used to investigate these relationships for constructing nonparametric tolerance intervals.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See the help file for tolIntNpar.

See Also

tolIntNpar, tolIntNparConfLevel, tolIntNparCoverage, plotTolIntNparDesign.

Examples

```
# Look at how the required sample size for a nonparametric tolerance interval increases
# with increasing confidence level:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

tolIntNparN(conf.level = seq(0.5, 0.9, by = 0.1))
#[1] 34 40 49 59 77

#----------

# Look at how the required sample size for a nonparametric tolerance interval increases
# with increasing coverage:

tolIntNparN(coverage = seq(0.5, 0.9, by = 0.1))
#[1]  8 10 14 22 46

#----------

# Look at how the required sample size for a nonparametric tolerance interval increases
```

```
# with the rank of the lower tolerance limit:

tolIntNparN(ltl.rank = 1:5)
#[1]  93 124 153 181 208

#==========

# Example 17-4 on page 17-21 of USEPA (2009) uses copper concentrations (ppb) from 3
# background wells to set an upper limit for 2 compliance wells.  The maximum value from
# the 3 wells is set to the 95% confidence upper tolerance limit, and we need to
# determine the coverage of this tolerance interval.

tolIntNparCoverage(n = 24, conf.level = 0.95, ti.type = "upper")
#[1] 0.8826538

# Here we will modify the example and determine the sample size required to produce
# a tolerance interval with 95% confidence level AND 95% coverage.

tolIntNparN(coverage = 0.95, conf.level = 0.95, ti.type = "upper")
#[1] 59
```

`tolIntPois` *Tolerance Interval for a Poisson Distribution*

Description

Construct a β-content or β-expectation tolerance interval for a Poisson distribution.

Usage

```
tolIntPois(x, coverage = 0.95, cov.type = "content", ti.type = "two-sided",
  conf.level = 0.95)
```

Arguments

`x`	numeric vector of observations, or an object resulting from a call to an estimating function that assumes a Poisson distribution (i.e., `epois` or `epoisCensored`). If `cov.type="content"` then `x` must be a numeric vector. If `x` is a numeric vector, missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`coverage`	a scalar between 0 and 1 indicating the desired coverage of the tolerance interval. The default value is `coverage=0.95`. If `cov.type="expectation"`, this argument is ignored.
`cov.type`	character string specifying the coverage type for the tolerance interval. The possible values are `"content"` (β-content; the default), and `"expectation"` (β-expectation). See the DETAILS section for more information.
`ti.type`	character string indicating what kind of tolerance interval to compute. The possible values are `"two-sided"` (the default), `"lower"`, and `"upper"`.

`conf.level` a scalar between 0 and 1 indicating the confidence level associated with the tolerance interval. The default value is `conf.level=0.95`.

Details

If `x` contains any missing (`NA`), undefined (`NaN`) or infinite (`Inf`, `-Inf`) values, they will be removed prior to performing the estimation.

A tolerance interval for some population is an interval on the real line constructed so as to contain $100\beta\%$ of the population (i.e., $100\beta\%$ of all future observations), where $0 < \beta < 1$. The quantity $100\beta\%$ is called the *coverage*.

There are two kinds of tolerance intervals (Guttman, 1970):

- A β-content tolerance interval with confidence level $100(1-\alpha)\%$ is constructed so that it contains at least $100\beta\%$ of the population (i.e., the coverage is at least $100\beta\%$) with probability $100(1-\alpha)\%$, where $0 < \alpha < 1$. The quantity $100(1-\alpha)\%$ is called the confidence level or confidence coefficient associated with the tolerance interval.
- A β-expectation tolerance interval is constructed so that the *average* coverage of the interval is $100\beta\%$.

Note: A β-expectation tolerance interval with coverage $100\beta\%$ is equivalent to a prediction interval for one future observation with associated confidence level $100\beta\%$. Note that there is no explicit confidence level associated with a β-expectation tolerance interval. If a β-expectation tolerance interval is treated as a β-content tolerance interval, the confidence level associated with this tolerance interval is usually around 50% (e.g., Guttman, 1970, Table 4.2, p.76).

Because of the discrete nature of the Poisson distribution, even true tolerance intervals (tolerance intervals based on the true value of λ) will usually not contain exactly $\beta\%$ of the population. For example, for the Poisson distribution with parameter `lambda=2`, the interval [0, 4] contains 94.7% of this distribution and the interval [0, 5] contains 98.3% of this distribution. Thus, no interval can contain exactly 95% of this distribution.

β-Content Tolerance Intervals for a Poisson Distribution

Zacks (1970) showed that for monotone likelihood ratio (MLR) families of discrete distributions, a uniformly most accurate upper $\beta 100\%$ β-content tolerance interval with associated confidence level $(1-\alpha)100\%$ is constructed by finding the upper $(1-\alpha)100\%$ confidence limit for the parameter associated with the distribution, and then computing the β'th quantile of the distribution assuming the true value of the parameter is equal to the upper confidence limit. This idea can be extended to one-sided lower and two-sided tolerance limits.

It can be shown that all distributions that are one parameter exponential families have the MLR property, and the Poisson distribution is a one-parameter exponential family, so the method of Zacks (1970) can be applied to a Poisson distribution.

Let X denote a Poisson random variable with parameter `lambda=`λ. Let $x_{p|\lambda}$ denote the p'th quantile of this distribution. That is,

$$Pr(X < x_{p|\lambda}) \leq p \leq Pr(X \leq x_{p|\lambda}) \qquad (1)$$

Note that due to the discrete nature of the Poisson distribution, there will be several values of p associated with one value of X. For example, for $\lambda = 2$, the value 1 is the p'th quantile for any value of p between 0.140 and 0.406.

Let $\underline{x}$ denote a vector of n observations from a Poisson distribution with parameter `lambda=`λ. When `ti.type="upper"`, the first step is to compute the one-sided upper $(1-\alpha)100\%$ confidence limit for λ based on the observations $\underline{x}$ (see the help file for `epois`). Denote this upper confidence limit by UCL. The one-sided upper $\beta 100\%$ tolerance limit is then given by:

$$[0, x_{\beta|\lambda=UCL}] \qquad (2)$$

Similarly, when `ti.type="lower"`, the first step is to compute the one-sided lower $(1-\alpha)100\%$ confidence limit for λ based on the observations $\underline{x}$. Denote this lower confidence limit by LCL. The one-sided lower $\beta 100\%$ tolerance limit is then given by:

$$[x_{1-\beta|\lambda=LCL}, \infty] \qquad (3)$$

Finally, when `ti.type="two-sided"`, the first step is to compute the two-sided $(1-\alpha)100\%$ confidence limits for λ based on the observations $\underline{x}$. Denote these confidence limits by LCL and UCL. The two-sided $\beta 100\%$ tolerance limit is then given by:

$$[x_{\frac{1-\beta}{2}|\lambda=LCL}, x_{\frac{1+\beta}{2}|\lambda=UCL}] \qquad (4)$$

Note that the function `tolIntPois` uses the exact confidence limits for λ when computing β-content tolerance limits (see `epois`).

β-Expectation Tolerance Intervals for a Poisson Distribution

As stated above, a β-expectation tolerance interval with coverage $\beta 100\%$ is equivalent to a prediction interval for one future observation with associated confidence level $\beta 100\%$. This is because the probability that any single future observation will fall into this interval is $\beta 100\%$, so the distribution of the number of N future observations that will fall into this interval is binomial with parameters `size=`N and `prob=`β. Hence the expected proportion of future observations that fall into this interval is $\beta 100\%$ and is independent of the value of N. See the help file for `predIntPois` for information on how these intervals are constructed.

Value

If `x` is a numeric vector, `tolIntPois` returns a list of class `"estimate"` containing the estimated parameters, a component called `interval` containing the tolerance interval information, and other information. See `estimate.object` for details.

If `x` is the result of calling an estimation function, `tolIntPois` returns a list whose class is the same as `x`. The list contains the same components as `x`. If `x` already has a component called `interval`, this component is replaced with the tolerance interval information.

Note

Tolerance intervals have long been applied to quality control and life testing problems (Hahn, 1970b,c; Hahn and Meeker, 1991; Krishnamoorthy and Mathew, 2009). References that discuss tolerance intervals in the context of environmental monitoring include: Berthouex and Brown (2002, Chapter 21), Gibbons et al. (2009), Millard and Neerchal (2001, Chapter 6), Singh et al. (2010b), and USEPA (2009).

Gibbons (1987b) used the Poisson distribution to model the number of detected compounds per scan of the 32 volatile organic priority pollutants (VOC), and also to model the distribution of chemical

concentration (in ppb). He explained the derivation of a one-sided upper β-content tolerance limit for a Poisson distribution based on the work of Zacks (1970) using the Pearson-Hartley approximation to the confidence limits for the mean parameter λ (see the help file for epois). Note that there are several typographical errors in the derivation and examples on page 575 of Gibbons (1987b) because there is confusion between where the value of β (the coverage) should be and where the value of $1 - \alpha$ (the confidence level) should be. Gibbons et al. (2009, pp.103-104) gives correct formulas.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Gibbons, R.D. (1987b). Statistical Models for the Analysis of Volatile Organic Compounds in Waste Disposal Sites. *Ground Water* **25**, 572–580.

Gibbons, R.D., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*, Second Edition. John Wiley & Sons, Hoboken.

Guttman, I. (1970). *Statistical Tolerance Regions: Classical and Bayesian*. Hafner Publishing Co., Darien, CT.

Hahn, G.J., and W.Q. Meeker. (1991). *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, New York.

Johnson, N. L., S. Kotz, and A. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, Chapter 4.

Krishnamoorthy K., and T. Mathew. (2009). *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley and Sons, Hoboken.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton.

Zacks, S. (1970). Uniformly Most Accurate Upper Tolerance Limits for Monotone Likelihood Ratio Families of Discrete Distributions. *Journal of the American Statistical Association* **65**, 307–316.

See Also

Poisson, epois, eqpois, estimate.object, Tolerance Intervals, Estimating Distribution Parameters, Estimating Distribution Quantiles.

Examples

```
# Generate 20 observations from a Poisson distribution with parameter
# lambda=2. The interval [0, 4] contains 94.7% of this distribution and
# the interval [0,5] contains 98.3% of this distribution.  Thus, because
# of the discrete nature of the Poisson distribution, no interval contains
# exactly 95% of this distribution.  Use tolIntPois to estimate the mean
# parameter of the true distribution, and construct a one-sided upper 95%
# beta-content tolerance interval with associated confidence level 90%.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)
```

```
dat <- rpois(20, 2)
tolIntPois(dat, conf.level = 0.9)

#Results of Distribution Parameter Estimation
#--------------------------------------------
#
#Assumed Distribution:            Poisson
#
#Estimated Parameter(s):          lambda = 1.8
#
#Estimation Method:               mle/mme/mvue
#
#Data:                            dat
#
#Sample Size:                     20
#
#Tolerance Interval Coverage:     95%
#
#Coverage Type:                   content
#
#Tolerance Interval Method:       Zacks
#
#Tolerance Interval Type:         two-sided
#
#Confidence Level:                90%
#
#Tolerance Interval:              LTL = 0
#                                 UTL = 6

#------

# Clean up
rm(dat)
```

`Total.P.df` *Total Phosphorus Data from Chesapeake Bay*

Description

Monthly estimated total phosphorus mass (mg) within a water column at two different stations for the 5-year time period October 1984 to September 1989 from a study on phosphorus concentration conducted in the Chesapeake Bay.

Usage

```
Total.P.df
```

Format

A data frame with 60 observations on the following 4 variables.

`CB3.1` a numeric vector of phosphorus concentrations at station CB3.1

`CB3.3e` a numeric vector phosphorus concentrations at station CB3.3e

`Month` a factor indicating the month the observation was taken

`Year` a numeric vector indicating the year an observation was taken

Source

Neerchal, N. K., and S. L. Brunenmeister. (1993). Estimation of Trend in Chesapeake Bay Water Quality Data. In Patil, G.P., and C.R. Rao, eds., *Handbook of Statistics, Vol. 6: Multivariate Environmental Statistics*. North-Holland, Amsterdam, Chapter 19, 407-422.

Triangular *The Triangular Distribution*

Description

Density, distribution function, quantile function, and random generation for the triangular distribution with parameters `min`, `max`, and `mode`.

Usage

```
dtri(x, min = 0, max = 1, mode = 1/2)
ptri(q, min = 0, max = 1, mode = 1/2)
qtri(p, min = 0, max = 1, mode = 1/2)
rtri(n, min = 0, max = 1, mode = 1/2)
```

Arguments

`x`	vector of quantiles. Missing values (`NA`s) are allowed.
`q`	vector of quantiles. Missing values (`NA`s) are allowed.
`p`	vector of probabilities between 0 and 1. Missing values (`NA`s) are allowed.
`n`	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
`min`	vector of minimum values of the distribution of the random variable. The default value is `min=0`.
`max`	vector of maximum values of the random variable. The default value is `max=1`.
`mode`	vector of modes of the random variable. The default value is `mode=1/2`.

Details

Let X be a triangular random variable with parameters `min`=a, `max`=b, and `mode`=c.

Probability Density and Cumulative Distribution Function
The density function of X is given by:

$$f(x; a, b, c) = \begin{array}{ll} \frac{2(x-a)}{(b-a)(c-a)} & \text{for } a \leq x \leq c \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{for } c \leq x \leq b \end{array}$$

where $a < c < b$.

The cumulative distribution function of X is given by:

$$F(x; a, b, c) = \begin{array}{ll} \frac{(x-a)^2}{(b-a)(c-a)} & \text{for } a \leq x \leq c \\ 1 - \frac{(b-x)^2}{(b-a)(b-c)} & \text{for } c \leq x \leq b \end{array}$$

where $a < c < b$.

Quantiles
The p^th quantile of X is given by:

$$x_p = \begin{array}{ll} a + \sqrt{(b-a)(c-a)p} & \text{for } 0 \leq p \leq F(c) \\ b - \sqrt{(b-a)(b-c)(1-p} & \text{for } F(c) \leq p \leq 1 \end{array}$$

where $0 \leq p \leq 1$.

Random Numbers
Random numbers are generated using the inverse transformation method:

$$x = F^{-1}(u)$$

where u is a random deviate from a uniform $[0, 1]$ distribution.

Mean and Variance
The mean and variance of X are given by:

$$E(X) = \frac{a + b + c}{3}$$

$$Var(X) = \frac{a^2 + b^2 + c^2 - ab - ac - bc}{18}$$

Value

`dtri` gives the density, `ptri` gives the distribution function, `qtri` gives the quantile function, and `rtri` generates random deviates.

Note

The triangular distribution is so named because of the shape of its probability density function. The average of two independent identically distributed uniform random variables with parameters min=α and max=β has a triangular distribution with parameters min=α, max=β, and mode=$(\beta-\alpha)/2$.

The triangular distribution is sometimes used as an input distribution in probability risk assessment.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Forbes, C., M. Evans, N. Hastings, and B. Peacock. (2011). Statistical Distributions. Fourth Edition. John Wiley and Sons, Hoboken, NJ.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York.

See Also

Uniform, Probability Distributions and Random Numbers.

Examples

```
# Density of a triangular distribution with parameters
# min=10, max=15, and mode=12, evaluated at 12, 13 and 14:

dtri(12:14, 10, 15, 12)
#[1] 0.4000000 0.2666667 0.1333333

#----------

# The cdf of a triangular distribution with parameters
# min=2, max=7, and mode=5, evaluated at 3, 4, and 5:

ptri(3:5, 2, 7, 5)
#[1] 0.06666667 0.26666667 0.60000000

#----------

# The 25'th percentile of a triangular distribution with parameters
# min=1, max=4, and mode=3:

qtri(0.25, 1, 4, 3)
#[1] 2.224745

#----------

# A random sample of 4 numbers from a triangular distribution with
# parameters min=3 , max=20, and mode=12.
# (Note: the call to set.seed simply allows you to reproduce this example.)
```

```
set.seed(10)
rtri(4, 3, 20, 12)
#[1] 11.811593  9.850955 11.081885 13.539496
```

---

tTestAlpha *Type I Error Level for a One- or Two-Sample t-Test*

Description

Compute the Type I Error level necessary to achieve a specified power for a one- or two-sample t-test, given the sample size(s) and scaled difference.

Usage

```
tTestAlpha(n.or.n1, n2 = n.or.n1, delta.over.sigma = 0, power = 0.95,
 sample.type = ifelse(!missing(n2) && !is.null(n2), "two.sample", "one.sample"),
  alternative = "two.sided", approx = FALSE, tol = 1e-07, maxiter = 1000)
```

Arguments

n.or.n1	numeric vector of sample sizes. When sample.type="one.sample", n.or.n1 denotes n, the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes n_1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
n2	numeric vector of sample sizes for group 2. The default value is the value of n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
delta.over.sigma	numeric vector specifying the ratio of the true difference (δ) to the population standard deviation (σ). This is also called the "scaled difference".
power	numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.
sample.type	character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
approx	logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.
tol	numeric scalar indicating the tolerance argument to pass to the uniroot function. The default value is tol=1e-7.

`maxiter` positive integer indicating the maximum number of iterations argument to pass to the `uniroot` function. The default value is `maxiter=1000`.

Details

Formulas for the power of the t-test for specified values of the sample size, scaled difference, and Type I error level are given in the help file for `tTestPower`. The function `tTestAlpha` uses the `uniroot` search algorithm to determine the required Type I error level for specified values of the sample size, power, and scaled difference.

Value

numeric vector of Type I error levels.

Note

See `tTestPower`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See `tTestPower`.

See Also

`tTestPower`, `tTestScaledMdd`, `tTestN`, `plotTTestDesign`, Normal, `t.test`, Hypothesis Tests.

Examples

```
# Look at how the required Type I error level for the one-sample t-test
# decreases with increasing sample size.  Set the power to 80% and
# the scaled difference to 0.5.

seq(5, 30, by = 5)
#[1] 5 10 15 20 25 30

alpha <- tTestAlpha(n.or.n1 = seq(5, 30, by = 5),
  power = 0.8, delta.over.sigma = 0.5)

round(alpha, 2)
#[1] 0.65 0.45 0.29 0.18 0.11 0.07


#----------


# Repeat the last example, but use the approximation.
# Note how the approximation underestimates the power
# for the smaller sample sizes.
```

```
#--------------------------------------------------

alpha <- tTestAlpha(n.or.n1 = seq(5, 30, by = 5),
  power = 0.8, delta.over.sigma = 0.5, approx = TRUE)

round(alpha, 2)
#[1] 0.63 0.46 0.30 0.18 0.11 0.07


#----------


# Look at how the required Type I error level for the two-sample
# t-test decreases with increasing scaled difference.  Use
# a power of 90% and a sample size of 10 in each group.

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

alpha <- tTestAlpha(10, sample.type = "two.sample",
  power = 0.9, delta.over.sigma = seq(0.5, 2, by = 0.5))

round(alpha, 2)
#[1] 0.82 0.35 0.06 0.01


#----------


# Look at how the required Type I error level for the two-sample
# t-test increases with increasing values of required power.  Use
# a sample size of 20 for each group and a scaled difference of
# 1.

alpha <- tTestAlpha(20, sample.type = "two.sample", delta.over.sigma = 1,
  power = c(0.8, 0.9, 0.95))

round(alpha, 2)
#[1] 0.03 0.07 0.14


#----------


# Clean up
#---------
rm(alpha)
```

---

tTestLnormAltN *Sample Size for a One- or Two-Sample t-Test, Assuming Lognormal Data*

Description

Compute the sample size necessary to achieve a specified power for a one- or two-sample t-test, given the ratio of means, coefficient of variation, and significance level, assuming lognormal data.

Usage

```
tTestLnormAltN(ratio.of.means, cv = 1, alpha = 0.05, power = 0.95,
  sample.type = ifelse(!is.null(n2), "two.sample", "one.sample"),
  alternative = "two.sided", approx = FALSE, n2 = NULL, round.up = TRUE,
  n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

ratio.of.means	numeric vector specifying the ratio of the first mean to the second mean. When sample.type="one.sample", this is the ratio of the population mean to the hypothesized mean. When sample.type="two.sample", this is the ratio of the mean of the first population to the mean of the second population. The default value is ratio.of.means=1.
cv	numeric vector of positive value(s) specifying the coefficient of variation. When sample.type="one.sample", this is the population coefficient of variation. When sample.type="two.sample", this is the coefficient of variation for both the first and second population. The default value is cv=1.
alpha	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.
power	numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.
sample.type	character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
approx	logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.
n2	numeric vector of sample sizes for group 2. The default value is NULL in which case it is assumed that the sample sizes for groups 1 and 2 are equal. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
round.up	logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is TRUE.
n.max	positive integer greater than 1 indicating the maximum sample size when sample.type="one.sample" or the maximum sample size for group 1 when sample.type="two.sample". The default value is n.max=5000.

tol	numeric scalar indicating the toloerance to use in the uniroot search algorithm. The default value is tol=1e-7.
maxiter	positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

Details

If the arguments ratio.of.means, cv, alpha, power, and n2 are not all the same length, they are replicated to be the same length as the length of the longest argument.

Formulas for the power of the t-test for lognormal data for specified values of the sample size, ratio of means, and Type I error level are given in the help file for tTestLnormAltPower. The function tTestLnormAltN uses the uniroot search algorithm to determine the required sample size(s) for specified values of the power, scaled difference, and Type I error level.

Value

When sample.type="one.sample", or sample.type="two.sample" and n2 is not supplied (so equal sample sizes for each group is assumed), tTestLnormAltN returns a numeric vector of sample sizes. When sample.type="two.sample" and n2 is supplied, tTestLnormAltN returns a list with two components called n1 and n2, specifying the sample sizes for each group.

Note

See tTestLnormAltPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See tTestLnormAltPower.

See Also

tTestLnormAltPower, tTestLnormAltRatioOfMeans, plotTTestLnormAltDesign, LognormalAlt, t.test, Hypothesis Tests.

Examples

```
# Look at how the required sample size for the one-sample test increases with
# increasing required power:

seq(0.5, 0.9, by = 0.1)
# [1] 0.5 0.6 0.7 0.8 0.9

tTestLnormAltN(ratio.of.means = 1.5, power = seq(0.5, 0.9, by = 0.1))
# [1] 19 23 28 36 47

#----------
```

```
# Repeat the last example, but compute the sample size based on the approximate
# power instead of the exact power:

tTestLnormAltN(ratio.of.means = 1.5, power = seq(0.5, 0.9, by = 0.1), approx = TRUE)
# [1] 19 23 29 36 47

#==========

# Look at how the required sample size for the two-sample t-test decreases with
# increasing ratio of means:

seq(1.5, 2, by = 0.1)
#[1] 1.5 1.6 1.7 1.8 1.9 2.0

tTestLnormAltN(ratio.of.means = seq(1.5, 2, by = 0.1), sample.type = "two")
#[1] 111  83  65  54  45  39

#----------

# Look at how the required sample size for the two-sample t-test decreases with
# increasing values of Type I error:

tTestLnormAltN(ratio.of.means = 1.5, alpha = c(0.001, 0.01, 0.05, 0.1),
  sample.type = "two")
#[1] 209 152 111  92

#----------

# For the two-sample t-test, compare the total sample size required to detect a
# ratio of means of 2 for equal sample sizes versus the case when the sample size
# for the second group is constrained to be 30.  Assume a coefficient of variation
# of 1, a 5% significance level, and 95% power.  Note that for the case of equal
# sample sizes, a total of 78 samples (39+39) are required, whereas when n2 is
# constrained to be 30, a total of 84 samples (54 + 30) are required.

tTestLnormAltN(ratio.of.means = 2, sample.type = "two")
#[1] 39

tTestLnormAltN(ratio.of.means = 2, n2 = 30)
#$n1:
#[1] 54
#
#$n2:
#[1] 30

#==========

# The guidance document Soil Screening Guidance: Technical Background Document
# (USEPA, 1996c, Part 4) discusses sampling design and sample size calculations
# for studies to determine whether the soil at a potentially contaminated site
# needs to be investigated for possible remedial action. Let 'theta' denote the
# average concentration of the chemical of concern.  The guidance document
```

```
# establishes the following goals for the decision rule (USEPA, 1996c, p.87):
#
#     Pr[Decide Don't Investigate | theta > 2 * SSL] = 0.05
#
#     Pr[Decide to Investigate | theta <= (SSL/2)] = 0.2
#
# where SSL denotes the pre-established soil screening level.
#
# These goals translate into a Type I error of 0.2 for the null hypothesis
#
#     H0: [theta / (SSL/2)] <= 1
#
# and a power of 95% for the specific alternative hypothesis
#
#     Ha: [theta / (SSL/2)] = 4
#
# Assuming a lognormal distribution and the above values for Type I error and
# power, determine the required samples sizes associated with various values of
# the coefficient of variation for the one-sample test.  Based on these calculations,
# you need to take at least 6 soil samples to satisfy the requirements for the
# Type I and Type II errors when the coefficient of variation is 2.

cv <- c(0.5, 1, 2)
N <- tTestLnormAltN(ratio.of.means = 4, cv = cv, alpha = 0.2,
  alternative = "greater")

names(N) <- paste("CV=", cv, sep = "")
N
#CV=0.5   CV=1   CV=2
#     2      3      6

#----------

# Repeat the last example, but use the approximate power calculation instead of the
# exact. Using the approximate power calculation, you need 7 soil samples when the
# coefficient of variation is 2 (because the approximation underestimates the
# true power).

N <- tTestLnormAltN(ratio.of.means = 4, cv = cv, alpha = 0.2,
  alternative = "greater", approx = TRUE)

names(N) <- paste("CV=", cv, sep = "")
N
#CV=0.5   CV=1   CV=2
#     3      5      7

#----------

# Repeat the last example, but use a Type I error of 0.05.

N <- tTestLnormAltN(ratio.of.means = 4, cv = cv, alternative = "greater",
  approx = TRUE)
```

```
names(N) <- paste("CV=", cv, sep = "")
N
#CV=0.5   CV=1   CV=2
#     4      6     12

#==========

# Reproduce the second column of Table 2 in van Belle and Martin (1993, p.167).

tTestLnormAltN(ratio.of.means = 1.10, cv = seq(0.1, 0.8, by = 0.1),
  power = 0.8, sample.type = "two.sample", approx = TRUE)
#[1]  19  69 150 258 387 533 691 856

#==========

# Clean up
#---------
rm(cv, N)
```

---

tTestLnormAltPower *Power of a One- or Two-Sample t-Test Assuming Lognormal Data*

Description

Compute the power of a one- or two-sample t-test, given the sample size, ratio of means, coefficient of variation, and significance level, assuming lognormal data.

Usage

```
tTestLnormAltPower(n.or.n1, n2 = n.or.n1, ratio.of.means = 1, cv = 1, alpha = 0.05,
   sample.type = ifelse(!missing(n2), "two.sample", "one.sample"),
   alternative = "two.sided", approx = FALSE)
```

Arguments

n.or.n1	numeric vector of sample sizes. When sample.type="one.sample", n.or.n1 denotes n, the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes n_1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
n2	numeric vector of sample sizes for group 2. The default value is the value of n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
ratio.of.means	numeric vector specifying the ratio of the first mean to the second mean. When sample.type="one.sample", this is the ratio of the population mean to the hypothesized mean. When sample.type="two.sample", this is the ratio of the mean of the first population to the mean of the second population. The default value is ratio.of.means=1.

cv	numeric vector of positive value(s) specifying the coefficient of variation. When sample.type="one.sample", this is the population coefficient of variation. When sample.type="two.sample", this is the coefficient of variation for both the first and second population. The default value is cv=1.
alpha	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.
sample.type	character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".
approx	logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.

Details

If the arguments n.or.n1, n2, ratio.of.means, cv, and alpha are not all the same length, they are replicated to be the same length as the length of the longest argument.

One-Sample Case (sample.type="one.sample")
Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a lognormal distribution with mean θ and coefficient of variation τ, and consider the null hypothesis:

$$H_0 : \theta = \theta_0 \qquad (1)$$

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

$$H_a : \theta > \theta_0 \qquad (2)$$

the lower one-sided alternative (alternative="less")

$$H_a : \theta < \theta_0 \qquad (3)$$

and the two-sided alternative (alternative="two.sided")

$$H_a : \theta \neq \theta_0 \qquad (4)$$

To test the null hypothesis (1) versus any of the three alternatives (2)-(4), one might be tempted to use Student's t-test based on the log-transformed observations. Unlike the two-sample case with equal coefficients of variation (see below), in the one-sample case Student's t-test applied to the log-transformed observations will not test the correct hypothesis, as now explained.

Let

$$y_i = log(x_i), \quad i = 1, 2, \ldots, n \qquad (5)$$

Then $\underline{y} = y_1, y_2, \ldots, y_n$ denote n observations from a normal distribution with mean μ and standard deviation σ, where

$$\mu = log(\frac{\theta}{\sqrt{\tau^2 + 1}}) \qquad (6)$$

$$\sigma = [log(\tau^2 + 1)]^{1/2} \quad (7)$$

$$\theta = exp[\mu + (\sigma^2/2)] \quad (8)$$

$$\tau = [exp(\sigma^2) - 1]^{1/2} \quad (9)$$

(see the help file for LognormalAlt). Hence, by Equations (6) and (8) above, the Student's t-test on the log-transformed data would involve a test of hypothesis on both the parameters θ and τ, not just on θ.

To test the null hypothesis (1) above versus any of the alternatives (2)-(4), you can use the function `elnormAlt` to compute a confidence interval for θ, and use the relationship between confidence intervals and hypothesis tests. To test the null hypothesis (1) above versus the upper one-sided alternative (2), you can also use Chen's modified t-test for skewed distributions.

Although you can't use Student's t-test based on the log-transformed observations to test a hypothesis about θ, you can use the t-distribution to estimate the power of a test about θ that is based on confidence intervals or Chen's modified t-test, if you are willing to assume the population coefficient of variation τ stays constant for all possible values of θ you are interested in, and you are willing to postulate possible values for τ.

First, let's re-write the hypotheses (1)-(4) as follows. The null hypothesis (1) is equivalent to:

$$H_0 : \frac{\theta}{\theta_0} = 1 \quad (10)$$

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`)

$$H_a : \frac{\theta}{\theta_0} > 1 \quad (11)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : \frac{\theta}{\theta_0} < 1 \quad (12)$$

and the two-sided alternative (`alternative="two.sided"`)

$$H_a : \frac{\theta}{\theta_0} \neq 1 \quad (13)$$

For a constant coefficient of variation τ, the standard deviation of the log-transformed observations σ is also constant (see Equation (7) above). Hence, by Equation (8), the ratio of the true mean to the hypothesized mean can be written as:

$$R = \frac{\theta}{\theta_0} = \frac{exp[\mu + (\sigma^2/2)]}{exp[\mu_0 + (\sigma^2/2)]} = \frac{e^{\mu}}{e_0^{\mu}} = e^{\mu - \mu_0} \quad (14)$$

which only involves the difference

$$\mu - \mu_0 \quad (15)$$

Thus, for given values of R and τ, the power of the test of the null hypothesis (10) against any of the alternatives (11)-(13) can be computed based on the power of a one-sample t-test with

$$\frac{\delta}{\sigma} = \frac{log(R)}{\sqrt{log(\tau^2 + 1)}} \quad (16)$$

(see the help file for tTestPower). Note that for the function tTestLnormAltPower, R corresponds to the argument ratio.of.means, and τ corresponds to the argument cv.

Two-Sample Case (sample.type="two.sample")
Let $\underline{x}_1 = x_{11}, x_{12}, \ldots, x_{1n_1}$ denote a vector of n_1 observations from a lognormal distribution with mean θ_1 and coefficient of variaiton τ, and let $\underline{x}_2 = x_{21}, x_{22}, \ldots, x_{2n_2}$ denote a vector of n_2 observations from a lognormal distribution with mean θ_2 and coefficient of variation τ, and consider the null hypothesis:

$$H_0 : \theta_1 = \theta_2 \qquad (17)$$

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater"):

$$H_a : \theta_1 > \theta_2 \qquad (18)$$

the lower one-sided alternative (alternative="less")

$$H_a : \theta_1 < \theta_2 \qquad (19)$$

and the two-sided alternative (alternative="two.sided")

$$H_a : \theta_1 \neq \theta_2 \qquad (20)$$

Because we are assuming the coefficient of variation τ is the same for both populations, the test of the null hypothesis (17) versus any of the three alternatives (18)-(20) can be based on the Student t-statistic using the log-transformed observations.

To show this, first, let's re-write the hypotheses (17)-(20) as follows. The null hypothesis (17) is equivalent to:

$$H_0 : \frac{\theta_1}{\theta_2} = 1 \qquad (21)$$

The three possible alternative hypotheses are the upper one-sided alternative (alternative="greater")

$$H_a : \frac{\theta_1}{\theta_2} > 1 \qquad (22)$$

the lower one-sided alternative (alternative="less")

$$H_a : \frac{\theta_1}{\theta_2} < 1 \qquad (23)$$

and the two-sided alternative (alternative="two.sided")

$$H_a : \frac{\theta_1}{\theta_2} \neq 1 \qquad (24)$$

If coefficient of variation τ is the same for both populations, then the standard deviation of the log-transformed observations σ is also the same for both populations (see Equation (7) above). Hence, by Equation (8), the ratio of the means can be written as:

$$R = \frac{\theta_1}{\theta_2} = \frac{exp[\mu_1 + (\sigma^2/2)]}{exp[\mu_2 + (\sigma^2/2)]} = \frac{e^{\mu}_1}{e^{\mu}_2} = e^{\mu_1 - \mu_2} \qquad (25)$$

which only involves the difference

$$\mu_1 - \mu_2 \qquad (26)$$

Thus, for given values of R and τ, the power of the test of the null hypothesis (21) against any of the alternatives (22)-(24) can be computed based on the power of a two-sample t-test with

$$\frac{\delta}{\sigma} = \frac{log(R)}{\sqrt{log(\tau^2 + 1)}} \qquad (27)$$

(see the help file for `tTestPower`). Note that for the function `tTestLnormAltPower`, R corresponds to the argument `ratio.of.means`, and τ corresponds to the argument `cv`.

Value

a numeric vector powers.

Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. Often, you need to determine whether a population mean is significantly different from a specified standard (e.g., an MCL or ACL, USEPA, 1989b, Section 6), or whether two different means are significantly different from each other (e.g., USEPA 2009, Chapter 16). When you have lognormally-distributed data, you have to be careful about making statements regarding inference for the mean. For the two-sample case with assumed equal coefficients of variation, you can perform the Student's t-test on the log-transformed observations. For the one-sample case, you can perform a hypothesis test by constructing a confidence interval for the mean using `elnormAlt`, or use Chen's t-test modified for skewed data.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether a mean differs from a specified level or two means differ from each other. The functions `tTestLnormAltPower`, `tTestLnormAltN`, `tTestLnormAltRatioOfMeans`, and `plotTTestLnormAltDesign` can be used to investigate these relationships for the case of lognormally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

van Belle, G., and D.C. Martin. (1993). Sample Size as a Function of Coefficient of Variation and Ratio of Means. *The American Statistician* **47**(3), 165–167.

Also see the list of references in the help file for `tTestPower`.

See Also

`tTestLnormAltN`, `tTestLnormAltRatioOfMeans`, `plotTTestLnormAltDesign`, LognormalAlt, `t.test`, Hypothesis Tests.

Examples

```
# Look at how the power of the one-sample test increases with increasing
# sample size:

seq(5, 30, by = 5)
#[1]  5 10 15 20 25 30

power <- tTestLnormAltPower(n.or.n1 = seq(5, 30, by = 5),
  ratio.of.means = 1.5, cv = 1)

round(power, 2)
#[1] 0.14 0.28 0.42 0.54 0.65 0.73

#----------

# Repeat the last example, but use the approximation to the power instead of the
# exact power.  Note how the approximation underestimates the true power for
# the smaller sample sizes:

power <- tTestLnormAltPower(n.or.n1 = seq(5, 30, by = 5),
  ratio.of.means = 1.5, cv = 1, approx = TRUE)

round(power, 2)
#[1] 0.09 0.25 0.40 0.53 0.64 0.73

#==========

# Look at how the power of the two-sample t-test increases with increasing
# ratio of means:

power <- tTestLnormAltPower(n.or.n1 = 20, sample.type = "two",
  ratio.of.means = c(1.1, 1.5, 2), cv = 1)

round(power, 2)
#[1] 0.06 0.32 0.73

#----------

# Look at how the power of the two-sample t-test increases with increasing
# values of Type I error:

power <- tTestLnormAltPower(30, sample.type = "two", ratio.of.means = 1.5,
  cv = 1, alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
#[1] 0.07 0.23 0.46 0.59

#==========

# The guidance document Soil Screening Guidance: Technical Background Document
# (USEPA, 1996c, Part 4) discusses sampling design and sample size calculations
# for studies to determine whether the soil at a potentially contaminated site
```

```
# needs to be investigated for possible remedial action. Let 'theta' denote the
# average concentration of the chemical of concern.  The guidance document
# establishes the following goals for the decision rule (USEPA, 1996c, p.87):
#
#     Pr[Decide Don't Investigate | theta > 2 * SSL] = 0.05
#
#     Pr[Decide to Investigate | theta <= (SSL/2)] = 0.2
#
# where SSL denotes the pre-established soil screening level.
#
# These goals translate into a Type I error of 0.2 for the null hypothesis
#
#     H0: [theta / (SSL/2)] <= 1
#
# and a power of 95% for the specific alternative hypothesis
#
#     Ha: [theta / (SSL/2)] = 4
#
# Assuming a lognormal distribution with a coefficient of variation of 2,
# determine the power associated with various sample sizes for this one-sample test.
# Based on these calculations, you need to take at least 6 soil samples to
# satisfy the requirements for the Type I and Type II errors.

power <- tTestLnormAltPower(n.or.n1 = 2:8, ratio.of.means = 4, cv = 2,
  alpha = 0.2, alternative = "greater")

names(power) <- paste("N=", 2:8, sep = "")

round(power, 2)
# N=2  N=3  N=4  N=5  N=6  N=7  N=8
#0.65 0.80 0.88 0.93 0.96 0.97 0.98

#----------

# Repeat the last example, but use the approximate power calculation instead of
# the exact one.  Using the approximate power calculation, you need at least
# 7 soil samples instead of 6 (because the approximation underestimates the power).

power <- tTestLnormAltPower(n.or.n1 = 2:8, ratio.of.means = 4, cv = 2,
  alpha = 0.2, alternative = "greater", approx = TRUE)

names(power) <- paste("N=", 2:8, sep = "")

round(power, 2)
# N=2  N=3  N=4  N=5  N=6  N=7  N=8
#0.55 0.75 0.84 0.90 0.93 0.95 0.97

#==========

# Clean up
#---------
rm(power)
```

tTestLnormAltRatioOfMeans

Minimal or Maximal Detectable Ratio of Means for One- or Two-Sample t-Test, Assuming Lognormal Data

Description

Compute the minimal or maximal detectable ratio of means associated with a one- or two-sample t-test, given the sample size, coefficient of variation, significance level, and power, assuming lognormal data.

Usage

```
tTestLnormAltRatioOfMeans(n.or.n1, n2 = n.or.n1, cv = 1, alpha = 0.05, power = 0.95,
  sample.type = ifelse(!missing(n2), "two.sample", "one.sample"),
  alternative = "two.sided", two.sided.direction = "greater", approx = FALSE,
  tol = 1e-07, maxiter = 1000)
```

Arguments

n.or.n1	numeric vector of sample sizes. When sample.type="one.sample", n.or.n1 denotes n, the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes n_1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
n2	numeric vector of sample sizes for group 2. The default value is the value of n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
cv	numeric vector of positive value(s) specifying the coefficient of variation. When sample.type="one.sample", this is the population coefficient of variation. When sample.type="two.sample", this is the coefficient of variation for both the first and second population. The default value is cv=1.
alpha	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.
power	numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.
sample.type	character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "greater", and "less".

`two.sided.direction`
: character string indicating the direction (greater than 1 or less than 1) for the detectable ratio of means when `alternative="two.sided"`. When `two.sided.direction="greater"` (the default), the detectable ratio of means is greater than 1. When `two.sided.direction="less"`, the detectable ratio of means is less than 1 (but greater than 0). This argument is ignored if `alternative="less"` or `alternative="greater"`.

`approx`
: logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `FALSE`.

`tol`
: numeric scalar indicating the toloerance to use in the `uniroot` search algorithm. The default value is `tol=1e-7`.

`maxiter`
: positive integer indicating the maximum number of iterations argument to pass to the `uniroot` function. The default value is `maxiter=1000`.

Details

If the arguments `n.or.n1`, `n2`, `cv`, `alpha`, and `power` are not all the same length, they are replicated to be the same length as the length of the longest argument.

Formulas for the power of the t-test for lognormal data for specified values of the sample size, ratio of means, and Type I error level are given in the help file for `tTestLnormAltPower`. The function `tTestLnormAltRatioOfMeans` uses the `uniroot` search algorithm to determine the required ratio of means for specified values of the power, sample size, and Type I error level.

Value

a numeric vector of computed minimal or maximal detectable ratios of means. When `alternative="less"`, or `alternative="two.sided"` and `two.sided.direction="less"`, the computed ratios are less than 1 (but greater than 0). Otherwise, the ratios are greater than 1.

Note

See `tTestLnormAltPower`.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See `tTestLnormAltPower`.

See Also

`tTestLnormAltPower`, `tTestLnormAltN`, `plotTTestLnormAltDesign`, LognormalAlt, `t.test`, Hypothesis Tests.

Examples

```
# Look at how the minimal detectable ratio of means for the one-sample t-test
# increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = 20,
  power = seq(0.5, 0.9, by = 0.1))

round(ratio.of.means, 2)
#[1] 1.47 1.54 1.63 1.73 1.89

#----------

# Repeat the last example, but compute the minimal detectable ratio of means
# based on the approximate power instead of the exact:

ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = 20,
  power = seq(0.5, 0.9, by = 0.1), approx = TRUE)

round(ratio.of.means, 2)
#[1] 1.48 1.55 1.63 1.73 1.89

#==========

# Look at how the minimal detectable ratio of means for the two-sample t-test
# decreases with increasing sample size:

seq(10, 50, by = 10)
#[1] 10 20 30 40 50

ratio.of.means <- tTestLnormAltRatioOfMeans(seq(10, 50, by = 10), sample.type="two")

round(ratio.of.means, 2)
#[1] 4.14 2.65 2.20 1.97 1.83

#----------

# Look at how the minimal detectable ratio of means for the two-sample t-test
# decreases with increasing values of Type I error:

ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = 20,
  alpha = c(0.001, 0.01, 0.05, 0.1), sample.type = "two")

round(ratio.of.means, 2)
#[1] 4.06 3.20 2.65 2.42

#==========

# The guidance document Soil Screening Guidance: Technical Background Document
# (USEPA, 1996c, Part 4) discusses sampling design and sample size calculations
```

```
# for studies to determine whether the soil at a potentially contaminated site
# needs to be investigated for possible remedial action. Let 'theta' denote the
# average concentration of the chemical of concern.  The guidance document
# establishes the following goals for the decision rule (USEPA, 1996c, p.87):
#
#     Pr[Decide Don't Investigate | theta > 2 * SSL] = 0.05
#
#     Pr[Decide to Investigate | theta <= (SSL/2)] = 0.2
#
# where SSL denotes the pre-established soil screening level.
#
# These goals translate into a Type I error of 0.2 for the null hypothesis
#
#     H0: [theta / (SSL/2)] <= 1
#
# and a power of 95% for the specific alternative hypothesis
#
#     Ha: [theta / (SSL/2)] = 4
#
# Assuming a lognormal distribution, the above values for Type I and power, and a
# coefficient of variation of 2, determine the minimal detectable increase above
# the soil screening level associated with various sample sizes for the one-sample
# test.  Based on these calculations, you need to take at least 6 soil samples to
# satisfy the requirements for the Type I and Type II errors when the coefficient
# of variation is 2.

N <- 2:8
ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = N, cv = 2, alpha = 0.2,
  alternative = "greater")

names(ratio.of.means) <- paste("N=", N, sep = "")
round(ratio.of.means, 1)
# N=2  N=3  N=4  N=5  N=6  N=7  N=8
#19.9  7.7  5.4  4.4  3.8  3.4  3.1

#----------

# Repeat the last example, but use the approximate power calculation instead of
# the exact.  Using the approximate power calculation, you need 7 soil samples
# when the coefficient of variation is 2.  Note how poorly the approximation
# works in this case for small sample sizes!

ratio.of.means <- tTestLnormAltRatioOfMeans(n.or.n1 = N, cv = 2, alpha = 0.2,
  alternative = "greater", approx = TRUE)

names(ratio.of.means) <- paste("N=", N, sep = "")
round(ratio.of.means, 1)
#  N=2   N=3   N=4   N=5   N=6   N=7   N=8
#990.8  18.5   8.3   5.7   4.6   3.9   3.5

#==========

# Clean up
```

```
#---------
rm(ratio.of.means, N)
```

`tTestN` *Sample Size for a One- or Two-Sample t-Test*

Description

Compute the sample size necessary to achieve a specified power for a one- or two-sample t-test, given the scaled difference and significance level.

Usage

```
tTestN(delta.over.sigma, alpha = 0.05, power = 0.95,
  sample.type = ifelse(!is.null(n2), "two.sample", "one.sample"),
  alternative = "two.sided", approx = FALSE, n2 = NULL, round.up = TRUE,
  n.max = 5000, tol = 1e-07, maxiter = 1000)
```

Arguments

`delta.over.sigma`
: numeric vector specifying the ratio of the true difference δ ($\delta = \mu - \mu_0$ for the one-sample case and $\delta = \mu_1 - \mu_2$ for the two-sample case) to the population standard deviation (σ). This is also called the "scaled difference".

`alpha`
: numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is `alpha=0.05`.

`power`
: numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is `power=0.95`.

`sample.type`
: character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When `sample.type="one.sample"`, the computed power is based on a hypothesis test for a single mean. When `sample.type="two.sample"`, the computed power is based on a hypothesis test for the difference between two means. The default value is `sample.type="one.sample"` unless the argument `n2` is supplied.

`alternative`
: character string indicating the kind of alternative hypothesis. The possible values are:

  - `"two.sided"` (the default). $H_a : \mu \neq \mu_0$ for the one-sample case and $H_a : \mu_1 \neq \mu_2$ for the two-sample case.
  - `"greater"`. $H_a : \mu > \mu_0$ for the one-sample case and $H_a : \mu_1 > \mu_2$ for the two-sample case.
  - `"less"`. $H_a : \mu < \mu_0$ for the one-sample case and $H_a : \mu_1 < \mu_2$ for the two-sample case.

`approx`
: logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `FALSE`.

n2	numeric vector of sample sizes for group 2. The default value is NULL in which case it is assumed that the sample sizes for groups 1 and 2 are equal. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
round.up	logical scalar indicating whether to round up the values of the computed sample size(s) to the next smallest integer. The default value is TRUE.
n.max	positive integer greater than 1 indicating the maximum sample size when sample.type="one.sample" or the maximum sample size for group 1 when sample.type="two.sample". The default value is n.max=5000.
tol	numeric scalar indicating the toloerance to use in the uniroot search algorithm. The default value is tol=1e-7.
maxiter	positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

Details

Formulas for the power of the t-test for specified values of the sample size, scaled difference, and Type I error level are given in the help file for tTestPower. The function tTestN uses the uniroot search algorithm to determine the required sample size(s) for specified values of the power, scaled difference, and Type I error level.

Value

When sample.type="one.sample", tTestN returns a numeric vector of sample sizes.

When sample.type="two.sample" and n2 is not supplied, equal sample sizes for each group is assumed and tTestN returns a numeric vector of sample sizes indicating the required sample size *for each group*.

When sample.type="two.sample" and n2 is supplied, tTestN returns a list with two components called n1 and n2, specifying the sample sizes for each group.

Note

See tTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See tTestPower.

See Also

tTestPower, tTestScaledMdd, tTestAlpha, plotTTestDesign, Normal, t.test, Hypothesis Tests.

Examples

```
# Look at how the required sample size for the one-sample t-test
# increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

tTestN(delta.over.sigma = 0.5, power = seq(0.5, 0.9, by = 0.1))
#[1] 18 22 27 34 44

#----------

# Repeat the last example, but compute the sample size based on the
# approximation to the power instead of the exact method:

tTestN(delta.over.sigma = 0.5, power = seq(0.5, 0.9, by = 0.1),
  approx = TRUE)
#[1] 18 22 27 34 45

#==========

# Look at how the required sample size for the two-sample t-test
# decreases with increasing scaled difference:

seq(0.5, 2,by = 0.5)
#[1] 0.5 1.0 1.5 2.0

tTestN(delta.over.sigma = seq(0.5, 2, by = 0.5), sample.type = "two")
#[1] 105  27  13   8

#----------

# Look at how the required sample size for the two-sample t-test decreases
# with increasing values of Type I error:

tTestN(delta.over.sigma = 0.5, alpha = c(0.001, 0.01, 0.05, 0.1),
  sample.type="two")
#[1] 198 145 105  88

#----------

# For the two-sample t-test, compare the total sample size required to
# detect a scaled difference of 1 for equal sample sizes versus the case
# when the sample size for the second group is constrained to be 20.
# Assume a 5% significance level and 95% power.  Note that for the case
# of equal sample sizes, a total of 54 samples (27+27) are required,
# whereas when n2 is constrained to be 20, a total of 62 samples
# (42 + 20) are required.

tTestN(1, sample.type="two")
#[1] 27
```

```
tTestN(1, n2 = 20)
#$n1
#[1] 42
#
#$n2
#[1] 20

#==========

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009), determine the
# required sample size to detect a mean aldicarb level greater than the MCL
# of 7 ppb at the third compliance well with a power of 95%, assuming the
# true mean is 10 or 14.  Use the estimated standard deviation from the
# first four months of data to estimate the true population standard
# deviation, use a Type I error level of alpha=0.01, and assume an
# upper one-sided alternative (third compliance well mean larger than 7).
# (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
# Note that the required sample size changes from 11 to 5 as the true mean
# increases from 10 to 14.


EPA.09.Ex.21.1.aldicarb.df
#   Month   Well Aldicarb.ppb
#1      1 Well.1         19.9
#2      2 Well.1         29.6
#3      3 Well.1         18.7
#4      4 Well.1         24.2
#5      1 Well.2         23.7
#6      2 Well.2         21.9
#7      3 Well.2         26.9
#8      4 Well.2         26.1
#9      1 Well.3          5.6
#10     2 Well.3          3.3
#11     3 Well.3          2.3
#12     4 Well.3          6.9

sigma <- with(EPA.09.Ex.21.1.aldicarb.df,
  sd(Aldicarb.ppb[Well == "Well.3"]))

sigma
#[1] 2.101388

tTestN(delta.over.sigma = (c(10, 14) - 7)/sigma,
  alpha = 0.01, sample.type="one", alternative="greater")
#[1] 11  5


# Clean up
#---------
rm(sigma)
```

tTestPower *Power of a One- or Two-Sample t-Test*

Description

Compute the power of a one- or two-sample t-test, given the sample size, scaled difference, and significance level.

Usage

```
tTestPower(n.or.n1, n2 = n.or.n1, delta.over.sigma = 0, alpha = 0.05,
  sample.type = ifelse(!missing(n2), "two.sample", "one.sample"),
  alternative = "two.sided", approx = FALSE)
```

Arguments

n.or.n1	numeric vector of sample sizes. When sample.type="one.sample", n.or.n1 denotes n, the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes n_1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
n2	numeric vector of sample sizes for group 2. The default value is the value of n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
delta.over.sigma	numeric vector specifying the ratio of the true difference δ ($\delta = \mu - \mu_0$ for the one-sample case and $\delta = \mu_1 - \mu_2$ for the two-sample case) to the population standard deviation (σ). This is also called the "scaled difference". The default value is delta.over.sigma=0.
alpha	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.
sample.type	character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.
alternative	character string indicating the kind of alternative hypothesis. The possible values are: • "two.sided" (the default). $H_a : \mu \neq \mu_0$ for the one-sample case and $H_a : \mu_1 \neq \mu_2$ for the two-sample case. • "greater". $H_a : \mu > \mu_0$ for the one-sample case and $H_a : \mu_1 > \mu_2$ for the two-sample case. • "less". $H_a : \mu < \mu_0$ for the one-sample case and $H_a : \mu_1 < \mu_2$ for the two-sample case.

`approx` logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is `FALSE`.

Details

If the arguments `n.or.n1`, `n2`, `delta.over.sigma`, and `alpha` are not all the same length, they are replicated to be the same length as the length of the longest argument.

One-Sample Case (`sample.type="one.sample"`)
Let $\underline{x} = x_1, x_2, \ldots, x_n$ denote a vector of n observations from a normal distribution with mean μ and standard deviation σ, and consider the null hypothesis:

$$H_0 : \mu = \mu_0 \qquad (1)$$

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`):

$$H_a : \mu > \mu_0 \qquad (2)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : \mu < \mu_0 \qquad (3)$$

and the two-sided alternative (`alternative="two.sided"`)

$$H_a : \mu \neq \mu_0 \qquad (4)$$

The test of the null hypothesis (1) versus any of the three alternatives (2)-(4) is based on the Student t-statistic:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} \qquad (5)$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (6)$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (7)$$

Under the null hypothesis (1), the t-statistic in (5) follows a Student's t-distribution with $n - 1$ degrees of freedom (Zar, 2010, Chapter 7; Johnson et al., 1995, pp.362-363).

The formula for the power of the test depends on which alternative is being tested. The two subsections below describe exact and approximate formulas for the power of the one-sample t-test. Note that none of the equations for the power of the t-test requires knowledge of the values δ (Equation (12) below) or σ (the population standard deviation), only the ratio δ/σ. The argument `delta.over.sigma` is this ratio, and it is referred to as the "scaled difference".

Exact Power Calculations (`approx=FALSE`)
This subsection describes the exact formulas for the power of the one-sample t-test.

Upper one-sided alternative (`alternative="greater"`)
The standard Student's t-test rejects the null hypothesis (1) in favor of the upper alternative hypothesis (2) at level-α if

$$t \geq t_\nu(1 - \alpha) \qquad (8)$$

where

$$\nu = n - 1 \qquad (9)$$

and $t_\nu(p)$ denotes the p'th quantile of Student's t-distribution with ν degrees of freedom (Zar, 2010; Berthouex and Brown, 2002). The power of this test, denoted by $1 - \beta$, where β denotes the probability of a Type II error, is given by:

$$1 - \beta = Pr[t_{\nu,\Delta} \geq t_\nu(1 - \alpha)] = 1 - G[t_\nu(1 - \alpha), \nu, \Delta] \qquad (10)$$

where

$$\Delta = \sqrt{n}(\frac{\delta}{\sigma}) \qquad (11)$$

$$\delta = \mu - \mu_0 \qquad (12)$$

and $t_{\nu,\Delta}$ denotes a non-central Student's t-random variable with ν degrees of freedom and non-centrality parameter Δ, and $G(x, \nu, \Delta)$ denotes the cumulative distribution function of this random variable evaluated at x (Johnson et al., 1995, pp.508-510).

Lower one-sided alternative (`alternative="less"`)
The standard Student's t-test rejects the null hypothesis (1) in favor of the lower alternative hypothesis (3) at level-α if

$$t \leq t_\nu(\alpha) \qquad (13)$$

and the power of this test is given by:

$$1 - \beta = Pr[t_{\nu,\Delta} \leq t_\nu(\alpha)] = G[t_\nu(\alpha), \nu, \Delta] \qquad (14)$$

Two-sided alternative (`alternative="two.sided"`)
The standard Student's t-test rejects the null hypothesis (1) in favor of the two-sided alternative hypothesis (4) at level-α if

$$|t| \geq t_\nu(1 - \alpha/2) \qquad (15)$$

and the power of this test is given by:

$$1 - \beta = Pr[t_{\nu,\Delta} \leq t_\nu(\alpha/2)] + Pr[t_{\nu,\Delta} \geq t_\nu(1 - \alpha/2)]$$

$$= G[t_\nu(\alpha/2), \nu, \Delta] + 1 - G[t_\nu(1 - \alpha/2), \nu, \Delta] \qquad (16)$$

The power of the t-test given in Equation (16) can also be expressed in terms of the cumulative distribution function of the non-central F-distribution as follows. Let $F_{\nu_1,\nu_2,\Delta}$ denote a non-central F random variable with ν_1 and ν_2 degrees of freedom and non-centrality parameter Δ, and let $H(x, \nu_1, \nu_2, \Delta)$ denote the cumulative distribution function of this random variable evaluated at x. Also, let $F_{\nu_1,\nu_2}(p)$ denote the p'th quantile of the central F-distribution with ν_1 and ν_2 degrees of freedom. It can be shown that

$$(t_{\nu,\Delta})^2 \cong F_{1,\nu,\Delta^2} \qquad (17)$$

where $\cong$ denotes "equal in distribution". Thus, it follows that

$$[t_\nu(1 - \alpha/2)]^2 = F_{1,\nu}(1 - \alpha) \qquad (18)$$

so the formula for the power of the t-test given in Equation (16) can also be written as:

$$1-\beta = Pr\{(t_{\nu,\Delta})^2 \geq [t_\nu(1-\alpha/2)]^2\}$$
$$= Pr[F_{1,\nu,\Delta^2} \geq F_{1,\nu}(1-\alpha)] = 1 - H[F_{1,\nu}(1-\alpha), 1, \nu, \Delta^2] \qquad (19)$$

Approximate Power Calculations (`approx=TRUE`)
Zar (2010, pp.115–118) presents an approximation to the power for the t-test given in Equations (10), (14), and (16) above. His approximation to the power can be derived by using the approximation

$$\sqrt{n}(\frac{\delta}{s}) \approx \sqrt{n}(\frac{\delta}{\sigma}) = \Delta \qquad (20)$$

where $\approx$ denotes "approximately equal to". Zar's approximation can be summarized in terms of the cumulative distribution function of the non-central t-distribution as follows:

$$G(x, \nu, \Delta) \approx G(x - \Delta, \nu, 0) = G(x - \Delta, \nu) \qquad (21)$$

where $G(x, \nu)$ denotes the cumulative distribution function of the central Student's t-distribution with ν degrees of freedom evaluated at x.

The following three subsections explicitly derive the approximation to the power of the t-test for each of the three alternative hypotheses.

Upper one-sided alternative (`alternative="greater"`)
The power for the upper one-sided alternative (2) given in Equation (10) can be approximated as:

$$1-\beta = Pr[t \geq t_\nu(1-\alpha)]$$
$$= Pr[\frac{\bar{x}-\mu}{s/\sqrt{n}} \geq t_\nu(1-\alpha) - \sqrt{n}\frac{\delta}{s}]$$
$$\approx Pr[t_\nu \geq t_\nu(1-\alpha) - \Delta]$$
$$= 1 - Pr[t_\nu \leq t_\nu(1-\alpha) - \Delta]$$
$$= 1 - G[t_\nu(1-\alpha) - \Delta, \nu] \qquad (22)$$

where t_ν denotes a central Student's t-random variable with ν degrees of freedom.

Lower one-sided alternative (`alternative="less"`)
The power for the lower one-sided alternative (3) given in Equation (14) can be approximated as:

$$1-\beta = Pr[t \leq t_\nu(\alpha)]$$
$$= Pr[\frac{\bar{x}-\mu}{s/\sqrt{n}} \leq t_\nu(\alpha) - \sqrt{n}\frac{\delta}{s}]$$
$$\approx Pr[t_\nu \leq t_\nu(\alpha) - \Delta]$$
$$= G[t_\nu(\alpha) - \Delta, \nu] \qquad (23)$$

Two-sided alternative (`alternative="two.sided"`)
The power for the two-sided alternative (4) given in Equation (16) can be approximated as:

$$1-\beta = Pr[t \le t_\nu(\alpha/2)] + Pr[t \ge t_\nu(1-\alpha/2)]$$

$$= Pr[\frac{\bar{x}-\mu}{s/\sqrt{n}} \le t_\nu(\alpha/2) - \sqrt{n}\frac{\delta}{s}] + Pr[\frac{\bar{x}-\mu}{s/\sqrt{n}} \ge t_\nu(1-\alpha) - \sqrt{n}\frac{\delta}{s}]$$

$$\approx Pr[t_\nu \le t_\nu(\alpha/2) - \Delta] + Pr[t_\nu \ge t_\nu(1-\alpha/2) - \Delta]$$

$$= G[t_\nu(\alpha/2) - \Delta, \nu] + 1 - G[t_\nu(1-\alpha/2) - \Delta, \nu] \qquad (24)$$

Two-Sample Case (`sample.type="two.sample"`)
Let $\underline{x}_1 = x_{11}, x_{12}, \ldots, x_{1n_1}$ denote a vector of n_1 observations from a normal distribution with mean μ_1 and standard deviation σ, and let $\underline{x}_2 = x_{21}, x_{22}, \ldots, x_{2n_2}$ denote a vector of n_2 observations from a normal distribution with mean μ_2 and standard deviation σ, and consider the null hypothesis:

$$H_0 : \mu_1 = \mu_2 \qquad (25)$$

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`):

$$H_a : \mu_1 > \mu_2 \qquad (26)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : \mu_1 < \mu_2 \qquad (27)$$

and the two-sided alternative (`alternative="two.sided"`)

$$H_a : \mu_1 \ne \mu_2 \qquad (28)$$

The test of the null hypothesis (25) versus any of the three alternatives (26)-(28) is based on the Student t-statistic:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \qquad (29)$$

where

$$\bar{x}_1 = \frac{1}{n_1}\sum_{i=1}^{n_1} x_{1i} \qquad (30)$$

$$\bar{x}_2 = \frac{1}{n_2}\sum_{i=1}^{n_2} x_{2i} \qquad (31)$$

$$s_p^2 = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2} \qquad (32)$$

$$s_1^2 = \frac{1}{n_1-1}\sum_{i=1}^{n_1}(x_{1i}-\bar{x}_1)^2 \qquad (33)$$

$$s_2^2 = \frac{1}{n_2-1}\sum_{i=1}^{n_2}(x_{2i}-\bar{x}_2)^2 \qquad (34)$$

Under the null hypothesis (25), the t-statistic in (29) follows a Student's t-distribution with $n_1 + n_2 - 2$ degrees of freedom (Zar, 2010, Chapter 8; Johnson et al., 1995, pp.508–510, Helsel and Hirsch, 1992, pp.124–128).

The formulas for the power of the two-sample t-test are precisely the same as those for the one-sample case, with the following modifications:

$$\nu = n_1 + n_2 - 2 \qquad (35)$$

$$\Delta = \sqrt{\frac{n_1 n_2}{n_1 + n_2}}(\frac{\delta}{\sigma}) \qquad (36)$$

$$\delta = \mu_1 - \mu_2 \qquad (37)$$

Note that none of the equations for the power of the t-test requires knowledge of the values δ or σ (the population standard deviation for both populations), only the ratio δ/σ. The argument `delta.over.sigma` is this ratio, and it is referred to as the "scaled difference".

Value

a numeric vector powers.

Note

The normal distribution and lognormal distribution are probably the two most frequently used distributions to model environmental data. Often, you need to determine whether a population mean is significantly different from a specified standard (e.g., an MCL or ACL, USEPA, 1989b, Section 6), or whether two different means are significantly different from each other (e.g., USEPA 2009, Chapter 16). In this case, assuming normally distributed data, you can perform the Student's t-test.

In the course of designing a sampling program, an environmental scientist may wish to determine the relationship between sample size, significance level, power, and scaled difference if one of the objectives of the sampling program is to determine whether a mean differs from a specified level or two means differ from each other. The functions `tTestPower`, `tTestN`, `tTestScaledMdd`, and `plotTTestDesign` can be used to investigate these relationships for the case of normally-distributed observations.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Berthouex, P.M., and L.C. Brown. (2002). *Statistics for Environmental Engineers*. Second Edition. Lewis Publishers, Boca Raton, FL.

Helsel, D.R., and R.M. Hirsch. (1992). *Statistical Methods in Water Resources Research*. Elsevier, New York, NY, Chapter 7.

Johnson, N. L., S. Kotz, and N. Balakrishnan. (1995). *Continuous Univariate Distributions, Volume 2*. Second Edition. John Wiley and Sons, New York, Chapters 28, 31

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

USEPA. (1989b). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities, Interim Final Guidance*. EPA/530-SW-89-026. Office of Solid Waste, U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

tTestN, tTestScaledMdd, tTestAlpha, plotTTestDesign, Normal, t.test, Hypothesis Tests.

Examples

```
# Look at how the power of the one-sample t-test increases with
# increasing sample size:

seq(5, 30, by = 5)
#[1] 5 10 15 20 25 30

power <- tTestPower(n.or.n1 = seq(5, 30, by = 5), delta.over.sigma = 0.5)

round(power, 2)
#[1] 0.14 0.29 0.44 0.56 0.67 0.75

#----------

# Repeat the last example, but use the approximation.
# Note how the approximation underestimates the power
# for the smaller sample sizes.
#---------------------------------------------------

power <- tTestPower(n.or.n1 = seq(5, 30, by = 5), delta.over.sigma = 0.5,
  approx = TRUE)

round(power, 2)
#[1] 0.10 0.26 0.42 0.56 0.67 0.75

#----------

# Look at how the power of the two-sample t-test increases with increasing
# scaled difference:

seq(0.5, 2, by = 0.5)
#[1] 0.5 1.0 1.5 2.0

power <- tTestPower(10, sample.type = "two.sample",
  delta.over.sigma = seq(0.5, 2, by = 0.5))

round(power, 2)
```

```
#[1] 0.19 0.56 0.89 0.99

#----------

# Look at how the power of the two-sample t-test increases with increasing values
# of Type I error:

power <- tTestPower(20, sample.type = "two.sample", delta.over.sigma = 0.5,
  alpha = c(0.001, 0.01, 0.05, 0.1))

round(power, 2)
#[1] 0.03 0.14 0.34 0.46

#==========

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009), determine how
# adding another four months of observations to increase the sample size from
# 4 to 8 for any one particular compliance well will affect the power of a
# one-sample t-test that compares the mean for the well with the MCL of
# 7 ppb.  Use alpha = 0.01, assume an upper one-sided alternative
# (i.e., compliance well mean larger than 7 ppb), and assume a scaled
# difference of 2.  (The data are stored in EPA.09.Ex.21.1.aldicarb.df.)
# Note that the power changes from 49% to 98% by increasing the sample size
# from 4 to 8.

tTestPower(n.or.n1 = c(4, 8), delta.over.sigma = 2, alpha = 0.01,
  sample.type = "one.sample", alternative = "greater")
#[1] 0.4865800 0.9835401

#==========

# Clean up
#---------
rm(power)
```

`tTestScaledMdd` *Scaled Minimal Detectable Difference for One- or Two-Sample t-Test*

Description

Compute the scaled minimal detectable difference necessary to achieve a specified power for a one- or two-sample t-test, given the sample size(s) and Type I error level.

Usage

```
tTestScaledMdd(n.or.n1, n2 = n.or.n1, alpha = 0.05, power = 0.95,
 sample.type = ifelse(!missing(n2) && !is.null(n2), "two.sample", "one.sample"),
  alternative = "two.sided", two.sided.direction = "greater",
  approx = FALSE, tol = 1e-07, maxiter = 1000)
```

Arguments

n.or.n1	numeric vector of sample sizes. When sample.type="one.sample", n.or.n1 denotes n, the number of observations in the single sample. When sample.type="two.sample", n.or.n1 denotes n_1, the number of observations from group 1. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
n2	numeric vector of sample sizes for group 2. The default value is the value of n.or.n1. This argument is ignored when sample.type="one.sample". Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are ***not*** allowed.
alpha	numeric vector of numbers between 0 and 1 indicating the Type I error level associated with the hypothesis test. The default value is alpha=0.05.
power	numeric vector of numbers between 0 and 1 indicating the power associated with the hypothesis test. The default value is power=0.95.
sample.type	character string indicating whether to compute power based on a one-sample or two-sample hypothesis test. When sample.type="one.sample", the computed power is based on a hypothesis test for a single mean. When sample.type="two.sample", the computed power is based on a hypothesis test for the difference between two means. The default value is sample.type="one.sample" unless the argument n2 is supplied.
alternative	character string indicating the kind of alternative hypothesis. The possible values are: • "two.sided" (the default). $H_a : \mu \neq \mu_0$ for the one-sample case and $H_a : \mu_1 \neq \mu_2$ for the two-sample case. • "greater". $H_a : \mu > \mu_0$ for the one-sample case and $H_a : \mu_1 > \mu_2$ for the two-sample case. • "less". $H_a : \mu < \mu_0$ for the one-sample case and $H_a : \mu_1 < \mu_2$ for the two-sample case.
two.sided.direction	character string indicating the direction (positive or negative) for the scaled minimal detectable difference when alternative="two.sided". When two.sided.direction="greater" (the default), the scaled minimal detectable difference is positive. When two.sided.direction="less", the scaled minimal detectable difference is negative. This argument is ignored if alternative="less" or alternative="greater".
approx	logical scalar indicating whether to compute the power based on an approximation to the non-central t-distribution. The default value is FALSE.
tol	numeric scalar indicating the tolerance argument to pass to the uniroot function. The default value is tol=1e-7.
maxiter	positive integer indicating the maximum number of iterations argument to pass to the uniroot function. The default value is maxiter=1000.

Details

Formulas for the power of the t-test for specified values of the sample size, scaled difference, and Type I error level are given in the help file for tTestPower. The function tTestScaledMdd uses

the uniroot search algorithm to determine the required scaled minimal detectable difference for specified values of the sample size, power, and Type I error level.

Value

numeric vector of scaled minimal detectable differences.

Note

See tTestPower.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

See tTestPower.

See Also

tTestPower, tTestAlpha, tTestN, plotTTestDesign, Normal, t.test, Hypothesis Tests.

Examples

```
# Look at how the scaled minimal detectable difference for the
# one-sample t-test increases with increasing required power:

seq(0.5, 0.9, by = 0.1)
#[1] 0.5 0.6 0.7 0.8 0.9

scaled.mdd <- tTestScaledMdd(n.or.n1 = 20, power = seq(0.5,0.9,by=0.1))

round(scaled.mdd, 2)
#[1] 0.46 0.52 0.59 0.66 0.76

#----------

# Repeat the last example, but compute the scaled minimal detectable
# differences based on the approximation to the power instead of the
# exact formula:

scaled.mdd <- tTestScaledMdd(n.or.n1 = 20, power = seq(0.5, 0.9, by = 0.1),
  approx = TRUE)

round(scaled.mdd, 2)
#[1] 0.47 0.53 0.59 0.66 0.76

#==========

# Look at how the scaled minimal detectable difference for the two-sample
# t-test decreases with increasing sample size:
```

```
seq(10,50,by=10)
#[1] 10 20 30 40 50

scaled.mdd <- tTestScaledMdd(seq(10, 50, by = 10), sample.type = "two")

round(scaled.mdd, 2)
#[1] 1.71 1.17 0.95 0.82 0.73

#----------

# Look at how the scaled minimal detectable difference for the two-sample
# t-test decreases with increasing values of Type I error:

scaled.mdd <- tTestScaledMdd(20, alpha = c(0.001, 0.01, 0.05, 0.1),
  sample.type="two")

round(scaled.mdd, 2)
#[1] 1.68 1.40 1.17 1.06

#==========

# Modifying the example on pages 21-4 to 21-5 of USEPA (2009),
# determine the minimal mean level of aldicarb at the third compliance
# well necessary to detect a mean level of aldicarb greater than the
# MCL of 7 ppb, assuming 90%, 95%, and 99% power.  Use a 99% significance
# level and assume an upper one-sided alternative (third compliance well
# mean larger than 7).  Use the estimated standard deviation from the
# first four months of data to estimate the true population standard
# deviation in order to determine the minimal detectable difference based
# on the computed scaled minimal detectable difference, then use this
# minimal detectable difference to determine the mean level of aldicarb
# necessary to detect a difference.  (The data are stored in
# EPA.09.Ex.21.1.aldicarb.df.)
#
# Note that the scaled minimal detectable difference changes from 3.4 to
# 3.9 to 4.7 as the power changes from 90% to 95% to 99%.  Thus, the
# minimal detectable difference changes from 7.2 to 8.1 to 9.8, and the
# minimal mean level of aldicarb changes from 14.2 to 15.1 to 16.8.

EPA.09.Ex.21.1.aldicarb.df
#   Month   Well Aldicarb.ppb
#1      1 Well.1         19.9
#2      2 Well.1         29.6
#3      3 Well.1         18.7
#4      4 Well.1         24.2
#5      1 Well.2         23.7
#6      2 Well.2         21.9
#7      3 Well.2         26.9
#8      4 Well.2         26.1
#9      1 Well.3          5.6
#10     2 Well.3          3.3
#11     3 Well.3          2.3
```

```
#12     4 Well.3          6.9

sigma <- with(EPA.09.Ex.21.1.aldicarb.df,
  sd(Aldicarb.ppb[Well == "Well.3"]))

sigma
#[1] 2.101388

scaled.mdd <- tTestScaledMdd(n.or.n1 = 4, alpha = 0.01,
  power = c(0.90, 0.95, 0.99), sample.type="one", alternative="greater")

scaled.mdd
#[1] 3.431501 3.853682 4.668749

mdd <- scaled.mdd * sigma
mdd
#[1] 7.210917 8.098083 9.810856

minimal.mean <- mdd + 7

minimal.mean
#[1] 14.21092 15.09808 16.81086


#==========

# Clean up
#---------
rm(scaled.mdd, sigma, mdd, minimal.mean)
```

`twoSampleLinearRankTest` *Two-Sample Linear Rank Test to Detect a Difference Between Two Distributions*

Description

Two-sample linear rank test to detect a difference (usually a shift) between two distributions. The Wilcoxon Rank Sum test is a special case of a linear rank test. The function `twoSampleLinearRankTest` is part of **EnvStats** mainly because this help file gives the necessary background to explain two-sample linear rank tests for censored data (see `twoSampleLinearRankTestCensored`).

Usage

```
twoSampleLinearRankTest(x, y, location.shift.null = 0, scale.shift.null = 1,
  alternative = "two.sided", test = "wilcoxon", shift.type = "location")
```

Arguments

x	numeric vector of values for the first sample. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
y	numeric vector of values for the second sample. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
location.shift.null	numeric scalar indicating the hypothesized value of Δ, the location shift between the two distributions, under the null hypothesis. The default value is location.shift.null=0. This argument is ignored if shift.type="scale".
scale.shift.null	numeric scalar indicating the hypothesized value of τ, the scale shift between the two distributions, under the null hypothesis. The default value is scale.shift.null=1. This argument is ignored if shift.type="location".
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater". See the DETAILS section below for more information.
test	character string indicating which linear rank test to use. The possible values are: "wilcoxon" (the default), "normal.scores", "moods.median", and "savage.scores".
shift.type	character string indicating which kind of shift is being tested. The possible values are "location" (the default) and "scale".

Details

The function twoSampleLinearRankTest allows you to compare two samples using a locally most powerful rank test (LMPRT) to determine whether the two samples come from the same distribution. The sections below explain the concepts of location and scale shifts, linear rank tests, and LMPRT's.

Definitions of Location and Scale Shifts

Let X denote a random variable representing measurements from group 1 with cumulative distribution function (cdf):

$$F_1(t) = Pr(X \leq t) \qquad (1)$$

and let $x_1, x_2, \ldots, x_m$ denote m independent observations from this distribution. Let Y denote a random variable from group 2 with cdf:

$$F_2(t) = Pr(Y \leq t) \qquad (2)$$

and let $y_1, y_2, \ldots, y_n$ denote n independent observations from this distribution. Set $N = m + n$.

General Hypotheses to Test Differences Between Two Populations

A very general hypothesis to test whether two distributions are the same is given by:

$$H_0 : F_1(t) = F_2(t), -\infty < t < \infty \qquad (3)$$

versus the two-sided alternative hypothesis:

$$H_a : F_1(t) \neq F_2(t) \qquad (4)$$

with strict inequality for at least one value of t. The two possible one-sided hypotheses would be:

$$H_0 : F_1(t) \geq F_2(t) \qquad (5)$$

versus the alternative hypothesis:

$$H_a : F_1(t) < F_2(t) \qquad (6)$$

and

$$H_0 : F_1(t) \leq F_2(t) \qquad (7)$$

versus the alternative hypothesis:

$$H_a : F_1(t) > F_2(t) \qquad (8)$$

A similar set of hypotheses to test whether the two distributions are the same are given by (Conover, 1980, p. 216):

$$H_0 : Pr(X < Y) = 1/2 \qquad (9)$$

versus the two-sided alternative hypothesis:

$$H_a : Pr(X < Y) \neq 1/2 \qquad (10)$$

or

$$H_0 : Pr(X < Y) \geq 1/2 \qquad (11)$$

versus the alternative hypothesis:

$$H_a : Pr(X < Y) < 1/2 \qquad (12)$$

or

$$H_0 : Pr(X < Y) \leq 1/2 \qquad (13)$$

versus the alternative hypothesis:

$$H_a : Pr(X < Y) > 1/2 \qquad (14)$$

Note that this second set of hypotheses (9)–(14) is **not** equivalent to the set of hypotheses (3)–(8). For example, if X takes on the values 1 and 4 with probability 1/2 for each, and Y only takes on values in the interval (1, 4) with strict inequality at the enpoints (e.g., Y takes on the values 2 and 3 with probability 1/2 for each), then the null hypothesis (9) is true but the null hypothesis (3) is not true. However, the null hypothesis (3) implies the null hypothesis (9), (5) implies (11), and (7) implies (13).

Location Shift

A special case of the alternative hypotheses (4), (6), and (8) above is the ***location shift*** alternative:

$$H_a : F_1(t) = F_2(t - \Delta) \qquad (15)$$

where Δ denotes the shift between the two groups. (Note: some references refer to (15) above as a shift in the median, but in fact this kind of shift represents a shift in every single quantile, not just the median.) If Δ is positive, this means that observations in group 1 tend to be larger

than observations in group 2, and if Δ is negative, observations in group 1 tend to be smaller than observations in group 2.

The alternative hypothesis (15) is called a ***location shift***: the only difference between the two distributions is a difference in location (e.g., the standard deviation is assumed to be the same for both distributions). A location shift is not applicable to distributions that are bounded below or above by some constant, such as a lognormal distribution. For lognormal distributions, the location shift could refer to a shift in location of the distribution of the log-transformed observations.

For a location shift, the null hypotheses (3) can be generalized as:

$$H_0 : F_1(t) = F_2(t - \Delta_0), -\infty < t < \infty \qquad (16)$$

where Δ_0 denotes the null shift between the two groups. Almost always, however, the null shift is taken to be 0 and we will assume this for the rest of this help file.

Alternatively, the null and alternative hypotheses can be written as

$$H_0 : \Delta = 0 \qquad (17)$$

versus the alternative hypothesis

$$H_a : \Delta > 0 \qquad (18)$$

The other one-sided alternative hypothesis ($\Delta < 0$) and two-sided alternative hypothesis ($\Delta \neq 0$) could be considered as well.

The general hypotheses (3)-(14) are *not* location shift hypotheses (e.g., the standard deviation does not have to be the same for both distributions), but they do allow for distributions that are bounded below or above by a constant (e.g., lognormal distributions).

Scale Shift

A special kind of scale shift replaces the alternative hypothesis (15) with the alternative hypothesis:

$$H_a : F_1(t) = F_2(t/\tau) \qquad (19)$$

where τ denotes the shift in scale between the two groups. Alternatively, the null and alternative hypotheses for this scale shift can be written as

$$H_0 : \tau = 1 \qquad (20)$$

versus the alternative hypothesis

$$H_a : \tau > 1 \qquad (21)$$

The other one-sided alternative hypothesis ($t < 1$) and two-sided alternative hypothesis ($t \neq 1$) could be considered as well.

This kind of scale shift often involves a shift in both location and scale. For example, suppose the underlying distribution for both groups is exponential, with parameter `rate`=λ. Then the mean and standard deviation of the reference group is $1/\lambda$, while the mean and standard deviation of the treatment group is τ/λ. In this case, the alternative hypothesis (21) implies the more general alternative hypothesis (8).

Linear Rank Tests

The usual nonparametric test to test the null hypothesis of the same distribution for both groups

versus the location-shift alternative (18) is the Wilcoxon Rank Sum test (Gilbert, 1987, pp.247-250; Helsel and Hirsch, 1992, pp.118-123; Hollander and Wolfe, 1999). Note that the Mann-Whitney U test is equivalent to the Wilcoxon Rank Sum test (Hollander and Wolfe, 1999; Conover, 1980, p.215, Zar, 2010). Hereafter, this test will be abbreviated as the MWW test. The MWW test is performed by combining the m X observations with the n Y observations and ranking them from smallest to largest, and then computing the statistic

$$W = \sum_{i=1}^{m} R_i \qquad (22)$$

where $R_1, R_2, \ldots, R_m$ denote the ranks of the X observations when the X and Y observations are combined ranked. The null hypothesis (5), (11), or (17) is rejected in favor of the alternative hypothesis (6), (12) or (18) if the value of W is too large. For small sample sizes, the exact distribution of W under the null hypothesis is fairly easy to compute and may be found in tables (e.g., Hollander and Wolfe, 1999; Conover, 1980, pp.448-452). For larger sample sizes, a normal approximation is usually used (Hollander and Wolfe, 1999; Conover, 1980, p.217). For the R function `wilcox.test`, an exact p-value is computed if the samples contain less than 50 finite values and there are no ties.

It is important to note that the MWW test is actually testing the more general hypotheses (9)-(14) (Conover, 1980, p.216; Divine et al., 2013), even though it is often presented as only applying to location shifts.

The MWW W-statistic in Equation (22) is an example of a ***linear rank statistic*** (Hettmansperger, 1984, p.147; Prentice, 1985), which is any statistic that can be written in the form:

$$L = \sum_{i=1}^{m} a(R_i) \qquad (23)$$

where $a()$ denotes a score function. Statistics of this form are also called ***general scores statistics*** (Hettmansperger, 1984, p.147). The MWW test uses the identity score function:

$$a(R_i) = R_i \qquad (24)$$

Any test based on a linear rank statistic is called a ***linear rank test***. Under the null hypothesis (3), (9), (17), or (20), the distribution of the linear rank statistic L does not depend on the form of the underlying distribution of the X and Y observations. Hence, tests based on L are nonparametric (also called distribution-free). If the null hypothesis is not true, however, the distribution of L will depend not only on the distributions of the X and Y observations, but also upon the form the score function $a()$.

Locally Most Powerful Linear Rank Tests

The decision of what scores to use may be based on considering the power of the test. A locally most powerful rank test (LMPRT) of the null hypothesis (17) versus the alternative (18) maximizes the slope of the power (as a function of Δ) in the neighborhood where $\Delta = 0$. A LMPRT of the null hypothesis (20) versus the alternative (21) maximizes the slope of the power (as a function of τ) in the neighborhood where $\tau = 1$. That is, LMPRT's are the best linear rank test you can use for detecting small shifts in location or scale.

Table 1 below shows the score functions associated with the LMPRT's for various assumed underlying distributions (Hettmansperger, 1984, Chapter 3; Millard and Deverel, 1988, p.2090). A test based on the identity score function of Equation (24) is equivalent to a test based on the score shown

in Table 1 associated with the logistic distribution, thus the MWW test is the LMPRT for detecting a location shift when the underlying observations follow the logistic distribution. When the underlying distribution is normal or lognormal, the LMPRT for a location shift uses the "Normal scores" shown in Table 1. When the underlying distribution is exponential, the LMPRT for detecting a scale shift is based on the "Savage scores" shown in Table 1.

Table 1. Scores of LMPRT's for Various Distributions

Distribution	**Score** $a(R_i)$	**Shift Type**	**Test Name**
Logistic	$[2/(N+1)]R_i - 1$	Location	Wilcoxon Rank Sum
Normal or Lognormal (log-scale)	$\Phi^{-1}[R_i/(N+1)]$*	Location	Van der Waerden or Normal scores
Double Exponential	$sign[R_i - (N+1)/2]$	Location	Mood's Median
Exponential or Extreme Value	$\sum_{j=1}^{R_i}(N-j+1)^{-1}$	Scale	Savage scores

* Denotes an approximation to the true score. The symbol Φ denotes the cumulative distribution function of the standard normal distribution, and $sign$ denotes the `sign` function.

A large sample normal approximation to the distribution of the linear rank statistic L for arbitrary score functions is given by Hettmansperger (1984, p.148). Under the null hypothesis (17) or (20), the mean and variance of L are given by:

$$E(L) = \mu_L = \frac{m}{N}\sum_{i=1}^{N} a_i = m\bar{a} \qquad (24)$$

$$Var(L) = \sigma_L^2 = \frac{mn}{N(N-1)}\sum_{i=1}^{N}(a_i - \bar{a})^2 \qquad (25)$$

Hettmansperger (1984, Chapter 3) shows that under the null hypothesis of no difference between the two groups, the statistic

$$z = \frac{L - \mu_L}{\sigma_L} \qquad (26)$$

is approximately distributed as a standard normal random variable for "large" sample sizes. This statistic will tend to be large if the observations in group 1 tend to be larger than the observations in group 2.

Value

a list of class `"htestEnvStats"` containing the results of the hypothesis test. See the help file for `htestEnvStats.object` for details.

Note

The Wilcoxon Rank Sum test, also known as the Mann-Whitney U test, is the standard nonparametric test used to test for differences between two groups (e.g., Zar, 2010; USEPA, 2009, pp.16-14 to 16-20). Other possible nonparametric tests include linear rank tests based on scores other than the ranks, including the "normal scores" test and the "Savage scores" tests. The normal scores test is actually slightly more powerful than the Wilcoxon Rank Sum test for detecting small shifts in location if the underlying distribution is normal or lognormal. In general, however, there will be little difference between these two tests.

The results of calling the function `twoSampleLinearRankTest` with the argument `test="wilcoxon"` will match those of calling the built-in R function `wilcox.test` with the arguments `exact=FALSE` and `correct=FALSE`. In general, it is better to use the built-in function `wilcox.test` for performing the Wilcoxon Rank Sum test, since this function can compute exact (rather than approximate) p-values.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York, Chapter 4.

Divine, G., H.J. Norton, R. Hunt, and J. Dinemann. (2013). A Review of Analysis and Sample Size Calculation Considerations for Wilcoxon Tests. *Anesthesia & Analgesia* **117**, 699–710.

Hettmansperger, T.P. (1984). *Statistical Inference Based on Ranks*. John Wiley and Sons, New York, 323pp.

Hollander, M., and D.A. Wolfe. (1999). *Nonparametric Statistical Methods, Second Edition*. John Wiley and Sons, New York.

Millard, S.P., and S.J. Deverel. (1988). Nonparametric Statistical Methods for Comparing Two Sites Based on Data With Multiple Nondetect Limits. *Water Resources Research*, **24**(12), 2087–2098.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.432–435.

Prentice, R.L. (1985). Linear Rank Tests. In Kotz, S., and N.L. Johnson, eds. *Encyclopedia of Statistical Science*. John Wiley and Sons, New York. Volume 5, pp.51–58.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

`wilcox.test`, `twoSampleLinearRankTestCensored`, `htestEnvStats.object`.

Examples

```
# Generate 15 observations from a normal distribution with parameters
# mean=3 and sd=1.  Call these the observations from the reference group.
# Generate 10 observations from a normal distribution with parameters
# mean=3.5 and sd=1.  Call these the observations from the treatment group.
# Compare the results of calling wilcox.test to those of calling
# twoSampleLinearRankTest with test="normal.scores".
# (The call to set.seed allows you to reproduce this example.)

set.seed(346)
x <- rnorm(15, mean = 3)
y <- rnorm(10, mean = 3.5)

wilcox.test(x, y)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 location shift = 0
#
#Alternative Hypothesis:          True location shift is not equal to 0
#
#Test Name:                       Wilcoxon rank sum test
#
#Data:                            x and y
#
#Test Statistic:                  W = 32
#
#P-value:                         0.0162759


twoSampleLinearRankTest(x, y, test = "normal.scores")

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 Fy(t) = Fx(t)
#
#Alternative Hypothesis:          Fy(t) != Fx(t) for at least one t
#
#Test Name:                       Two-Sample Linear Rank Test:
#                                 Normal Scores Test
#                                 Based on Normal Approximation
#
#Data:                            x = x
#                                 y = y
#
#Sample Sizes:                    nx = 15
#                                 ny = 10
#
#Test Statistic:                  z = -2.431099
#
```

```
#P-value:                          0.01505308

#----------

# Clean up
#---------
rm(x, y)

#==========

# Following Example 6.6 on pages 6.22-6.26 of USEPA (1994b), perform the
# Wilcoxon Rank Sum test for the TcCB data (stored in EPA.94b.tccb.df).
# There are m=47 observations from the reference area and n=77 observations
# from the cleanup unit.  Then compare the results using the other available
# linear rank tests.  Note that Mood's median test yields a p-value less
# than 0.10, while the other tests yield non-significant p-values.
# In this case, Mood's median test is picking up the residual contamination
# in the cleanup unit. (See the example in the help file for quantileTest.)

names(EPA.94b.tccb.df)
#[1] "TcCB.orig" "TcCB"      "Censored"  "Area"

summary(EPA.94b.tccb.df$Area)
#  Cleanup Reference
#       77        47

with(EPA.94b.tccb.df,
  twoSampleLinearRankTest(TcCB[Area=="Cleanup"], TcCB[Area=="Reference"]))

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 Fy(t) = Fx(t)
#
#Alternative Hypothesis:          Fy(t) != Fx(t) for at least one t
#
#Test Name:                       Two-Sample Linear Rank Test:
#                                 Wilcoxon Rank Sum Test
#                                 Based on Normal Approximation
#
#Data:                            x = TcCB[Area == "Cleanup"]
#                                 y = TcCB[Area == "Reference"]
#
#Sample Sizes:                    nx = 77
#                                 ny = 47
#
#Test Statistic:                  z = -1.171872
#
#P-value:                         0.2412485

with(EPA.94b.tccb.df,
  twoSampleLinearRankTest(TcCB[Area=="Cleanup"],
    TcCB[Area=="Reference"], test="normal.scores"))$p.value
```

```
#[1] 0.3399484

with(EPA.94b.tccb.df,
  twoSampleLinearRankTest(TcCB[Area=="Cleanup"],
    TcCB[Area=="Reference"], test="moods.median"))$p.value
#[1] 0.09707393

with(EPA.94b.tccb.df,
  twoSampleLinearRankTest(TcCB[Area=="Cleanup"],
    TcCB[Area=="Reference"], test="savage.scores"))$p.value
#[1] 0.2884351
```

`twoSampleLinearRankTestCensored` *Two-Sample Linear Rank Test to Detect a Difference Between Two Distributions Based on Censored Data*

Description

Two-sample linear rank test to detect a difference (usually a shift) between two distributions based on censored data.

Usage

```
twoSampleLinearRankTestCensored(x, x.censored, y, y.censored,
  censoring.side = "left", location.shift.null = 0, scale.shift.null = 1,
  alternative = "two.sided", test = "logrank", variance = "hypergeometric",
  surv.est = "prentice", shift.type = "location")
```

Arguments

`x`	numeric vector of values for the first sample. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`x.censored`	numeric or logical vector indicating which values of `x` are censored. This must be the same length as `x`. If the mode of `x.censored` is `"logical"`, `TRUE` values correspond to elements of `x` that are censored, and `FALSE` values correspond to elements of `x` that are not censored. If the mode of `x.censored` is `"numeric"`, it must contain only `1`'s and `0`'s; `1` corresponds to `TRUE` and `0` corresponds to `FALSE`. Missing (`NA`) values are allowed but will be removed.
`y`	numeric vector of values for the second sample. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`y.censored`	numeric or logical vector indicating which values of `y` are censored. This must be the same length as `y`. If the mode of `y.censored` is `"logical"`, `TRUE` values correspond to elements of `y` that are censored, and `FALSE` values correspond to elements of `y` that are not censored. If the mode of `y.censored` is `"numeric"`, it must contain only `1`'s and `0`'s; `1` corresponds to `TRUE` and `0` corresponds to `FALSE`. Missing (`NA`) values are allowed but will be removed.

`censoring.side` character string indicating on which side the censoring occurs for the data in `x` and `y`. The possible values are `"left"` (the default) and `"right"`.

`location.shift.null`
numeric scalar indicating the hypothesized value of Δ, the location shift between the two distributions, under the null hypothesis. The default value is `location.shift.null=0`. This argument is ignored if `shift.type="scale"`.

`scale.shift.null`
numeric scalar indicating the hypothesized value of τ, the scale shift between the two distributions, under the null hypothesis. The default value is `scale.shift.null=1`. This argument is ignored if `shift.type="location"`.

`alternative` character string indicating the kind of alternative hypothesis. The possible values are `"two.sided"` (the default), `"less"`, and `"greater"`. See the DETAILS section below for more information.

`test` character string indicating which linear rank test to use. The possible values are: `"logrank"` (the default), `"tarone-ware"`, `"gehan"`, `"peto-peto"`, `"normal.scores.1"`, `"normal.scores.2"`, and `"generalized.sign"`. See the DETAILS section below for more information.

`variance` character string indicating which kind of variance to compute for the test. The possible values are: `"hypergeometric"` (the default), `"permutation"`, and `"asymptotic"`. See the DETAILS section below for more information.

`surv.est` character string indicating what method to use to estimate the survival function. The possible values are `"prentice"` (the default), `"kaplan-meier"`, `"peto-peto"`, and `"altshuler"`. When `test="logrank"` the argument `surv.est` is automatically set to `"altshuler"` and cannot be changed by the user. See the DETAILS section below for more information.

`shift.type` character string indicating which kind of shift is being tested. The possible values are `"location"` (the default) and `"scale"`.

Details

The function `twoSampleLinearRankTestCensored` allows you to compare two samples containing censored observations using a linear rank test to determine whether the two samples came from the same distribution. The help file for `twoSampleLinearRankTest` explains linear rank tests for complete data (i.e., no censored observations are present), and here we assume you are familiar with that material The sections below explain how linear rank tests can be extended to the case of censored data.

Notation

Several authors have proposed extensions of the MWW test to the case of censored data, mainly in the context of survival analysis (e.g., Breslow, 1970; Cox, 1972; Gehan, 1965; Mantel, 1966; Peto and Peto, 1972; Prentice, 1978). Prentice (1978) showed how all of these proposed tests are extensions of a linear rank test to the case of censored observations.

Survival analysis usually deals with right-censored data, whereas environmental data is rarely right-censored but often left-censored (some observations are reported as less than some detection limit). Fortunately, all of the methods developed for right-censored data can be applied to left-censored data as well. (See the sub-section *Left-Censored Data below*.)

In order to explain Prentice's (1978) generalization of linear rank tests to censored data, we will use the following notation that closely follows Prentice (1978), Prentice and Marek (1979), and Latta (1981). Let X denote a random variable representing measurements from group 1 with cumulative distribution function (cdf):

$$F_1(t) = Pr(X \leq t) \qquad (1)$$

and let $x_1, x_2, \ldots, x_m$ denote m independent observations from this distribution. Let Y denote a random variable from group 2 with cdf:

$$F_2(t) = Pr(Y \leq t) \qquad (2)$$

and let $y_1, y_2, \ldots, y_n$ denote n independent observations from this distribution. Set $N = m + n$, the total number of observations. Assume the data are right-censored so that some observations are only recorded as greater than some censoring level, with possibly several different censoring levels. Let $t_1, t_2, \ldots, t_k$ denote the k ordered, unique, uncensored observations for the combined samples (in the context of survival data, t usually stands for "time of death"). For $i = 1, 2, \ldots, k$, let d_{1i} denote the number of observations from sample 1 (the X observations) that are equal to t_i, and let d_{2i} denote the observations from sample 2 (the Y observations) equal to this value. Set

$$d_i = d_{1i} + d_{2i} \qquad (3)$$

the total number of observations equal to t_i. If there are no tied uncensored observations, then $t_i = 1$ for $i = 1, 2, \ldots, k$, otherwise it is greater than 1 for at least on value of i.

For $i = 1, 2, \ldots, k$, let e_{1i} denote the number of censored observations from sample 1 (the X observations) with censoring levels that fall into the interval $[t_i, t_{i+1})$ where $t_{k+1} = \infty$ by definition, and let e_{2i} denote the number of censored observations from sample 2 (the Y observations) with censoring levels that fall into this interval. Set

$$e_i = e_{1i} + e_{2i} \qquad (4)$$

the total number of censoring levels that fall into this interval.

Finally, set n_{1i} equal to the number of observations from sample 1 (uncensored and censored) known to be greater than or equal to t_i, i.e., that lie in the interval $[t_i, \infty)$, set n_{2i} equal to the number of observations from sample 2 (uncensored and censored) that lie in this interval, and set

$$n_i = n_{1i} + n_{2i} \qquad (5)$$

In survival analysis jargon, n_{1i} denotes the number of people from sample 1 who are "at risk" at time t_i, that is, these people are known to still be alive at this time. Similarly, n_{2i} denotes the number of people from sample 2 who are at risk at time t_i, and n_i denotes the total number of people at risk at time t_i.

Score Statistics for Multiply Censored Data

Prentice's (1978) generalization of the two-sample score (linear rank) statistic is given by:

$$\nu = \sum_{i=1}^{k} (d_{1i} c_i + e_{1i} C_i) \qquad (6)$$

where c_i and C_i denote the scores associated with the uncensored and censored observations, respectively. As for complete data, the form of the scores depends upon the assumed underlying

distribution. Table 1 below shows scores for various assumed distributions as presented in Prentice (1978) and Latta (1981) (also see Table 5 of Millard and Deverel, 1988, p.2091). The last column shows what these tests reduce to in the case of complete data (no censored observations).

Table 1. Scores Associated with Various Censored Data Rank Tests

Distribution	**Uncensored Score** (c_i)	**Censored Score** (C_i)	**Test Name**	**Uncensored Analogue**
Logistic	$2\hat{F}_i - 1$	$\hat{F}_i$	Peto-Peto	Wilcoxon Rank Sum
"	$i - n_i$	i	Gehan or Breslow	"
"	$i - \sqrt{n_i}$	i	Tarone-Ware	"
Normal, Lognormal	$\Phi^{-1}(\hat{F}_i)$	$\phi(c_i)/\hat{S}_i$	Normal Scores 1	Normal Scores
"	"	$\frac{n_i C_{i-1} - c_i}{n_i - 1}$	Normal Scores 2	"
Double Exponential	$sign(\hat{F}_i - 0.5)$	$\frac{\hat{F}_i}{1-\hat{F}_i}, if \hat{F}_i < 0.5$ $1, if \hat{F}_i \geq 0.5$	Generalized Sign	Mood's Median
Exponential, Extreme Value	$-log(\tilde{S}_i) - 1$	$-log(\tilde{S}_i)$	Logrank	Savage Scores

In Table 1 above, Φ denotes the cumulative distribution function of the standard normal distribution, ϕ denotes the probability density function of the standard normal distribution, and $sign$ denotes the `sign` function. Also, the quantities $\hat{F}_i$ and $\hat{S}_i$ denote the estimates of the cumulative distribution function (cdf) and survival function, respectively, at time t_i for the combined sample. The estimated cdf is related to the estimated survival function by:

$$\hat{F}_i = 1 - \hat{S}_i \qquad (7)$$

The quantity $\tilde{S}_i$ denotes the Altshuler (1970) estimate of the survival function at time t_i for the combined sample (see below).

The argument `surv.est` determines what method to use estimate the survival function. When `surv.est="prentice"` (the default), the survival function is estimated as:

$$\hat{S}_{i,P} = \prod_{j=1}^{i} \frac{n_j - d_j + 1}{n_j + 1} \qquad (8)$$

(Prentice, 1978). When `surv.est="kaplan-meier"`, the survival function is estimated as:

$$\hat{S}_{i,KM} = \prod_{j=1}^{i} \frac{n_j - d_j}{n_j} \qquad (9)$$

(Kaplan and Meier, 1958), and when `surv.est="peto-peto"`, the survival function is estimated as:

$$\hat{S}_{i,PP} = \frac{1}{2}(\hat{S}_{i,KM} + \hat{S}_{i-1,KM}) \qquad (10)$$

where $\hat{S}_{0,KM} = 0$ (Peto and Peto, 1972). All three of these estimators of the survival function should produce very similar results. When `surv.est="altshuler"`, the survival function is estimated as:

$$\tilde{S}_i = exp(-\sum_{j=1}^{i} \frac{d_j}{n_j}) \qquad (11)$$

(Altshuler, 1970). The scores for the logrank test use this estimator of survival.

Lee and Wang (2003, p. 116) present a slightly different version of the Peto-Peto test. They use the Peto-Peto estimate of the survival function for c_i, but use the Kaplan-Meier estimate of the survival function for C_i.

The scores for the "Normal Scores 1" test shown in Table 1 above are based on the approximation (30) of Prentice (1978). The scores for the "Normal Scores 2" test are based on equation (7) of Prentice and Marek (1979). For the "Normal Scores 2" test, the following rules are used to construct the scores for the censored observations: $C_0 = 0$, and $C_k = 0$ if $n_k = 1$.

The Distribution of the Score Statistic

Under the null hypothesis that the two distributions are the same, the expected value of the score statistic ν in Equation (6) is 0. The variance of ν can be computed in at least three different ways. If the censoring mechanism is the same for both groups, the ***permutation variance*** is appropriate (`variance="permutation"`) and its estimate is given by:

$$\hat{\sigma}_\nu^2 = \frac{mn}{N(N-1)} \sum_{i=1}^{k} (d_i c_i^2 + e_i C_i^2) \qquad (12)$$

Often, however, it is not clear whether this assumption is valid, and both Prentice (1978) and Prentice and Marek (1979) caution against using the permuation variance (Prentice and Marek, 1979, state it can lead to inflated estimates of variance).

If the censoring mechanisms for the two groups are not necessarily the same, a more general estimator of the variance is based on a conditional permutation approach. In this case, the statistic ν in Equation (6) is re-written as:

$$\nu = \sum_{i=1}^{k} w_i [d_{1i} - d_i \frac{n_{1i}}{n_i}] \qquad (13)$$

where

$$w_i = c_i - C_i \qquad (14)$$

c_i and C_i are given above in Table 1, and the conditional permutation or ***hypergeometric estimate*** (`variance="hypergeometric"`) is given by:

$$\hat{\sigma}_\nu^2 = \sum_{i=1}^{k} d_i w_i^2 (\frac{n_{1i}}{n_i})(1 - \frac{n_{1i}}{n_i})(\frac{n_i - d_i}{n_i - 1}) \qquad (15)$$

(Prentice and Marek, 1979; Latta, 1981; Millard and Deverel, 1988). Note that Equation (13) can be thought of as the sum of weighed values of observed minus expected observations.

Prentice (1978) derived an asymptotic estimator of the variance of the score statistic ν given in Equation (6) above based on the log likelihood of the rank vector (`variance="asymptotic"`). This estimator is the same as the hypergeometric variance estimator for the logrank and Gehan tests (assuming no tied uncensored observations), but for the Peto-Peto test, this estimator is given by:

$$\hat{\sigma}_\nu^2 = \sum_{i=1}^{k} \{\hat{S}_i(1 - a_i)b_i - (a_i - \hat{S}_i)b_i[\hat{S}_i b_i + 2 \sum_{j=i+1}^{k} \hat{S}_j b_j]\} \quad (16)$$

where

$$a_i = \prod_{j=1}^{i} \frac{n_j + 1}{n_j + 2} \quad (17)$$

$$b_i = 2d_{1i} + e_{1i} \quad (18)$$

(Prentice, 1978; Latta, 1981; Millard and Deverel, 1988). Note that equation (14) of Millard and Deverel (1988) contains a typographical error.

The Treatment of Ties

If the hypergeometric estimator of variance is being used, no modifications need to be made for ties; Equations (13)-(15) already account for ties. For the case of the permutation or asymptotic variance estimators, Equations (6), (12), and (16) all assume no ties in the uncensored observations. If ties exist in the uncensored observations, Prentice (1978) suggests computing the scores shown in Table 1 above as if there were no ties, and then assigning average scores to the tied observations. (This modification also applies to the quantities a_i and $\hat{S}_i$ in Equation (16) above.) For this algorithm, the statistic in Equation (6) is not in general the same as the one in Equation (13).

Computing a Test Statistic

Under the null hypothesis that the two distributions are the same, the statistic

$$z = \frac{\nu}{\hat{\sigma}_\nu} \quad (19)$$

is approximately distributed as a standard normal random variable for "large" sample sizes. **This statistic will tend to be large if the observations in group 1 (the X observations) tend to be larger than the observations in group 2 (the Y observations)**.

Left-Censored Data

Most of the time, if censored observations occur in environmental data, they are left-censored (e.g., observations are reported as less than one or more detection limits). For the two-sample test of differences between groups, the methods that apply to right-censored data are easily adapted to left-censored data: simply multiply the observations by -1, compute the z-statistic shown in Equation (20), then reverse the sign of this statistic before computing the p-value.

Value

a list of class `"htestCensored"` containing the results of the hypothesis test. See the help file for `htestCensored.object` for details.

Note

All of the tests computed by `twoSampleLinearRankTestCensored` (logrank, Tarone-Ware, Gehan, Peto-Peto, normal scores, and generalized sign) are based on a statistic that is essentially the sum over all uncensored time points of the weighted difference between the observed and expected number of observations at each time point (see Equation (15) above). The tests differ in how they weight the differences between the observed and expected number of observations.

Prentice and Marek (1979) point out that the Gehan test uses weights that depend on the censoring rates within each group and can lead to non-significant outcomes in the case of heavy censoring when in fact a very large difference between the two groups exists.

Latta (1981) performed a Monte Carlo simulation to study the power of the Gehan, logrank, and Peto-Peto tests using all three different estimators of variance (permutation, hypergeometric, and asymptotic). He used lognormal, Weibull, and exponential distributions to generate the observations, and studied two different cases of censoring: uniform censoring for both samples vs. no censoring in the first sample and uniform censoring in the second sample. Latta (1981) used sample sizes of 10 and 50 (both the equal and unequal cases were studied). Latta (1981) found that all three tests maintained the nominal Type I error level (α-level) in the case of equal sample sizes and equal censoring. Also, the Peto-Peto test based on the asymptotic variance appeared to maintain the nominal α-level in all situations, but the other tests were slightly biased in the case of unequal sample sizes and/or unequal censoring. In particular, tests based on the hypergeometric variance are slightly biased for unequal sample sizes. Latta (1981) concludes that if there is no censoring or light censoring, any of the tests may be used (but the hypergeometric variance should not be used if the sample sizes are very different). In the case of heavy censoring where sample sizes are far apart and/or the censoring is very different between samples, the Peto-Peto test based on the asymptotic variance should be used.

Millard and Deverel (1988) also performed a Monte Carlo simulation similar to Latta's (1981) study. They only used the lognormal distribution to generate observations, but also looked at the normal scores test and two ad-hoc modifications of the MWW test. They found the "Normal Scores 2" test shown in Table 1 above to be the best behaved test in terms of maintaining the nominal α-level, but the other tests behaved almost as well. As Latta (1981) found, when sample sizes and censoring are very different between the two groups, the nominal α-level of most of the tests is slightly biased. In the cases where the nominal α-level was maintained, the Peto-Peto test based on the asymptotic variance appeared to be as powerful or more powerful than the normal scores tests.

Neither of the Monte Carlo studies performed by Latta (1981) and Millard and Deverel (1988) looked at the behavior of the two-sample linear rank tests in the presence of several tied uncensored observations (because both studies generated observations from continuous distributions). Note that the results shown in Table 9 of Millard and Deverel (1988, p.2097) are not all correct because they did not allow for tied uncensored values. The last example in the EXAMPLES section below shows the correct values that should appear in that table.

Heller and Venkatraman (1996) performed a Monte Carlo simulation study to compare the behaviors of the Peto-Peto test (using the Prentice, 1978, estimator of survival; they call this the Prentice-Wilcoxon test) and logrank test under varying censoring conditions with sample sizes of 20 and 50 per group based on using the following methods to compute p-values: the asymptotic standard normal approximation, a permutation test approach (this is **NOT** the same as the permutation variance), and a bootstrap approach. Observed times were generated from Weibull and lognormal survival time distributions with independent uniform censoring. They found that for the Peto-Peto test, "the asymptotic test procedure was the most accurate; resampling procedures did not improve

upon its accuracy." For the logrank test, with sample sizes of 20 per group, the usual test based on the asymptotic standard normal approximation tended to have a very slightly higher Type I error rate than assumed (however, for an assumed Type I error rate of 0.05, the largest Type I error rate observed was less than 0.065), whereas the permuation and bootstrap tests performed better; with sample sizes of 50 per group there was no difference in test performance.

Fleming and Harrington (1981) introduced a family of tests (sometimes called G-rho tests) that contain the logrank and Peto-Peto tests as special cases. A single parameter ρ (rho) controls the weights given to the uncensored and censored observations. Positive values of ρ produce tests more sensitive to early differences in the survival function, that is, differences in the cdf at small values. Negative values of ρ produce tests more sensitive to late differences in the survival function, that is, differences in the cdf at large values.

The function `survdiff` in the R package **survival** implements the G-rho family of tests suggested by Flemming and Harrington (1981). Calling `survdiff` with `rho=0` (the default) yields the logrank test. Calling `survdiff` with `rho=1` yields the Peto-Peto test based on the Kaplan-Meier estimate of survival. The function `survdiff` always uses the hypergeometric estimate of variance and the Kaplan-Meier estimate of survival, but it uses the "left-continuous" version of the Kaplan-Meier estimate. The left-continuous K-M estimate of survival is defined as follows: at each death (unique uncensored observation), the estimated survival is equal to the estimated survival based on the ordinary K-M estimate at the prior death time (or 1 for the first death).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Altshuler, B. (1970). Theory for the Measurement of Competing Risks in Animal Experiments. *Mathematical Biosciences* **6**, 1–11.

Breslow, N.E. (1970). A Generalized Kruskal-Wallis Test for Comparing K Samples Subject to Unequal Patterns of Censorship. *Biometrika* **57**, 579–594.

Conover, W.J. (1980). *Practical Nonparametric Statistics*. Second Edition. John Wiley and Sons, New York, Chapter 4.

Cox, D.R. (1972). Regression Models and Life Tables (with Discussion). *Journal of the Royal Statistical Society of London, Series B* **34**, 187–220.

Divine, G., H.J. Norton, R. Hunt, and J. Dinemann. (2013). A Review of Analysis and Sample Size Calculation Considerations for Wilcoxon Tests. *Anesthesia & Analgesia* **117**, 699–710.

Fleming, T.R., and D.P. Harrington. (1981). A Class of Hypothesis Tests for One and Two Sample Censored Survival Data. *Communications in Statistics – Theory and Methods* **A10**(8), 763–794.

Fleming, T.R., and D.P. Harrington. (1991). *Counting Processes & Survival Analysis*. John Wiley and Sons, New York, Chapter 7.

Gehan, E.A. (1965). A Generalized Wilcoxon Test for Comparing Arbitrarily Singly-Censored Samples. *Biometrika* **52**, 203–223.

Harrington, D.P., and T.R. Fleming. (1982). A Class of Rank Test Procedures for Censored Survival Data. *Biometrika* **69**(3), 553–566.

Heller, G., and E. S. Venkatraman. (1996). Resampling Procedures to Compare Two Survival Distributions in the Presence of Right-Censored Data. *Biometrics* **52**, 1204–1213.

Hettmansperger, T.P. (1984). *Statistical Inference Based on Ranks*. John Wiley and Sons, New York, 323pp.

Hollander, M., and D.A. Wolfe. (1999). *Nonparametric Statistical Methods, Second Edition*. John Wiley and Sons, New York.

Kaplan, E.L., and P. Meier. (1958). Nonparametric Estimation From Incomplete Observations. *Journal of the American Statistical Association* **53**, 457–481.

Latta, R.B. (1981). A Monte Carlo Study of Some Two-Sample Rank Tests with Censored Data. *Journal of the American Statistical Association* **76**(375), 713–719.

Mantel, N. (1966). Evaluation of Survival Data and Two New Rank Order Statistics Arising in its Consideration. *Cancer Chemotherapy Reports* **50**, 163-170.

Millard, S.P., and S.J. Deverel. (1988). Nonparametric Statistical Methods for Comparing Two Sites Based on Data With Multiple Nondetect Limits. *Water Resources Research*, **24**(12), 2087–2098.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.432–435.

Peto, R., and J. Peto. (1972). Asymptotically Efficient Rank Invariant Test Procedures (with Discussion). *Journal of the Royal Statistical Society of London, Series A* **135**, 185–206.

Prentice, R.L. (1978). Linear Rank Tests with Right Censored Data. *Biometrika* **65**, 167–179.

Prentice, R.L. (1985). Linear Rank Tests. In Kotz, S., and N.L. Johnson, eds. *Encyclopedia of Statistical Science*. John Wiley and Sons, New York. Volume 5, pp.51–58.

Prentice, R.L., and P. Marek. (1979). A Qualitative Discrepancy Between Censored Data Rank Tests. *Biometrics* **35**, 861–867.

Tarone, R.E., and J. Ware. (1977). On Distribution-Free Tests for Equality of Survival Distributions. *Biometrika* **64**(1), 156–160.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

`twoSampleLinearRankTest`, `survdiff`, `wilcox.test`, `htestCensored.object`.

Examples

```
# The last part of the EXAMPLES section in the help file for
# cdfCompareCensored compares the empirical distribution of copper and zinc
# between two sites:  Alluvial Fan and Basin-Trough (Millard and Deverel, 1988).
# The data for this example are stored in Millard.Deverel.88.df.  Perform a
# test to determine if there is a significant difference between these two
# sites (perform a separate test for the copper and the zinc).

Millard.Deverel.88.df
#    Cu.orig Cu Cu.censored Zn.orig  Zn Zn.censored        Zone Location
```

```
#1       < 1  1        TRUE     <10  10        TRUE Alluvial.Fan        1
#2       < 1  1        TRUE       9   9       FALSE Alluvial.Fan        2
#3         3  3       FALSE      NA  NA       FALSE Alluvial.Fan        3
#.
#.
#.
#116       5  5       FALSE      50  50       FALSE Basin.Trough       48
#117      14 14       FALSE      90  90       FALSE Basin.Trough       49
#118       4  4       FALSE      20  20       FALSE Basin.Trough       50


#------------------------------
# First look at the copper data
#------------------------------

Cu.AF <- with(Millard.Deverel.88.df,
  Cu[Zone == "Alluvial.Fan"])

Cu.AF.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Alluvial.Fan"])

Cu.BT <- with(Millard.Deverel.88.df,
  Cu[Zone == "Basin.Trough"])

Cu.BT.cen <- with(Millard.Deverel.88.df,
  Cu.censored[Zone == "Basin.Trough"])

# Note the large number of tied observations in the copper data
#--------------------------------------------------------------

table(Cu.AF[!Cu.AF.cen])
# 1  2  3  4  5  7  8  9 10 11 12 16 20
# 5 21  6  3  3  3  1  1  1  1  1  1  1

table(Cu.BT[!Cu.BT.cen])
# 1  2  3  4  5  6  8  9 12 14 15 17 23
# 7  4  8  5  1  2  1  2  1  1  1  1  1


# Logrank test with hypergeometric variance:
#-------------------------------------------
twoSampleLinearRankTestCensored(x = Cu.AF, x.censored = Cu.AF.cen,
  y = Cu.BT, y.censored = Cu.BT.cen)

#Results of Hypothesis Test
#Based on Censored Data
#--------------------------
#
#Null Hypothesis:                 Fy(t) = Fx(t)
#
#Alternative Hypothesis:          Fy(t) != Fx(t) for at least one t
#
#Test Name:                       Two-Sample Linear Rank Test:
```

```
#                                  Logrank Test
#                                  with Hypergeometric Variance
#
#Censoring Side:                   left
#
#Censoring Level(s):               x =  1  5 10 20
#                                  y =  1  2  5 10 15
#
#Data:                             x = Cu.AF
#                                  y = Cu.BT
#
#Censoring Variable:               x = Cu.AF.cen
#                                  y = Cu.BT.cen
#
#Number NA/NaN/Inf's Removed:      x = 3
#                                  y = 1
#
#Sample Sizes:                     nx = 65
#                                  ny = 49
#
#Percent Censored:                 x = 26.2%
#                                  y = 28.6%
#
#Test Statistics:                  nu     = -1.8791355
#                                  var.nu = 13.6533490
#                                  z      = -0.5085557
#
#P-value:                          0.6110637


# Compare the p-values produced by the Normal Scores 2 test
# using the hypergeomtric vs. permutation variance estimates.
# Note how much larger the estimated variance is based on
# the permuation variance estimate:
#-----------------------------------------------------------

twoSampleLinearRankTestCensored(x = Cu.AF, x.censored = Cu.AF.cen,
  y = Cu.BT, y.censored = Cu.BT.cen,
  test = "normal.scores.2")$p.value
#[1] 0.2008913

twoSampleLinearRankTestCensored(x = Cu.AF, x.censored = Cu.AF.cen,
  y = Cu.BT, y.censored = Cu.BT.cen,
  test = "normal.scores.2", variance = "permutation")$p.value
#[1] [1] 0.657001


#--------------------------
# Now look at the zinc data
#--------------------------

Zn.AF <- with(Millard.Deverel.88.df,
  Zn[Zone == "Alluvial.Fan"])
```

```
Zn.AF.cen <- with(Millard.Deverel.88.df,
  Zn.censored[Zone == "Alluvial.Fan"])

Zn.BT <- with(Millard.Deverel.88.df,
  Zn[Zone == "Basin.Trough"])

Zn.BT.cen <- with(Millard.Deverel.88.df,
  Zn.censored[Zone == "Basin.Trough"])

# Note the moderate number of tied observations in the zinc data,
# and the "outlier" of 620 in the Alluvial Fan data.
#--------------------------------------------------------------

table(Zn.AF[!Zn.AF.cen])
#  5   7   8   9  10  11  12  17  18  19  20  23  29  30  33  40  50 620
#  1   1   1   1  20   2   1   1   1   1  14   1   1   1   1   1   1   1

table(Zn.BT[!Zn.BT.cen])
# 3  4  5  6  8 10 11 12 13 14 15 17 20 25 30 40 50 60 70 90
# 2  2  2  1  1  5  1  2  1  1  1  2 11  1  4  3  2  2  1  1


# Logrank test with hypergeometric variance:
#-------------------------------------------
twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
  y = Zn.BT, y.censored = Zn.BT.cen)

#Results of Hypothesis Test
#Based on Censored Data
#--------------------------
#
#Null Hypothesis:                 Fy(t) = Fx(t)
#
#Alternative Hypothesis:          Fy(t) != Fx(t) for at least one t
#
#Test Name:                       Two-Sample Linear Rank Test:
#                                 Logrank Test
#                                 with Hypergeometric Variance
#
#Censoring Side:                  left
#
#Censoring Level(s):              x =  3 10
#                                 y =  3 10
#
#Data:                            x = Zn.AF
#                                 y = Zn.BT
#
#Censoring Variable:              x = Zn.AF.cen
#                                 y = Zn.BT.cen
#
#Number NA/NaN/Inf's Removed:     x = 1
#                                 y = 0
```

```
#
#Sample Sizes:                    nx = 67
#                                 ny = 50
#
#Percent Censored:                x = 23.9%
#                                 y =  8.0%
#
#Test Statistics:                 nu     = -6.992999
#                                 var.nu = 17.203227
#                                 z      = -1.686004
#
#P-value:                         0.09179512

#----------

# Compare the p-values produced by the Logrank, Gehan, Peto-Peto,
# and Tarone-Ware tests using the hypergeometric variance.
#-----------------------------------------------------------

twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
  y = Zn.BT, y.censored = Zn.BT.cen,
  test = "logrank")$p.value
#[1] 0.09179512

twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
  y = Zn.BT, y.censored = Zn.BT.cen,
  test = "gehan")$p.value
#[1] 0.0185445

twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
  y = Zn.BT, y.censored = Zn.BT.cen,
  test = "peto-peto")$p.value
#[1] 0.009704529

twoSampleLinearRankTestCensored(x = Zn.AF, x.censored = Zn.AF.cen,
  y = Zn.BT, y.censored = Zn.BT.cen,
  test = "tarone-ware")$p.value
#[1] 0.03457803

#----------

# Clean up
#---------

rm(Cu.AF, Cu.AF.cen, Cu.BT, Cu.BT.cen,
   Zn.AF, Zn.AF.cen, Zn.BT, Zn.BT.cen)


#==========

# Example 16.5 on pages 16-22 to 16.23 of USEPA (2009) shows how to perform
# the Tarone-Ware two sample linear rank test based on censored data using
# observations on tetrachloroethylene (PCE) (ppb) collected at one background
```

```
# and one compliance well.  The data for this example are stored in
# EPA.09.Ex.16.5.PCE.df.

EPA.09.Ex.16.5.PCE.df

#    Well.type PCE.Orig.ppb PCE.ppb Censored
#1  Background           <4     4.0     TRUE
#2  Background          1.5     1.5    FALSE
#3  Background           <2     2.0     TRUE
#4  Background          8.7     8.7    FALSE
#5  Background          5.1     5.1    FALSE
#6  Background           <5     5.0     TRUE
#7  Compliance          6.4     6.4    FALSE
#8  Compliance         10.9    10.9    FALSE
#9  Compliance            7     7.0    FALSE
#10 Compliance         14.3    14.3    FALSE
#11 Compliance          1.9     1.9    FALSE
#12 Compliance           10    10.0    FALSE
#13 Compliance          6.8     6.8    FALSE
#14 Compliance           <5     5.0     TRUE

with(EPA.09.Ex.16.5.PCE.df,
  twoSampleLinearRankTestCensored(
    x = PCE.ppb[Well.type == "Compliance"],
    x.censored = Censored[Well.type == "Compliance"],
    y = PCE.ppb[Well.type == "Background"],
    y.censored = Censored[Well.type == "Background"],
    test = "tarone-ware", alternative = "greater"))

#Results of Hypothesis Test
#Based on Censored Data
#--------------------------
#
#Null Hypothesis:                 Fy(t) = Fx(t)
#
#Alternative Hypothesis:          Fy(t) > Fx(t) for at least one t
#
#Test Name:                       Two-Sample Linear Rank Test:
#                                 Tarone-Ware Test
#                                 with Hypergeometric Variance
#
#Censoring Side:                  left
#
#Censoring Level(s):              x = 5
#                                 y = 2 4 5
#
#Data:                            x = PCE.ppb[Well.type == "Compliance"]
#                                 y = PCE.ppb[Well.type == "Background"]
#
#Censoring Variable:              x = Censored[Well.type == "Compliance"]
#                                 y = Censored[Well.type == "Background"]
#
#Sample Sizes:                    nx = 8
```

```
#                                ny = 6
#
#Percent Censored:               x = 12.5%
#                                y = 50.0%
#
#Test Statistics:                nu     =  8.458912
#                                var.nu = 20.912407
#                                z      =  1.849748
#
#P-value:                        0.03217495

# Compare the p-value for the Tarone-Ware test with p-values from
# the logrank, Gehan, and Peto-Peto tests
#------------------------------------------------------------------

with(EPA.09.Ex.16.5.PCE.df,
  twoSampleLinearRankTestCensored(
    x = PCE.ppb[Well.type == "Compliance"],
    x.censored = Censored[Well.type == "Compliance"],
    y = PCE.ppb[Well.type == "Background"],
    y.censored = Censored[Well.type == "Background"],
    test = "tarone-ware", alternative = "greater"))$p.value
#[1] 0.03217495


with(EPA.09.Ex.16.5.PCE.df,
  twoSampleLinearRankTestCensored(
    x = PCE.ppb[Well.type == "Compliance"],
    x.censored = Censored[Well.type == "Compliance"],
    y = PCE.ppb[Well.type == "Background"],
    y.censored = Censored[Well.type == "Background"],
    test = "logrank", alternative = "greater"))$p.value
#[1] 0.02752793


with(EPA.09.Ex.16.5.PCE.df,
  twoSampleLinearRankTestCensored(
    x = PCE.ppb[Well.type == "Compliance"],
    x.censored = Censored[Well.type == "Compliance"],
    y = PCE.ppb[Well.type == "Background"],
    y.censored = Censored[Well.type == "Background"],
    test = "gehan", alternative = "greater"))$p.value
#[1] 0.03656224

with(EPA.09.Ex.16.5.PCE.df,
  twoSampleLinearRankTestCensored(
    x = PCE.ppb[Well.type == "Compliance"],
    x.censored = Censored[Well.type == "Compliance"],
    y = PCE.ppb[Well.type == "Background"],
    y.censored = Censored[Well.type == "Background"],
    test = "peto-peto", alternative = "greater"))$p.value
#[1] 0.03127296
```

```
#==========

# The results shown in Table 9 of Millard and Deverel (1988, p.2097) are correct
# only for the hypergeometric variance and the modified MWW tests; the other
# results were computed as if there were no ties.  Re-compute the correct
# z-statistics and p-values for the copper and zinc data.

test <- c(rep(c("gehan", "logrank", "peto-peto"), 2), "peto-peto",
  "normal.scores.1", "normal.scores.2", "normal.scores.2")

variance <- c(rep("permutation", 3), rep("hypergeometric", 3),
  "asymptotic", rep("permutation", 2), "hypergeometric")

stats.mat <- matrix(as.numeric(NA), ncol = 4, nrow = 10)

for(i in 1:10) {
  dum.list <- with(Millard.Deverel.88.df,
      twoSampleLinearRankTestCensored(
        x = Cu[Zone == "Basin.Trough"],
        x.censored = Cu.censored[Zone == "Basin.Trough"],
        y = Cu[Zone == "Alluvial.Fan"],
        y.censored = Cu.censored[Zone == "Alluvial.Fan"],
        test = test[i], variance = variance[i]))
  stats.mat[i, 1:2] <- c(dum.list$statistic["z"], dum.list$p.value)

  dum.list <- with(Millard.Deverel.88.df,
      twoSampleLinearRankTestCensored(
        x = Zn[Zone == "Basin.Trough"],
        x.censored = Zn.censored[Zone == "Basin.Trough"],
        y = Zn[Zone == "Alluvial.Fan"],
        y.censored = Zn.censored[Zone == "Alluvial.Fan"],
        test = test[i], variance = variance[i]))
  stats.mat[i, 3:4] <- c(dum.list$statistic["z"], dum.list$p.value)
}

dimnames(stats.mat) <- list(paste(test, variance, sep = "."),
  c("Cu.Z", "Cu.p.value", "Zn.Z", "Zn.p.value"))

round(stats.mat, 2)
#                               Cu.Z Cu.p.value Zn.Z Zn.p.value
#gehan.permutation              0.87       0.38 2.49       0.01
#logrank.permutation            0.79       0.43 1.75       0.08
#peto-peto.permutation          0.92       0.36 2.42       0.02
#gehan.hypergeometric           0.71       0.48 2.35       0.02
#logrank.hypergeometric         0.51       0.61 1.69       0.09
#peto-peto.hypergeometric       1.03       0.30 2.59       0.01
#peto-peto.asymptotic           0.90       0.37 2.37       0.02
#normal.scores.1.permutation    0.94       0.34 2.37       0.02
#normal.scores.2.permutation    0.98       0.33 2.39       0.02
#normal.scores.2.hypergeometric 1.28       0.20 2.48       0.01

#----------
```

```
# Clean up
#---------
rm(test, variance, stats.mat, i, dum.list)
```

`twoSamplePermutationTestLocation` *Two-Sample or Paired-Sample Randomization (Permutation) Test for Location*

Description

Perform a two-sample or paired-sample randomization (permutation) test for location based on either means or medians.

Usage

```
twoSamplePermutationTestLocation(x, y, fcn = "mean", alternative = "two.sided",
  mu1.minus.mu2 = 0, paired = FALSE, exact = FALSE, n.permutations = 5000,
  seed = NULL, tol = sqrt(.Machine$double.eps))
```

Arguments

`x`	numeric vector of observations from population 1. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`y`	numeric vector of observations from population 2. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed. In the case when `paired=TRUE`, the argument `y` must have the same number of elements as the argument `x`.
`fcn`	character string indicating which location parameter to compare between the two groups. The possible values are `fcn="mean"` (the default) and `fcn="median"`. This argument is ignored when `paired=TRUE`.
`alternative`	character string indicating the kind of alternative hypothesis. The possible values are `"two.sided"` (the default), `"less"`, and `"greater"`.
`mu1.minus.mu2`	numeric scalar indicating the hypothesized value of the difference between the means or medians. The default value is `mu1.minus.mu2=0`.
`paired`	logical scalar indicating whether to perform a paired or two-sample permutation test. The possible values are `paired=FALSE` (the default; indicates a two-sample permutation test) and `paired=TRUE` (indicates take differences of pairs and perform a one-sample permutation test).
`exact`	logical scalar indicating whether to perform the exact permutation test (i.e., enumerate all possible permutations) or simply sample from the permutation distribution. The default value is `exact=FALSE`.
`n.permutations`	integer indicating how many times to sample from the permutation distribution when `exact=FALSE`. The default value is `n.permutations=5000`. This argument is ignored when `exact=TRUE`.

`seed` positive integer to pass to the R function `set.seed`. The default is `seed=NULL`, in which case the current value of `.Random.seed` is used. Using the `seed` argument lets you reproduce the exact same result if all other arguments stay the same.

`tol` numeric scalar indicating the tolerance to use for computing the p-value for the two-sample permutation test. The default value is `tol=sqrt(.Machine$double.eps)`. See the DETAILS section below for more information.

Details

Randomization Tests

In 1935, R.A. Fisher introduced the idea of a ***randomization test*** (Manly, 2007, p. 107; Efron and Tibshirani, 1993, Chapter 15), which is based on trying to answer the question: "Did the observed pattern happen by chance, or does the pattern indicate the null hypothesis is not true?" A randomization test works by simply enumerating all of the possible outcomes under the null hypothesis, then seeing where the observed outcome fits in. A randomization test is also called a ***permutation test***, because it involves permuting the observations during the enumeration procedure (Manly, 2007, p. 3).

In the past, randomization tests have not been used as extensively as they are now because of the "large" computing resources needed to enumerate all of the possible outcomes, especially for large sample sizes. The advent of more powerful personal computers and software has allowed randomization tests to become much easier to perform. Depending on the sample size, however, it may still be too time consuming to enumerate all possible outcomes. In this case, the randomization test can still be performed by sampling from the randomization distribution, and comparing the observed outcome to this sampled permutation distribution.

Two-Sample Randomization Test for Location (`paired=FALSE`)

Let $\underline{x} = x_1, x_2, \ldots, x_{n1}$ be a vector of $n1$ independent and identically distributed (i.i.d.) observations from some distribution with location parameter (e.g., mean or median) θ_1, and let $\underline{y} = y_1, y_2, \ldots, y_{n2}$ be a vector of $n2$ i.i.d. observations from the same distribution with possibly different location parameter θ_2.

Consider the test of the null hypothesis that the difference in the location parameters is equal to some specified value:

$$H_0 : \delta = \delta_0 \qquad (1)$$

where

$$\delta = \theta_1 - \theta_2 \qquad (2)$$

and δ_0 denotes the hypothesized difference in the meansures of location (usually $\delta_0 = 0$).

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`)

$$H_a : \delta > \delta_0 \qquad (3)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : \delta < \delta_0 \qquad (4)$$

and the two-sided alternative

$$H_a : \delta \neq \delta_0 \qquad (5)$$

To perform the test of the null hypothesis (1) versus any of the three alternatives (3)-(5), you can use the two-sample permutation test. The two sample permutation test is based on trying to answer the question, "Did the observed difference in means or medians happen by chance, or does the observed difference indicate that the null hypothesis is not true?" Under the null hypothesis, the underlying distributions for each group are the same, therefore it should make no difference which group an observation gets assigned to. The two-sample permutation test works by simply enumerating all possible permutations of group assignments, and for each permutation computing the difference between the measures of location for each group (Manly, 2007, p. 113; Efron and Tibshirani, 1993, p. 202). The measure of location for a group could be the mean, median, or any other measure you want to use. For example, if the observations from Group 1 are 3 and 5, and the observations from Group 2 are 4, 6, and 7, then there are 10 different ways of splitting these five observations into one group of size 2 and another group of size 3. The table below lists all of the possible group assignments, along with the differences in the group means.

Group 1	**Group 2**	**Mean 1 - Mean 2**
3, 4	5, 6, 7	-2.5
3, 5	4, 6, 7	-1.67
3, 6	4, 5, 7	-0.83
3, 7	4, 5, 6	0
4, 5	3, 6, 7	-0.83
4, 6	3, 5, 7	0
4, 7	3, 5, 6	0.83
5, 6	3, 4, 7	0.83
5, 7	3, 4, 6	1.67
6, 7	3, 4, 5	2.5

In this example, the *observed* group assignments and difference in means are shown in the second row of the table.

For a one-sided upper alternative (Equation (3)), the p-value is computed as the proportion of times that the differences of the means (or medians) in the permutation distribution are greater than or equal to the observed difference in means (or medians). For a one-sided lower alternative hypothesis (Equation (4)), the p-value is computed as the proportion of times that the differences in the means (or medians) in the permutation distribution are less than or equal to the observed difference in the means (or medians). For a two-sided alternative hypothesis (Equation (5)), the p-value is computed as the proportion of times the absolute values of the differences in the means (or medians) in the permutation distribution are greater than or equal to the absolute value of the observed difference in the means (or medians).

For this simple example, the one-sided upper, one-sided lower, and two-sided p-values are 0.9, 0.2 and 0.4, respectively.

Note: Because of the nature of machine arithmetic and how the permutation distribution is computed, a one-sided upper p-value is computed as the proportion of times that the differences of the means (or medians) in the permutation distribution are greater than or equal to [the observed difference in means (or medians) - a small tolerance value], where the tolerance value is determined by the argument `tol`. Similarly, a one-sided lower p-value is computed as the proportion of times that the differences in the means (or medians) in the permutation distribution are less than or equal to [the observed difference in the means (or medians) + a small tolerance value]. Finally, a two-sided p-value is computed as the proportion of times the absolute values of the differences in the means

(or medians) in the permutation distribution are greater than or equal to [the absolute value of the observed difference in the means (or medians) - a small tolerance value].

In this simple example, we assumed the hypothesized differences in the means under the null hypothesis was $\delta_0 = 0$. If we had hypothesized a different value for δ_0, then we would have had to subtract this value from each of the observations in Group 1 before permuting the group assignments to compute the permutation distribution of the differences of the means. As in the case of the one-sample permutation test, if the sample sizes for the groups become too large to compute all possible permutations of the group assignments, the permutation test can still be performed by sampling from the permutation distribution and comparing the observed difference in locations to the sampled permutation distribution of the difference in locations.

Unlike the two-sample Student's t-test, we do not have to worry about the normality assumption when we use a permutation test. The permutation test still assumes, however, that under the null hypothesis, the distributions of the observations from each group are exactly the same, and under the alternative hypothesis there is simply a shift in location (that is, the whole distribution of group 1 is shifted by some constant relative to the distribution of group 2). Mathematically, this can be written as follows:

$$F_1(t) = F_2(t - \delta), \quad -\infty < t < \infty \qquad (6)$$

where F_1 and F_2 denote the cumulative distribution functions for group 1 and group 2, respectively. If $\delta > 0$, this implies that the observations in group 1 tend to be larger than the observations in group 2, and if $\delta < 0$, this implies that the observations in group 1 tend to be smaller than the observations in group 2. Thus, the shape and spread (variance) of the two distributions should be the same whether the null hypothesis is true or not. Therefore, the Type I error rate for a permutation test can be affected by differences in variances between the two groups.

Confidence Intervals for the Difference in Means or Medians
Based on the relationship between hypothesis tests and confidence intervals, it is possible to construct a two-sided or one-sided $(1 - \alpha)100\%$ confidence interval for the difference in means or medians based on the two-sample permutation test by finding the values of δ_0 that correspond to obtaining a p-value of α (Manly, 2007, pp. 18–20, 114). A confidence interval based on the bootstrap however, will yield a similar type of confidence interval (Efron and Tibshirani, 1993, p. 214); see the help file for `boot` in the R package **boot**.

Paired-Sample Randomization Test for Location (`paired=TRUE`)
When the argument `paired=TRUE`, the arguments `x` and `y` are assumed to have the same length, and the $n1 = n2 = n$ differences $y_i = x_i - y_i$, $i = 1, 2, \ldots, n$ are assumed to be independent observations from some symmetric distribution with mean μ. The one-sample permutation test can then be applied to the differences.

Value

A list of class `"permutationTest"` containing the results of the hypothesis test. See the help file for `permutationTest.object` for details.

Note

A frequent question in environmental statistics is "Is the concentration of chemical X in Area A greater than the concentration of chemical X in Area B?". For example, in groundwater detection

monitoring at hazardous and solid waste sites, the concentration of a chemical in the groundwater at a downgradient well must be compared to "background". If the concentration is "above" the background then the site enters assessment monitoring. As another example, soil cleanup at a Superfund site may involve comparing the concentration of a chemical in the soil at a "cleaned up" site with the concentration at a "background" site. If the concentration at the "cleaned up" site is "greater" than the background concentration, then further investigation and remedial action may be required. Determining what it means for the chemical concentration to be "greater" than background is a policy decision: you may want to compare averages, medians, 95'th percentiles, etc.

Hypothesis tests you can use to compare "location" between two groups include: Student's t-test, Fisher's randomization test (described in this help file), the Wilcoxon rank sum test, other two-sample linear rank tests, the quantile test, and a test based on a bootstrap confidence interval.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, Chapter 15.

Manly, B.F.J. (2007). *Randomization, Bootstrap and Monte Carlo Methods in Biology*. Third Edition. Chapman & Hall, New York, Chapter 6.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.426–431.

See Also

permutationTest.object, plot.permutationTest, oneSamplePermutationTest, twoSamplePermutationTestProportion, Hypothesis Tests, boot.

Examples

```
# Generate 10 observations from a lognormal distribution with parameters
# mean=5 and cv=2, and and 20 observations from a lognormal distribution with
# parameters mean=10 and cv=2.  Test the null hypothesis that the means of the
# two distributions are the same against the alternative that the mean for
# group 1 is less than the mean for group 2.
# (Note: the call to set.seed allows you to reproduce the same data
# (dat1 and dat2), and setting the argument seed=732 in the call to
# twoSamplePermutationTestLocation() lets you reproduce this example by
# getting the same sample from the permutation distribution).

set.seed(256)
dat1 <- rlnormAlt(10, mean = 5, cv = 2)
dat2 <- rlnormAlt(20, mean = 10, cv = 2)

test.list <- twoSamplePermutationTestLocation(dat1, dat2,
  alternative = "less", seed = 732)
```

```
# Print the results of the test
#------------------------------
test.list

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 mu.x-mu.y = 0
#
#Alternative Hypothesis:          True mu.x-mu.y is less than 0
#
#Test Name:                       Two-Sample Permutation Test
#                                 Based on Differences in Means
#                                 (Based on Sampling
#                                 Permutation Distribution
#                                 5000 Times)
#
#Estimated Parameter(s):          mean of x =  2.253439
#                                 mean of y = 11.825430
#
#Data:                            x = dat1
#                                 y = dat2
#
#Sample Sizes:                    nx = 10
#                                 ny = 20
#
#Test Statistic:                  mean.x - mean.y = -9.571991
#
#P-value:                         0.001


# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

#==========

# The guidance document "Statistical Methods for Evaluating the Attainment of
# Cleanup Standards, Volume 3: Reference-Based Standards for Soils and Solid
# Media" (USEPA, 1994b, pp. 6.22-6.25) contains observations of
# 1,2,3,4-Tetrachlorobenzene (TcCB) in ppb at a Reference Area and a Cleanup Area.
# These data are stored in the data frame EPA.94b.tccb.df.  Use the
# two-sample permutation test to test for a difference in means between the
# two areas vs. the alternative that the mean in the Cleanup Area is greater.
# Do the same thing for the medians.
#
# The permutation test based on comparing means shows a significant differnce,
# while the one based on comparing medians does not.


# First test for a difference in the means.
#------------------------------------------
```

```
mean.list <- with(EPA.94b.tccb.df,
  twoSamplePermutationTestLocation(
    TcCB[Area=="Cleanup"], TcCB[Area=="Reference"],
    alternative = "greater", seed = 47))

mean.list

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 mu.x-mu.y = 0
#
#Alternative Hypothesis:          True mu.x-mu.y is greater than 0
#
#Test Name:                       Two-Sample Permutation Test
#                                 Based on Differences in Means
#                                 (Based on Sampling
#                                 Permutation Distribution
#                                 5000 Times)
#
#Estimated Parameter(s):          mean of x = 3.9151948
#                                 mean of y = 0.5985106
#
#Data:                            x = TcCB[Area == "Cleanup"]
#                                 y = TcCB[Area == "Reference"]
#
#Sample Sizes:                    nx = 77
#                                 ny = 47
#
#Test Statistic:                  mean.x - mean.y = 3.316684
#
#P-value:                         0.0206

dev.new()
plot(mean.list)


#----------

# Now test for a difference in the medians.
#------------------------------------------

median.list <- with(EPA.94b.tccb.df,
  twoSamplePermutationTestLocation(
    TcCB[Area=="Cleanup"], TcCB[Area=="Reference"],
    fcn = "median", alternative = "greater", seed = 47))

median.list

#Results of Hypothesis Test
#--------------------------
#
```

```
#Null Hypothesis:                 mu.x-mu.y = 0
#
#Alternative Hypothesis:          True mu.x-mu.y is greater than 0
#
#Test Name:                       Two-Sample Permutation Test
#                                 Based on Differences in Medians
#                                 (Based on Sampling
#                                 Permutation Distribution
#                                 5000 Times)
#
#Estimated Parameter(s):          median of x = 0.43
#                                 median of y = 0.54
#
#Data:                            x = TcCB[Area == "Cleanup"]
#                                 y = TcCB[Area == "Reference"]
#
#Sample Sizes:                    nx = 77
#                                 ny = 47
#
#Test Statistic:                  median.x - median.y = -0.11
#
#P-value:                         0.936

dev.new()
plot(median.list)

#==========

# Clean up
#---------
rm(test.list, mean.list, median.list)
graphics.off()
```

`twoSamplePermutationTestProportion` *Randomization (Permutation) Test to Compare Two Proportions (Fisher's Exact Test)*

Description

Perform a two-sample randomization (permutation) test to compare two proportions. This is also called Fisher's exact test.

Note: You can perform Fisher's exact test in R using the function `fisher.test`.

Usage

```
twoSamplePermutationTestProportion(x, y, x.and.y = "Binomial Outcomes",
  alternative = "two.sided", tol = sqrt(.Machine$double.eps))
```

Arguments

x, y	When x.and.y="Binomial Outcomes" (the default), x and y are vectors of observations from groups 1 and 2, respectively. The vectors x and y must contain no more than 2 unique values (e.g., 0 and 1, FALSE and TRUE, "No" and "Yes", etc.). In this case, the result of sort(unique(x))[2] is taken to be the value that indicates a "success" for x and the result of sort(unique(y))[2] is taken to be the value that indicates a "success" for y. For example, x = c(FALSE, TRUE, FALSE, TRUE, TRUE) indicates 3 successes in 5 trials, and y = c(1, 0, 0, 0) indicates 1 success in 4 trials. When x.and.y="Binomial Outcomes", missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed. When x.and.y="Number of Successes and Trials", x must be a vector of length 2 containing the number of successes for groups 1 and 2, respectively, and y must be a vector of length 2 that contains the number of trials for groups 1 and 2, respectively. For example, x = c(3, 1) and y = c(5, 4) indicates 3 successes in 5 trials for group 1 and 1 success in 4 trials for group 2.
x.and.y	character string indicating the kind of data stored in the vectors x and y. The possible values are x.and.y="Binomial Outcomes" (the default), and x.and.y="Number Successes and Trials".
alternative	character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (the default), "less", and "greater".
tol	numeric scalar indicating the tolerance to use for computing the p-value for the two-sample permutation test. The default value is tol=sqrt(.Machine$double.eps). See the DETAILS section below for more information.

Details

Randomization Tests

In 1935, R.A. Fisher introduced the idea of a ***randomization test*** (Manly, 2007, p. 107; Efron and Tibshirani, 1993, Chapter 15), which is based on trying to answer the question: "Did the observed pattern happen by chance, or does the pattern indicate the null hypothesis is not true?" A randomization test works by simply enumerating all of the possible outcomes under the null hypothesis, then seeing where the observed outcome fits in. A randomization test is also called a ***permutation test***, because it involves permuting the observations during the enumeration procedure (Manly, 2007, p. 3).

In the past, randomization tests have not been used as extensively as they are now because of the "large" computing resources needed to enumerate all of the possible outcomes, especially for large sample sizes. The advent of more powerful personal computers and software has allowed randomization tests to become much easier to perform. Depending on the sample size, however, it may still be too time consuming to enumerate all possible outcomes. In this case, the randomization test can still be performed by sampling from the randomization distribution, and comparing the observed outcome to this sampled permutation distribution.

Two-Sample Randomization Test for Proportions

Let $\underline{x} = x_1, x_2, \ldots, x_{n_1}$ be a vector of n_1 independent and identically distributed (i.i.d.) observations from a binomial distribution with parameter size=1 and probability of success prob=p_1,

and let $\underline{y} = y_1, y_2, \ldots, y_{n_2}$ be a vector of n_2 i.i.d. observations from a binomial distribution with parameter `size=1` and probability of success `prob=`p_2.

Consider the test of the null hypothesis:

$$H_0 : p_1 = p_2 \qquad (1)$$

The three possible alternative hypotheses are the upper one-sided alternative (`alternative="greater"`)

$$H_a : p_1 > p_2 \qquad (2)$$

the lower one-sided alternative (`alternative="less"`)

$$H_a : p_1 < p_2 \qquad (3)$$

and the two-sided alternative

$$H_a : p_1 \neq p_2 \qquad (4)$$

To perform the test of the null hypothesis (1) versus any of the three alternatives (2)-(4), you can use the two-sample permutation test, which is also called Fisher's exact test. When the observations are from a B(1, p) distribution, the sample mean is an estimate of p. Fisher's exact test is simply a permutation test for the difference between two means from two different groups (see `twoSamplePermutationTestLocation`), where the underlying populations are binomial with size parameter `size=1`, but possibly different values of the `prob` parameter p. Fisher's exact test is usually described in terms of testing hypotheses concerning a 2 x 2 contingency table (van Bell et al., 2004, p. 157; Hollander and Wolfe, 1999, p. 473; Sheskin, 2011; Zar, 2010, p. 561). The probabilities associated with the permutation distribution can be computed by using the hypergeometric distribution.

Value

A list of class `"permutationTest"` containing the results of the hypothesis test. See the help file for `permutationTest.object` for details.

Note

Sometimes in environmental data analysis we are interested in determining whether two probabilities or rates or proportions differ from each other. For example, we may ask the question: "Does exposure to pesticide X increase the risk of developing cancer Y?", where cancer Y may be liver cancer, stomach cancer, or some other kind of cancer. One way environmental scientists attempt to answer this kind of question is by conducting experiments on rodents in which one group (the "treatment" or "exposed" group) is exposed to the pesticide and the other group (the control group) is not. The incidence of cancer Y in the exposed group is compared with the incidence of cancer Y in the control group. (See Rodricks (2007) for a discussion of extrapolating results from experiments involving rodents to consequences in humans and the associated difficulties).

Hypothesis tests you can use to compare proportions or probability of "success" between two groups include Fisher's exact test and the test based on the normal approximation (see the R help file for `prop.test`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Efron, B., and R.J. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, Chapter 15.

Hollander, M., and D.A. Wolfe. (1999). *Nonparametric Statistical Methods*. Second Edition. John Wiley and Sons, New York, p.473.

Manly, B.F.J. (2007). *Randomization, Bootstrap and Monte Carlo Methods in Biology*. Third Edition. Chapman & Hall, New York, Chapter 6.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL, pp.441–446.

Graham, S.L., K.J. Davis, W.H. Hansen, and C.H. Graham. (1975). Effects of Prolonged Ethylene Thiourea Ingestion on the Thyroid of the Rat. *Food and Cosmetics Toxicology*, **13**(5), 493–499.

Rodricks, J.V. (1992). *Calculated Risks: The Toxicity and Human Health Risks of Chemicals in Our Environment*. Cambridge University Press, New York.

Rodricks, J.V. (2007). *Calculated Risks: The Toxicity and Human Health Risks of Chemicals in Our Environment*. Second Edition. Cambridge University Press, New York.

Sheskin, D.J. (2011). *Handbook of Parametric and Nonparametric Statistical Procedures* Fifth Edition. CRC Press, Boca Raton, FL.

van Belle, G., L.D. Fisher, Heagerty, P.J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences, 2nd Edition*. John Wiley & Sons, New York, p. 157.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ, p. 561.

See Also

permutationTest.object, plot.permutationTest, twoSamplePermutationTestLocation, oneSamplePermutationTest, Hypothesis Tests, boot.

Examples

```
# Generate 10 observations from a binomial distribution with parameters
# size=1 and prob=0.3, and 20 observations from a binomial distribution
# with parameters size=1 and prob=0.5.  Test the null hypothesis that the
# probability of "success" for the two distributions is the same against the
# alternative that the probability of "success" for group 1 is less than
# the probability of "success" for group 2.
# (Note: the call to set.seed allows you to reproduce this example).

set.seed(23)
dat1 <- rbinom(10, size = 1, prob = 0.3)
dat2 <- rbinom(20, size = 1, prob = 0.5)

test.list <- twoSamplePermutationTestProportion(
  dat1, dat2, alternative = "less")

#----------

# Print the results of the test
```

```
#------------------------------
test.list

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 p.x - p.y = 0
#
#Alternative Hypothesis:          True p.x - p.y is less than 0
#
#Test Name:                       Two-Sample Permutation Test
#                                 Based on Differences in Proportions
#                                 (Fisher's Exact Test)
#
#Estimated Parameter(s):          p.hat.x = 0.60
#                                 p.hat.y = 0.65
#
#Data:                            x = dat1
#                                 y = dat2
#
#Sample Sizes:                    nx = 10
#                                 ny = 20
#
#Test Statistic:                  p.hat.x - p.hat.y = -0.05
#
#P-value:                         0.548026

#----------

# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

#----------

# Compare to the results of fisher.test
#--------------------------------------
x11 <- sum(dat1)
x21 <- length(dat1) - sum(dat1)
x12 <- sum(dat2)
x22 <- length(dat2) - sum(dat2)
mat <- matrix(c(x11, x12, x21, x22), ncol = 2)
fisher.test(mat, alternative = "less")

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 odds ratio = 1
#
#Alternative Hypothesis:          True odds ratio is less than 1
#
#Test Name:                       Fisher's Exact Test for Count Data
```

```
#
#Estimated Parameter(s):          odds ratio = 0.8135355
#
#Data:                            mat
#
#P-value:                         0.548026
#
#95% Confidence Interval:         LCL = 0.000000
#                                 UCL = 4.076077

#==========

# Rodricks (1992, p. 133) presents data from an experiment by
# Graham et al. (1975) in which different groups of rats were exposed to
# various concentration levels of ethylene thiourea (ETU), a decomposition
# product of a certain class of fungicides that can be found in treated foods.
# In the group exposed to a dietary level of 250 ppm of ETU, 16 out of 69 rats
# (23%) developed thyroid tumors, whereas in the control group
# (no exposure to ETU) only 2 out of 72 (3%) rats developed thyroid tumors.
# If we use Fisher's exact test to test the null hypothesis that the proportion
# of rats exposed to 250 ppm of ETU who will develop thyroid tumors over their
# lifetime is no greater than the proportion of rats not exposed to ETU who will
# develop tumors, we get a one-sided upper p-value of 0.0002.  Therefore, we
# conclude that the true underlying rate of tumor incidence in the exposed group
# is greater than in the control group.
#
# The data for this example are stored in Graham.et.al.75.etu.df.

# Look at the data
#-----------------

Graham.et.al.75.etu.df
#  dose tumors  n proportion
#1    0      2 72 0.02777778
#2    5      2 75 0.02666667
#3   25      1 73 0.01369863
#4  125      2 73 0.02739726
#5  250     16 69 0.23188406
#6  500     62 70 0.88571429

# Perform the test for a difference in tumor rates
#-------------------------------------------------

Num.Tumors <- with(Graham.et.al.75.etu.df, tumors[c(5, 1)])
Sample.Sizes <- with(Graham.et.al.75.etu.df, n[c(5, 1)])

test.list <- twoSamplePermutationTestProportion(
  x = Num.Tumors, y = Sample.Sizes,
  x.and.y="Number Successes and Trials", alternative = "greater")

#----------

# Print the results of the test
```

```
#------------------------------
test.list

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 p.x - p.y = 0
#
#Alternative Hypothesis:          True p.x - p.y is greater than 0
#
#Test Name:                       Two-Sample Permutation Test
#                                 Based on Differences in Proportions
#                                 (Fisher's Exact Test)
#
#Estimated Parameter(s):          p.hat.x = 0.23188406
#                                 p.hat.y = 0.02777778
#
#Data:                            x = Num.Tumors
#                                 n = Sample.Sizes
#
#Sample Sizes:                    nx = 69
#                                 ny = 72
#
#Test Statistic:                  p.hat.x - p.hat.y = 0.2041063
#
#P-value:                         0.0002186462

#----------

# Plot the results of the test
#-----------------------------
dev.new()
plot(test.list)

#==========

# Clean up
#---------
rm(test.list, x11, x12, x21, x22, mat, Num.Tumors, Sample.Sizes)
#graphics.off()
```

---

varGroupTest *Test for Homogeneity of Variance Among Two or More Groups*

Description

Test the null hypothesis that the variances of two or more normal distributions are the same using Levene's or Bartlett's test.

Usage

```
varGroupTest(object, ...)

## S3 method for class 'formula'
varGroupTest(object, data = NULL, subset,
  na.action = na.pass, ...)

## Default S3 method:
varGroupTest(object, group, test = "Levene",
  correct = TRUE, data.name = NULL, group.name = NULL,
  parent.of.data = NULL, subset.expression = NULL, ...)

## S3 method for class 'data.frame'
varGroupTest(object, ...)

## S3 method for class 'matrix'
varGroupTest(object, ...)

## S3 method for class 'list'
varGroupTest(object, ...)
```

Arguments

object	an object containing data for 2 or more groups whose variances are to be compared. In the default method, the argument object must be a numeric vector. When object is a data frame, all columns must be numeric. When object is a matrix, it must be a numeric matrix. When object is a list, all components must be numeric vectors. In the formula method, a symbolic specification of the form y ~ g can be given, indicating the observations in the vector y are to be grouped according to the levels of the factor g. Missing (NA), undefined (NaN), and infinite (Inf, -Inf) values are allowed but will be removed.
data	when object is a formula, data specifies an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which summaryStats is called.
subset	when object is a formula, subset specifies an optional vector specifying a subset of observations to be used.
na.action	when object is a formula, na.action specifies a function which indicates what should happen when the data contain NAs. The default is na.pass.
group	when object is a numeric vector, group is a factor or character vector indicating which group each observation belongs to. When object is a matrix or data frame this argument is ignored and the columns define the groups. When object is a list this argument is ignored and the components define the groups. When object is a formula, this argument is ignored and the right-hand side of the formula specifies the grouping variable.

test	character string indicating which test to use. The possible values are "Levene" (Levene's test; the default) and "Bartlett" (Bartlett's test).
correct	logical scalar indicating whether to use the correction factor for Bartlett's test. The default value is correct=TRUE. This argument is ignored if test="Levene".
data.name	character string indicating the name of the data used for the group variance test. The default value is data.name=deparse(substitute(object)).
group.name	character string indicating the name of the data used to create the groups. The default value is group.name=deparse(substitute(group)).
parent.of.data	character string indicating the source of the data used for the group variance test.
subset.expression	character string indicating the expression used to subset the data.
...	additional arguments affecting the group variance test.

Details

The function varGroupTest performs Levene's or Bartlett's test for homogeneity of variance among two or more groups. The R function var.test compares two variances.

Bartlett's test is very sensitive to the assumption of normality and will tend to give significant results even when the null hypothesis is true if the underlying distributions have long tails (e.g., are leptokurtic). Levene's test is almost as powerful as Bartlett's test when the underlying distributions are normal, and unlike Bartlett's test it tends to maintain the assumed $alpha$-level when the underlying distributions are not normal (Snedecor and Cochran, 1989, p.252; Milliken and Johnson, 1992, p.22; Conover et al., 1981). Thus, Levene's test is generally recommended over Bartlett's test.

Value

a list of class "htestEnvStats" containing the results of the group variance test. Objects of class "htestEnvStats" have special printing and plotting methods. See the help file for htestEnvStats.object for details.

Note

Chapter 11 of USEPA (2009) discusses using Levene's test to test the assumption of equal variances between monitoring wells or to test that the variance is stable over time when performing intrawell tests.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Conover, W.J., M.E. Johnson, and M.M. Johnson. (1981). A Comparative Study of Tests for Homogeneity of Variances, with Applications to the Outer Continental Shelf Bidding Data. *Technometrics* **23**(4), 351-361.

Davis, C.B. (1994). Environmental Regulatory Statistics. In Patil, G.P., and C.R. Rao, eds., *Handbook of Statistics, Vol. 12: Environmental Statistics*. North-Holland, Amsterdam, a division of Elsevier, New York, NY, Chapter 26, 817-865.

Milliken, G.A., and D.E. Johnson. (1992). *Analysis of Messy Data, Volume I: Designed Experiments*. Chapman & Hall, New York.

Snedecor, G.W., and W.G. Cochran. (1989). *Statistical Methods, Eighth Edition*. Iowa State University Press, Ames Iowa.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

USEPA. (2010). *Errata Sheet - March 2009 Unified Guidance*. EPA 530/R-09-007a, August 9, 2010. Office of Resource Conservation and Recovery, Program Information and Implementation Division. U.S. Environmental Protection Agency, Washington, D.C.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

var.test, varTest.

Examples

```
# Example 11-2 of USEPA (2009, page 11-7) gives an example of
# testing the assumption of equal variances across wells for arsenic
# concentrations (ppb) in groundwater collected at 6 monitoring
# wells over 4 months.  The data for this example are stored in
# EPA.09.Ex.11.1.arsenic.df.

head(EPA.09.Ex.11.1.arsenic.df)
#  Arsenic.ppb Month Well
#1        22.9     1    1
#2         3.1     2    1
#3        35.7     3    1
#4         4.2     4    1
#5         2.0     1    2
#6         1.2     2    2

longToWide(EPA.09.Ex.11.1.arsenic.df, "Arsenic.ppb", "Month", "Well",
  paste.row.name = TRUE, paste.col.name = TRUE)
#        Well.1 Well.2 Well.3 Well.4 Well.5 Well.6
#Month.1   22.9    2.0    2.0    7.8   24.9    0.3
#Month.2    3.1    1.2  109.4    9.3    1.3    4.8
#Month.3   35.7    7.8    4.5   25.9    0.8    2.8
#Month.4    4.2   52.0    2.5    2.0   27.0    1.2

varGroupTest(Arsenic.ppb ~ Well, data = EPA.09.Ex.11.1.arsenic.df)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 Ratio of each pair of variances = 1
#
#Alternative Hypothesis:          At least one variance differs
```

```
#
#Test Name:                      Levene's Test for
#                                Homogenity of Variance
#
#Estimated Parameter(s):         Well.1 =  246.8158
#                                Well.2 =  592.6767
#                                Well.3 = 2831.4067
#                                Well.4 =  105.2967
#                                Well.5 =  207.4467
#                                Well.6 =    3.9025
#
#Data:                           Arsenic.ppb
#
#Grouping Variable:              Well
#
#Data Source:                    EPA.09.Ex.11.1.arsenic.df
#
#Sample Sizes:                   Well.1 = 4
#                                Well.2 = 4
#                                Well.3 = 4
#                                Well.4 = 4
#                                Well.5 = 4
#                                Well.6 = 4
#
#Test Statistic:                 F = 4.564176
#
#Test Statistic Parameters:      num df   =  5
#                                denom df = 18
#
#P-value:                        0.007294084
```

`varTest` *One-Sample Chi-Squared Test on Variance*

Description

Estimate the variance, test the null hypothesis using the chi-squared test that the variance is equal to a user-specified value, and create a confidence interval for the variance.

Usage

```
varTest(x, alternative = "two.sided", conf.level = 0.95,
  sigma.squared = 1, data.name = NULL)
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`alternative`	character string indicating the kind of alternative hypothesis. The possible values are `"two.sided"` (the default), `"greater"`, and `"less"`.

conf.level	numeric scalar between 0 and 1 indicating the confidence level associated with the confidence interval for the population variance. The default value is conf.level=0.95.
sigma.squared	a numeric scalar indicating the hypothesized value of the variance. The default value is sigma.squared=1.
data.name	character string indicating the name of the data used for the test of variance.

Details

The function varTest performs the one-sample chi-squared test of the hypothesis that the population variance is equal to the user specified value given by the argument sigma.squared, and it also returns a confidence interval for the population variance. The R function var.test performs the F-test for comparing two variances.

Value

A list of class "htest" containing the results of the hypothesis test. See the help file for htest.object for details.

Note

Just as you can perform tests of hypothesis on measures of location (mean, median, percentile, etc.), you can do the same thing for measures of spread or variability. Usually, we are interested in estimating variability only because we want to quantify the uncertainty of our estimated location or percentile. Sometimes, however, we are interested in estimating variability and quantifying the uncertainty in our estimate of variability (for example, for performing a sensitivity analysis for power or sample size calculations), or testing whether the population variability is equal to a certain value. There are at least two possible methods of performing a one-sample hypothesis test on variability:

- Perform a hypothesis test for the population variance based on the chi-squared statistic, assuming the underlying population is normal.
- Perform a hypothesis test for any kind of measure of spread assuming any kind of underlying distribution based on a bootstrap confidence interval (using, for example, the package **boot**).

You can use varTest for the first method.

Note: For a one-sample test of location, Student's t-test is fairly robust to departures from normality (i.e., the Type I error rate is maintained), as long as the sample size is reasonably "large." The chi-squared test on the population variance, however, is extremely sensitive to departures from normality. For example, if the underlying population is skewed, the actual Type I error rate will be larger than assumed.

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

van Belle, G., L.D. Fisher, Heagerty, P.J., and Lumley, T. (2004). *Biostatistics: A Methodology for the Health Sciences, 2nd Edition*. John Wiley & Sons, New York.

Millard, S.P., and N.K. Neerchal. (2001). *Environmental Statistics with S-PLUS*. CRC Press, Boca Raton, FL.

Zar, J.H. (2010). *Biostatistical Analysis*. Fifth Edition. Prentice-Hall, Upper Saddle River, NJ.

See Also

var.test, varGroupTest.

Examples

```
# Generate 20 observations from a normal distribution with parameters
# mean=2 and sd=1.  Test the null hypothesis that the true variance is
# equal to 0.5 against the alternative that the true variance is not
# equal to 0.5.
# (Note: the call to set.seed allows you to reproduce this example).

set.seed(23)
dat <- rnorm(20, mean = 2, sd = 1)
varTest(dat, sigma.squared = 0.5)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 variance = 0.5
#
#Alternative Hypothesis:          True variance is not equal to 0.5
#
#Test Name:                       Chi-Squared Test on Variance
#
#Estimated Parameter(s):          variance = 0.753708
#
#Data:                            dat
#
#Test Statistic:                  Chi-Squared = 28.64090
#
#Test Statistic Parameter:        df = 19
#
#P-value:                         0.1436947
#
#95% Confidence Interval:         LCL = 0.4359037
#                                 UCL = 1.6078623

# Note that in this case we would not reject the
# null hypothesis at the 5% or even the 10% level.

# Clean up
rm(dat)
```

`ZeroModifiedLognormal` *The Zero-Modified Lognormal (Delta) Distribution*

Description

Density, distribution function, quantile function, and random generation for the zero-modified lognormal distribution with parameters `meanlog`, `sdlog`, and `p.zero`.

The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0.

Usage

```
dzmlnorm(x, meanlog = 0, sdlog = 1, p.zero = 0.5)
pzmlnorm(q, meanlog = 0, sdlog = 1, p.zero = 0.5)
qzmlnorm(p, meanlog = 0, sdlog = 1, p.zero = 0.5)
rzmlnorm(n, meanlog = 0, sdlog = 1, p.zero = 0.5)
```

Arguments

`x`	vector of quantiles.
`q`	vector of quantiles.
`p`	vector of probabilities between 0 and 1.
`n`	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
`meanlog`	vector of means of the normal (Gaussian) part of the distribution on the log scale. The default is `meanlog=0`.
`sdlog`	vector of (positive) standard deviations of the normal (Gaussian) part of the distribution on the log scale. The default is `sdlog=1`.
`p.zero`	vector of probabilities between 0 and 1 indicating the probability the random variable equals 0. For `rzmlnorm` this must be a single, non-missing number.

Details

The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0. This distribution was introduced without a name by Aitchison (1955), and the name Δ-distribution was coined by Aitchison and Brown (1957, p.95). It is a special case of a "zero-modified" distribution (see Johnson et al., 1992, p. 312).

Let $f(x; \mu, \sigma)$ denote the density of a lognormal random variable X with parameters `meanlog`=μ and `sdlog`=σ. The density function of a zero-modified lognormal (delta) random variable Y with parameters `meanlog`=μ, `sdlog`=σ, and `p.zero`=p, denoted $h(y; \mu, \sigma, p)$, is given by:

$$h(y; \mu, \sigma, p) = \begin{array}{ll} p & \text{for } y = 0 \\ (1-p) f(y; \mu, \sigma) & \text{for } y > 0 \end{array}$$

Note that μ is *not* the mean of the zero-modified lognormal distribution on the log scale; it is the mean of the lognormal part of the distribution on the log scale. Similarly, σ is *not* the standard deviation of the zero-modified lognormal distribution on the log scale; it is the standard deviation of the lognormal part of the distribution on the log scale.

Let γ and δ denote the mean and standard deviation of the overall zero-modified lognormal distribution on the log scale. Aitchison (1955) shows that:

$$E[log(Y)] = \gamma = (1-p)\mu$$

$$Var[log(Y)] = \delta^2 = (1-p)\sigma^2 + p(1-p)\mu^2$$

Note that when `p.zero`=p=`0`, the zero-modified lognormal distribution simplifies to the lognormal distribution.

Value

`dzmlnorm` gives the density, `pzmlnorm` gives the distribution function, `qzmlnorm` gives the quantile function, and `rzmlnorm` generates random deviates.

Note

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as "Below Detection Limit" (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see `qqPlotCensored`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J. (1955). On the Distribution of a Positive Random Variable Having a Discrete Probability Mass at the Origin. *Journal of the American Statistical Association* **50**, 901-908.

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special reference to its uses in economics)*. Cambridge University Press, London. pp.94-99.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, pp.47-51.

Gibbons, RD., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley and Sons, Hoboken, NJ.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135-146.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R*. Second Edition. John Wiley and Sons, Hoboken, NJ, Chapter 1.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, p.312.

Owen, W., and T. DeRouen. (1980). Estimation of the Mean for Lognormal Data Containing Zeros and Left-Censored Values, with Applications to the Measurement of Worker Exposure to Air Contaminants. *Biometrics* **36**, 707-719.

USEPA (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, Permits and State Programs Division, US Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

Zero-Modified Lognormal (Alternative Parameterization), Lognormal, LognormalAlt, Zero-Modified Normal, `ezmlnorm`, Probability Distributions and Random Numbers.

Examples

```
# Density of the zero-modified lognormal (delta) distribution with
# parameters meanlog=0, sdlog=1, and p.zero=0.5, evaluated at
# 0, 0.5, 1, 1.5, and 2:

dzmlnorm(seq(0, 2, by = 0.5))
#[1] 0.50000000 0.31374804 0.19947114 0.12248683
#[5] 0.07843701

#----------

# The cdf of the zero-modified lognormal (delta) distribution with
# parameters meanlog=1, sdlog=2, and p.zero=0.1, evaluated at 4:

pzmlnorm(4, 1, 2, .1)
#[1] 0.6189203

#----------

# The median of the zero-modified lognormal (delta) distribution with
# parameters meanlog=2, sdlog=3, and p.zero=0.1:

qzmlnorm(0.5, 2, 3, 0.1)
#[1] 4.859177

#----------
```

```
# Random sample of 3 observations from the zero-modified lognormal
# (delta) distribution with parameters meanlog=1, sdlog=2, and p.zero=0.4.
# (Note: The call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rzmlnorm(3, 1, 2, 0.4)
#[1] 0.000000 0.000000 3.146641
```

`ZeroModifiedLognormalAlt` *The Zero-Modified Lognormal (Delta) Distribution (Alternative Parameterization)*

Description

Density, distribution function, quantile function, and random generation for the zero-modified lognormal distribution with parameters `mean`, `cv`, and `p.zero`.

The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0.

Usage

```
dzmlnormAlt(x, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)
pzmlnormAlt(q, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)
qzmlnormAlt(p, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)
rzmlnormAlt(n, mean = exp(1/2), cv = sqrt(exp(1) - 1), p.zero = 0.5)
```

Arguments

`x`	vector of quantiles.
`q`	vector of quantiles.
`p`	vector of probabilities between 0 and 1.
`n`	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.
`mean`	vector of means of the lognormal part of the distribution on the. The default is `mean=exp(1/2)`.
`cv`	vector of (positive) coefficients of variation of the lognormal part of the distribution. The default is `cv=sqrt(exp(1) - 1)`.
`p.zero`	vector of probabilities between 0 and 1 indicating the probability the random variable equals 0. For `rzmlnormAlt` this must be a single, non-missing number.

Details

The zero-modified lognormal (delta) distribution is the mixture of a lognormal distribution with a positive probability mass at 0. This distribution was introduced without a name by Aitchison (1955), and the name Δ-distribution was coined by Aitchison and Brown (1957, p.95). It is a special case of a "zero-modified" distribution (see Johnson et al., 1992, p. 312).

Let $f(x; \theta, \tau)$ denote the density of a lognormal random variable X with parameters `mean`=θ and `cv`=τ. The density function of a zero-modified lognormal (delta) random variable Y with parameters `mean`=θ, `cv`=τ, and `p.zero`=p, denoted $h(y; \theta, \tau, p)$, is given by:

$$h(y; \theta, \tau, p) = \begin{array}{ll} p & \text{for } y = 0 \\ (1-p) f(y; \theta, \tau) & \text{for } y > 0 \end{array}$$

Note that θ is *not* the mean of the zero-modified lognormal distribution; it is the mean of the lognormal part of the distribution. Similarly, τ is *not* the coefficient of variation of the zero-modified lognormal distribution; it is the coefficient of variation of the lognormal part of the distribution.

Let γ, δ, and ω denote the mean, standard deviation, and coefficient of variation of the overall zero-modified lognormal distribution. Let η denote the standard deviation of the lognormal part of the distribution, so that $\eta = \theta\tau$. Aitchison (1955) shows that:

$$E(Y) = \gamma = (1-p)\theta$$

$$Var(Y) = \delta^2 = (1-p)\eta^2 + p(1-p)\theta^2$$

so that

$$\omega = \sqrt{(\tau^2 + p)/(1-p)}$$

Note that when `p.zero`=p=`0`, the zero-modified lognormal distribution simplifies to the lognormal distribution.

Value

`dzmlnormAlt` gives the density, `pzmlnormAlt` gives the distribution function, `qzmlnormAlt` gives the quantile function, and `rzmlnormAlt` generates random deviates.

Note

The zero-modified lognormal (delta) distribution is sometimes used to model chemical concentrations for which some observations are reported as "Below Detection Limit" (the nondetects are assumed equal to 0). See, for example, Gilliom and Helsel (1986), Owen and DeRouen (1980), and Gibbons et al. (2009, Chapter 12). USEPA (2009, Chapter 15) recommends this strategy only in specific situations, and Helsel (2012, Chapter 1) strongly discourages this approach to dealing with non-detects.

A variation of the zero-modified lognormal (delta) distribution is the zero-modified normal distribution, in which a normal distribution is mixed with a positive probability mass at 0.

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see `qqPlotCensored`).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J. (1955). On the Distribution of a Positive Random Variable Having a Discrete Probability Mass at the Origin. *Journal of the American Statistical Association* **50**, 901-908.

Aitchison, J., and J.A.C. Brown (1957). *The Lognormal Distribution (with special reference to its uses in economics)*. Cambridge University Press, London. pp.94-99.

Crow, E.L., and K. Shimizu. (1988). *Lognormal Distributions: Theory and Applications*. Marcel Dekker, New York, pp.47-51.

Gibbons, RD., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley and Sons, Hoboken, NJ.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135-146.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R*. Second Edition. John Wiley and Sons, Hoboken, NJ, Chapter 1.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, p.312.

Owen, W., and T. DeRouen. (1980). Estimation of the Mean for Lognormal Data Containing Zeros and Left-Censored Values, with Applications to the Measurement of Worker Exposure to Air Contaminants. *Biometrics* **36**, 707-719.

USEPA (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, Permits and State Programs Division, US Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

Zero-Modified Lognormal, LognormalAlt, `ezmlnormAlt`, Probability Distributions and Random Numbers.

Examples

```
# Density of the zero-modified lognormal (delta) distribution with
# parameters mean=10, cv=1, and p.zero=0.5, evaluated at
# 9, 10, and 11:

dzmlnormAlt(9:11, mean = 10, cv = 1, p.zero = 0.5)
#[1] 0.02552685 0.02197043 0.01891924

#----------

# The cdf of the zero-modified lognormal (delta) distribution with
```

```
# parameters mean=10, cv=2, and p.zero=0.1, evaluated at 8:

pzmlnormAlt(8, 10, 2, .1)
#[1] 0.709009

#----------

# The median of the zero-modified lognormal (delta) distribution with
# parameters mean=10, cv=2, and p.zero=0.1:

qzmlnormAlt(0.5, 10, 2, 0.1)
#[1] 3.74576

#----------

# Random sample of 3 observations from the zero-modified lognormal
# (delta) distribution with parameters mean=10, cv=2, and p.zero=0.4.
# (Note: The call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rzmlnormAlt(3, 10, 2, 0.4)
#[1] 0.000000 0.000000 4.907131
```

`ZeroModifiedNormal` *The Zero-Modified Normal Distribution*

Description

Density, distribution function, quantile function, and random generation for the zero-modified normal distribution with parameters `mean`, `sd`, and `p.zero`.

The zero-modified normal distribution is the mixture of a normal distribution with a positive probability mass at 0.

Usage

```
dzmnorm(x, mean = 0, sd = 1, p.zero = 0.5)
pzmnorm(q, mean = 0, sd = 1, p.zero = 0.5)
qzmnorm(p, mean = 0, sd = 1, p.zero = 0.5)
rzmnorm(n, mean = 0, sd = 1, p.zero = 0.5)
```

Arguments

`x`	vector of quantiles.
`q`	vector of quantiles.
`p`	vector of probabilities between 0 and 1.
`n`	sample size. If `length(n)` is larger than 1, then `length(n)` random values are returned.

mean	vector of means of the normal (Gaussian) part of the distribution. The default is mean=0.
sd	vector of (positive) standard deviations of the normal (Gaussian) part of the distribution. The default is sd=1.
p.zero	vector of probabilities between 0 and 1 indicating the probability the random variable equals 0. For rzmnorm this must be a single, non-missing number.

Details

The zero-modified normal distribution is the mixture of a normal distribution with a positive probability mass at 0.

Let $f(x; \mu, \sigma)$ denote the density of a normal (Gaussian) random variable X with parameters mean=μ and sd=σ. The density function of a zero-modified normal random variable Y with parameters mean=μ, sd=σ, and p.zero=p, denoted $h(y; \mu, \sigma, p)$, is given by:

$$h(y; \mu, \sigma, p) = \begin{cases} p & \text{for } y = 0 \\ (1-p) f(y; \mu, \sigma) & \text{for } y \neq 0 \end{cases}$$

Note that μ is *not* the mean of the zero-modified normal distribution; it is the mean of the normal part of the distribution. Similarly, σ is *not* the standard deviation of the zero-modified normal distribution; it is the standard deviation of the normal part of the distribution.

Let γ and δ denote the mean and standard deviation of the overall zero-modified normal distribution. Aitchison (1955) shows that:

$$E(Y) = \gamma = (1-p)\mu$$

$$Var(Y) = \delta^2 = (1-p)\sigma^2 + p(1-p)\mu^2$$

Note that when p.zero=p=0, the zero-modified normal distribution simplifies to the normal distribution.

Value

dzmnorm gives the density, pzmnorm gives the distribution function, qzmnorm gives the quantile function, and rzmnorm generates random deviates.

Note

The zero-modified normal distribution is sometimes used to model chemical concentrations for which some observations are reported as "Below Detection Limit". See, for example USEPA (1992c, pp.27-34) and Gibbons et al. (2009, Chapter 12). Note, however, that USEPA (1992c) has been superseded by USEPA (2009) which recommends this strategy only in specific situations (see Chapter 15 of the document). This strategy is strongly discouraged by Helsel (2012, Chapter 1).

In cases where you want to model chemical concentrations for which some observations are reported as "Below Detection Limit" and you want to treat the non-detects as equal to 0, it will usually be more appropriate to model the data with a zero-modified lognormal (delta) distribution since chemical concentrations are bounded below at 0 (e.g., Gilliom and Helsel, 1986; Owen and DeRouen, 1980).

One way to try to assess whether a zero-modified lognormal (delta), zero-modified normal, censored normal, or censored lognormal is the best model for the data is to construct both censored and detects-only probability plots (see qqPlotCensored).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Aitchison, J. (1955). On the Distribution of a Positive Random Variable Having a Discrete Probability Mass at the Origin. *Journal of the American Statistical Association* **50**, 901-908.

Gilliom, R.J., and D.R. Helsel. (1986). Estimation of Distributional Parameters for Censored Trace Level Water Quality Data: 1. Estimation Techniques. *Water Resources Research* **22**, 135-146.

Gibbons, RD., D.K. Bhaumik, and S. Aryal. (2009). *Statistical Methods for Groundwater Monitoring*. Second Edition. John Wiley and Sons, Hoboken, NJ.

Helsel, D.R. (2012). *Statistics for Censored Environmental Data Using Minitab and R*. Second Edition. John Wiley and Sons, Hoboken, NJ, Chapter 1.

Johnson, N. L., S. Kotz, and A.W. Kemp. (1992). *Univariate Discrete Distributions*. Second Edition. John Wiley and Sons, New York, p.312.

Owen, W., and T. DeRouen. (1980). Estimation of the Mean for Lognormal Data Containing Zeros and Left-Censored Values, with Applications to the Measurement of Worker Exposure to Air Contaminants. *Biometrics* **36**, 707-719.

USEPA (1992c). *Statistical Analysis of Ground-Water Monitoring Data at RCRA Facilities: Addendum to Interim Final Guidance*. Office of Solid Waste, Permits and State Programs Division, US Environmental Protection Agency, Washington, D.C.

USEPA. (2009). *Statistical Analysis of Groundwater Monitoring Data at RCRA Facilities, Unified Guidance*. EPA 530/R-09-007, March 2009. Office of Resource Conservation and Recovery Program Implementation and Information Division. U.S. Environmental Protection Agency, Washington, D.C.

See Also

Zero-Modified Lognormal, Normal, ezmnorm, Probability Distributions and Random Numbers.

Examples

```
# Density of the zero-modified normal distribution with parameters
# mean=2, sd=1, and p.zero=0.5, evaluated at 0, 0.5, 1, 1.5, and 2:

dzmnorm(seq(0, 2, by = 0.5), mean = 2)
#[1] 0.5000000 0.0647588 0.1209854 0.1760327 0.1994711

#----------

# The cdf of the zero-modified normal distribution with parameters
# mean=3, sd=2, and p.zero=0.1, evaluated at 4:
```

```
pzmnorm(4, 3, 2, .1)
#[1] 0.7223162

#----------

# The median of the zero-modified normal distribution with parameters
# mean=3, sd=1, and p.zero=0.1:

qzmnorm(0.5, 3, 1, 0.1)
#[1] 2.86029

#----------

# Random sample of 3 observations from the zero-modified normal distribution
# with parameters mean=3, sd=1, and p.zero=0.4.
# (Note: The call to set.seed simply allows you to reproduce this example.)

set.seed(20)
rzmnorm(3, 3, 1, 0.4)
#[1] 0.000000 0.000000 3.073168
```

`zTestGevdShape` *Test Whether the Shape Parameter of a Generalized Extreme Value Distribution is Equal to 0*

Description

Estimate the shape parameter of a generalized extreme value distribution and test the null hypothesis that the true value is equal to 0.

Usage

```
zTestGevdShape(x, pwme.method = "unbiased",
  plot.pos.cons = c(a = 0.35, b = 0), alternative = "two.sided")
```

Arguments

`x`	numeric vector of observations. Missing (`NA`), undefined (`NaN`), and infinite (`Inf`, `-Inf`) values are allowed but will be removed.
`pwme.method`	character string specifying the method of estimating the probability-weighted moments. Possible values are `"unbiased"` (method based on the U-statistic; the default), and `"plotting.position"` (plotting position). See the help file for `egevd` for more information on these estimation methods.
`plot.pos.cons`	numeric vector of length 2 specifying the constants used in the formula for the plotting positions. The default value is `plot.pos.cons=c(a=0.35, b=0)`. If this vector has a names attribute with the value `c("a","b")` or `c("b","a")`, then the elements will be matched by name in the formula for computing the plotting positions. Otherwise, the first element is mapped to the name `"a"` and

the second element to the name "b". See the help file for egevd for more information. This argument is ignored if pwme.method is not equal to "plotting.position".

alternative — character string indicating the kind of alternative hypothesis. The possible values are "two.sided" (shape not equal to 0; the default), "less" (shape less than 0), and "greater" (shape greater than 0).

Details

Let $\underline{x} = x_1, x_2, \ldots, x_n$ be a vector of n observations from a generalized extreme value distribution with parameters location=η, scale=θ, and shape=κ. Furthermore, let $\hat{\kappa}_{pwme}$ denote the probability-weighted moments estimator (PWME) of the shape parameter κ (see the help file for egevd). Then the statistic

$$z = \frac{\hat{\kappa}_{pwme}}{\sqrt{0.5633/n}} \qquad (1)$$

is asymptotically distributed as a N(0,1) random variable under the null hypothesis $H_0 : \kappa = 0$ (Hosking et al., 1985). The function zTestGevdShape performs the usual one-sample z-test using the statistic computed in Equation (1). The PWME of κ may be computed using either U-statistic type probability-weighted moments estimators or plotting-position type estimators (see egevd). Although Hosking et al. (1985) base their statistic on plotting-position type estimators, Hosking and Wallis (1995) recommend using the U-statistic type estimators for almost all applications.

This test is only asymptotically correct. Hosking et al. (1985), however, found that the α-level is adequately maintained for samples as small as $n = 25$.

Value

A list of class "htestEnvStats" containing the results of the hypothesis test. See the help file for htestEnvStats.object for details.

Note

Two-parameter extreme value distributions (EVD) have been applied extensively since the 1930's to several fields of study, including the distributions of hydrological and meteorological variables, human lifetimes, and strength of materials. The three-parameter generalized extreme value distribution (GEVD) was introduced by Jenkinson (1955) to model annual maximum and minimum values of meteorological events. Since then, it has been used extensively in the hydological and meteorological fields.

The three families of EVDs are all special kinds of GEVDs. When the shape parameter $\kappa = 0$, the GEVD reduces to the Type I extreme value (Gumbel) distribution. When $\kappa > 0$, the GEVD is the same as the Type II extreme value distribution, and when $\kappa < 0$ it is the same as the Type III extreme value distribution.

Hosking et al. (1985) introduced the test used by the function zTestGevdShape to test the null hypothesis $H_0 : \kappa = 0$. They found this test has power comparable to the modified likelihood-ratio test, which was found by Hosking (1984) to be the best overall test the thirteen tests he considered.

Fill and Stedinger (1995) denote this test the "kappa test" and compare it with the *L-C*s test suggested by Chowdhury et al. (1991) and the probability plot correlation coefficient goodness-of-fit test for the Gumbel distribution given by Vogel (1986) (see the sub-section for test="ppcc" under the Details section of the help file for gofTest).

Author(s)

Steven P. Millard (<EnvStats@ProbStatInfo.com>)

References

Chowdhury, J.U., J.R. Stedinger, and L. H. Lu. (1991). Goodness-of-Fit Tests for Regional Generalized Extreme Value Flood Distributions. *Water Resources Research* **27**(7), 1765–1776.

Fill, H.D., and J.R. Stedinger. (1995). L Moment and Probability Plot Correlation Coefficient Goodness-of-Fit Tests for the Gumbel Distribution and Impact of Autocorrelation. *Water Resources Research* **31**(1), 225–229.

Hosking, J.R.M. (1984). Testing Whether the Shape Parameter is Zero in the Generalized Extreme-Value Distribution. *Biometrika* **71**(2), 367–374.

Hosking, J.R.M., and J.R. Wallis (1995). A Comparison of Unbiased and Plotting-Position Estimators of L Moments. *Water Resources Research* **31**(8), 2019–2025.

Hosking, J.R.M., J.R. Wallis, and E.F. Wood. (1985). Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments. *Technometrics* **27**(3), 251–261.

Jenkinson, A.F. (1955). The Frequency Distribution of the Annual Maximum (or Minimum) of Meteorological Events. *Quarterly Journal of the Royal Meteorological Society* **81**, 158–171.

Vogel, R.M. (1986). The Probability Plot Correlation Coefficient Test for the Normal, Lognormal, and Gumbel Distributional Hypotheses. *Water Resources Research* **22**(4), 587–590. (Correction, *Water Resources Research* **23**(10), 2013, 1987.)

See Also

GEVD, egevd, EVD, eevd, Goodness-of-Fit Tests, htestEnvStats.object.

Examples

```
# Generate 25 observations from a generalized extreme value distribution with
# parameters location=2, scale=1, and shape=1, and test the null hypothesis
# that the shape parameter is equal to 0.
# (Note: the call to set.seed simply allows you to reproduce this example.)

set.seed(250)

dat <- rgevd(25, location = 2, scale = 1, shape = 1)

zTestGevdShape(dat)

#Results of Hypothesis Test
#--------------------------
#
#Null Hypothesis:                 shape = 0
#
#Alternative Hypothesis:          True shape is not equal to 0
#
#Test Name:                       Z-test of shape=0 for GEVD
#
```

```
#Estimated Parameter(s):          shape = 0.6623014
#
#Estimation Method:               Unbiased pwme
#
#Data:                            dat
#
#Sample Size:                     25
#
#Test Statistic:                  z = 4.412206
#
#P-value:                         1.023225e-05

#----------

# Clean up
#---------
rm(dat)
```

Index

∗ **Abstract**
CastilloAndHadi1994, 62
HoskingEtAl1985, 670
∗ **Castillo**
CastilloAndHadi1994, 62
∗ **EnvStats Functions By Category**
FcnsByCat, 555
∗ **Estimation**
FcnsByCatEstDistParams, 560
FcnsByCatEstDistQuants, 561
∗ **Euler's Constant**
EulersConstant, 531
∗ **Hosking**
HoskingEtAl1985, 670
∗ **Inverse Prediction**
FcnsByCatCalibration, 556
∗ **Less Than Detection Limit**
FcnsByCatCensoredData, 556
∗ **Percentile**
FcnsByCatEstDistQuants, 561
∗ **Quantile**
FcnsByCatEstDistQuants, 561
∗ **aplot**
stat_mean_sd_text, 1176
stat_median_iqr_text, 1180
stat_n_text, 1184
stat_test_text, 1188
∗ **classes**
boxcox.object, 35
boxcoxCensored.object, 47
boxcoxLm.object, 51
calibrate.object, 60
distChoose.object, 147
distChooseCensored.object, 165
estimate.object, 522
estimateCensored.object, 527
gof.object, 595
gofCensored.object, 599
gofGroup.object, 602
gofOutlier.object, 612
gofTwoSample.object, 657
htest.htestEnvStats.object, 675
htestCensored.object, 679
permutationTest.object, 773
summaryStats.object, 1222
∗ **datagen**
Chi, 87
Empirical, 302
EVD, 535
GammaAlt, 575
GEVD, 592
Lognormal3, 734
LognormalAlt, 738
LognormalMix, 741
LognormalMixAlt, 743
LognormalTrunc, 745
LognormalTruncAlt, 748
NormalMix, 755
NormalTrunc, 757
Pareto, 767
simulateMvMatrix, 1160
simulateVector, 1168
Triangular, 1286
ZeroModifiedLognormal, 1367
ZeroModifiedLognormalAlt, 1370
ZeroModifiedNormal, 1373
∗ **datasets**
ACE.13.TCE.df, 10
Beal.2010.Pb.df, 22
Benthic.df, 22
BJC.2000.df, 25
Distribution.df, 168
Environmental, 355
EPA.02d.Ex.2.ug.per.L.vec, 357
EPA.02d.Ex.4.mg.per.kg.vec, 357
EPA.02d.Ex.6.mg.per.kg.vec, 358
EPA.02d.Ex.9.mg.per.L.vec, 358
EPA.09.Ex.10.1.nickel.df, 359

EPA.09.Ex.11.1.arsenic.df, 359
EPA.09.Ex.12.1.ccl4.df, 360
EPA.09.Ex.12.4.naphthalene.df, 361
EPA.09.Ex.13.1.iron.df, 361
EPA.09.Ex.14.1.manganese.df, 362
EPA.09.Ex.14.3.alkalinity.df, 363
EPA.09.Ex.14.4.arsenic.df, 363
EPA.09.Ex.14.8.df, 364
EPA.09.Ex.15.1.manganese.df, 365
EPA.09.Ex.16.1.sulfate.df, 366
EPA.09.Ex.16.2.benzene.df, 366
EPA.09.Ex.16.4.copper.df, 367
EPA.09.Ex.16.5.PCE.df, 368
EPA.09.Ex.17.1.loglead.df, 369
EPA.09.Ex.17.2.toluene.df, 369
EPA.09.Ex.17.3.chrysene.df, 370
EPA.09.Ex.17.3.log.chrysene.df, 371
EPA.09.Ex.17.4.copper.df, 372
EPA.09.Ex.17.5.chloride.df, 372
EPA.09.Ex.17.6.sulfate.df, 373
EPA.09.Ex.17.7.sodium.df, 374
EPA.09.Ex.18.1.arsenic.df, 375
EPA.09.Ex.18.2.chrysene.df, 375
EPA.09.Ex.18.3.TCE.df, 376
EPA.09.Ex.18.4.xylene.df, 377
EPA.09.Ex.19.1.sulfate.df, 378
EPA.09.Ex.19.2.chloride.df, 379
EPA.09.Ex.19.5.mercury.df, 379
EPA.09.Ex.20.1.nickel.df, 380
EPA.09.Ex.21.1.aldicarb.df, 381
EPA.09.Ex.21.2.benzene.df, 382
EPA.09.Ex.21.5.beryllium.df, 382
EPA.09.Ex.21.6.nitrate.df, 383
EPA.09.Ex.21.7.TCE.df, 384
EPA.09.Ex.22.1.VC.df, 385
EPA.09.Ex.22.2.Specific.Conductance.df, 385
EPA.09.Ex.6.3.sulfate.df, 386
EPA.09.Ex.7.1.arsenic.df, 387
EPA.09.Table.9.1.TCE.df, 388
EPA.09.Table.9.3.df, 388
EPA.09.Table.9.4.nickel.vec, 389
EPA.89b.aldicarb1.df, 390
EPA.89b.aldicarb2.df, 391
EPA.89b.benzene.df, 391
EPA.89b.cadmium.df, 392
EPA.89b.chlordane1.df, 392
EPA.89b.chlordane2.df, 393
EPA.89b.edb.df, 394
EPA.89b.lead.df, 394
EPA.89b.loglead.df, 395
EPA.89b.manganese.df, 395
EPA.89b.sulfate.df, 396
EPA.89b.t29.df, 397
EPA.89b.toc.vec, 397
EPA.92c.arsenic1.df, 398
EPA.92c.arsenic2.df, 398
EPA.92c.arsenic3.df, 399
EPA.92c.benzene1.df, 400
EPA.92c.benzene2.df, 400
EPA.92c.ccl4.df, 401
EPA.92c.chrysene.df, 402
EPA.92c.copper1.df, 402
EPA.92c.copper2.df, 403
EPA.92c.lognickel1.df, 404
EPA.92c.nickel1.df, 404
EPA.92c.nickel2.df, 405
EPA.92c.toluene.df, 405
EPA.92c.zinc.df, 406
EPA.92d.chromium.df, 407
EPA.92d.chromium.vec, 407
EPA.94b.lead.df, 408
EPA.94b.tccb.df, 408
EPA.97.cadmium.111.df, 409
Gibbons.et.al.09.Alkilinity.vec, 594
Gibbons.et.al.09.Vinyl.Chloride.vec, 595
Graham.et.al.75.etu.df, 666
Grice.Bain.80.mat, 667
Helsel.Cohn.88.app.b.df, 668
Helsel.Cohn.88.silver.df, 669
Helsel.Hirsch.02.Mayfly.df, 669
Lin.Evans.80.df, 715
Millard.Deverel.88.df, 752
Modified.TcCB.df, 753
NIOSH.89.air.lead.vec, 755
Olympic.NH4.df, 760
Ozone.NE.df, 766
ProUCL.5.2.TRS.df, 1091
ProUCL.Crit.Vals.for.AD.Test.for.Gamma.array, 1092
ProUCL.Crit.Vals.for.KS.Test.for.Gamma.array, 1093
Refinery.CO.df, 1127

Skagit.NH3_N.df, 1172
Total.P.df, 1285
* **design**
aovN, 13
aovPower, 16
ciBinomHalfWidth, 89
ciBinomN, 95
ciNormHalfWidth, 103
ciNormN, 107
ciNparConfLevel, 111
ciNparN, 115
ciTableMean, 117
ciTableProp, 124
linearTrendTestN, 716
linearTrendTestPower, 718
linearTrendTestScaledMds, 726
plotAovDesign, 811
plotCiBinomDesign, 815
plotCiNormDesign, 821
plotCiNparDesign, 826
plotLinearTrendTestDesign, 829
plotPredIntNormDesign, 841
plotPredIntNormSimultaneousTestPowerCurve, 845
plotPredIntNparDesign, 853
plotPredIntNparSimultaneousDesign, 857
plotPredIntNparSimultaneousTestPowerCurve, 860
plotPropTestDesign, 865
plotTolIntNormDesign, 870
plotTolIntNparDesign, 874
plotTTestDesign, 878
plotTTestLnormAltDesign, 882
predIntNormHalfWidth, 954
predIntNormN, 963
predIntNormSimultaneousTestPower, 987
predIntNparConfLevel, 1008
predIntNparN, 1011
predIntNparSimultaneous, 1014
predIntNparSimultaneousConfLevel, 1027
predIntNparSimultaneousN, 1031
propTestMdd, 1072
propTestN, 1078
propTestPower, 1085
tolIntNormHalfWidth, 1254
tolIntNormN, 1263
tolIntNparConfLevel, 1273
tolIntNparCoverage, 1276
tolIntNparN, 1278
tTestAlpha, 1289
tTestLnormAltN, 1291
tTestLnormAltPower, 1296
tTestLnormAltRatioOfMeans, 1303
tTestN, 1307
tTestPower, 1311
tTestScaledMdd, 1318
* **distribution**
CastilloAndHadi1994, 62
cdfCompare, 64
cdfCompareCensored, 69
cdfPlot, 78
Chi, 87
ciBinomHalfWidth, 89
ciBinomN, 95
ciNormHalfWidth, 103
ciNormN, 107
ciNparConfLevel, 111
ciNparN, 115
ciTableMean, 117
ciTableProp, 124
ebeta, 176
ebinom, 179
ecdfPlot, 186
ecdfPlotCensored, 190
eevd, 196
eexp, 201
egamma, 204
egammaAltCensored, 214
egammaCensored, 223
egeom, 232
egevd, 234
ehyper, 241
elnorm, 244
elnorm3, 246
elnormAlt, 259
elnormAltCensored, 272
elnormCensored, 289
elogis, 298
Empirical, 302
enbinom, 307
enorm, 310
enormCensored, 315
enpar, 335

enparCensored, 341
epareto, 410
epdfPlot, 412
epois, 415
epoisCensored, 419
eqbeta, 427
eqbinom, 429
eqevd, 432
eqexp, 435
eqgamma, 437
eqgeom, 444
eqgevd, 446
eqhyper, 449
eqlnorm, 452
eqlnorm3, 456
eqlnormCensored, 460
eqlogis, 467
eqnbinom, 469
eqnorm, 472
eqnormCensored, 478
eqnpar, 487
eqpareto, 503
eqpois, 505
equnif, 509
eqweibull, 511
eqzmlnorm, 514
eqzmnorm, 517
errorBar, 519
EulersConstant, 531
eunif, 532
EVD, 535
evNormOrdStats, 538
eweibull, 542
ezmlnorm, 545
ezmnorm, 550
GammaAlt, 575
GEVD, 592
gpqCiNormCensored, 659
gpqTolIntNormCensored, 663
HoskingEtAl1985, 670
linearTrendTestN, 716
linearTrendTestPower, 718
linearTrendTestScaledMds, 726
lMoment, 729
Lognormal3, 734
LognormalAlt, 738
LognormalMix, 741
LognormalMixAlt, 743
LognormalTrunc, 745
LognormalTruncAlt, 748
NormalMix, 755
NormalTrunc, 757
Pareto, 767
pdfPlot, 770
plotCiBinomDesign, 815
plotCiNormDesign, 821
plotCiNparDesign, 826
plotLinearTrendTestDesign, 829
plotPredIntLnormAltSimultaneousTestPowerCurve, 832
plotPredIntLnormAltTestPowerCurve, 837
plotPredIntNormDesign, 841
plotPredIntNormSimultaneousTestPowerCurve, 845
plotPredIntNormTestPowerCurve, 850
plotPredIntNparDesign, 853
plotPredIntNparSimultaneousDesign, 857
plotPredIntNparSimultaneousTestPowerCurve, 860
plotPropTestDesign, 865
plotTolIntNormDesign, 870
plotTolIntNparDesign, 874
plotTTestDesign, 878
plotTTestLnormAltDesign, 882
ppointsCensored, 891
predIntGamma, 908
predIntGammaSimultaneous, 915
predIntLnorm, 925
predIntLnormAltSimultaneousTestPower, 934
predIntLnormAltTestPower, 938
predIntLnormSimultaneous, 940
predIntNorm, 947
predIntNormHalfWidth, 954
predIntNormK, 957
predIntNormN, 963
predIntNormSimultaneous, 967
predIntNormSimultaneousK, 978
predIntNormSimultaneousTestPower, 987
predIntNormTestPower, 995
predIntNpar, 999
predIntNparConfLevel, 1008
predIntNparN, 1011

predIntNparSimultaneous, 1014
predIntNparSimultaneousConfLevel, 1027
predIntNparSimultaneousN, 1031
predIntNparSimultaneousTestPower, 1036
predIntPois, 1043
pwMoment, 1094
qqPlot, 1098
qqPlotCensored, 1107
qqPlotGestalt, 1116
simulateMvMatrix, 1160
simulateVector, 1168
tolIntGamma, 1224
tolIntLnorm, 1232
tolIntLnormCensored, 1238
tolIntNorm, 1243
tolIntNormCensored, 1248
tolIntNormHalfWidth, 1254
tolIntNormK, 1257
tolIntNormN, 1263
tolIntNpar, 1267
tolIntNparConfLevel, 1273
tolIntNparCoverage, 1276
tolIntNparN, 1278
tolIntPois, 1281
Triangular, 1286
tTestAlpha, 1289
tTestLnormAltN, 1291
tTestLnormAltPower, 1296
tTestLnormAltRatioOfMeans, 1303
tTestN, 1307
tTestPower, 1311
tTestScaledMdd, 1318
ZeroModifiedLognormal, 1367
ZeroModifiedLognormalAlt, 1370
ZeroModifiedNormal, 1373

∗ **dplot**

ppointsCensored, 891

∗ **hplot**

cdfCompare, 64
cdfCompareCensored, 69
cdfPlot, 78
ecdfPlot, 186
ecdfPlotCensored, 190
epdfPlot, 412
errorBar, 519
geom_stripchart, 579
pdfPlot, 770
plotAovDesign, 811
plotCiBinomDesign, 815
qqPlot, 1098
qqPlotCensored, 1107
qqPlotGestalt, 1116
stripChart, 1193

∗ **htestEnvStats**

chenTTest, 81
kendallSeasonalTrendTest, 688
kendallTrendTest, 703
quantileTest, 1120
serialCorrelationTest, 1141
signTest, 1152
twoSampleLinearRankTest, 1322
varGroupTest, 1360
zTestGevdShape, 1376

∗ **htest**

aovN, 13
aovPower, 16
ciBinomHalfWidth, 89
ciBinomN, 95
ciNormHalfWidth, 103
ciNormN, 107
ciNparConfLevel, 111
ciNparN, 115
ciTableMean, 117
ciTableProp, 124
distChoose, 135
distChooseCensored, 151
ebeta, 176
ebinom, 179
eevd, 196
eexp, 201
egamma, 204
egammaAltCensored, 214
egammaCensored, 223
egeom, 232
egevd, 234
ehyper, 241
elnorm, 244
elnorm3, 246
elnormAlt, 259
elnormAltCensored, 272
elnormCensored, 289
elogis, 298
enbinom, 307
enorm, 310

enormCensored, 315
enpar, 335
enparCensored, 341
epareto, 410
epois, 415
epoisCensored, 419
eqbeta, 427
eqbinom, 429
eqevd, 432
eqexp, 435
eqgamma, 437
eqgeom, 444
eqgevd, 446
eqhyper, 449
eqlnorm, 452
eqlnorm3, 456
eqlnormCensored, 460
eqlogis, 467
eqnbinom, 469
eqnorm, 472
eqnormCensored, 478
eqnpar, 487
eqpareto, 503
eqpois, 505
equnif, 509
eqweibull, 511
eqzmlnorm, 514
eqzmnorm, 517
eunif, 532
eweibull, 542
ezmlnorm, 545
ezmnorm, 550
geom_stripchart, 579
gofGroupTest, 605
gofTest, 615
gofTestCensored, 640
gpqCiNormCensored, 659
gpqTolIntNormCensored, 663
linearTrendTestN, 716
linearTrendTestPower, 718
linearTrendTestScaledMds, 726
lMoment, 729
oneSamplePermutationTest, 761
plotAovDesign, 811
plotCiBinomDesign, 815
plotCiNormDesign, 821
plotCiNparDesign, 826
plotLinearTrendTestDesign, 829
plotPredIntLnormAltSimultaneousTestPowerCurve, 832
plotPredIntLnormAltTestPowerCurve, 837
plotPredIntNormDesign, 841
plotPredIntNormSimultaneousTestPowerCurve, 845
plotPredIntNormTestPowerCurve, 850
plotPredIntNparDesign, 853
plotPredIntNparSimultaneousDesign, 857
plotPredIntNparSimultaneousTestPowerCurve, 860
plotPropTestDesign, 865
plotTolIntNormDesign, 870
plotTolIntNparDesign, 874
plotTTestDesign, 878
plotTTestLnormAltDesign, 882
predIntGamma, 908
predIntGammaSimultaneous, 915
predIntLnorm, 925
predIntLnormAltSimultaneousTestPower, 934
predIntLnormAltTestPower, 938
predIntLnormSimultaneous, 940
predIntNorm, 947
predIntNormHalfWidth, 954
predIntNormK, 957
predIntNormN, 963
predIntNormSimultaneous, 967
predIntNormSimultaneousK, 978
predIntNormSimultaneousTestPower, 987
predIntNormTestPower, 995
predIntNpar, 999
predIntNparConfLevel, 1008
predIntNparN, 1011
predIntNparSimultaneous, 1014
predIntNparSimultaneousConfLevel, 1027
predIntNparSimultaneousN, 1031
predIntNparSimultaneousTestPower, 1036
predIntPois, 1043
propTestMdd, 1072
propTestN, 1078
propTestPower, 1085
pwMoment, 1094

quantileTestPValue, 1125
rosnerTest, 1128
stripChart, 1193
summaryStats, 1211
tolIntGamma, 1224
tolIntLnorm, 1232
tolIntLnormCensored, 1238
tolIntNorm, 1243
tolIntNormCensored, 1248
tolIntNormHalfWidth, 1254
tolIntNormK, 1257
tolIntNormN, 1263
tolIntNpar, 1267
tolIntNparConfLevel, 1273
tolIntNparCoverage, 1276
tolIntNparN, 1278
tolIntPois, 1281
tTestAlpha, 1289
tTestLnormAltN, 1291
tTestLnormAltPower, 1296
tTestLnormAltRatioOfMeans, 1303
tTestN, 1307
tTestPower, 1311
tTestScaledMdd, 1318
twoSampleLinearRankTestCensored, 1331
twoSamplePermutationTestLocation, 1347
twoSamplePermutationTestProportion, 1354
varTest, 1364

∗ **manip**
longToWide, 750

∗ **math**
base, 20

∗ **models**
anovaPE, 10
aovN, 13
aovPower, 16
boxcox, 26
boxcoxCensored, 38
boxcoxTransform, 53
calibrate, 57
chenTTest, 81
detectionLimitCalibrate, 131
distChoose, 135
distChooseCensored, 151
gofGroupTest, 605
gofTest, 615
gofTestCensored, 640
inversePredictCalibrate, 682
oneSamplePermutationTest, 761
plotAovDesign, 811
pointwise, 887
predict, 903
predict.lm, 906
propTestMdd, 1072
propTestN, 1078
propTestPower, 1085
quantileTest, 1120
quantileTestPValue, 1125
rosnerTest, 1128
serialCorrelationTest, 1141
signTest, 1152
twoSamplePermutationTestLocation, 1347
twoSamplePermutationTestProportion, 1354
varGroupTest, 1360
varTest, 1364
zTestGevdShape, 1376

∗ **nonparametric**
kendallSeasonalTrendTest, 688
kendallTrendTest, 703
twoSampleLinearRankTest, 1322
twoSampleLinearRankTestCensored, 1331

∗ **package**
FcnsByCat, 555
FcnsByCatCalibration, 556
FcnsByCatCensoredData, 556
FcnsByCatDataTrans, 559
FcnsByCatEstDistParams, 560
FcnsByCatEstDistQuants, 561
FcnsByCatGOFTests, 562
FcnsByCatHypothTests, 563
FcnsByCatMCandRisk, 564
FcnsByCatPlotProbDists, 564
FcnsByCatPlotUsingggplot2, 565
FcnsByCatPower, 566
FcnsByCatPredInts, 570
FcnsByCatPrintPlot, 571
FcnsByCatProbDists, 572
FcnsByCatSumStats, 573
FcnsByCatTolInts, 574
FcnsByCatTrend, 575

newsEnvStats, 754
∗ **plot**
plot.boxcox, 776
plot.boxcoxCensored, 779
plot.boxcoxLm, 783
plot.gof, 787
plot.gofCensored, 792
plot.gofGroup, 798
plot.gofTwoSample, 802
plot.permutationTest, 808
∗ **print**
print, 1052
print.boxcox, 1054
print.boxcoxCensored, 1055
print.boxcoxLm, 1056
print.distChoose, 1057
print.distChooseCensored, 1058
print.estimate, 1059
print.estimateCensored, 1060
print.gof, 1061
print.gofCensored, 1062
print.gofGroup, 1063
print.gofOutlier, 1064
print.gofTwoSample, 1065
print.htest.htestEnvStats, 1066
print.htestCensored, 1069
print.permutationTest, 1070
print.summaryStats, 1071
∗ **regression**
anovaPE, 10
aovN, 13
aovPower, 16
calibrate, 57
detectionLimitCalibrate, 131
inversePredictCalibrate, 682
kendallSeasonalTrendTest, 688
kendallTrendTest, 703
plotAovDesign, 811
pointwise, 887
predict, 903
predict.lm, 906
twoSampleLinearRankTest, 1322
twoSampleLinearRankTestCensored, 1331
∗ **univar**
boxcox, 26
boxcoxCensored, 38
boxcoxTransform, 53
cv, 128
geoMean, 577
geoSD, 590
iqr, 686
kurtosis, 711
serialCorrelationTest, 1141
skewness, 1173
summaryFull, 1205
summaryStats, 1211
(Alternative), *171–176*
(Delta), *173*, *176*
.Random.seed, *762*, *1348*

abstract for Castillo and Hadi (1994), *237*
abstract for Hosking et al. (1985), *237*, *238*
ACE.13.TCE.df, 10
acf, *1148*
ad.test, *625*
Air.df *(*Environmental*)*, 355
Aldicarb *(*EPA.89b.aldicarb2.df*)*, 391
anova, *11*
anova.lm, *11*, *1214*, *1216*
anovaPE, 10, *59*, *60*, *556*
aov, *15*, *18*, *813*, *1190*, *1191*, *1197*, *1201*
aovN, 13, *18*, *566*, *813*
aovPower, *14*, *15*, 16, *566*, *813*
approx, *305*, *1103*
ar, *1148*
arima, *1144*, *1146*, *1148*
arima.sim, *1148*
as.data.frame, *57*
axis, *1194*, *1200*

barplot, *574*
base, 20
Beal.2010.Pb.df, 22
Benthic.df, 22
Benzene *(*EPA.89b.benzene.df*)*, 391
Beta, *171*, *173*, *178*, *428*, *560*, *561*, *1161*, *1169*
beta, *572*
beta distribution, *176*, *177*, *427*, *982*, *991*, *1018*, *1145*, *1269*
beta function, *982*, *991*, *997*, *1095*
beta random variable, *1269*, *1279*
binom, *572*
binom.test, *91*, *93*, *99*, *125*, *126*, *180*, *182*, *819*, *869*, *1075*, *1080*, *1082*, *1087*,

1088, *1214*, *1216*
Binomial, *79*, *171–173*, *175*, *182*, *432*, *560*, *561*, *770*, *869*, *1044*, *1260*
binomial, *1046*, *1154*, *1269*, *1283*
binomial distribution, *242*, *417*, *429*, *451*, *508*, *1355*, *1356*
binomial random variable, *489*, *948*, *1154*
binomial test, *1154*
BJC.2000.df, 25
boot, *763*, *764*, *1350*, *1351*, *1357*
Box-Cox *(*boxcox*)*, 26
Box-Cox Censored *(*boxcoxCensored*)*, 38
Box-Cox Transformation *(*boxcox*)*, 26
Box-Cox Transformation for Censored Data *(*boxcoxCensored*)*, 38
BoxCox *(*boxcox*)*, 26
boxcox, 26, *35*, *36*, *44*, *51*, *52*, *54*, *56*, *560*, *776*, *778*, *783*, *786*, *1054*, *1056*
BoxCox Censored *(*boxcoxCensored*)*, 38
BoxCox Transformation *(*boxcox*)*, 26
BoxCox Transformation for Censored Data *(*boxcoxCensored*)*, 38
boxcox.default *(*boxcox*)*, 26
boxcox.lm *(*boxcox*)*, 26
boxcox.object, *29*, *31*, 35, *52*, *776*, *778*, *1054*
boxcoxCensored, 38, *47*, *49*, *556*, *779*, *780*, *782*, *1055*
boxcoxCensored.object, *42*, *44*, 47, *780*, *782*, *1055*
boxcoxLm.object, *29*, *31*, *36*, 51, *784–786*, *1056*
boxcoxTransform, *27*, *31*, *40*, 53, *560*
boxplot, *574*, *686*
boxplots, *414*, *1193*

Cadmium *(*EPA.89b.cadmium.df*)*, 392
calibrate, *11*, 57, *61*, *131*, *135*, *556*, *682–684*, *888*, *889*, *907*
calibrate.object, *60*, 60
Calibration, *555*
Calibration *(*FcnsByCatCalibration*)*, 556
Castillo and Hadi (1994), *238*
Castillo and Hadi 1994 *(*CastilloAndHadi1994*)*, 62
CastilloAndHadi1994, 62
Cauchy, *171*, *173*
cauchy, *572*
cdfCompare, 64, *67*, *81*, *189*, *305*, *414*, *565*, *791*, *807*
cdfCompareCensored, 69, *193*, *194*, *558*, *797*
cdfPlot, *67*, *68*, *72*, *74*, 78, *189*, *193*, *194*, *414*, *565*, *772*
ceiling, *1280*
Censored Data, *71*, *166*, *169*, *171*, *555*, *601*, *681*, *797*, *1059–1061*, *1063*, *1070*
Censored Data *(*FcnsByCatCensoredData*)*, 556
Chen's modified t-test, *764*, *1155*
Chen's modified t-test for skewed distributions, *1298*
Chen's t-test modified for skewed data, *1300*
chenTTest, 81, *563*, *675*
Chi, 87, *171*, *173*
chi, *572*
chi random variable, *960*
Chi-square, *171*, *173*
chi-square, *209*, *911*, *918*, *1226*
chi-square distribution, *202*, *203*, *253*, *263*, *416*, *576*
chi-square distributions, *439*
chi-square random variable, *694*
chi-squared, *88*
chi-squared distribution, *208*, *217*, *218*, *226*, *279*, *280*, *324*, *325*, *422*
chisq, *572*
chisq.test, *628*, *633*, *1214*, *1216*
Chisquare, *88*
Chisquare distribution, *801*
Chlordane *(*EPA.89b.chlordane1.df*)*, 392
chol, *1163*
ciBinomHalfWidth, 89, *97–99*, *126*, *182*, *566*, *818*, *819*
ciBinomN, *92*, *93*, 95, *98*, *126*, *182*, *566*, *818*, *819*
ciNormHalfWidth, 103, *108*, *109*, *120*, *566*, *823*, *824*
ciNormN, *105*, 107, *109*, *120*, *566*, *823*, *824*
ciNparConfLevel, 111, *116*, *566*, *828*
ciNparN, *112*, 115, *566*, *828*
ciTableMean, 117, *126*, *566*
ciTableProp, *120*, 124, *566*
class, *11*, *26*, *59*, *61*, *903*, *1052*, *1053*, *1141*, *1193*, *1205*, *1211*
Coefficient of Variation *(*cv*)*, 128
coefficient of variation *(*cv*)*, 128
Compare CDFs *(*cdfCompare*)*, 64

Compare CDFs for Censored Data *(*cdfCompareCensored*)*, 69
cor, *620*, *1166*
cor.test, *700*, *704*, *709, 710*
Cumulative Distribution *(*cdfPlot*)*, 78
CV *(*cv*)*, 128
cv, 128, *573*, *714*, *733*, *1176*, *1207, 1208*
cvm.test, *626*

Data Transformations, *31*, *44*, *56*, *555*, *776*, *778*, *782, 783*, *786*, *1054–1056*
Data Transformations *(*FcnsByCatDataTrans*)*, 559
data.frame, *11*, *751, 752*
dbeta, *171*
dchi *(*Chi*)*, 87
Delta Distribution *(*ZeroModifiedLognormal*)*, 1367
DeltaDist *(*ZeroModifiedLognormal*)*, 1367
DeltaDistAlt *(*ZeroModifiedLognormalAlt*)*, 1370
demp *(*Empirical*)*, 302
density, *302, 303*, *305*, *413, 414*
Detection Limit *(*detectionLimitCalibrate*)*, 131
detection limit *(*detectionLimitCalibrate*)*, 131
detectionLimitCalibrate, *60, 61*, 131, *556*, *684*, *888, 889*, *907*
devAskNewPage, *777*, *780*, *784*, *788*, *794*, *799*, *804*, *1118*
devd *(*EVD*)*, 535
dgamma, *576*
dgammaAlt *(*GammaAlt*)*, 575
dgevd *(*GEVD*)*, 592
diff, *1208*
digamma, *536*
digamma function, *177*, *532*
distChoose, 135, *148, 149*, *155*, *563*, *1057*
distChoose.object, *138*, *140*, 147, *1057, 1058*
distChooseCensored, 150, *165, 166*, *1058, 1059*
distChooseCensored.object, *153*, *155*, 165, *1059*
Distribution.df, *65, 66*, *70, 71*, *79*, *81*, *136*, *148, 149*, *151*, *165, 166*, 168, *523*, *525*, *527*, *530*, *572*, *596, 597*, *599*, *601*, *603, 604*, *607, 608*, *613*, *617–619*, *624, 625*, *641*, *676*, *680*, *770*, *772*, *1099*, *1105*, *1108*, *1116, 1117*, *1160, 1161*, *1169*
dlnorm, *735*, *739*
dlnorm3 *(*Lognormal3*)*, 734
dlnormAlt *(*LognormalAlt*)*, 738
dlnormMix *(*LognormalMix*)*, 741
dlnormMixAlt *(*LognormalMixAlt*)*, 743
dlnormTrunc *(*LognormalTrunc*)*, 745
dlnormTruncAlt *(*LognormalTruncAlt*)*, 748
dnormMix *(*NormalMix*)*, 755
dnormTrunc *(*NormalTrunc*)*, 757
dotchart, *574*
dpareto *(*Pareto*)*, 767
dtri *(*Triangular*)*, 1286
dzmlnorm *(*ZeroModifiedLognormal*)*, 1367
dzmlnormAlt *(*ZeroModifiedLognormalAlt*)*, 1370
dzmnorm *(*ZeroModifiedNormal*)*, 1373

ebeta, *171*, 176, *427, 428*, *560*
ebinom, *91*, *93*, *99*, 179, *429, 430*, *432*, *560*, *819*, *1046*
ecdfPlot, *65*, *67, 68*, *70*, *81*, 186, *192*, *194*, *219*, *227*, *282*, *305*, *327*, *337*, *347*, *414*, *423*, *565*, *789*, *795*, *805*, *893*, *895*, *900*, *1101*, *1105*
ecdfPlotCensored, *70*, *72*, *74*, *189*, 190, *344*, *350*, *558*, *899, 900*, *1113*
eevd, 195, *239*, *433*, *435*, *449*, *532*, *537*, *560*, *1097*, *1378*
eexp, 201, *436, 437*, *560*
egamma, *137*, 204, *221*, *230*, *438, 439*, *441*, *560*, *607*, *617*, *628*, *667*, *909*, *912*, *917*, *921*, *1225*, *1227*
egammaAlt, *438*, *560*, *577*, *921*
egammaAlt *(*egamma*)*, 204
egammaAltCensored, 214, *230*, *557*
egammaCensored, *221*, 223, *557*
egeom, 232, *308*, *444, 445*, *471*, *560*
egevd, *62*, *64*, 234, *446, 447*, *449*, *560*, *594*, *674*, *1097*, *1376–1378*
ehyper, 241, *450, 451*, *560*
elnorm, *86*, 244, *295*, *452*, *454*, *560*, *591*, *633*, *926*, *928*, *941*, *944*, *1232*, *1235*
elnorm3, 246, *457*, *459*, *541*, *560*, *623*, *630*, *633*, *737*

elnormAlt, *30*, *55*, *86*, 259, *275, 276*, *281*, *286*, *560*, *578*, *633*, *740*, *926*, *928*, *941, 942*, *944*, *1232, 1233*, *1235*, *1298*, *1300*
elnormAltCensored, *43*, *152*, 271, *557*, *641*, *648*, *926*, *941*, *1232*
elnormCensored, *276*, *286*, 289, *452*, *464*, *557*, *648*, *926*, *941*, *1232*
elogis, 298, *468, 469*, *560*
Empirical, 302, *414*, *564*, *1161*, *1166*, *1169*, *1171*
empirical cdf plots, *80*
empirical PDF *(*epdfPlot*)*, 412
empirical pdf plots, *772*
enbinom, *233*, 307, *445*, *470, 471*, *560*
enorm, *105*, *109*, *120*, *245*, 310, *321*, *323*, *332*, *339*, *472–474*, *523*, *540*, *560*, *633*, *824*, *947*, *950*, *962*, *967*, *972*, *985*, *1243*, *1246*, *1262*
enormCensored, *42*, *275*, *286*, *292*, *295*, 314, *464*, *472*, *480*, *483*, *527*, *557*, *648*, *659–661*, *663, 664*, *947*, *967*, *1109*, *1238*, *1240*, *1243*, *1249–1251*
enpar, 335, *343*, *346*, *350*, *560*
enparCensored, *339*, 341, *557*
Environmental, 355
EnvStats Functions for Censored Data, *152*, *527*, *530*, *679*, *1069*, *1109*, *1113*
EnvStats Functions for Censored Data *(*FcnsByCatCensoredData*)*, 556
EnvStats Functions for Goodness-of-Fit Tests *(*FcnsByCatGOFTests*)*, 562
EPA.02d.Ex.2.ug.per.L.vec, 357
EPA.02d.Ex.4.mg.per.kg.vec, 357
EPA.02d.Ex.6.mg.per.kg.vec, 358
EPA.02d.Ex.9.mg.per.L.vec, 358
EPA.09.Ex.10.1.nickel.df, 359
EPA.09.Ex.11.1.arsenic.df, 359
EPA.09.Ex.12.1.ccl4.df, 360
EPA.09.Ex.12.4.naphthalene.df, 361
EPA.09.Ex.13.1.iron.df, 361
EPA.09.Ex.14.1.manganese.df, 362
EPA.09.Ex.14.3.alkalinity.df, 363
EPA.09.Ex.14.4.arsenic.df, 363
EPA.09.Ex.14.8.df, 364
EPA.09.Ex.15.1.manganese.df, 365
EPA.09.Ex.16.1.sulfate.df, 366
EPA.09.Ex.16.2.benzene.df, 366
EPA.09.Ex.16.4.copper.df, 367
EPA.09.Ex.16.5.PCE.df, 368
EPA.09.Ex.17.1.loglead.df, 369
EPA.09.Ex.17.2.toluene.df, 369
EPA.09.Ex.17.3.chrysene.df, 370
EPA.09.Ex.17.3.log.chrysene.df, 371
EPA.09.Ex.17.4.copper.df, 372
EPA.09.Ex.17.5.chloride.df, 372
EPA.09.Ex.17.6.sulfate.df, 373
EPA.09.Ex.17.7.sodium.df, 374
EPA.09.Ex.18.1.arsenic.df, 375
EPA.09.Ex.18.2.chrysene.df, 375
EPA.09.Ex.18.3.TCE.df, 376
EPA.09.Ex.18.4.xylene.df, 377
EPA.09.Ex.19.1.sulfate.df, 378
EPA.09.Ex.19.2.chloride.df, 379
EPA.09.Ex.19.5.mercury.df, 379
EPA.09.Ex.20.1.nickel.df, 380
EPA.09.Ex.21.1.aldicarb.df, 381
EPA.09.Ex.21.2.benzene.df, 382
EPA.09.Ex.21.5.beryllium.df, 382
EPA.09.Ex.21.6.nitrate.df, 383
EPA.09.Ex.21.7.TCE.df, 384
EPA.09.Ex.22.1.VC.df, 385
EPA.09.Ex.22.2.Specific.Conductance.df, 385
EPA.09.Ex.6.3.sulfate.df, 386
EPA.09.Ex.7.1.arsenic.df, 387
EPA.09.Table.9.1.TCE.df, 388
EPA.09.Table.9.3.df, 388
EPA.09.Table.9.4.nickel.vec, 389
EPA.89b.aldicarb1.df, 390
EPA.89b.aldicarb2.df, 391
EPA.89b.benzene.df, 391
EPA.89b.cadmium.df, 392
EPA.89b.chlordane1.df, 392
EPA.89b.chlordane2.df, 393
EPA.89b.edb.df, 394
EPA.89b.lead.df, 394
EPA.89b.loglead.df, 395
EPA.89b.manganese.df, 395
EPA.89b.sulfate.df, 396
EPA.89b.t29.df, 397
EPA.89b.toc.vec, 397
EPA.92c.arsenic1.df, 398
EPA.92c.arsenic2.df, 398
EPA.92c.arsenic3.df, 399
EPA.92c.benzene1.df, 400

EPA.92c.benzene2.df, 400
EPA.92c.ccl4.df, 401
EPA.92c.chrysene.df, 402
EPA.92c.copper1.df, 402
EPA.92c.copper2.df, 403
EPA.92c.lognickel1.df, 404
EPA.92c.nickel1.df, 404
EPA.92c.nickel2.df, 405
EPA.92c.toluene.df, 405
EPA.92c.zinc.df, 406
EPA.92d.chromium.df, 407
EPA.92d.chromium.vec, 407
EPA.94b.lead.df, 408
EPA.94b.tccb.df, 408, *753*, *754*
EPA.97.cadmium.111.df, 409
epareto, 410, *503*, *504*, *560*, *769*
epdfPlot, *305*, 412, *565*, *772*
epois, 415, *422*, *426*, *506–508*, *560*, *1043*, *1049*, *1281*, *1283*, *1284*
epoisCensored, 419, *557*, *1043*, *1281*
eqbeta, 427, *561*
eqbinom, 429, *561*
eqevd, 432, *561*
eqexp, 435, *561*
eqgamma, *210*, 437, *561*
eqgammaAlt, *561*
eqgammaAlt *(*eqgamma*)*, 437
eqgeom, 444, *561*
eqgevd, 446, *561*
eqhyper, 449, *561*
eqlnorm, *265*, 452, *561*, *1235*
eqlnorm3, 456, *561*
eqlnormCensored, 460, *557*
eqlogis, 467, *561*
eqnbinom, 469, *561*
eqnorm, *439*, *441*, *453*, *454*, 472, *480*, *523*, *561*, *947*, *950*, *962*, *967*, *1246*, *1262*
eqnormCensored, *461*, *464*, 478, *527*, *557*, *664*, *1240*, *1251*
eqnpar, *112*, *115*, *116*, *259*, *263*, *305*, *339*, 487, *558*, *828*, *1153*, *1155*, *1156*, *1271*
eqpareto, 503, *561*, *769*
eqpois, 505, *561*, *1284*
equnif, 509, *561*
eqweibull, 511, *562*
eqzmlnorm, 514, *562*
eqzmlnormAlt, *562*
eqzmlnormAlt *(*eqzmlnorm*)*, 514
eqzmnorm, 517, *562*
errorBar, 519
estimate *(*estimate.object*)*, 522
estimate.object, *177*, *181*, *199*, *203*, *209*, *210*, *233*, *238*, *242*, *245*, *254*, *265*, *300*, *308*, *312*, *339*, *411*, *417*, *428*, *430*, *432*, *433*, *435–437*, *439*, *441*, *445*, *447*, *449–451*, *453*, *454*, *457*, *468–471*, *473*, *474*, *493*, *494*, *504*, *507*, *508*, *510–513*, *515*, *518*, 522, *530*, *533*, *534*, *543*, *544*, *548*, *552*, *553*, *910*, *912*, *918*, *921*, *926*, *928*, *942*, *944*, *949*, *950*, *962*, *970*, *972*, *985*, *1001*, *1002*, *1019*, *1020*, *1048*, *1049*, *1059*, *1060*, *1225*, *1227*, *1233*, *1235*, *1245*, *1246*, *1262*, *1270*, *1271*, *1283*, *1284*
estimateCensored *(*estimateCensored.object*)*, 527
estimateCensored.object, *220*, *221*, *229*, *230*, *283*, *286*, *292*, *295*, *329*, *332*, *349*, *350*, *425*, *426*, *461*, *464*, *481*, *483*, *525*, 527, *661*, *664*, *1060*, *1061*, *1239*, *1240*, *1250*, *1251*
Estimating Distribution Parameters, *66*, *105*, *109*, *137*, *169*, *171*, *523*, *525*, *555*, *607*, *617*, *824*, *1049*, *1059*, *1060*, *1099*, *1117*, *1235*, *1246*, *1262*, *1271*, *1284*
Estimating Distribution Parameters *(*FcnsByCatEstDistParams*)*, 560
Estimating Distribution Quantiles, *169*, *494*, *523*, *525*, *555*, *557*, *558*, *1059*, *1060*, *1235*, *1246*, *1262*, *1271*, *1284*
Estimating Distribution Quantiles *(*FcnsByCatEstDistQuants*)*, 561
Euler's Constant, *200*
Euler's constant, *197–199*, *536*
Eulers Constant *(*EulersConstant*)*, 531
EulersConstant, 531
eunif, *510*, *511*, 532, *560*
EVD, *513*, 535, *544*, *594*, *1378*
evd, *572*
evNormOrdStats, *250*, 538, *573*
evNormOrdStatsScalar, *573*, *1037*, *1040*
evNormOrdStatsScalar *(*evNormOrdStats*)*, 538

eweibull, *512, 513*, 542, *561*
exp, *572*
Exponential, *171*, *173*, *203*, *437*, *513*, *544*, *560, 561*, *769*
exponential, *209*, *439*, *536*, *911*, *918*, *1226*, *1325*
exponential distribution, *199*, *201–203*, *411*, *434–436*, *504*, *512*, *536*, *544*, *576*, *769*
exponential random variable, *199*, *434*
Extreme, *171*, *174*
Extreme Value, *560, 561*
Extreme Value Distribution, *200*, *239*, *435*, *449*, *532*
Extreme Value Distribution *(*EVD*)*, 535
extreme value distribution, *196, 197*, *203*, *432*, *436*, *593*
extreme value distributions, *238*, *447*
extreme value distributions (EVD), *593*
ezmlnorm, *514–516*, *518*, 545, *553*, *561*, *633*, *1369*
ezmlnormAlt, *514–516*, *561*, *633*, *1372*
ezmlnormAlt *(*ezmlnorm*)*, 545
ezmnorm, *517*, 550, *561*, *633*, *1375*

F, *171*, *174*
f, *572*
F-distribution, *17*, *1046*
FcnsByCat, 555
FcnsByCatCalibration, 556
FcnsByCatCensoredData, 556, *899*
FcnsByCatDataTrans, 559
FcnsByCatEstDistParams, 560
FcnsByCatEstDistQuants, 561
FcnsByCatGOFTests, 562
FcnsByCatHypothTests, 563
FcnsByCatMCandRisk, 564
FcnsByCatPlotProbDists, 564
FcnsByCatPlotUsingggplot2, 565
FcnsByCatPower, 566
FcnsByCatPredInts, 570
FcnsByCatPrintPlot, 571
FcnsByCatProbDists, 572
FcnsByCatSumStats, 573
FcnsByCatTolInts, 574
FcnsByCatTrend, 575
Fisher's exact test, *1356*
Fisher's randomization test, *85*, *1155*
fisher.test, *1214*, *1216*, *1354*
Fishers's exact test, *242*, *451*
floor, *242*
format, *581*, *762*, *1054–1059*, *1061–1066*, *1069–1071*, *1177*, *1181*, *1190*, *1196*
formula, *57*
Functions By Category *(*FcnsByCat*)*, 555
Functions for Censored Data *(*FcnsByCatCensoredData*)*, 556

Gamma, *136–138*, *151*, *153*, *171*, *174*, *204*, *560, 561*, *641*, *644*, *646*
gamma, *54*, *572*
Gamma Distribution *(*GammaAlt*)*, 575
gamma distribution, *202, 203*, *205*, *214*, *216*, *223, 224*, *436*, *438*, *911*, *915*, *918*, *1224*
gamma distribution (alternative parameterization), *915*
Gamma function, *208*
gamma function, *543*, *672*
GammaAlt, 575, *912*, *921*
GammaDist, *210*, *221*, *230*, *441*, *577*, *912*, *921*, *1227*
gammAlt, *572*
Generalized, *171*, *174*
Generalized Extreme Value, *560, 561*
Generalized Extreme Value Distribution, *64*, *239*, *449*, *674*
Generalized Extreme Value Distribution *(*GEVD*)*, 592
generalized extreme value distribution, *62, 63*, *235, 236*, *238*, *446, 447*, *670*, *1096*, *1376, 1377*
geom, *572*
geom_errorbar, *580*
geom_jitter, *580*, *582*, *584*
geom_label, *582–584*, *1177, 1178*, *1181, 1182*, *1185, 1186*, *1188–1191*
geom_line, *580*, *582*, *584*
geom_point, *580*, *582*, *584*, *1176*, *1180*, *1184*, *1188*
geom_stripchart, *565*, *574*, 579, *1178*, *1182*, *1185, 1186*, *1191*
geom_text, *581*, *583, 584*, *1177, 1178*, *1181, 1182*, *1185, 1186*, *1188–1191*
geoMean, *573*, 577, *591*, *1207*
Geometric, *172*, *174*, *233*, *308*, *445*, *471*, *560, 561*

geometric distribution, *232*, *233*, *308*, *444*, *445*, *470*
Geometric Mean (geoMean), 577
geometric mean (geoMean), 577
Geometric SD (geoSD), 590
geometric SD (geoSD), 590
Geometric Standard Deviation (geoSD), 590
geometric standard deviation (geoSD), 590
geoSD, *574*, *579*, 590, *1206*, *1208*
GEVD, *236*, *537*, 592, *1378*
gevd, *572*
ggplot, *574*
ggplot2, *565*, *580*, *1176*, *1180*, *1184*, *1188*
Gibbons.et.al.09.Alkilinity.vec, 594
Gibbons.et.al.09.Vinyl.Chloride.vec, 595
GOF (FcnsByCatGOFTests), 562
gof, *148*
gof (gof.object), 595
gof.object, 595, *630*, *633*, *788*, *791*, *803*, *1061*, *1062*
gofCensored, *166*
gofCensored (gofCensored.object), 599
gofCensored.object, *597*, 599, *646*, *648*, *793*, *796*, *797*, *1062*, *1063*
gofGroup (gofGroup.object), 602
gofGroup.object, 602, *609*, *610*, *799*, *801*, *1063*, *1064*
gofGroupTest, *562*, *602–604*, 605, *631*, *798*, *801*, *1063*
gofOutlier (gofOutlier.object), 612
gofOutlier.object, 612, *1064*, *1065*, *1137*, *1139*
gofTest, *28*, *42*, *67*, *137*, *138*, *140*, *153*, *189*, *251*, *254*, *540*, *541*, *562*, *595*, *597*, *609*, *610*, 615, *643–645*, *648*, *657*, *787*, *791*, *802*, *804*, *805*, *807*, *1065*, *1092*, *1093*, *1138*, *1139*, *1175*, *1377*
gofTestCensored, *41*, *152*, *153*, *155*, *558*, *599*, *601*, 640, *792*, *797*, *1062*
gofTwoSample (gofTwoSample.object), 657
gofTwoSample.object, *630*, 657, *806*, *807*, *1065*
Goodness-of-Fit Tests, *31*, *44*, *56*, *149*, *166*, *555*, *563*, *597*, *601*, *604*, *614*, *658*, *787*, *791*, *798*, *801*, *803*, *807*, *1057*, *1061*, *1062*, *1064*, *1065*, *1378*
Goodness-of-Fit Tests (FcnsByCatGOFTests), 562
goodness-of-fit tests, *675*
gpqCiNormCensored, 659
gpqCiNormMultiplyCensored, *329*, *557*
gpqCiNormMultiplyCensored (gpqCiNormCensored), 659
gpqCiNormSinglyCensored, *328*, *557*
gpqCiNormSinglyCensored (gpqCiNormCensored), 659
gpqTolIntNormCensored, 663
gpqTolIntNormMultiplyCensored, *559*, *1239*, *1249*
gpqTolIntNormMultiplyCensored (gpqTolIntNormCensored), 663
gpqTolIntNormSinglyCensored, *559*, *1239*, *1240*, *1249*, *1251*
gpqTolIntNormSinglyCensored (gpqTolIntNormCensored), 663
Graham.et.al.75.etu.df, 666
Grice.Bain.80.mat, *209*, 667
Gumbel Distribution (EVD), 535

Helsel.Cohn.88.app.b.df, 668
Helsel.Cohn.88.silver.df, 669
Helsel.Hirsch.02.Mayfly.df, 669
hist, *574*, *791*, *796*
histograms, *414*
Hosking et al 1985 (HoskingEtAl1985), 670
Hosking et al., 1985), *62*, *64*
HoskingEtAl1985, 670
htest, *1066*
htest.htestEnvStats.object, 675
htest.object, *773*, *1066*, *1067*, *1127*, *1365*
htest.object (htest.htestEnvStats.object), 675
htestCensored.object, 679, *1069*, *1070*, *1336*, *1339*
htestEnvStats, *1066*
htestEnvStats.object, *84*, *698*, *700*, *708*, *710*, *1066*, *1067*, *1123*, *1124*, *1146*, *1148*, *1155*, *1156*, *1327*, *1328*, *1362*, *1377*, *1378*
htestEnvStats.object (htest.htestEnvStats.object), 675

hyper, *572*
Hypergeometric, *172*, *174*, *243*, *451*, *560, 561*
hypergeometric distribution, *241, 242*, *449–451*, *1356*
Hypothesis Tests, *555*, *575*, *675*, *677*, *764*, *774*, *809*, *1067*, *1071*, *1124*, *1127*, *1148*, *1156*, *1290*, *1293*, *1300*, *1304*, *1308*, *1317*, *1320*, *1351*, *1357*
Hypothesis Tests *(*FcnsByCatHypothTests*)*, 563

integrate, *834*, *846*, *858*, *862*, *935*, *979*, *989*, *1015*, *1028*, *1033*, *1037*, *1258*
Interquartile Range *(*iqr*)*, 686
inversePredictCalibrate, *59–61*, *135*, *556*, 682, *888, 889*, *907*
invisible, *1052, 1053*
iqr, *574*, 686, *1182*, *1208*

Kendall's nonparametric test for trend, *724*
kendallSeasonalTrendTest, *563*, *675*, 688, *689*, *708*, *710*
kendallTrendTest, *563*, *675*, *689–693*, *700*, 703, *717*, *724*, *728*
kruskal.test, *1190, 1191*, *1197*, *1201*, *1214*, *1216*
ks.test, *608*, *618*, *628*, *633*
Kurtosis *(*kurtosis*)*, 711
kurtosis, *130*, *574*, 711, *733*, *1176*, *1207*

L-moments, *1096*
lag.plot, *1148*
layer, *580*, *1178*, *1182*, *1185*, *1190*
lillie.test, *626*
Lin.Evans.80.df, 715
linearTrendTestN, *567*, 716, *724*, *728*, *831*
linearTrendTestPower, *567*, *717*, 718, *727, 728*, *831*
linearTrendTestScaledMds, *567*, *717*, *724*, 726, *831*
lines, *66*, *72*, *80*, *187*, *192*, *413*
lm, *11*, *26*, *58–61*, *684*, *717*, *724*, *728*, *889*, *907*, *1142*
lmle, mme,, *174*
lMoment, *130*, *574*, *713*, 729, *1097*, *1175*
lmsreg, *64*, *237*
lnorm, *573*
lnorm3, *573*
lnormAlt, *573*
lnormMix, *573*
lnormMixAlt, *573*
lnormTrunc, *573*
lnormTruncAlt, *573*
logis, *572*
Logistic, *172*, *174*, *301*, *469*, *560, 561*
logistic distribution, *299*, *301*, *411*, *467, 468*, *504*, *769*
Lognormal, *79*, *136–138*, *151–153*, *172–176*, *246*, *255*, *267*, *295*, *454*, *459*, *464*, *516*, *549*, *557*, *560, 561*, *591*, *607*, *617*, *619*, *624–627*, *633*, *641*, *648*, *737*, *740*, *742*, *744*, *747*, *770*, *928*, *944*, *1161*, *1169*, *1235*, *1369*
lognormal, *789*
lognormal distribution, *54*, *244, 245*, *259, 260*, *272*, *274*, *289*, *292*, *452*, *460*, *576*, *591*, *622*, *624*, *640*, *735*, *739*, *744*, *834*, *925*, *934*, *936*, *939, 940*, *1232*, *1297*, *1299, 1300*, *1316*
lognormal distribution (alternative parameterization), *640*, *925*, *940*, *1232*
Lognormal distribution, alternative parameterization, *641*
lognormal random variable, *741*, *744*, *1367*, *1371*
Lognormal3, *255*, *459*, *633*, 734
LognormalAlt, *246*, *255*, *260*, *262*, *267*, *274*, *286*, *459*, 738, *744*, *749, 750*, *835*, *839*, *937*, *939*, *944*, *1235*, *1293*, *1298*, *1300*, *1304*, *1369*, *1372*
LognormalMix, 741, *744*, *757*
LognormalMixAlt, 743
LognormalTrunc, 745
LognormalTruncAlt, 748
longToWide, 750
lse, mle, *175*
ltsreg, *237*

mad, *1208*
matrix, *752*
max, *1208*
mean, *339*, *573*, *579*, *1178*, *1207*
median, *237*, *573*, *579*, *1182*, *1207*
methods, *26*, *903*, *1052, 1053*, *1141*, *1193*, *1205*, *1211*
Millard.Deverel.88.df, 752

min, *1208*
Mixture, *172*, *174*, *175*
mle, bcmle, mme, mmue, *174*
mle, mme, mmue, *173–175*
mle, mme, mmue,, *174*
mle, mme, mmue, pwme, *174*
mle, mvue, *174*
mle, pwme, tsoe, *174*
mle/mme, *173*
mle/mme, mvue, *174*, *175*
mle/mme/mvue, *173*, *175*
mmue, mmme,, *174*
model.matrix, *58*
Modified.TcCB.df, 753
Monte Carlo Simulation and Risk Assessment, *555*
Monte Carlo Simulation and Risk Assessment *(*FcnsByCatMCandRisk*)*, 564
mvue, *176*
mvue, qmle, *174*

na.exclude, *58*
na.fail, *58*
na.omit, *58*
na.pass, *136*, *607*, *616*, *689*, *704*, *1206*, *1213*, *1361*
nbinom, *573*
Negative, *172*, *175*
Negative Binomial, *560*, *561*
Negative Binomial distribution, *1018*
negative binomial distribution, *233*, *307*, *308*, *445*, *469*, *470*
negative binomial distributions, *307*
NegBinomial, *233*, *308*, *445*, *471*
news, *754*
newsEnvStats, 754
NIOSH.89.air.lead.vec, 755
nlminb, *28*, *40*, *236*, *249*, *982*
non-central F random variable, *17*, *722*, *1313*
non-central F-distribution, *722*, *1313*
non-central Student's t-distribution, *982*, *983*, *991*, *997*
non-central Student's t-random variable, *721*, *1313*
Nonparametric Prediction Interval *(*predIntNpar*)*, 999
Nonparametric Simultaneous Prediction Interval *(*predIntNparSimultaneous*)*, 1014
norm, *573*
Normal, *15*, *18*, *88*, *105*, *109*, *136–138*, *151–153*, *170*, *172*, *173*, *175*, *176*, *246*, *255*, *267*, *313*, *332*, *459*, *474*, *483*, *518*, *541*, *553*, *557*, *560*, *561*, *607*, *617*, *619*, *624–627*, *633*, *641*, *648*, *717*, *724*, *728*, *757*, *759*, *813*, *824*, *843*, *847*, *852*, *873*, *950*, *962*, *972*, *985*, *992*, *998*, *1139*, *1246*, *1255*, *1262*, *1265*, *1290*, *1308*, *1317*, *1320*, *1375*
normal, *88*, *789*, *860*, *1036*
normal (Gaussian), *1118*
normal (Gaussian) distribution, *118*, *245*, *292*, *310*, *315*, *619*, *642*
normal (Gaussian) random variable, *1374*
normal distribuiton, *263*
Normal distribution, *659*
normal distribution, *263*, *318*, *472*, *478*, *538*, *539*, *640*, *935*, *947*, *948*, *957–959*, *967–969*, *978*, *980*, *982*, *989*, *991*, *996*, *997*, *1145*, *1243*, *1300*, *1312*, *1315*, *1316*
normal probability plot, *625*
normal random variable, *756*
Normal(0,1) distribution, *801*
NormalMix, *742*, *744*, 755
NormalTrunc, 757
normMix, *573*
normTrunc, *573*

Olympic.NH4.df, 760
one-sample permutation test, *1350*
oneSamplePermutationTest, *21*, *563*, 761, *773*, *774*, *808*, *809*, *1070*, *1071*, *1351*, *1357*
options, *58*, *812*, *818*, *823*, *827*, *831*, *834*, *839*, *843*, *847*, *851*, *855*, *859*, *862*, *868*, *872*, *876*, *880*, *885*
Outlier Test *(*rosnerTest*)*, 1128
outlier test *(*rosnerTest*)*, 1128
Ozone.NE.df, 766

par, *66*, *71*, *72*, *79*, *80*, *186*, *187*, *191*, *192*, *413*, *521*, *771*, *777*, *778*, *780–782*, *784*, *785*, *789–791*, *794–796*, *800*,

804–806, *808*, *809*, *812*, *818*, *819*, *823*, *827*, *831*, *834*, *839*, *842*, *843*, *846*, *847*, *851*, *855*, *859*, *862*, *868*, *872*, *876*, *880*, *884*, *885*, *1100*, *1110*, *1118*, *1195*, *1200*
Parameter, *172*, *174*
Pareto, *172*, *175*, *410*, *412*, *504*, *560*, *561*, 767
pareto, *573*
Pareto distribution, *410*, *503*
pbeta, *171*
pchi *(*Chi*)*, 87
pdfPlot, *81*, *414*, *565*, 770, *791*, *796*
pemp *(*Empirical*)*, 302
permutationTest.object, *762–764*, 773, *808*, *809*, *1070*, *1071*, *1350*, *1351*, *1356*, *1357*
pevd *(*EVD*)*, 535
pgamma, *576*
pgammaAlt *(*GammaAlt*)*, 575
pgevd *(*GEVD*)*, 592
plnorm, *735*, *739*
plnorm3 *(*Lognormal3*)*, 734
plnormAlt *(*LognormalAlt*)*, 738
plnormMix *(*LognormalMix*)*, 741
plnormMixAlt *(*LognormalMixAlt*)*, 743
plnormTrunc *(*LognormalTrunc*)*, 745
plnormTruncAlt *(*LognormalTruncAlt*)*, 748
plot, *36*, *49*, *52*, *521*, *597*, *600*, *603*, *658*, *774*, *776*, *778*, *779*, *782*, *783*, *785–787*, *791*, *792*, *796–798*, *801*, *803*, *806–809*, *1165*, *1194*
Plot CDF *(*cdfPlot*)*, 78
Plot Cumulative Distribution *(*cdfPlot*)*, 78
Plot PDF *(*pdfPlot*)*, 770
Plot Probability Density *(*pdfPlot*)*, 770
Plot Probability Distributions *(*FcnsByCatPlotProbDists*)*, 564
Plot Using ggplot2 *(*FcnsByCatPlotUsingggplot2*)*, 565
plot.boxcox, *31*, *36*, *560*, *572*, 776, *1054*
plot.boxcoxCensored, *44*, *48*, *49*, *556*, *572*, 779, *1055*
plot.boxcoxLm, *31*, *52*, *560*, *572*, 783, *1056*
plot.default, *771*, *1195–1197*, *1199*
plot.gof, *562*, *572*, *597*, *633*, 787
plot.gofCensored, *558*, *572*, *601*, *648*, 792
plot.gofGroup, *562*, *572*, *604*, *610*, 798
plot.gofTwoSample, *562*, *572*, *658*, 802
plot.permutationTest, *572*, *774*, 808, *1351*, *1357*
plot.ts, *1148*
plot.window, *1195*
plotAovDesign, *14*, *15*, *18*, *567*, 811
plotCiBinomDesign, *92*, *93*, *98*, *99*, *126*, *182*, *566*, 815, *818*
plotCiNormDesign, *105*, *109*, *120*, *566*, 821
plotCiNparDesign, *112*, *116*, *566*, 826
plotLinearTrendTestDesign, *567*, *717*, *724*, *728*, 829
plotPredIntLnormAltSimultaneousTestPowerCurve, *568*, 832, *937*
plotPredIntLnormAltTestPowerCurve, *568*, 837, *939*
plotPredIntNormDesign, *568*, 841, *955*, *965*
plotPredIntNormSimultaneousTestPowerCurve, *568*, 845, *937*, *992*
plotPredIntNormTestPowerCurve, *568*, *839*, 850, *939*, *998*
plotPredIntNparDesign, *569*, 853, *1002*, *1009*, *1012*
plotPredIntNparSimultaneousDesign, *569*, 857, *863*, *1020*, *1029*, *1033*, *1040*
plotPredIntNparSimultaneousTestPowerCurve, *569*, 860, *1040*
plotPropTestDesign, *567*, 865, *1075*, *1081*, *1082*, *1088*
Plotting Probability Distributions, *555*, *574*
Plotting Probability Distributions *(*FcnsByCatPlotProbDists*)*, 564
Plotting Using ggplot2, *555*, *574*
Plotting Using ggplot2 *(*FcnsByCatPlotUsingggplot2*)*, 565
plotTolIntNormDesign, *569*, 870, *873*, *1255*, *1264*, *1265*
plotTolIntNparDesign, *570*, 874, *1274*, *1275*, *1277*, *1280*
plotTTestDesign, *567*, 878, *1290*, *1308*, *1316*, *1317*, *1320*
plotTTestLnormAltDesign, *567*, 882, *1293*, *1300*, *1304*
pnormMix *(*NormalMix*)*, 755
pnormTrunc *(*NormalTrunc*)*, 757

points, *72*, *192*, *1110*, *1194*
pointwise, *135*, *521*, *556*, *684*, 887, *906*
pois, *573*
Poisson, *173*, *175*, *418*, *426*, *508*, *560*, *561*, *1049*, *1284*
Poisson distribution, *54*, *415–417*, *419*, *420*, *505*, *508*, *1044*, *1045*, *1281–1283*
Poisson random variable, *506*, *1282*
Power and Sample Size, *570*, *574*, *717*, *724*, *728*
Power and Sample Size *(*FcnsByCatPower*)*, 566
Power and Sample Size Calculations, *555*, *563*
Power and Sample Size Calculations *(*FcnsByCatPower*)*, 566
Power t-test *(*tTestPower*)*, 1311
power t-test *(*tTestPower*)*, 1311
ppareto *(*Pareto*)*, 767
ppoints, *189*, *194*, *541*, *900*, *1105*
ppointsCensored, *73*, *152*, *192–194*, *274*, *277*, *291*, *317*, *320*, *321*, *350*, *558*, *641*, *642*, *645*, *795*, 891, *1111–1113*
predict, *887*, *889*, *903*, 903, *904*, *907*
predict.lm, *556*, *889*, *903*, *904*, *906*, 906, *907*
Prediction Intervals, *169*, *523*, *525*, *555*, *559*, *835*, *839*, *847*, *852*, *937*, *939*, *992*, *998*, *1049*, *1059*, *1060*, *1262*
Prediction Intervals *(*FcnsByCatPredInts*)*, 570
predIntGamma, *210*, *570*, 908, *1227*
predIntGammaAlt, *570*
predIntGammaAlt *(*predIntGamma*)*, 908
predIntGammaAltSimultaneous, *570*
predIntGammaAltSimultaneous *(*predIntGammaSimultaneous*)*, 915
predIntGammaSimultaneous, *570*, 915
predIntLnorm, *570*, 925, *950*, *962*, *1235*
predIntLnormAlt, *570*, *835*, *839*, *939*
predIntLnormAlt *(*predIntLnorm*)*, 925
predIntLnormAltSimultaneous, *570*, *835*, *839*, *928*, *935–937*, *939*
predIntLnormAltSimultaneous *(*predIntLnormSimultaneous*)*, 940
predIntLnormAltSimultaneousTestPower, *568*, *834*, *835*, *839*, 934, *939*, *944*
predIntLnormAltTestPower, *568*, *835*, *839*, *938*
predIntLnormSimultaneous, *570*, *928*, 940, *972*, *985*
predIntNorm, *88*, *523*, *570*, *839*, *843*, *847*, *852*, *910*, *912*, *921*, *926*, *928*, *935*, *939*, *944*, 947, *954*, *955*, *957*, *959*, *962*, *964*, *965*, *968*, *970*, *972*, *980*, *981*, *985*, *989*, *991*, *992*, *996*, *998*, *1045*, *1047*, *1246*, *1262*
predIntNormHalfWidth, *567*, *843*, 953, *965*
predIntNormK, *567*, *570*, *839*, *843*, *847*, *852*, *926*, *928*, *939*, *947*, *948*, *950*, *954*, *955*, 957, *964*, *965*, *969*, *981*, *985*, *992*, *997*, *998*, *1260*
predIntNormN, *568*, *843*, *955*, 963
predIntNormSimultaneous, *570*, *834*, *835*, *847*, *852*, *917*, *921*, *937*, *942*, *944*, *950*, *962*, 967, *978*, *980*, *985*, *989*, *992*, *998*
predIntNormSimultaneousK, *570*, *847*, *852*, *969*, *972*, 978, *990*, *992*, *997*, *998*
predIntNormSimultaneousTestPower, *568*, *847*, *852*, *921*, *936*, *944*, *972*, *982*, *985*, 987, *998*, *1037*, *1039*
predIntNormTestPower, *568*, *839*, *847*, *851*, *852*, *939*, *992*, 995
predIntNpar, *559*, *570*, *855*, *856*, *859*, *863*, 999, *1009*, *1012*, *1016*, *1017*, *1020*, *1029*, *1033*, *1038*, *1040*, *1269*
predIntNparConfLevel, *569*, *855*, *856*, *1002*, 1008, *1012*, *1028*
predIntNparN, *569*, *855*, *856*, *1002*, *1009*, 1011, *1033*
predIntNparSimultaneous, *559*, *571*, *859*, *863*, 1014, *1028*, *1029*, *1033*, *1038*, *1040*
predIntNparSimultaneousConfLevel, *569*, *859*, *863*, *1020*, 1027, *1033*, *1040*
predIntNparSimultaneousN, *569*, *859*, *863*, *1020*, *1029*, 1031, *1040*
predIntNparSimultaneousTestPower, *541*, *569*, *859*, *862*, *863*, *1020*, *1029*, *1033*, 1036
predIntPois, *571*, 1043, *1283*
print, *36*, *49*, *52*, *149*, *166*, *525*, *529*, *597*, *600*, *603*, *613*, *658*, *676*, *680*, *774*, 1052, *1053–1067*, *1069–1072*, *1223*
print.boxcox, *31*, *36*, *560*, *571*, *778*, 1054

print.boxcoxCensored, *44*, *49*, *556*, *571*, *782*, 1055
print.boxcoxLm, *31*, *52*, *560*, *571*, *786*, 1056
print.default, *1052*, *1053*
print.distChoose, *140*, *149*, *563*, 1057
print.distChooseCensored, *155*, *166*, 1058
print.estimate, *571*, 1059
print.estimateCensored, *557*, *571*, 1060
print.gof, *562*, *571*, *597*, *633*, *791*, 1061
print.gofCensored, *558*, *571*, *601*, *648*, *797*, 1062
print.gofGroup, *562*, *571*, *604*, *610*, *801*, 1063
print.gofOutlier, *562*, *614*, 1064, *1139*
print.gofTwoSample, *562*, *571*, *658*, *807*, 1065
print.htest, *571*, *677*, *1052*, *1053*, *1066*
print.htest (print.htest.htestEnvStats), 1066
print.htest.htestEnvStats, 1066
print.htestCensored, *558*, *571*, *681*, 1069
print.htestEnvStats, *677*
print.htestEnvStats (print.htest.htestEnvStats), 1066
print.permutationTest, *571*, *774*, *809*, 1070
print.summaryStats, *571*, 1071, *1207*, *1208*, *1213*, *1215*
Printing and Plotting Methods, *555*
Printing and Plotting Methods (FcnsByCatPrintPlot), 571
Probability Density (pdfPlot), 770
Probability Distributions (FcnsByCatProbDists), 572
Probability Distributions and Random Numbers, *88*, *537*, *555*, *577*, *594*, *737*, *740*, *742*, *744*, *747*, *750*, *757*, *759*, *769*, *1166*, *1171*, *1288*, *1369*, *1372*, *1375*
Probability Distributions and Random Numbers (FcnsByCatProbDists), 572
probability plots, *304*
probability-weighted moment, *671*
prop.test, *91*, *93*, *99*, *125*, *126*, *180*, *182*, *819*, *869*, *1075*, *1080*, *1082*, *1087*, *1088*, *1214*, *1356*
propTestMdd, *567*, *868*, *869*, 1072, *1081*, *1082*, *1088*
propTestN, *567*, *868*, *869*, *1075*, 1078, *1088*
propTestPower, *567*, *868*, *869*, *1074*, *1075*, *1081*, *1082*, 1085
ProUCL.5.2.TRS.df, 1091
ProUCL.Crit.Vals.for.AD.Test.for.Gamma.array, *628*, 1092
ProUCL.Crit.Vals.for.KS.Test.for.Gamma.array, *628*, 1093
ptri (Triangular), 1286
pwMoment, *196*, *198*, *235*, *574*, *731*, *733*, 1094
pzmlnorm (ZeroModifiedLognormal), 1367
pzmlnormAlt (ZeroModifiedLognormalAlt), 1370
pzmnorm (ZeroModifiedNormal), 1373

qbeta, *171*, *428*
qbinom, *430*
qchi (Chi), 87
qemp, *1161*, *1169*
qemp (Empirical), 302
qevd, *433*
qevd (EVD), 535
qexp, *436*
qgamma, *439*, *576*
qgammaAlt (GammaAlt), 575
qgeom, *444*
qgevd, *447*
qgevd (GEVD), 592
qhyper, *450*
qlnorm, *735*, *739*
qlnorm3, *457*
qlnorm3 (Lognormal3), 734
qlnormAlt (LognormalAlt), 738
qlnormMix (LognormalMix), 741
qlnormMixAlt (LognormalMixAlt), 743
qlnormTrunc (LognormalTrunc), 745
qlnormTruncAlt (LognormalTruncAlt), 748
qlogis, *468*
qnbinom, *470*
qnorm, *473*, *480*
qnormMix (NormalMix), 755
qnormTrunc (NormalTrunc), 757
qpareto, *504*
qpareto (Pareto), 767
qpois, *507*
qqnorm, *1098*, *1105*

qqPlot, *65*, *67, 68*, *70*, *138*, *188, 189*, *193, 194*, *320*, *414*, *540, 541*, *565*, *610*, *625*, *631*, *633*, *777, 778*, *780*, *784*, *786*, *789*, *791*, *795*, *801*, *805*, *807*, *900*, 1098, *1111*, *1113*, *1117*, *1119*, *1138, 1139*
qqplot, *305*, *1098*, *1103*
qqPlotCensored, *73, 74*, *154*, *193, 194*, *344*, *350*, *515*, *518*, *548*, *552*, *558*, *646*, *648*, *780–782*, *797*, *899, 900*, *1105*, 1107, *1368*, *1371*, *1375*
qqPlotGestalt, *565*, *1105*, *1113*, 1116
qtri *(*Triangular*)*, 1286
quantile, *219*, *228*, *282*, *305*, *327*, *338*, *347*, *424*, *488, 489*, *494*, *573*, *660*, *664*, *686*, *1208*
quantile test, *1125*, *1351*
quantile-quantile plot, *28*, *41*
Quantile-Quantile plots, *895*
quantile-quantile plots, *54*
quantileTest, *563*, *675*, 1120, *1126, 1127*
quantileTestPValue, *563*, *1122*, *1124*, 1125
qunif, *510*
qweibull, *512*
qzmlnorm, *515*
qzmlnorm *(*ZeroModifiedLognormal*)*, 1367
qzmlnormAlt, *515*
qzmlnormAlt *(*ZeroModifiedLognormalAlt*)*, 1370
qzmnorm, *517*
qzmnorm *(*ZeroModifiedNormal*)*, 1373

range, *573*, *1208*
Rank Sum, *173*, *176*
rbeta, *171*
rchi *(*Chi*)*, 87
Refinery.CO.df, 1127
remp, *1170*
remp *(*Empirical*)*, 302
reshape, *750*, *752*
revd *(*EVD*)*, 535
rgamma, *576*
rgammaAlt *(*GammaAlt*)*, 575
rgevd *(*GEVD*)*, 592
rlnorm, *735*, *739*, *1170*
rlnorm3 *(*Lognormal3*)*, 734
rlnormAlt *(*LognormalAlt*)*, 738
rlnormMix *(*LognormalMix*)*, 741
rlnormMixAlt *(*LognormalMixAlt*)*, 743
rlnormTrunc *(*LognormalTrunc*)*, 745
rlnormTruncAlt *(*LognormalTruncAlt*)*, 748
rnormMix *(*NormalMix*)*, 755
rnormTrunc *(*NormalTrunc*)*, 757
Rosner *(*rosnerTest*)*, 1128
rosnerTest, *562*, *613, 614*, *633*, 1128
royston.skew,, *174*
rpareto *(*Pareto*)*, 767
rtri *(*Triangular*)*, 1286
rzmlnorm *(*ZeroModifiedLognormal*)*, 1367
rzmlnormAlt *(*ZeroModifiedLognormalAlt*)*, 1370
rzmnorm *(*ZeroModifiedNormal*)*, 1373

sample, *305*
sample kurtosis *(*kurtosis*)*, 711
sd, *130*, *339*, *687*, *714*, *1176*, *1178*, *1208*
segments, *521*
serial correlation *(*serialCorrelationTest*)*, 1141
serial correlation test *(*serialCorrelationTest*)*, 1141
serialCorrelationTest, *563*, *675*, 1141
set.seed, *291*, *316*, *336*, *343*, *461*, *480*, *539*, *580*, *660*, *664*, *762*, *1161*, *1166*, *1169*, *1171*, *1194*, *1348*
shapiro.test, *633*, *648*
sign, *691*, *705*, *1327*, *1334*
sign test, *764*
sign test *(*signTest*)*, 1152
signTest, *563*, *675*, 1152
simulateMvMatrix, *305*, *564*, 1160, *1171*
simulateVector, *305*, *564*, *1163*, *1166*, 1168
Skagit.NH3_N.df, 1172
Skew *(*skewness*)*, 1173
skew *(*skewness*)*, 1173
Skewness *(*skewness*)*, 1173
skewness, *83*, *130*, *574*, *714*, *733*, 1173, *1207*
smedian.hilow, *581*
standard normal distribution, *218*, *227*, *280*, *325*, *337*, *346*, *423*
stat_mean_sd_text, *565*, *580*, *583, 584*, 1176, *1182*, *1186*, *1191*
stat_median_iqr_text, *565*, *580*, *583, 584*, *1178*, 1180, *1186*, *1191*
stat_n_text, *565*, *580*, *583, 584*, *1178*, *1182*, 1184, *1191*
stat_summary, *580*, *582*, *584*

stat_test_text, *565*, *580*, *583, 584*, *1178*, *1182*, *1186*, 1188
stripChart, *521*, *565*, *574*, *580*, 1193
stripchart, *574*, *1193, 1194*, *1201*
Student t-statistic, *1312*, *1315*
Student's t, *173*, *175*
Student's t-distribuiton, *218*, *227*, *280, 281*, *325*, *337*, *346*, *423*
Student's t-distribution, *83*, *104*, *118, 119*, *198*, *252*, *263*, *300*, *311*, *324*, *490*, *547*, *552*, *720, 721*, *959*, *1047*, *1129*, *1312*, *1316*
Student's t-test, *85*, *764*, *1155*, *1297*, *1300*, *1350, 1351*
summary, *573*, *1207, 1208*, *1216*
Summary Plots *(*FcnsByCatSumStats*)*, 573
Summary Statistics, *130*, *555*, *687*, *714*, *1176*
Summary Statistics *(*FcnsByCatSumStats*)*, *573*
summary.aov, *1190, 1191*
summary.lm, *717*, *724*, *728*
summaryFull, *130*, *574*, *579*, *591*, *687*, *714*, *1071, 1072*, *1176*, 1205, *1215, 1216*, *1222, 1223*
summaryStats, *574*, *1071, 1072*, *1208*, 1211, *1222, 1223*
summaryStats.object, *1071, 1072*, *1208*, *1215*, 1222
survdiff, *1338, 1339*
survfit, *899, 900*

t, *573*
T-test power *(*tTestPower*)*, 1311
t-test Power *(*tTestPower*)*, 1311
t-test power *(*tTestPower*)*, 1311
t.test, *83*, *86*, *105*, *109*, *120*, *675*, *717*, *724*, *728*, *824*, *881*, *885*, *1189–1191*, *1197*, *1201*, *1214*, *1216*, *1290*, *1293*, *1300*, *1304*, *1308*, *1317*, *1320*
TDist, *1259*
Tests for Outliers, *1064, 1065*
the lognormal distribution, *746*
the normal distribution, *758*
the sign test, *85*
the Wilcoxon signed rank test, *85*
Three Parameter Lognormal *(*Lognormal3*)*, *734*
Three-, *172*, *174*
three-parameter generalized extreme value distribution (GEVD), *1377*
Three-Parameter Lognormal, *137*, *560, 561*, *607*, *617*, *619*
three-parameter lognormal distribution, *246*, *248*, *456*, *622, 623*
three-parameter lognormal distributions, *789*
title, *1194, 1195*
Tolerance Intervals, *494*, *523*, *525*, *555*, *559*, *1059, 1060*, *1235*, *1246*, *1262*, *1271*, *1284*
Tolerance Intervals *(*FcnsByCatTolInts*)*, *574*
tolIntGamma, *210*, *441*, *574*, *912*, *921*, 1224
tolIntGammaAlt, *574*
tolIntGammaAlt *(*tolIntGamma*)*, 1224
tolIntLnorm, *574*, *928*, *944*, 1232, *1246*
tolIntLnormAlt, *574*, *928*
tolIntLnormAlt *(*tolIntLnorm*)*, 1232
tolIntLnormCensored, *559*, 1237
tolIntNorm, *473, 474*, *523*, *574*, *872, 873*, *948*, *950*, *958*, *962*, *969*, *972*, *981*, *985*, *990*, *996*, *1224, 1225*, *1227*, *1233*, *1235*, 1243, *1250*, *1255*, *1258*, *1262*, *1264, 1265*
tolIntNormCensored, *464*, *481*, *483*, *527*, *559*, *664*, *1239, 1240*, 1248
tolIntNormHalfWidth, *569*, *872*, 1254, *1264, 1265*
tolIntNormK, *569*, *574*, *872, 873*, *1244*, *1246*, *1255*, 1257, *1264, 1265*
tolIntNormN, *569*, *872, 873*, *1255*, 1263
tolIntNpar, *494*, *559*, *574*, *859*, *863*, *876, 877*, *1001*, *1020*, *1029*, *1033*, *1040*, 1267, *1274, 1275*, *1277*, *1279, 1280*
tolIntNparConfLevel, *569*, *876, 877*, 1273, *1277*, *1280*
tolIntNparCoverage, *569*, *876, 877*, *1274, 1275*, 1276, *1280*
tolIntNparN, *569*, *876, 877*, *1271*, *1274, 1275*, *1277*, 1278
tolIntPois, *507*, *575*, *1049*, 1281
Total.P.df, 1285
Trend Analysis, *555*
Trend Analysis *(*FcnsByCatTrend*)*, 575
tri, *573*

Triangular, *173*, *175*, 1286
Truncated, *172*, *175*
ts.plot, *1148*
tTestAlpha, *567*, 1289, *1308*, *1317*, *1320*
tTestLnormAltN, *567*, *885*, 1291, *1300*, *1304*
tTestLnormAltPower, *567*, *839*, *885*, *939*, *1293*, 1296, *1304*
tTestLnormAltRatioOfMeans, *567*, *885*, *1293*, *1300*, 1303
tTestN, *567*, *880, 881*, *1290*, 1307, *1316, 1317*, *1320*
tTestPower, *567*, *880, 881*, *1290*, *1299, 1300*, *1308*, 1311, *1319, 1320*
tTestScaledMdd, *567*, *880, 881*, *1290*, *1308*, *1316, 1317*, 1318
Two-parameter extreme value distributions (EVD), *1377*
two-parameter lognormal distribution, *735, 736*
two-sample linear rank tests, *1351*
twoSampleLinearRankTest, *563*, *675*, 1322, *1332*, *1339*
twoSampleLinearRankTestCensored, *558*, *679*, *1069*, *1322*, *1328*, 1331
twoSamplePermutationTestLocation, *563*, *773, 774*, *808, 809*, *1070, 1071*, 1347, *1356, 1357*
twoSamplePermutationTestProportion, *564*, *773, 774*, *808, 809*, *1070, 1071*, *1351*, 1354
Type I (Gumbel) extreme value distribution, *237*
Type I Extreme Value (Gumbel) distribution, *1096*
Type I extreme value (Gumbel) distribution, *513*, *532*, *544*, *593*, *789*
Type I, also called the Gumbel extreme value distribution or simply Gumbel distribution, *199*, *434*

unif, *573*
Uniform, *169, 170*, *173*, *175*, *177*, *428*, *511*, *534*, *560, 561*, *1288*
uniform (0,1), *1102*
Uniform [0,1] distribution, *609*, *801*
uniform distribution, *509, 510*, *532–534*
uniroot, *14*, *97*, *99*, *108*, *716, 717*, *727*, *812*, *823*, *830*, *842*, *854*, *858*, *871*, *880*, *884*, *964*, *1012*, *1033*, *1074*, *1080*, *1263, 1264*, *1289, 1290*, *1293*, *1304*, *1308*, *1319, 1320*

Value, *171*, *174*
var, *130*, *573*, *687*, *714*, *1176*
var.test, *1362, 1363*, *1365, 1366*
varGroupTest, *564*, *675*, 1360, *1366*
varTest, *564*, *675*, *1363*, 1364

Weibull, *173*, *175*, *513*, *544*, *561, 562*
weibull, *573*
Weibull distribution, *199*, *434*, *512*, *537*, *542, 543*
Weibull random variable, *199*, *434*, *537*
wilcox, *573*
wilcox.test, *92*, *98*, *818*, *1124*, *1127*, *1156*, *1190, 1191*, *1197, 1198*, *1201*, *1214*, *1216*, *1326*, *1328*, *1339*
Wilcoxon, *173*, *176*
Wilcoxon Rank Sum test, *1322*, *1326*, *1328*
Wilcoxon rank sum test, *1122*, *1351*
Wilcoxon signed rank test, *764*, *1155*

Zero Modified Lognormal *(*ZeroModifiedLognormal*)*, 1367
Zero Modified Normal *(*ZeroModifiedNormal*)*, 1373
Zero-Modified, *173*, *176*
Zero-Modified Lognormal, *516*, *549*, *1372*, *1375*
Zero-Modified Lognormal *(*ZeroModifiedLognormal*)*, 1367
Zero-Modified Lognormal (Alternative Parameterization), *516*, *1369*
Zero-Modified Lognormal (Delta), *137*, *561, 562*, *607*, *617*, *619*, *624–627*, *633*
zero-modified lognormal (delta), *518*, *552*
zero-modified lognormal (delta) distribution, *1374*
zero-modified lognormal distribution, *514*, *545, 546*
zero-modified lognormal distribution (alternative parameterization), *514*, *545, 546*
Zero-Modified Normal, *137*, *516*, *549*, *561*, *562*, *607*, *617*, *619*, *624–627*, *633*, *1369*

Zero-Modified Normal
(ZeroModifiedNormal), 1373
zero-modified normal, *518*, *552*
zero-modified normal distribution, *515*, *517*, *518*, *548*, *550–552*, *1368*, *1371*
zero.skew, *175*
ZeroModifiedLognormal, *518*, *553*, 1367
ZeroModifiedLognormalAlt, 1370
ZeroModifiedNormal, *518*, *553*, 1373
zmlnorm, *573*
zmlnormAlt, *573*
zmnorm, *573*
zTestGevdShape, *238*, *239*, *447*, *564*, *593*, *594*, *671*, *675*, 1376