Package 'RESI'

July 24, 2026

Type Package

Title Robust Effect Size Index (RESI) Estimation

Version 1.4.2

Description Summarize model output using a robust effect size index. The index is introduced in Vandekar, Tao, & Blume (2020, <doi:10.1007/s11336-020-09698-2>). Software paper available at <doi:10.18637/jss.v112.i03>.

License GPL-3

Encoding UTF-8

LazyData true

Collate 'internal_functions.R' 'summary.resi.R' 'resi.R' 'resi_pe.R' 'Anova.resi.R' 'data.R' 'chisq2S.R' 'f2S.R' 't2S.R' 'z2S.R' 'Rsq2S.R' 'S2Rsq.R' 'S2d.R' 'S2fsq.R' 'S2z.R' 'S2chisq.R' 'd2S.R' 'fsq2S.R' 'plot.R' 'ggplot.R' 'print.R' 'omnibus.R' 'simulations.R' 'resi_asymptotic.R'

Imports aod, boot, car, clubSandwich, CompQuadForm, ggplot2, lmtest, nlme, parallel, sandwich, splines

Suggests gee, geepack, robustbase, glmmTMB, glmtoolbox, knitr, lme4, pscl, R.rsp, regtools, rmarkdown, survival, tibble, testthat (>= 3.0.0)

RoxygenNote 7.3.3

Depends R (>= 2.10)

Config/testthat/edition 3

URL https://statimagcoll.github.io/RESI/, https://github.com/statimagcoll/RESI

BugReports https://github.com/statimagcoll/RESI/issues

NeedsCompilation no

Author Megan Jones [aut], Kaidi Kang [aut], Simon Vandekar [aut, cre], Gina Yu [ctb], Xinyu Zhang [ctb]

Maintainer Simon Vandekar <simon.vandekar@vumc.org>

Repository CRAN

Date/Publication 2026-07-24 21:20:08 UTC

Contents

`anova.resi` *Anova method for resi objects*

Description

After running the `resi` function on a fitted model, this function can be used to print the Anova-style table component. If the resi function was run with the 'store.boot = TRUE' option to store the full matrix of bootstrapped estimates, the user can specify a different alpha level for this function's confidence intervals.

Usage

```
## S3 method for class 'resi'
anova(object, alpha = NULL, ...)
```

Arguments

object	an object resulting from resi function
alpha	an optional new specification for the confidence level. Can be vector-valued
...	ignored

Details

The resi function uses the car::Anova function to compute the Anova table.

Value

Returns an ‘anova‘ object containing the computed Anova-style table

Examples

```
# fit a model
mod = lm(charges ~ bmi + sex, data = RESI::insurance)

# run resi with the store.boot = TRUE option
resi.obj = resi(mod, nboot = 100, store.boot = TRUE, alpha = 0.01)

# run anova, specifying a different alpha level if desired
anova(resi.obj, alpha = 0.05)
```

---

chisq2S *Compute the robust effect size index estimate from chi-squared statistic.*

Description

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R. For mixed effects models, RESI is conditional on the average correlation structure within subjects.

Usage

```
chisq2S(chisq, df, n)
```

Arguments

chisq	The chi-square statistic for the parameter of interest.
df	Number of degrees of freedom of the chi-square statistic.
n	Number of independent samples.

Details

The formula for converting a Chi-square statistic to RESI is:
$S = \sqrt{(max(0, (chisq - df)/n))}$

Value

Returns a scalar or vector argument of the robust effect size index estimate.

Examples

```
# obtain Chi-sq value by fitting an lm and running a Wald test
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)

# run a Wald test with robust variance
wt = lmtest::waldtest(mod, vcov = sandwich::vcovHC, test = "Chisq")

# get Chi-sq value and degrees of freedom
chisq = wt$Chisq[2]
df = abs(wt$Df[2])

# run chisq2S to convert to RESI
chisq2S(chisq, df = df, n = nrow(mod$model))
```

---

d2S *Covert Cohen's* d *to* |*S*|

Description

Converts Cohen's d robust effect size index (S) using the formula from Vandekar, Tao, & Blume (2020).

Usage

```
d2S(d, pi = 0.5)
```

Arguments

d	Numeric, value of Cohen's d.
pi	Numeric, the sampling proportions.

Details

The pi parameter comes from the fact that Cohen's d doesn't account for unequal sample proportions in the population, but S does.

The default is set to a natural value 1/2, which corresponds to a case control design, for example, where sampling proportions always are controlled by the experimenter.

The formula to convert Cohen's d to S is:
$S = d/\sqrt{(1/\pi + 1/(1-\pi))}$

Value

Returns an estimate the robust effect size index

Examples

```
# Consider an experiment with equal sampling proportions and a medium effect size
# corresponding to a Cohen's d of 0.5.
# convert to RESI (S)
d2S(d = 0.5)

# This corresponds to a RESI of 0.25.
```

---

depression *Depression Treatment Data*

Description

A longitudinal dataset comparing two treatments for depression.

Usage

```
depression
```

Format

A data frame with 1020 rows and 5 variables:

diagnose diagnosed depression severity

drug treatment; standard or new

id patient id

time time point of treatment

depression depression response at time of treatment. 1 = Normal, 0 = Abnormal

Source

http://static.lib.virginia.edu/statlab/materials/data/depression.csv

References

Agresti, A. (2002). Categorical Data Analysis. Wiley, 2nd Edition.

f2S *Compute the robust effect size index estimate from F-statistic*

Description

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

Usage

```
f2S(f, df, rdf, n)
```

Arguments

f	The F statistic for the parameter of interest.
df	Number of degrees of freedom of the F statistic.
rdf	Model residual degrees of freedom.
n	Number of independent samples.

Details

The formula for converting an F statistic to S is:

$S = \sqrt{(max(0, (f * df * (rdf - 2)/rdf - df)/n))}$

The estimator is derived by setting the statistic equal to the expected value of the test statistic and solving for S.

Value

Returns a scalar or vector argument of the robust effect size index estimate.

Examples

```
# to obtain example F values, first fit a lm
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)

# run Anova, using a robust variance-covariance function
# get the F values and Df values
fs = car::Anova(mod, vcov. = sandwich::vcovHC)[1:5, "F"]
dfs = car::Anova(mod, vcov. = sandwich::vcovHC)[1:5, "Df"]

# get RESI estimates
f2S(fs, df = dfs, rdf = mod$df.residual, n = nrow(RESI::insurance))
```

fsq2S *Covert Cohen's f^2 to S*

Description

Converts Cohen's f^2 to robust effect size index (S) using the formula from Vandekar, Tao, & Blume (2020).

Usage

```
fsq2S(fsq)
```

Arguments

fsq	Numeric, value of Cohen's f^2.

Details

The formula for the conversion is:

$S = \sqrt{(f^2)}$

Value

Returns an estimate the robust effect size index

Examples

```
# consider a moderate effect size of f^2 = 0.3
fsq2S(0.3)
# This corresponds to a RESI of 0.5477226
```

ggplot.resi *Plotting RESI Estimates and CIs*

Description

This function uses ggplot2 graphics to plot robust effect size (RESI) estimates and confidence intervals from 'resi', 'summary_resi', and 'anova_resi' objects.

Usage

```
## S3 method for class 'resi'
ggplot(data, mapping, alpha = NULL, error.bars = TRUE, ..., environment)
```

Arguments

data	Object of ‘resi‘, ‘summary_resi‘, or ‘anova_resi‘ class
mapping	Ignored, included for consistency with ‘ggplot‘ generic
alpha	Numeric, desired alpha level for confidence intervals
error.bars	Logical, whether to include end caps on the confidence intervals. Default = ‘TRUE‘
...	Ignored
environment	Ignored, included for consistency with ‘ggplot‘ generic

Value

Returns a ggplot of RESI point estimates

Examples

```
# create a resi object
resi_obj <- resi(lm(charges ~ region * age + bmi + sex, data = RESI::insurance),
nboot = 10)

# plot ANOVA table
ggplot2::ggplot(anova(resi_obj))
```

---

insurance *US Health Insurance Data*

Description

A dataset with 1338 observations on health insurance charges and demographic factors.

Usage

```
insurance
```

Format

A data frame with 1338 rows and 7 variables:

age age of primary beneficiary in years
sex insurance contractor sex, male/female
bmi body mass index
children number of dependents
smoker smoker/non-smoker
region beneficiary's region of US
charges individual medical costs billed by health insurance

Source

Kaggle dataset teertha ushealthinsurancedataset. url checker thinks the link is broken.

`insurancePlasmodeSim` *Insurance Plasmode Simulation for RESI Evaluation*

Description

Runs a plasmode simulation study using the `insurance` dataset to evaluate RESI confidence interval performance. In each replicate, `n` observations are resampled with replacement from the full insurance dataset ($N = 1338$). The RESI point estimates from `resi_pe` applied to the full dataset are treated as the true parameter values for computing bias, MSE, CI coverage, and CI width.

Usage

```
insurancePlasmodeSim(
  nsim = 1000L,
  n.vec = c(50, 100, 200, 500, 1000, 2000, 5000),
  nboot = 500L,
  alpha = 0.05,
  ci.method = c("boot", "normal", "qf", "cf"),
  output.dir = NULL,
  fixed.knots = FALSE,
  mc.cores.settings = 1L,
  mc.cores.reps = 1L
)
```

Arguments

`nsim`	Integer, number of simulation replicates per (setting, `n`) cell. Default 1000. Use 10 for initial testing.
`n.vec`	Integer vector of sample sizes. Default `c(50, 100, 200, 500, 1000, 2000, 5000)`.
`nboot`	Integer, bootstrap replicates per internal `resi` call. Default 500. Use 10 for initial testing. Ignored when `ci.method != "boot"`.
`alpha`	Numeric, CI significance level. Default 0.05.
`ci.method`	Character, CI method passed to `resi`. One of `"boot"` (bootstrap, default), `"normal"` (asymptotic truncated-normal), or `"qf"` (asymptotic quadratic-form / Imhof). When `ci.method != "boot"`, `nboot` is ignored.
`output.dir`	Character, path to the directory where all results are saved. Created if it does not exist. Defaults to `"resiBootSim"`, `"resiAsympNormalSim"`, or `"resiAsympQFSim"` based on `ci.method` when `NULL`.
`fixed.knots`	Logical. If `TRUE`, spline knots are fixed at the empirical tertiles of `age` in the full insurance dataset rather than re-selected by `df = 3` in each bootstrap sample. Default `FALSE`.
`mc.cores.settings`	Integer, cores for the outer `mclapply` over (setting $\times$ sample size) combinations. Default 1.

`mc.cores.reps` Integer, cores for the inner `mclapply` over simulation replicates within each (setting, `n`) cell. Default 1.

Details

Two models are evaluated:

- **lm**: `log10(charges) ~ ns(age, df=3) * sex + bmi + smoker + region`
- **glm**: `I(charges > 10000) ~ ns(age, df=3) * sex + bmi + smoker + region` with `family = binomial()`

Each model is evaluated under both parametric (`vcovfunc = stats::vcov`) and robust (`vcovfunc = sandwich::vcovHC`) variance settings, yielding four simulation conditions.

Parallelization is via `mclapply`, which uses forking and is not supported on Windows (falls back to sequential evaluation on Windows).

Value

Invisibly returns the summary metrics `data.frame`. Side effects:

- `output.dir/sim_raw/<setting>_n<n>.rds`: list of per-replicate `anova` and `coefficients` tables.
- `output.dir/summary_table.rds`: combined metrics table with columns `model`, `vcov`, `n`, `n_success`, `table`, `term`, `bias`, `mse`, `coverage`, `width`.

See Also

`simFigures`, `simCompareMethodsFigures`, `resi`, `resi_pe`

`omnibus` *Omnibus (Overall) Wald Test for resi objects*

Description

After running the `resi` function on a fitted model, this function can be used to print the overall Wald test component. If the resi function was run with the ‘store.boot = TRUE‘ option to store the full matrix of bootstrapped estimates, the user can specify a different alpha level for this function’s confidence intervals.

Usage

```
omnibus(object, alpha = NULL, ...)
```

Arguments

`object`	an object resulting from resi function
`alpha`	an optional new specification for the confidence level. Can be vector-valued
`...`	ignored

Value

Returns a ‘omnibus_resi‘ object containing the computed omnibus Wald test

Examples

```
# fit a model
mod = lm(charges ~ bmi + sex, data = RESI::insurance)

# run resi with the store.boot = TRUE option (ci.method must be "boot")
resi_obj = resi(mod, nboot = 100, store.boot = TRUE, alpha = 0.01, ci.method = "boot")

# run summary, specifying a different alpha level if desired
omnibus(resi_obj, alpha = 0.05)
```

`plot.resi` *Plotting RESI Estimates and CIs*

Description

This function uses base graphics to plot robust effect size (RESI) estimates and confidence intervals from ‘resi‘, ‘summary_resi‘, and ‘anova_resi‘ objects.

Usage

```
## S3 method for class 'resi'
plot(
  x,
  alpha = NULL,
  ycex.axis = NULL,
  yaxis.args = list(),
  automar = TRUE,
  ...
)
```

Arguments

`x`	Object of ‘resi‘, ‘summary_resi‘, or ‘anova_resi‘ class
`alpha`	Numeric, desired alpha level for confidence intervals
`ycex.axis`	Numeric, scale specifically for the variable name labels
`yaxis.args`	List, other arguments to be passed to `axis` for the y-axis
`automar`	Logical, whether to automatically adjust the plotting margins to accommodate variable names. Default = ‘TRUE‘
`...`	Other graphical parameters passed to `plot` and `lines`

Details

This function creates a forest-like plot with RESI estimates for each variable or factor. The size of the left margin will be automatically adjusted (and returned to original after plotting) unless 'automar = FALSE'. Additional graphics parameters will be passed to the main plot function, the confidence intervals. Arguments specifically for the y-axis (variable names) can be specified using 'yaxis.args'. To manually adjust the size of the y-axis labels without affecting the x-axis, the user can specify a value for 'ycex.axis'.

Value

Returns a plot of RESI point estimates

Examples

```
# create a resi object
resi_obj <- resi(lm(charges ~ region * age + bmi + sex, data = RESI::insurance),
nboot = 10)

# plot coefficients table, changing size of labels for both axes in the usual way
plot(resi_obj, cex.axis = 0.7)

# plot ANOVA table, changing the size of just the y-axis
plot(resi_obj, ycex.axis = 0.8)
```

`resi` *Robust Effect Size Index (RESI) point and interval estimation for models*

Description

This function will estimate the robust effect size (RESI) from Vandekar, Tao, & Blume (2020) and its confidence interval in various ways for a fitted model object. The overall RESI is estimated via a Wald test. RESI is (optionally) estimated for each factor in coefficients-style table. RESI is (optionally) estimated for each variable/interaction in an Anova-style table for models with existing Anova methods. CIs can be calculated using either non-parametric or Bayesian bootstrapping.

Usage

```
resi(model.full, ...)

## Default S3 method:
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
```

```
  nboot = 1000,
  boot.method = "nonparam",
  vcovfunc = sandwich::vcovHC,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  long = FALSE,
  clvar = NULL,
  ci.method = c("boot", "qf", "cf", "normal"),
  ...
)

## S3 method for class 'glm'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = sandwich::vcovHC,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ci.method = "qf",
  ...
)

## S3 method for class 'lm'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  boot.method = "nonparam",
  vcovfunc = sandwich::vcovHC,
```

```
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ci.method = "qf",
  ...
)

## S3 method for class 'lmrob'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  boot.method = "nonparam",
  vcovfunc = stats::vcov,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ci.method = "boot",
  ...
)

## S3 method for class 'glmrob'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = stats::vcov,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
```

```
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ci.method = "boot",
  ...
)

## S3 method for class 'nls'
resi(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  boot.method = "nonparam",
  anova = FALSE,
  vcovfunc = r_nlshc,
  alpha = 0.05,
  store.boot = FALSE,
  vcov.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'survreg'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = vcov,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'coxph'
resi(
  model.full,
```

```
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = vcov,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'hurdle'
resi(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = sandwich::sandwich,
  anova = FALSE,
  alpha = 0.05,
  store.boot = FALSE,
  vcov.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'zeroinfl'
resi(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = sandwich::sandwich,
  anova = FALSE,
  alpha = 0.05,
  store.boot = FALSE,
  vcov.args = list(),
```

```
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'geeglm'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  alpha = 0.05,
  store.boot = FALSE,
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'glmgee'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = FALSE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  alpha = 0.05,
  store.boot = FALSE,
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'gee'
resi(
  model.full,
  data,
  nboot = 1000,
  alpha = 0.05,
  store.boot = FALSE,
  unbiased = TRUE,
```

```
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'lme'
resi(
  model.full,
  alpha = 0.05,
  nboot = 1000,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  vcov.args = list(),
  ...
)

## S3 method for class 'lmerMod'
resi(
  model.full,
  alpha = 0.05,
  nboot = 1000,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  vcov.args = list(),
  ...
)

## S3 method for class 'glmmTMB'
resi(
  model.full,
  alpha = 0.05,
  nboot = 1000,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  vcov.args = list(),
  ...
)
```

Arguments

model.full	lm, glm, nls, survreg, coxph, hurdle, zeroinfl, gee, geeglm or lme model object.
...	Ignored.
model.reduced	Fitted model object of same type as model.full. By default 'NULL'; the same model as the full model but only having intercept.
data	Data.frame or object coercible to data.frame of model.full data (required for some model types).

anova	Logical, whether to produce an Anova table with the RESI columns added. By default = 'TRUE'.
coefficients	Logical, whether to produce a coefficients (summary) table with the RESI columns added. By default = 'TRUE'.
overall	Logical, whether to produce an overall Wald test comparing full to reduced model with RESI columns added. By default = 'TRUE'.
nboot	Numeric, the number of bootstrap replicates. Used only when 'ci.method = "boot"'. By default, 1000.
boot.method	String, which type of bootstrap to use: 'nonparam' = non-parametric bootstrap (default); 'bayes' = Bayesian bootstrap.
vcovfunc	The variance estimator function for constructing the Wald test statistic. By default, vcovHC (the robust (sandwich) variance estimator).
alpha	Numeric, significance level of the constructed CIs. By default, 0.05.
store.boot	Logical, whether to store all the bootstrapped estimates. By default, 'FALSE'.
Anova.args	List, additional arguments to be passed to Anova function.
vcov.args	List, additional arguments to be passed to vcovfunc.
unbiased	Logical, whether to use the unbiased or alternative T/Z statistic to RESI conversion. By default, 'TRUE'. See details.
parallel	See documentation for boot.
ncpus	See documentation for boot.
long	Logical, whether the data is longitudinal/clustered. By default, 'FALSE'.
clvar	Character, the name of the cluster/id variable if data is clustered. By default, 'NULL'.
ci.method	Character, the method used to compute confidence intervals. One of '"boot"' (bootstrap), '"qf"' (quadratic-form inversion, default for lm and glm), '"normal"' (normal approximation), or '"cf"' (Cornish-Fisher inversion). See 'resi_pe_asymptotic' for details on the asymptotic methods.

Details

The RESI, denoted as S, is applicable across many model types. It is a unitless index and can be easily be compared across models. The RESI can also be converted to Cohen's *d* (S2d) under model homoskedasticity.

This function computes the RESI point estimates and bootstrapped confidence intervals based on Chi-square, F, T, or Z statistics. The robust (sandwich) variance is used by default, allowing for consistency under model-misspecification. The RESI is related to the non-centrality parameter of the test statistic. The RESI estimate is consistent for all four (Chi-square, F, T, and Z) types of statistics used. The Chi-square and F-based calculations rely on asymptotic theory, so they may be biased in small samples. When possible, the T and Z statistics are used. There are two formulas for both the T and Z statistic conversion. The first (default, unbiased = TRUE) are based on solving the expected value of the T or Z statistic for the RESI. The alternative is based on squaring the T or Z statistic and using the F or Chi-square statistic conversion. Both of these methods are consistent, but the alternative exhibits a notable amount of finite sample bias. The alternative may be appealing because its absolute value will be equal to the RESI based on the F or Chi-square statistic. The RESI

based on the Chi-Square and F statistics is always greater than or equal to 0. The type of statistic used is listed with the output. See `f2S`, `chisq2S`, `t2S`, and `z2S` for more details on the formulas.

For GEE (geeglm, glmgee) models, a longitudinal RESI (L-RESI) and a cross-sectional, per-measurement RESI (CS-RESI) is estimated. The longitudinal RESI takes the specified clustering into account, while the cross-sectional RESI is designed to estimate the effect size if a random observation for each participant were collected cross-sectionally.

For most `lm` and `nls` model types, there is a Bayesian bootstrap option available as an alternative to the default, standard non-parametric bootstrap. The interpretation of a Bayesian bootstrapped interval is similar to that of a credible interval.

Certain model types require the data used for the model be entered as an argument. These are: `nls`, `survreg`, and `coxph`. Additionally, if a model includes certain functions (splines, factor, I), the data needs to be provided.

If running into convergence issues with nls models, it is advised to refit the nls model with starting values equal to the estimates provided by the model and then try rerunning `resi`.

Value

Returns a list that includes function arguments, RESI point estimates, and confidence intervals in coefficients/anova-style tables

Methods (by class)

- `resi(default)`: RESI point and interval estimation for models
- `resi(glm)`: RESI point and interval estimation for models
- `resi(lm)`: RESI point and interval estimation for lm models
- `resi(lmrob)`: RESI point and interval estimation for lmrob models (robustbase)
- `resi(glmrob)`: RESI point and interval estimation for glmrob models (robustbase)
- `resi(nls)`: RESI point and interval estimation for nls models
- `resi(survreg)`: RESI point and interval estimation for survreg models
- `resi(coxph)`: RESI point and interval estimation for coxph models
- `resi(hurdle)`: RESI point and interval estimation for hurdle models
- `resi(zeroinfl)`: RESI point and interval estimation for zeroinfl models
- `resi(geeglm)`: RESI point and interval estimation for GEE models
- `resi(glmgee)`: RESI point and interval estimation for GEE models in glmtoolbox
- `resi(gee)`: RESI point and interval estimation for GEE models
- `resi(lme)`: RESI point and interval estimation for LME (nlme) models
- `resi(lmerMod)`: RESI point and interval estimation for lmerMod models
- `resi(glmmTMB)`: RESI point and interval estimation for glmmTMB models - Gaussian only

References

Vandekar S, Tao R, Blume J. A Robust Effect Size Index. *Psychometrika*. 2020 Mar;85(1):232-246. doi: 10.1007/s11336-020-09698-2.

Kang, K., Armstrong, K., Avery, S., McHugo, M., Heckers, S., & Vandekar, S. (2021). Accurate confidence interval estimation for non-centrality parameters and effect size indices. *arXiv preprint arXiv:2111.05966*.

Jones, M., Kang, K., Vandekar, S. (2025). *Journal of Statistical Software*. RESI: An R Package for Robust Effect Sizes.<doi:10.18637/jss.v112.i03>

See Also

resi_pe, vcovHC, f2S, chisq2S, z2S, t2S

Examples

```
## for timing purposes, a small number of bootstrap replicates is used in the
## examples. Run them with a higher or default `nboot` argument for better performance

## RESI on a linear model
# fit linear model
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)

# run resi on fitted model with desired number of bootstrap replicates
# store bootstrap results for calculating different CIs later
resi_obj = resi(mod, nboot = 50, store.boot = TRUE)
# print output
resi_obj

# fit a reduced model for comparison
mod_red = lm(charges ~ bmi, data = RESI::insurance)

# running resi and including the reduced model will provide almost the exact same
# output as not including a reduced model. The difference is that the "overall"
# portion of the output will compare the full model to the reduced model.
# The "summary" and "anova" RESI estimates will be the same. (The bootstrapped
# confidence intervals may differ.)
resi(model.full = mod, model.reduced = mod_red, nboot = 10)

# used stored bootstrap results to get a different alpha-level confidence interval
summary(resi_obj, alpha = c(0.01, 0.1))
car::Anova(resi_obj, alpha = c(0.01, 0.1))

# the result of resi, as well as the summary or Anova of a `resi` object can be plotted
# if the resi object was created with the store.boot = `TRUE` option, any alpha
# can be specified
plot(resi_obj, alpha = 0.01)
# if the variable names on the y-axis are too long, you can reduce their size with
# the ycex.axis argument (or use regular common solutions like changing the margins)
plot(resi_obj, alpha = 0.01, ycex.axis = 0.5)

# for some model types and formula structures, data argument is required
```

```
if(requireNamespace("splines")){
  # fit logistic regression model with splines
  mod = glm(smoker ~ splines::ns(age, df = 3) + region, data = RESI::insurance,
    family = "binomial")

  # specify additional arguments to the variance-covariance function via vcov.args
  resi_obj = resi(mod, data = RESI::insurance, alpha = 0.01,
    vcov.args = list(type = "HC0"), nboot = 25)
  summary(resi_obj)
  car::Anova(resi_obj)}


## RESI on a survival model with alternate Z2S
if(requireNamespace("survival")){
  # fit coxph model on example data from survival package
  # Note: for survival models, you need to specify robust variance in the model
  # creation. resi will ignore the vcovfunc argument for this reason.
  mod.coxph =  survival::coxph(survival::Surv(time, status) ~ age + sex + wt.loss,
   data=survival::lung, robust = TRUE)

  # run resi on the model
  # to use the alternative Z to RESI formula (which is equal in absolute value to the
  # chi-square to RESI (S) formula), specify unbiased = FALSE.
  resi(mod.coxph, data = survival::lung, unbiased = FALSE, nboot = 10)}
```

---

resi_pe *Robust Effect Size Index (RESI) Point Estimation*

Description

This function will estimate the robust effect size (RESI) from Vandekar, Tao, & Blume (2020). The overall RESI is estimated via a Wald test. RESI is (optionally) estimated for each factor in coefficients-style table. RESI is (optionally) estimated for each variable/interaction in an Anova-style table for models with existing Anova methods. This function is the building block for the resi function.

Usage

```
resi_pe(...)

## Default S3 method:
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
```

```
  vcovfunc = sandwich::vcovHC,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  waldtype = 0,
  ...
)

## S3 method for class 'glm'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  vcovfunc = sandwich::vcovHC,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  waldtype = 0,
  ...
)

## S3 method for class 'lm'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = sandwich::vcovHC,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'nls'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  anova = FALSE,
  vcovfunc = r_nlshc,
  vcov.args = list(),
```

```
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'survreg'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = vcov,
  Anova.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'coxph'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = vcov,
  Anova.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'hurdle'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  anova = TRUE,
  vcovfunc = sandwich::sandwich,
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'zeroinfl'
```

```
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  anova = TRUE,
  vcovfunc = sandwich::sandwich,
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'lmrob'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = stats::vcov,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'glmrob'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = stats::vcov,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'geeglm'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
```

```
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  unbiased = TRUE,
  ...
)

## S3 method for class 'glmgee'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  unbiased = TRUE,
  ...
)

## S3 method for class 'gee'
resi_pe(model.full, data, unbiased = TRUE, ...)

## S3 method for class 'lme'
resi_pe(
  model.full,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  Anova.args = list(),
  vcov.args = list(),
  ...
)

## S3 method for class 'lmerMod'
resi_pe(
  model.full,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  ...
)

## S3 method for class 'glmmTMB'
resi_pe(
  model.full,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
```

```
  Anova.args = list(),
  vcov.args = list(),
  ...
)

## S3 method for class 'emmGrid'
resi_pe(object, model, N = NULL, unbiased = TRUE, ...)
```

Arguments

`...`	Ignored.
`model.full`	`lm`, `glm`, `nls`, `survreg`, `coxph`, `hurdle`, `zeroinfl`, `gee`, `geeglm` or `lme` model object.
`model.reduced`	Fitted model object of same type as model.full. By default 'NULL'; the same model as the full model but only having intercept.
`data`	Data.frame or object coercible to data.frame of model.full data (required for some model types).
`anova`	Logical, whether to produce an Anova table with the RESI columns added. By default = 'TRUE'.
`coefficients`	Logical, whether to produce a coefficients (summary) table with the RESI columns added. By default = 'TRUE'.
`overall`	Logical, whether to produce an overall Wald test comparing full to reduced model with RESI columns added. By default = 'TRUE'.
`vcovfunc`	The variance estimator function for constructing the Wald test statistic. By default, sandwich::vcovHC (the robust (sandwich) variance estimator).
`Anova.args`	List, additional arguments to be passed to Anova function.
`vcov.args`	List, additional arguments to be passed to vcovfunc.
`unbiased`	Logical, whether to use the unbiased or alternative T/Z statistic to RESI conversion. By default, 'TRUE'. See details.
`waldtype`	Numeric, indicates which function to use for overall Wald test. 0 (default) = lmtest::waldtest Chi-square, 1 = lmtest::waldtest F, 2 = aod::wald.test
`object`	emmGrid object
`model`	model used to generate emmeans
`N`	sample size

Details

The Robust Effect Size Index (RESI) is an effect size measure based on M-estimators. This function is called by `resi` a specified number of times to form bootstrapped confidence intervals. Called by itself, this function will only calculate point estimates.

The RESI, denoted as S, is applicable across many model types. It is a unitless index and can be easily be compared across models. The RESI can also be converted to Cohen's *d* (`S2d`) under model homoskedasticity.

The RESI is related to the non-centrality parameter of the test statistic. The RESI estimate is consistent for all four (Chi-square, F, T, and Z) types of statistics used. The Chi-square and F-based calculations rely on asymptotic theory, so they may be biased in small samples. When possible, the T and Z statistics are used. There are two formulas for both the T and Z statistic conversion. The first (default, unbiased = TRUE) are based on solving the expected value of the T or Z statistic for the RESI. The alternative is based on squaring the T or Z statistic and using the F or Chi-square statistic conversion. Both of these methods are consistent, but the alternative exhibits a notable amount of finite sample bias. The alternative may be appealing because its absolute value will be equal to the RESI based on the F or Chi-square statistic. The RESI based on the Chi-Square and F statistics is always greater than or equal to 0. The type of statistic used is listed with the output. See `f2S`, `chisq2S`, `t2S`, and `z2S` for more details on the formulas.

For GEE (`geeglm`) models, a longitudinal RESI (L-RESI) and a cross-sectional, per-measurement RESI (CS-RESI) is estimated. The longitudinal RESI takes the specified clustering into account, while the cross-sectional RESI is estimated using a model where each measurement is its own cluster.

Value

Returns a list containing RESI point estimates

Methods (by class)

- `resi_pe(default)`: RESI point estimation
- `resi_pe(glm)`: RESI point estimation for generalized linear models
- `resi_pe(lm)`: RESI point estimation for linear models
- `resi_pe(nls)`: RESI point estimation for nonlinear least squares models
- `resi_pe(survreg)`: RESI point estimation for survreg
- `resi_pe(coxph)`: RESI point estimation for coxph models
- `resi_pe(hurdle)`: RESI point estimation for hurdle models
- `resi_pe(zeroinfl)`: RESI point estimation for zeroinfl models
- `resi_pe(lmrob)`: RESI point estimation for lmrob objects (robustbase package)
- `resi_pe(glmrob)`: RESI point estimation for glmrob objects (robustbase package)
- `resi_pe(geeglm)`: RESI point estimation for geeglm object
- `resi_pe(glmgee)`: RESI point estimation for glmgee object
- `resi_pe(gee)`: RESI point estimation for gee object
- `resi_pe(lme)`: RESI point estimation for lme object
- `resi_pe(lmerMod)`: RESI point estimation for lmerMod object
- `resi_pe(glmmTMB)`: RESI point estimation for glmmTMB object - Gaussian only
- `resi_pe(emmGrid)`: RESI point estimation for emmeans object

References

Vandekar S, Tao R, Blume J. A Robust Effect Size Index. *Psychometrika*. 2020 Mar;85(1):232-246. doi: 10.1007/s11336-020-09698-2.

Examples

```
# This function produces point estimates for the RESI. The resi function will
# provide the same point estimates but adds confidence intervals. See resi for
# more detailed examples.

## resi_pe for a linear model
# fit linear model
mod <- lm(charges ~ region * age + bmi + sex, data = RESI::insurance)
# run resi_pe on the model
resi_pe(mod)

# if you want to have RESI estimates in the coefficient table that are equal in absolute
# value to those in the Anova table (except for those with >1 df and/or included in other
# interaction terms), you can specify unbiased = FALSE to use the alternate conversion.
resi_pe(mod, unbiased = FALSE)
```

---

resi_pe_asymptotic *Robust Effect Size Index with Asymptotic Confidence Intervals*

Description

Computes RESI point estimates and asymptotic confidence intervals using either a normal approximation (Zhang et al., 2025) or a quadratic-form (Imhof/Davies) approach.

Usage

```
resi_pe_asymptotic(
  model.full,
  data,
  vcovfunc = sandwich::vcovHC,
  coefficients = TRUE,
  anova = TRUE,
  alpha = 0.05,
  ci.method = c("qf", "normal", "cf"),
  type = "HC3",
  unbiased = TRUE,
  Anova.args = list(),
  vcov.args = list(),
  ...
)
```

Arguments

model.full	Fitted lm or glm model object.
data	Data frame of model data.
vcovfunc	Variance estimator for RESI point estimates. Default: sandwich::vcovHC.
coefficients	Logical; include coefficient table. Default TRUE.

anova	Logical; include anova table. Default TRUE.
alpha	Numeric; significance level. Default 0.05.
ci.method	Character; "qf" (quadratic-form Imhof, default), "normal" (truncated normal), or "cf" (Cornish-Fisher test inversion).
type	Character; HC type for the sandwich variance used in CI construction (controls tau_i weights in SigmaXw). Default "HC3". Supported: "HC0"–"HC5", "const".
unbiased	Logical; use bias-corrected RESI point estimate. Default TRUE.
Anova.args	List; additional arguments passed to car::Anova.
vcov.args	List; additional arguments passed to vcovfunc.
...	Ignored.

Value

A list of class "resi" with coefficients and/or anova tables containing RESI point estimates and CIs.

Rsq2S *Covert R^2 to S*

Description

Converts R^2, the partial coefficient of determination, to robust effect size index (S) using the formula from Vandekar, Tao, & Blume (2020).

Usage

```
Rsq2S(Rsq)
```

Arguments

Rsq	Numeric, R^2

Details

The formula for the conversion is:

$S = \sqrt{((-R^2)/(R^2 - 1))}$

Value

Returns an estimate of R^2 based on the RESI

Examples

```
# consider a moderate effect size of R^2 = 0.1
Rsq2S(0.1)
# this corresponds to a RESI of 0.333
```

`S2chisq` *Convert non-zero S to Chi-square statistic*

Description

Converts the robust effect size index (S) to Chi-square statistic, given that S is greater than 0. For an S value of 0, only an upper bound on the Chi-square statistic can be computed. Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

Usage

```
S2chisq(S, df, n)
```

Arguments

`S`	The value of the RESI estimate.
`df`	Number of degrees of freedom of the chi-square statistic.
`n`	Number of independent samples.

Details

The formula for converting a RESI estimate above 0 to Chi-square statistic is:

$chisq = n * S^2 + df$

If the RESI estimate is 0, all that is known is that the Chi-square statistic is less than or equal to the degrees of freedom.

Value

Returns a scalar or vector argument of the Chi-square statistic.

Examples

```
# convert S estimates with corresponding degrees of freedom to Chi-square estimates
S_ests = c(0.2, 0.4, 0.6)
dfs = c(2, 1, 3)
S2chisq(S = S_ests, df = dfs, n = 300)
```

S2d *Convert S to Cohen's d*

Description

Converts the robust effect size index (S) to Cohen's d using the formula from Vandekar, Tao, & Blume (2020).

Usage

```
S2d(S, pi = 0.5)
```

Arguments

S	Numeric, the robust effect size index.
pi	Numeric, the sampling proportions.

Details

The pi parameter comes from the fact that Cohen's d doesn't account for unequal sample proportions in the population, but S does.

The default is set to a natural value 1/2, which corresponds to a case control design, for example, where sampling proportions always are controlled by the experimenter.

The formula for the conversion is:

$d = |S * \sqrt{(1/\pi + 1/(1-\pi))}|$

Value

Returns an estimate of Cohen's *d* based on the RESI.

Examples

```
# fit a simple linear regression with a binary predictor
mod = lm(charges ~ sex, data = RESI::insurance)

# calculate t-value
t = summary(mod)$coefficients[2, "t value"]

# calculate RESI (S)
S = t2S(t, n = 1338, rdf = 1336)

# determine sample proportions
pi = length(which(RESI::insurance[,"sex"]=="male"))/1338

# convert S to Cohen's d
S2d(S = S, pi = pi)
```

S2fsq *Covert S to Cohen's* f^2

Description

Converts robust effect size index (S) to Cohen's f^2 (effect size for multiple regression) using the formula from Vandekar, Tao, & Blume (2020).

Usage

```
S2fsq(S)
```

Arguments

S	Numeric,the robust effect size index.

Details

The formula for the conversion is:

$f^2 = S^2$

Value

Returns an estimate of Cohen's f^2 based on the RESI

Examples

```
# fit a linear regression model with continuous outcome and predictor
mod = lm(charges ~ age, data = RESI::insurance)

# obtain t value for calculating RESI
t = summary(mod)$coefficients[2, "t value"]

# calculate RESI
S = t2S(t, n = 1338, rdf = 1336)

# convert to f^2
S2fsq(S)
```

S2Rsq *Covert S to R^2*

Description

Converts robust effect size index (S) to R^2, the partial coefficient of determination, using the formula from Vandekar, Tao, & Blume (2020).

Usage

```
S2Rsq(S)
```

Arguments

S	Numeric, the robust effect size index.

Details

The formula for the conversion is:

$R^2 = S^2/(1 + S^2)$

Value

Returns an estimate of R^2 based on the RESI

Examples

```
# fit a simple linear regression with a binary predictor
mod = lm(charges ~ sex, data = RESI::insurance)

# calculate t-value
t = summary(mod)$coefficients[2, "t value"]

# calculate RESI (S)
S = t2S(t, n = 1338, rdf = 1336)

# convert S to R^2
S2Rsq(S)
```

S2z *Convert RESI (S) estimate to Z statistic*

Description

Converts the robust effect size index (S) to Z statistic. Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

Usage

```
S2z(S, n, unbiased = TRUE)
```

Arguments

S	The value of the RESI estimate.
n	Number of independent samples.
unbiased	Logical, whether the unbiased or alternative estimator was used to compute RESI estimate. Default is TRUE.

Details

The formula for converting a RESI estimate to a corresponding Z statistic depends on which estimator was used to compute the RESI estimate (unbiased vs. alternative, see z2S). For the unbiased estimator, the RESI can be positive or negative and there is a 1-1 transformation from S to Z. The formula for converting S (unbiased) to the Z statistic is:

$\sqrt(n) * S$

For the alternative formula, if the RESI estimate is 0, the Z statistic is only known within an interval, [-1, 1]. For a non-zero S, the formula is:

$\sqrt{S^2}/S\sqrt(n * abs(S) + 1)$

Value

Returns a scalar or vector argument of the Chi-square statistic.

Examples

```
# convert S estimates with corresponding degrees of freedom to
# Z statistics estimates (using unbiased formula)
S_ests = c(-0.2, 0, 0.1)
S2z(S = S_ests, n = 300, unbiased = TRUE)

# convert S estimates with corresponding degrees of freedom to
# Z statistics estimates (using alernative formula)
S_ests = c(-0.2, 0, 0.1)
S2z(S = S_ests, n = 300, unbiased = FALSE)
```

simCalibrationFigures *Calibration Figures for RESI Variance Estimates*

Description

Reads output from simCalibrationSim and produces one PDF per model setting showing convergence of point estimates, covariance estimates, RESI estimates, and RESI variance estimates to their population targets.

Usage

```
simCalibrationFigures(
  output.dir = "resiCalibrationSim",
  figures.dir = NULL,
  alpha = 0.05,
 deriv_method = c("extended", "corrected", "original", "population", "population2",
    "zeroB", "zero_phi_cross")
)
```

Arguments

output.dir	Character, directory containing output from simCalibrationSim. Default "resiCalibrationSim".
figures.dir	Character, output directory for PDFs. Default file.path(output.dir, "figures").
alpha	Numeric, nominal CI level. Default 0.05.
deriv_method	Character, type of derivative being evaluated. Package now only includes "extended" in resi implementation.

Value

Invisibly returns the summary data frame. Saves PDF figures to figures.dir.

See Also

simCalibrationSim, insurancePlasmodeSim

simCalibrationSim *Asymptotic Calibration Check for RESI Variance Estimates*

Description

Runs a plasmode simulation to verify that the asymptotic normal CI machinery is correctly calibrated for all four model settings (lm/glm x parametric/robust). For each (setting, sample size) cell the function checks:

1. **Bias(theta)**: $\hat{\theta} \to \theta_{\text{true}}$ – raw coefficients (and $\phi = \hat{\sigma}^2$ for `lm`) converge to the full-dataset values.
2. **vcov check A** (estimator consistency): $n \cdot \bar{V}_{\text{analytic}}/(n_{\text{full}} \cdot V_{\text{true}}) \to 1$.
3. **vcov check B** (calibration): $n \cdot \widehat{\text{Var}}(\hat{\beta})/(n_{\text{full}} \cdot V_{\text{true}}) \to 1$.
4. **Bias(R)**: RESI point estimates converge to the full-dataset values.
5. **sigma2S check A** (estimator consistency): $\bar{\hat{\sigma}}^2_S/\sigma^2_{S,\text{true}} \to 1$.
6. **sigma2S check B** (CI calibration): $n \cdot \widehat{\text{Var}}(\hat{R})/\sigma^2_{S,\text{true}} \to 1$.

True values are taken from the full `insurance` dataset using the same definitions as `insurancePlasmodeSim`. σ^2_S is extracted from the asymptotic-normal CI half-width: $\hat{\sigma}^2_S = n \cdot (\text{hw}/z_{\alpha/2})^2$.

Usage

```
simCalibrationSim(
  nsim = 500L,
  n.vec = c(100L, 200L, 500L, 1000L, 2000L, 5000L),
  alpha = 0.05,
  output.dir = "resiCalibrationSim",
  fixed.knots = FALSE,
 deriv_method = c("corrected", "original", "population", "population2", "zeroB",
    "zero_phi_cross", "extended"),
  mc.cores.reps = 1L
)
```

Arguments

`nsim`	Integer, replicates per (setting, n) cell. Default 500.
`n.vec`	Integer vector of sample sizes. Default `c(100, 200, 500, 1000, 2000, 5000)`.
`alpha`	Numeric, nominal CI level. Default 0.05.
`output.dir`	Character, directory for raw per-cell RDS files. Default `"resiCalibrationSim"`.
`fixed.knots`	Logical. Fix spline knots at full-dataset quantiles. Default `FALSE`.
`deriv_method`	Character, type of derivative being evaluated. Package now only includes "extended" in resi implementation.
`mc.cores.reps`	Integer, cores for within-cell parallelism. Default 1.

Value

Invisibly returns the combined metrics `data.frame`.

See Also

`insurancePlasmodeSim`, `simCompareMethodsFigures`

simCompareMethodsFigures
Per-Term CI Method Comparison Figures

Description

Reads simulation summary tables from multiple output directories (one per CI method) and produces PDF figures comparing CI methods across sample sizes. For each (model type x variance estimator x table) combination two PDFs are saved: an estimator comparison (Bias, MSE) and a CI comparison (SE calibration, coverage, width).

Usage

```
simCompareMethodsFigures(
  output.dirs = c(Bootstrap = "resiBootSim", Normal = "resiAsympNormalSim", QF =
    "resiAsympQFSim", CF = "resiAsympCFSim"),
  figures.dir = NULL,
  alpha = 0.05,
  fixed.knots = FALSE
)
```

Arguments

output.dirs	Named character vector mapping CI method labels to their simulation output directories. Default: c(boot = "resiBootSim", normal = "resiAsympNormalSim",qf = "resiAsympQFSim", cf = "resiAsympCFSim"). Directories that do not exist are silently skipped.
figures.dir	Character, directory where comparison figures are saved. Default: file.path(output.dirs[1], "figures", "method_comparison").
alpha	Numeric, nominal CI level used for coverage reference lines. Default 0.05.
fixed.knots	Logical. Must match the value used in the original insurancePlasmodeSim call. Default FALSE.

Value

Invisibly returns the combined summary data.frame. Saves estimator_<model>_<vcov>_<table>.pdf, compare_<model>_<vcov>_<table>.pdf, and covquant_<model>_<vcov>_<table>.pdf to figures.dir.

See Also

insurancePlasmodeSim, simFigures, simEstimatorFigures

`simEstimatorFigures` *Estimator Comparison Figures: t2S/f2S vs z2S/chisq2S*

Description

Reads the per-replicate raw `.rds` files from one simulation directory and produces Bias + MSE comparison figures contrasting the RESI estimators actually used for `lm` models against the simpler z-statistic / chi-squared alternatives:

- **Coefficients table**: `t2S` (used) vs `z2S` (alternative)
- **Anova table**: `f2S` (used) vs `chisq2S` (alternative)

Only `lm` models are included; GLM models are skipped because z2S and chisq2S are the natural estimators there.

Usage

```
simEstimatorFigures(
  sim.dir = "resiBootSim",
  figures.dir = NULL,
  alpha = 0.05,
  fixed.knots = FALSE
)
```

Arguments

`sim.dir`	Character, directory containing simulation output with a `sim_raw/` sub-directory. Default `"resiBootSim"`.
`figures.dir`	Character, output directory for PDFs. Default `file.path(sim.dir, "figures", "estimator_compare")`.
`alpha`	Numeric, nominal level (used only in figure titles). Default 0.05.
`fixed.knots`	Logical. Must match the value used in the original `insurancePlasmodeSim` call. Default `FALSE`.

Value

Invisibly returns the combined estimator-comparison `data.frame`.

See Also

`simCompareMethodsFigures`, `simRecomputeSummary`

`simFigures` *Simulation Performance Figures for RESI Evaluation*

Description

Produces performance figures from the output of `insurancePlasmodeSim`. Creates one PDF figure per (model type) $\times$ (variance estimator) combination (4 figures total by default). Each figure is a 2 $\times$ 4 panel grid:

- **Top row**: Anova-table RESI metrics (Bias, MSE, CI Coverage, CI Width)
- **Bottom row**: Coefficients-table RESI metrics (same metrics; intercept excluded)
- **Colors**: one colored line per model term (high-contrast matte palette)
- **x-axis**: sample size on a log scale

Dashed reference lines are drawn at zero for Bias and at $1 - \alpha$ for CI Coverage.

Usage

```
simFigures(output.dir = "resiBootSim", alpha = 0.05, ci.label = NULL)
```

Arguments

`output.dir`	Character, directory containing simulation output from `insurancePlasmodeSim`. Default `"resiBootSim"`.
`alpha`	Numeric, nominal CI level used for the coverage reference line. Default 0.05.
`ci.label`	Character, label for the CI method used in figure titles and file names. Default `NULL`, which auto-detects from `output.dir`: `"boot"` if the directory name contains "Boot"/"boot", `"normal"` if it contains "Normal"/"normal", `"qf"` if it contains "QF"/"qf", otherwise `basename(output.dir)`.

Value

Invisibly returns the summary metrics `data.frame`. Saves PDF figures to `file.path(output.dir, "figures")`, named `sim_<model>_<vcov>_<ci.label>.pdf`.

See Also

`insurancePlasmodeSim`

simRecomputeSummary *Recompute Summary Table from Raw Simulation Output*

Description

Reads the per-replicate raw `.rds` files saved by `insurancePlasmodeSim` and recomputes the summary metrics table (bias, MSE, coverage, upper_coverage, lower_coverage, width). Use this whenever `.simComputeMetrics` has been updated (e.g. new metrics added) without needing to rerun the full simulation.

Usage

```
simRecomputeSummary(
  output.dir = "resiBootSim",
  alpha = 0.05,
  fixed.knots = FALSE
)
```

Arguments

`output.dir`	Character, directory containing simulation output. Default `"resiBootSim"`.
`alpha`	Numeric, CI significance level used in the original simulation. Default 0.05.
`fixed.knots`	Logical. Must match the value used in the original `insurancePlasmodeSim` call so that the true RESI values are computed from the same model formula. Default `FALSE`.

Details

True RESI values are re-estimated from the full `insurance` dataset using the same formulae and variance functions as the original simulation.

Value

Invisibly returns the updated summary `data.frame`. Overwrites `output.dir/summary_table.rds`.

See Also

`insurancePlasmodeSim`, `simFigures`

summary.resi *Summary method for resi objects*

Description

After running the resi function on a fitted model, this function can be used to print the coefficients table component. If the resi function was run with the 'store.boot = TRUE' option to store the full matrix of bootstrapped estimates, the user can specify a different alpha level for this function's confidence intervals.

Usage

```
## S3 method for class 'resi'
summary(object, alpha = NULL, ...)
```

Arguments

object	an object resulting from resi function
alpha	an optional new specification for the confidence level. Can be vector-valued
...	ignored

Value

Returns a 'summary_resi' object containing the computed coefficients table

Examples

```
# fit a model
mod = lm(charges ~ bmi + sex, data = RESI::insurance)

# run resi with the store.boot = TRUE option
resi_obj = resi(mod, nboot = 100, store.boot = TRUE, alpha = 0.01)

# run summary, specifying a different alpha level if desired
summary(resi_obj, alpha = 0.05)
```

t2S *Compute the robust effect size index estimate from t statistic (default)*

Description

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

Usage

```
t2S(t, rdf, n, unbiased = TRUE)
```

Arguments

t	The t statistic for the parameter of interest.
rdf	Model residual degrees of freedom/degrees of freedom of the t statistic.
n	Number of independent samples.
unbiased	Logical, whether to use unbiased or alternative estimator. See details.

Details

This function computes S, the RESI, from a t statistic. The formula for the unbiased estimator (default) is derived by solving the expected value of the t statistic for S. It is unbiased and consistent.

The formula for the unbiased conversion is:

$S = (t * \sqrt{(}2) * \Gamma(rdf/2))/(\sqrt{(}n * rdf) * \Gamma((rdf - 1)/2))$

The formula for the alternative estimator is derived by squaring the t statistic and using the f2S formula. This estimator may be appealing for its intuitive relationship to the F statistic; the absolute value of RESI estimates using this formula will be equal to a RESI estimate using an F statistic for the same model. However, this estimator does have finite sample bias, which is an important consideration for the coverage of the bootstrapping that resi uses.

The formula for the alternative conversion is:

$\sqrt{(}max(0, (t^2 * (rdf - 2)/rdf - 1)/rdf))$

Value

Returns a scalar or vector argument of the robust effect size index estimate.

Examples

```
# to obtain t values, first fit a lm
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)
# run lmtest::coeftest to get t values, using a robust variance-covariance formula
ts = lmtest::coeftest(mod, vcov. = sandwich::vcovHC)[,'t value']

# get RESI estimates using unbiased estimator
t2S(ts, n = nrow(RESI::insurance), rdf = mod$df.residual)

# get RESI estimates using alternative estimator
t2S(ts, n = nrow(RESI::insurance), rdf = mod$df.residual, unbiased = FALSE)
```

z2S *Compute the robust effect size index estimate from Z statistic*

Description

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

Usage

```
z2S(z, n, unbiased = TRUE)
```

Arguments

z	The Z statistic for the parameter of interest.
n	Number of independent samples.
unbiased	Logical, whether to use unbiased or alternative estimator. See details.

Details

This function computes S, the RESI, from a Z statistic. The formula for the unbiased estimator (default) is derived by solving the expected value of the Z statistic for S. It is unbiased and consistent.

The formula for the unbiased conversion is:

$S = Z/\sqrt{(n)}$

The formula for the alternative estimator is derived by squaring the Z statistic and using the chisq2S formula. This estimator may be appealing for its intuitive relationship to the Chi-square statistic; the absolute value of RESI estimates using this formula will be equal to a RESI estimate using a Chi-square statistic for the same model. However, this estimator does have finite sample bias, which is an important consideration for the coverage of the bootstrapping that resi uses.

The formula for the alternative conversion is:

$\sqrt{(max(0, (Z^2 - 1)/n))} * sign(Z)$

Value

Returns a scalar or vector argument of the robust effect size index estimate.

Examples

```
# to obtain example z values, first fit a glm
mod = glm(charges ~ region * age + bmi + sex, data = RESI::insurance)
# run coeftest to get z values using a robust variance-covariance function
zs = lmtest::coeftest(mod, vcov. = sandwich::vcovHC)[,'z value']

# get RESI estimates using unbiased estimator
z2S(zs, n = nrow(RESI::insurance))
```

```
# get RESI estimates usng alternative estimator
z2S(zs, n = nrow(RESI::insurance), unbiased = FALSE)
```

Index