

Package ‘metafor’

September 15, 2026

Version 5.2-1

Date 2026-09-15

Title Meta-Analysis Package for R

Depends R (>= 4.0.0), methods, Matrix, metadat, numDeriv

Imports stats, utils, graphics, grDevices, nlme, mathjaxr, pbapply, digest

Suggests lme4, pracma, minqa, nloptr, dfoptim, ucminf, lbfgsb3c, subplex, BB, Rsolnp, alabama, optimParallel, optimx, CompQuadForm, mvtnorm, BiasedUrn, Epi, survival, GLM-Madaptive, glmmTMB, car, multcomp, gsl, sp, ape, boot, clubSandwich, crayon, R.rsp, testthat, rmarkdown, wildmeta, emmeans, estmeansd, metaBLUE, rstudioapi, glmulti, MuMIn, mice, Amelia, calculus

Description A comprehensive collection of functions for conducting meta-analyses in R. The package includes functions to calculate various effect sizes or outcome measures, fit equal-, fixed-, random-, and mixed-effects models to such data, carry out moderator and meta-regression analyses, and create various types of meta-analytical plots (e.g., forest, funnel, radial, L'Abbe, Baujat, bubble, and GOSH plots). For meta-analyses of binomial and person-time data, the package also provides functions that implement specialized methods, including the Mantel-Haenszel method, Peto's method, and a variety of suitable generalized linear (mixed-effects) models (i.e., mixed-effects logistic and Poisson regression models). Finally, the package provides functionality for fitting meta-analytic multivariate/multilevel models that account for non-independent sampling errors and/or true effects (e.g., due to the inclusion of multiple treatment studies, multiple endpoints, or other forms of clustering). Network meta-analyses and meta-analyses accounting for known correlation structures (e.g., due to phylogenetic relatedness) can also be conducted. An introduction to the package can be found in Viechtbauer (2010) <[doi:10.18637/jss.v036.i03](https://doi.org/10.18637/jss.v036.i03)>.

License GPL (>=2)

ByteCompile TRUE

Encoding UTF-8

RdMacros mathjaxr

VignetteBuilder R.rsp

BuildManual TRUE

URL <https://www.metafor-project.org> <https://github.com/wviechtb/metafor> <https://www.vvbauer.com>

BugReports <https://github.com/wviechtb/metafor/issues>

Contents

metafor-package	4
addpoly	9
addpoly.default	10
addpoly.predict.rma	12
addpoly.rma	14
aggregate.escalc	17
anova.rma	21
baujat	27
bldiag	29
blsplit	31
blup	32
cmicalc	34
coef.deltamethod	36
coef.matreg	37
coef.permutest.rma.uni	39
coef.rma	40
confint.rma	42
contrmat	48
conv.2x2	49
conv.delta	55
conv.fivenum	60
conv.wald	66
cumul	72
deltamethod	75
dfround	79
emmprep	80
escalc	83
fitstats	108
fitted.rma	110
forest	111
forest.cumul.rma	112
forest.default	116
forest.rma	122
formatters	133
formula.rma	136
fsn	137
funnel	141
gosh	147
hc	150
hettest	152
influence.rma.mv	155
influence.rma.uni	158
labbe	161
leave1out	164
llplot	167
matreg	169

metafor.news	175
methods.vcovmat	176
mfopt	177
misc-models	178
misc-options	183
misc-recs	189
model.matrix.rma	192
pairmat	193
permutest	195
plot.cumul.rma	200
plot.gosh.rma	202
plot.infl.rma.uni	204
plot.permutest.rma.uni	207
plot.rma	210
plot.rma.uni.selmodel	211
plot.vif.rma	214
predict.matreg	216
predict.rma	218
print.anova.rma	223
print.confint.rma	225
print.deltamethod	226
print.escalc	227
print.fsn	229
print.gosh.rma	230
print.hc.rma.uni	231
print.list.rma	232
print.matreg	233
print.permutest.rma.uni	234
print.ranktest	235
print.regtest	236
print.rma	237
profile.rma	241
qqnorm.rma	246
radial	249
ranef	252
ranktest	254
rcalc	256
regplot	260
regtest	266
replmiss	271
reporter	272
residuals.rma	274
rma.glmm	278
rma.mh	288
rma.mv	293
rma.peto	307
rma.uni	311
robust	325

se	330
selmodel	331
simulate.rma	349
tes	351
to.long	354
to.table	357
to.wide	360
transf	362
trimfill	369
update.rma	372
vcalc	373
vcov.rma	380
vec2mat	382
vif	383
weights.rma	389
Index	391

 metafor-package

metafor: A Meta-Analysis Package for R

Description

The **metafor** package provides a comprehensive collection of functions for conducting meta-analyses in R. The package can be used to calculate a wide variety of effect size measures and allows the user to fit equal-, fixed-, and random-effects models to these data. By including study-level variables ('moderators') as predictors in these models, (mixed-effects) meta-regression models can also be fitted. For meta-analyses of 2×2 tables, proportions, incidence rates, and incidence rate ratios, the package also provides specialized methods, including the Mantel-Haenszel method, Peto's method, and a variety of suitable generalized linear mixed-effects models (i.e., mixed-effects logistic and Poisson regression models). For non-independent estimates (e.g., due to correlated sampling errors, correlated true effects, or other forms of clustering), one can fit multilevel and multivariate models.

Various methods are available to assess model fit, to identify outliers and/or influential studies, and for conducting sensitivity analyses (e.g., standardized residuals, Cook's distances, leave-one-out analyses). Advanced techniques for hypothesis testing and obtaining confidence intervals (e.g., for the average true effect or for the model coefficients in a meta-regression model) have also been implemented (e.g., the Knapp and Hartung method, permutation tests, cluster-robust inference methods / robust variance estimation).

The package also provides functions for creating forest, funnel, radial, normal quantile-quantile, L'Abbé, Baujat, bubble, and GOSH plots. The presence of publication bias (or more precisely, funnel plot asymmetry or 'small-study effects') and its potential impact on the results can be examined via the rank correlation and Egger's regression test, the trim and fill method, the test of excess significance, and by applying a variety of selection models.

The `escalc` Function

[`escalc`] Before a meta-analysis can be conducted, the relevant results from each study must be quantified in such a way that the resulting values can be further aggregated and compared. The `escalc` function can be used to compute a wide variety of effect sizes or ‘outcome measures’ (and the corresponding sampling variances) that are often used in meta-analyses (e.g., risk ratios, odds ratios, risk differences, mean differences, standardized mean differences, response ratios / ratios of means, raw or r-to-z transformed correlation coefficients). Measures for quantifying some characteristic of individual groups (e.g., in terms of means, proportions, or incidence rates and transformations thereof), measures of change (e.g., raw and standardized mean changes), and measures of variability (e.g., variability ratios and coefficient of variation ratios) are also available.

The `rma.uni` Function

[`rma.uni`] The various meta-analytic models that are typically used in practice are special cases of the general linear (mixed-effects) model. The `rma.uni` function (with alias `rma`) provides a general framework for fitting such models. The function can be used in combination with any of the measures computed with the `escalc` function or, more generally, any set of estimates (with corresponding sampling variances or standard errors) one would like to analyze. The notation and models underlying the `rma.uni` function are explained below.

For a set of $i = 1, \dots, k$ independent studies, let y_i denote the effect size estimate in the i th study. Let θ_i denote the corresponding (unknown) true effect, such that

$$y_i \mid \theta_i \sim N(\theta_i, v_i).$$

In other words, the estimates are assumed to be unbiased and normally distributed estimates of the corresponding true effects with sampling variances equal to v_i (where v_i is the square of the standard errors of the estimates). The v_i values are assumed to be known. Depending on the effect size measure used, a bias correction, normalizing, and/or variance stabilizing transformation may be necessary to ensure that these assumptions are (at least approximately) true (e.g., the log transformation for odds/risk ratios, the bias correction for standardized mean differences, Fisher’s r-to-z transformation for correlations; see `escalc` for further details).

According to the **random-effects model**, we assume that $\theta_i \sim N(\mu, \tau^2)$, that is, the true effects are normally distributed with μ denoting the average true effect and τ^2 the variance in the true effects (τ^2 is therefore often referred to as the amount of ‘heterogeneity’ in the true effects or the ‘between-study variance’). The random-effects model can also be written as

$$y_i = \mu + u_i + \varepsilon_i,$$

where $u_i \sim N(0, \tau^2)$ and $\varepsilon_i \sim N(0, v_i)$. The fitted model provides estimates of μ and τ^2 , that is,

$$\hat{\mu} = \frac{\sum_{i=1}^k w_i y_i}{\sum_{i=1}^k w_i},$$

where $w_i = 1/(\hat{\tau}^2 + v_i)$ and $\hat{\tau}^2$ denotes an estimate of τ^2 obtained with one of the many estimators that have described in the literature for this purpose (this is sometimes called the standard ‘inverse-variance’ method for random-effects models or the ‘normal-normal’ model).

A special case of the model above is the **equal-effects model** (also sometimes called the common-effect(s) model) which arises when $\tau^2 = 0$. In this case, the true effects are homogeneous (i.e.,

$\theta_1 = \theta_2 = \dots = \theta_k \equiv \theta$) and hence we can write the model as

$$y_i = \theta + \varepsilon_i,$$

where θ denotes *the* true effect in the studies, which is estimated with

$$\hat{\theta} = \frac{\sum_{i=1}^k w_i y_i}{\sum_{i=1}^k w_i},$$

where $w_i = 1/v_i$ (again, this is the standard ‘inverse-variance’ method as described in the meta-analytic literature). Note that the commonly-used term ‘fixed-effects model’ is not used here – for an explanation, see [here](#).

Study-level variables (often referred to as ‘moderators’) can also be included as predictors in meta-analytic models, leading to so-called ‘meta-regression’ models (to examine whether the effects tend to be larger/smaller under certain conditions or circumstances). When including moderator variables in a random-effects model, we obtain a **mixed-effects meta-regression model**. This model can be written as

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_{p'} x_{ip'} + u_i + \varepsilon_i,$$

where $u_i \sim N(0, \tau^2)$ and $\varepsilon_i \sim N(0, v_i)$ as before and x_{ij} denotes the value of the j th moderator variable for the i th study (letting $p = p' + 1$ denote the total number of coefficients in the model including the model intercept). Therefore, β_j denotes how much the average true effect differs for studies that differ by one unit on x_{ij} and the model intercept β_0 denotes the average true effect when the values of all moderator variables are equal to zero. The value of τ^2 in the mixed-effects model denotes the amount of ‘residual heterogeneity’ in the true effects (i.e., the amount of variability in the true effects that is not accounted for by the moderators included in the model). In matrix notation, the model can also be written as

$$y = X\beta + u + \varepsilon,$$

where y is a $k \times 1$ column vector with the effect size estimates, X is the $k \times p$ model matrix (with the first column equal to 1s for the intercept term), β is a $p \times 1$ column vector with the model coefficients, and u and ε are $k \times 1$ column vectors for the random effects and sampling errors, where $\text{Var}[\varepsilon]$ is a $k \times k$ diagonal matrix with the v_i values along the diagonal and $\text{Var}[u] = \tau^2 I$, where I is a $k \times k$ identity matrix.

The `rma.mh` Function

[\[rma.mh\]](#) The Mantel-Haenszel method provides an alternative approach for fitting equal-effects models when dealing with studies providing data in the form of 2×2 tables or in the form of event counts (i.e., person-time data) for two groups (Mantel & Haenszel, 1959). The method is particularly advantageous when aggregating a large number of studies with small sample sizes (the so-called sparse data or increasing strata case). The Mantel-Haenszel method is implemented in the `rma.mh` function. It can be used in combination with risk ratios, odds ratios, risk differences, incidence rate ratios, and incidence rate differences.

The `rma.peto` Function

[\[rma.peto\]](#) Yet another method that can be used in the context of a meta-analysis of 2×2 table data is Peto’s method (see Yusuf et al., 1985), implemented in the `rma.peto` function. The method

provides an estimate of the (log) odds ratio under an equal-effects model. The method is particularly advantageous when the event of interest is rare, but see the documentation of the function for some caveats.

The `rma.glmm` Function

[`rma.glmm`] Dichotomous response variables and event counts (based on which one can calculate measures such as odds ratios, incidence rate ratios, proportions, and incidence rates) are often assumed to arise from binomial and Poisson distributed data. Meta-analytic models that are directly based on such distributions are implemented in the `rma.glmm` function. These models are essentially special cases of generalized linear mixed-effects models (i.e., mixed-effects logistic and Poisson regression models). For 2×2 table data, a mixed-effects conditional logistic model (based on the non-central hypergeometric distribution) is also available. Random/mixed-effects models with dichotomous data are often referred to as ‘binomial-normal’ models in the meta-analytic literature. Analogously, for event count data, such models could be referred to as ‘Poisson-normal’ models.

The `rma.mv` Function

[`rma.mv`] Standard meta-analytic models assume independence between the effect size estimates obtained from a set of studies. This assumption is often violated in practice. Dependencies can arise for a variety of reasons. For example, the sampling errors and/or true effects may be correlated in multiple treatment studies (e.g., when multiple treatment groups are compared with a common control/reference group, such that the data from the control/reference group is used multiple times to compute the estimates) or in multiple endpoint studies (e.g., when more than one effect size estimate is calculated based on the same sample of subjects due to the use of multiple endpoints or response variables). Correlation among the true effects can also arise due to other forms of clustering (e.g., when multiple effects derived from the same author, lab, or research group may be more similar to each other than effects derived from different authors, labs, or research groups). In ecology and related fields, the shared phylogenetic history of the organisms studied (e.g., plants, fungi, animals) can also induce correlation among the effects. The `rma.mv` function can be used to fit suitable meta-analytic multivariate/multilevel models to such data, so that the non-independence in the effects is accounted for. Network meta-analyses (also called multiple/mixed treatment comparisons) can also be carried out with this function.

Future Plans and Updates

The **metafor** package is a work in progress and is updated on a regular basis with new functions and options. The development version of the package can be found on GitHub at <https://github.com/wviechtb/metafor> and can be installed with:

```
install.packages("remotes")
remotes::install_github("wviechtb/metafor")
```

With `metafor.news()`, you can read the ‘NEWS’ file of the package after installation. Comments, feedback, and suggestions for improvements are always welcome.

Citing the Package

To cite the package, please use the following reference:

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1-48. doi:[10.18637/jss.v036.i03](https://doi.org/10.18637/jss.v036.i03)

Getting Started with the Package

The paper mentioned above is a good starting place for those interested in using the package. The purpose of the article is to provide a general overview of the package and its capabilities (as of version 1.4-0). Not all of the functions and options are described in the paper, but it should provide a useful introduction to the package. The paper can be freely downloaded from the URL given above or can be directly loaded with the command `vignette("metafor")`.

In addition to reading the paper, carefully read this page and then the help pages for the `escalc` and the `rma.uni` functions (or the `rma.mh`, `rma.peto`, `rma.glmm`, and/or `rma.mv` functions if you intend to use these models/methods). The help pages for these functions provide links to many additional functions, which can be used after fitting a model. You can also read the entire documentation online at <https://www.viechtb.github.io/metafor/> (where it is nicely formatted and the output from all examples is provided).

A (pdf) diagram showing the various functions in the metafor package (and how they are related to each other) can be opened with the command `vignette("diagram", package="metafor")`.

Finally, additional information about the package, several detailed analysis examples, examples of plots and figures provided by the package (with the corresponding code), some additional tips and notes, and a FAQ can be found on the package website at <https://www.metafor-project.org>.

Author(s)

Wolfgang Viechtbauer <wvb@metafor-project.org>
package website: <https://www.metafor-project.org>
author homepage: <https://www.wvbauer.com>

Suggestions on how to obtain help with using the package can found on the package website at: <https://www.metafor-project.org/doku.php/help>

References

- Cooper, H., Hedges, L. V., & Valentine, J. C. (Eds.) (2009). *The handbook of research synthesis and meta-analysis* (2nd ed.). New York: Russell Sage Foundation.
- Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. San Diego, CA: Academic Press.
- Mantel, N., & Haenszel, W. (1959). Statistical aspects of the analysis of data from retrospective studies of disease. *Journal of the National Cancer Institute*, **22**(4), 719-748. <https://doi.org/10.1093/jnci/22.4.719>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1-48. <https://doi.org/10.18637/jss.v036.i03>
- Yusuf, S., Peto, R., Lewis, J., Collins, R., & Sleight, P. (1985). Beta blockade during and after myocardial infarction: An overview of the randomized trials. *Progress in Cardiovascular Disease*, **27**(5), 335-371. [https://doi.org/10.1016/s0033-0620\(85\)80003-7](https://doi.org/10.1016/s0033-0620(85)80003-7)

Description

Function to add polygons (sometimes called ‘diamonds’) to a forest plot, for example to show pooled estimates for subgroups of studies or to show fitted/predicted values based on models involving moderators.

Usage

```
addpoly(x, ...)
```

Arguments

x	either an object of class "rma", an object of class "predict.rma", or the values at which polygons should be drawn. See ‘Details’.
...	other arguments.

Details

Currently, methods exist for three types of situations.

In the first case, object x is a fitted model coming from the [rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), or [rma.mv](#) functions. The model must either be an equal- or a random-effects model, that is, the model should not contain any moderators. The corresponding method is [addpoly.rma](#). It can be used to add a polygon to an existing forest plot (usually at the bottom), showing the pooled estimate (with its confidence interval) based on the fitted model.

Alternatively, x can be an object of class "predict.rma" obtained with the [predict](#) function. In this case, polygons based on the predicted values are drawn. The corresponding method is [addpoly.predict.rma](#).

Alternatively, object x can be a vector with the values at which one or more polygons should be drawn. The corresponding method is [addpoly.default](#).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[addpoly.rma](#), [addpoly.predict.rma](#), and [addpoly.default](#) for the specific method functions.
[forest](#) for functions to draw forest plots to which polygons can be added.

addpoly.default

*Add Polygons to Forest Plots (Default Method)***Description**

Function to add one or more polygons to a forest plot.

Usage

```
## Default S3 method:
addpoly(x, vi, sei, ci.lb, ci.ub, pi.lb, pi.ub,
        rows=-1, level, annotate, predstyle, predlim, digits, width, mlab,
        transf, atransf, targs, efac, col, border, lty, fonts, cex,
        constarea=FALSE, ...)
```

Arguments

<code>x</code>	vector with the values at which the polygons should be drawn.
<code>vi</code>	vector with the corresponding variances.
<code>sei</code>	vector with the corresponding standard errors (note: only one of the two, <code>vi</code> or <code>sei</code> , needs to be specified).
<code>ci.lb</code>	vector with the corresponding lower confidence interval bounds. Not needed if <code>vi</code> or <code>sei</code> is specified. See ‘Details’.
<code>ci.ub</code>	vector with the corresponding upper confidence interval bounds. Not needed if <code>vi</code> or <code>sei</code> is specified. See ‘Details’.
<code>pi.lb</code>	optional vector with the corresponding lower prediction interval bounds.
<code>pi.ub</code>	optional vector with the corresponding upper prediction interval bounds.
<code>rows</code>	vector to specify the rows (or more generally, the positions) for plotting the polygons (defaults is <code>-1</code>). Can also be a single value to specify the row of the first polygon (the remaining polygons are then plotted below this starting row). When <code>predstyle</code> is not <code>"line"</code> , can also be a vector of two numbers, the first for the position of the polygon, the second for the position of the prediction interval/distribution.
<code>level</code>	optional numeric value between 0 and 100 to specify the confidence interval level (see here for details).
<code>annotate</code>	optional logical to specify whether annotations should be added to the plot for the polygons that are drawn.
<code>predstyle</code>	character string to specify the style of the prediction interval (either <code>"line"</code> (the default), <code>"polygon"</code> , <code>"bar"</code> , <code>"shade"</code> , or <code>"dist"</code> ; the last three only when adding a single polygon). Can be abbreviated.
<code>predlim</code>	optional argument to specify the limits of the predictive distribution when <code>predstyle="dist"</code> .
<code>digits</code>	optional integer to specify the number of decimal places to which the annotations should be rounded.

width	optional integer to manually adjust the width of the columns for the annotations.
mlab	optional character vector of the same length as x giving labels for the polygons that are drawn.
transf	optional argument to specify a function to transform the x values and confidence interval bounds (e.g., transf=exp; see also transf).
atransf	optional argument to specify a function to transform the annotations (e.g., atransf=exp; see also transf).
targs	optional arguments needed by the function specified via transf or atransf.
efac	optional vertical expansion factor for the polygons.
col	optional character string to specify the color of the polygons.
border	optional character string to specify the border color of the polygons.
lty	optional argument to specify the line type for the prediction interval.
fonts	optional character string to specify the font for the labels and annotations.
cex	optional symbol expansion factor.
constarea	logical to specify whether the height of the polygons (when adding multiple) should be adjusted so that the area of the polygons is constant (the default is FALSE).
...	other arguments.

Details

The function can be used to add one or more polygons to an existing forest plot created with the [forest](#) function. For example, pooled estimates based on a model involving moderators can be added to the plot this way (see ‘Examples’).

To use the function, one should specify the values at which the polygons should be drawn (via the x argument) together with the corresponding variances (via the vi argument) or with the corresponding standard errors (via the sei argument). Alternatively, one can specify the values at which the polygons should be drawn together with the corresponding confidence interval bounds (via the ci.lb and ci.ub arguments). Optionally, one can also specify the bounds of the corresponding prediction interval bounds via the pi.lb and pi.ub arguments. If the latter are specified, then they are added by default as lines around the summary polygons. When adding a single polygon to the plot, one can also use the predstyle argument to change the way the prediction interval is visualized (see [forest.rma](#) for details).

If unspecified, arguments level, annotate, digits, width, transf, atransf, targs, efac, fonts, cex, annosym, and textpos are automatically set equal to the same values that were used when creating the forest plot.

Author(s)

Wolfgang Viechtbauer (<wvbw@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[forest](#) for functions to draw forest plots to which polygons can be added.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude as a moderator
res <- rma(yi, vi, mods = ~ ablat, slab=paste(author, year, sep=", "), data=dat)

### forest plot of the observed risk ratios
forest(res, addfit=FALSE, atransf=exp, xlim=c(-9,5), ylim=c(-5,16), cex=0.9,
        order=ablat, ilab=ablat, ilab.lab="Latitude", ilab.xpos=-4.5,
        header="Author(s) and Year")

### predicted average log risk ratios for 10, 30, and 50 degrees absolute latitude
x <- predict(res, newmods=c(10, 30, 50))

### add predicted average risk ratios to the forest plot
addpoly(x$pred, sei=x$se, rows=-2, mlab=c("- at 10 Degrees", "- at 30 Degrees", "- at 50 Degrees"))
abline(h=0)
text(-9, -1, "Model-Based Estimates:", pos=4, cex=0.9, font=2)
```

addpoly.predict.rma *Add Polygons to Forest Plots (Method for 'predict.rma' Objects)*

Description

Function to add one or more polygons to a forest plot based on an object of class "predict.rma".

Usage

```
## S3 method for class 'predict.rma'
addpoly(x, rows=-2, annotate,
        addpred=FALSE, predstyle, predlim, digits, width, mlab,
        transf, atransf, targs, efac, col, border, lty, fonts, cex,
        constarea=FALSE, ...)
```

Arguments

x	an object of class "predict.rma".
rows	vector to specify the rows (or more generally, the positions) for plotting the polygons (defaults is -2). Can also be a single value to specify the row of the first polygon (the remaining polygons are then plotted below this starting row).
annotate	optional logical to specify whether annotations should be added to the plot for the polygons that are drawn.

addpred	logical to specify whether the prediction interval should be added to the plot (the default is FALSE).
predstyle	character string to specify the style of the prediction interval (either "line", "polygon", "bar", "shade", or "dist"). Can be abbreviated. Setting this argument automatically sets addpred=TRUE.
predlim	optional argument to specify the limits of the predictive distribution when predstyle="dist".
digits	optional integer to specify the number of decimal places to which the annotations should be rounded.
width	optional integer to manually adjust the width of the columns for the annotations.
mlab	optional character vector of the same length as x giving labels for the polygons that are drawn.
transf	optional argument to specify a function to transform the x values and confidence interval bounds (e.g., transf=exp; see also transf).
atransf	optional argument to specify a function to transform the annotations (e.g., atransf=exp; see also transf).
targs	optional arguments needed by the function specified via transf or atransf.
efac	optional vertical expansion factor for the polygons.
col	optional character string to specify the color of the polygons.
border	optional character string to specify the border color of the polygons.
lty	optional argument to specify the line type for the prediction interval.
fonts	optional character string to specify the font for the labels and annotations.
cex	optional symbol expansion factor.
constarea	logical to specify whether the height of the polygons (when adding multiple) should be adjusted so that the area of the polygons is constant (the default is FALSE).
...	other arguments.

Details

The function can be used to add one or more polygons to an existing forest plot created with the [forest](#) function. For example, pooled estimates based on a model involving moderators can be added to the plot this way (see 'Examples').

To use the function, one should specify the values at which the polygons should be drawn (via the `x` argument) together with the corresponding variances (via the `vi` argument) or with the corresponding standard errors (via the `sei` argument). Alternatively, one can specify the values at which the polygons should be drawn together with the corresponding confidence interval bounds (via the `ci.lb` and `ci.ub` arguments). Optionally, one can also specify the bounds of the corresponding prediction interval bounds via the `pi.lb` and `pi.ub` arguments.

If unspecified, arguments `annotate`, `digits`, `width`, `transf`, `atransf`, `targs`, `efac`, `fonts`, `cex`, `annosym`, and `textpos` are automatically set equal to the same values that were used when creating the forest plot.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[forest](#) for functions to draw forest plots to which polygons can be added.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
             data=dat.bcg, slab=paste(author, year, sep=", "))

### forest plot of the observed risk ratios
with(dat, forest(yi, vi, atransf=exp, xlim=c(-9,5), ylim=c(-5,16),
               at=log(c(0.05, 0.25, 1, 4)), cex=0.9, order=alloc,
               ilab=alloc, ilab.lab="Allocation", ilab.xpos=-4.5,
               header="Author(s) and Year"))

### fit mixed-effects model with allocation method as a moderator
res <- rma(yi, vi, mods = ~ 0 + alloc, data=dat)

### predicted log risk ratios for the different allocation methods
x <- predict(res, newmods=diag(3))

### add predicted average risk ratios to the forest plot
addpoly(x, efac=1.2, col="gray", addpred=TRUE,
        mlab=c("Alternate Allocation", "Random Allocation", "Systematic Allocation"))
abline(h=0)
text(-9, -1, "Model-Based Estimates:", pos=4, cex=0.9, font=2)
```

addpoly.rma

Add Polygons to Forest Plots (Method for 'rma' Objects)

Description

Function to add a polygon to a forest plot showing the pooled estimate with corresponding confidence interval based on an object of class "rma".

Usage

```
## S3 method for class 'rma'
addpoly(x, row=-2, level=x$level, annotate,
        addpred=FALSE, predstyle, predlim, digits, width, mlab,
        transf, atransf, targ, efac, col, border, lty, fonts, cex, ...)
```

Arguments

x	an object of class "rma".
row	numeric value to specify the row (or more generally, the position) for plotting the polygon (the default is -2).
level	numeric value between 0 and 100 to specify the confidence interval level (see here for details). The default is to take the value from the object.
annotate	optional logical to specify whether annotations for the pooled estimate should be added to the plot.
addpred	logical to specify whether the prediction interval should be added to the plot (the default is FALSE).
predstyle	character string to specify the style of the prediction interval (either "line", "polygon", "bar", "shade", or "dist"). Can be abbreviated. Setting this argument automatically sets addpred=TRUE.
predlim	optional argument to specify the limits of the predictive distribution when predstyle="dist".
digits	optional integer to specify the number of decimal places to which the annotations should be rounded.
width	optional integer to manually adjust the width of the columns for the annotations.
mlab	optional character string giving a label for the pooled estimate. If unspecified, the function sets a default label.
transf	optional argument to specify a function to transform the pooled estimate and confidence interval bounds (e.g., transf=exp; see also transf).
atransf	optional argument to specify a function to transform the annotations (e.g., atransf=exp; see also transf).
targs	optional arguments needed by the function specified via transf or atransf.
efac	optional vertical expansion factor for the polygon.
col	optional character string to specify the color of the polygon.
border	optional character string to specify the border color of the polygon.
lty	optional argument to specify the line type for the prediction interval.
fonts	optional character string to specify the font for the label and annotations.
cex	optional symbol expansion factor.
...	other arguments.

Details

The function can be used to add a four-sided polygon, sometimes called a summary 'diamond', to an existing forest plot created with the [forest](#) function. The polygon shows the pooled estimate (with its confidence interval bounds) based on an equal- or a random-effects model. Using this function, pooled estimates based on different types of models can be shown in the same plot. Also, pooled estimates based on a subgrouping of the studies can be added to the plot this way. See 'Examples'.

If unspecified, arguments annotate, digits, width, transf, atransf, targs, efac, fonts, cex, annosym, and textpos are automatically set equal to the same values that were used when creating the forest plot.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[forest](#) for functions to draw forest plots to which polygons can be added.

Examples

```
### meta-analysis of the log risk ratios using the Mantel-Haenszel method
res <- rma.mh(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg,
             slab=paste(author, year, sep=", "))

### forest plot of the observed risk ratios with the pooled estimate
forest(res, atranf=exp, xlim=c(-8,6), ylim=c(-3,16))

### meta-analysis of the log risk ratios using a random-effects model
res <- rma(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### add the pooled estimate from the random-effects model to the forest plot
addpoly(res)

### forest plot with subgrouping of studies and summaries per subgroup
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg,
             slab=paste(author, year, sep=", "))
res <- rma(yi, vi, data=dat)
tmp <- forest(res, xlim=c(-16, 4.6), at=log(c(0.05, 0.25, 1, 4)), atranf=exp,
             ilab=cbind(tpos, tneg, cpos, cneg), ilab.lab=c("TB+", "TB-", "TB+", "TB-"),
             ilab.xpos=c(-9.5,-8,-6,-4.5), cex=0.75, ylim=c(-2, 27), order=alloc,
             rows=c(3:4,9:15,20:23), mlab="RE Model for All Studies",
             header="Author(s) and Year")
op <- par(cex=tmp$cex)
text(c(-8.75,-5.25), tmp$ylim[2]-0.2, c("Vaccinated", "Control"), font=2)
text(-16, c(24,16,5), c("Systematic Allocation", "Random Allocation",
                        "Alternate Allocation"), font=4, pos=4)
par(op)
res <- rma(yi, vi, data=dat, subset=(alloc=="systematic"))
addpoly(res, row=18.5, mlab="RE Model for Subgroup")
res <- rma(yi, vi, data=dat, subset=(alloc=="random"))
addpoly(res, row=7.5, mlab="RE Model for Subgroup")
res <- rma(yi, vi, data=dat, subset=(alloc=="alternate"))
addpoly(res, row=1.5, mlab="RE Model for Subgroup")
```

aggregate.escalc

*Aggregate Multiple Estimates Within Studies***Description**

Function to aggregate multiple estimates belonging to the same study (or to the same level of some other clustering variable) into a single combined estimate.

Usage

```
## S3 method for class 'escalc'
aggregate(x, cluster, time, obs, V, struct="CS", rho, phi,
          weighted=TRUE, checkpd=TRUE, fun, na.rm=TRUE,
          addk=FALSE, subset, select, digits, var.names, ...)
```

Arguments

<code>x</code>	an object of class "escalc".
<code>cluster</code>	vector to specify the clustering variable (e.g., study).
<code>time</code>	optional vector to specify the time points (only relevant when <code>struct="CAR"</code> , <code>"CS+CAR"</code> , or <code>"CS*CAR"</code>).
<code>obs</code>	optional vector to distinguish different estimates measured at the same time point (only relevant when <code>struct="CS*CAR"</code>).
<code>V</code>	optional argument to specify the variance-covariance matrix of the sampling errors. If unspecified, argument <code>struct</code> is used to specify the variance-covariance structure.
<code>struct</code>	character string to specify the variance-covariance structure of the sampling errors within the same cluster (either <code>"ID"</code> , <code>"CS"</code> , <code>"CAR"</code> , <code>"CS+CAR"</code> , or <code>"CS*CAR"</code>). See 'Details'.
<code>rho</code>	value of the correlation of the sampling errors within clusters (when <code>struct="CS"</code> , <code>"CS+CAR"</code> , or <code>"CS*CAR"</code>). Can also be a vector with the value of the correlation for each cluster.
<code>phi</code>	value of the autocorrelation of the sampling errors within clusters (when <code>struct="CAR"</code> , <code>"CS+CAR"</code> , or <code>"CS*CAR"</code>). Can also be a vector with the value of the autocorrelation for each cluster.
<code>weighted</code>	logical to specify whether estimates within clusters should be aggregated using inverse-variance weighting (the default is <code>TRUE</code>). If set to <code>FALSE</code> , unweighted averages are computed.
<code>checkpd</code>	logical to specify whether to check that the variance-covariance matrices of the sampling errors within clusters are positive definite (the default is <code>TRUE</code>).
<code>fun</code>	optional list with three functions for aggregating other variables besides the estimates within clusters (for numeric/integer variables, for logicals, and for all other types, respectively).

<code>na.rm</code>	logical to specify whether NA values should be removed before aggregating values within clusters (the default is TRUE). Can also be a vector with two logicals (the first pertaining to the estimates, the second to all other variables).
<code>addk</code>	logical to specify whether to add the cluster size as a new variable (called <code>ki</code>) to the dataset (the default is FALSE).
<code>subset</code>	optional (logical or numeric) vector to specify the subset of rows to include when aggregating the estimates.
<code>select</code>	optional vector to specify the names of the variables to include in the aggregated dataset.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
<code>var.names</code>	optional character vector with two elements to specify the name of the variable that contains the estimates and the name of the variable with the corresponding sampling variances (when unspecified, the function attempts to set these automatically based on the object).
<code>...</code>	other arguments.

Details

In meta-analyses, one can often extract multiple estimates from the same study. Ideally, such structures should be analyzed using an appropriate multilevel/multivariate model as can be fitted with the `rma.mv` function. However, there may occasionally be reasons for aggregating multiple estimates belonging to the same study (or to the same level of some other clustering variable) into a single combined estimate. The present function can be used for this purpose.

The input must be an object of class "escalc". The error 'Error in match.fun(FUN): argument "FUN" is missing, with no default' indicates that a regular data frame was passed to the function, but this does not work. One can turn a regular data frame (containing the estimates and the corresponding sampling variances) into an "escalc" object with the `escalc` function. See the 'Examples' below for an illustration of this.

The `cluster` variable is used to specify which estimates belong to the same study/cluster.

In the simplest case, the estimates within clusters (or, to be precise, their sampling errors) are assumed to be independent. This is usually a safe assumption as long as each study participant (or whatever the study units are) only contributes data to a single estimate. For example, if a study provides effect size estimates for male and female subjects separately, then the sampling errors can usually be assumed to be independent. In this case, one can set `struct="ID"` and multiple estimates within the same cluster are combined using standard inverse-variance weighting (i.e., using weighted least squares) under the assumption of independence.

In other cases, the estimates within clusters cannot be assumed to be independent. For example, if multiple effect size estimates are computed for the same group of subjects (e.g., based on different scales to measure some construct of interest), then the estimates are likely to be correlated. If the actual correlation between the estimates is unknown, one can often still make an educated guess and set argument `rho` to this value, which is then assumed to be the same for all pairs of estimates within clusters when `struct="CS"` (for a compound symmetric structure). Multiple estimates within the same cluster are then combined using inverse-variance weighting taking their correlation into consideration (i.e., using generalized least squares). One can also specify a different value of `rho` for each cluster by passing a vector (of the same length as the number of clusters) to this argument.

If multiple effect size estimates are computed for the same group of subjects at different time points, then it may be more sensible to assume that the correlation between the estimates decreases as a function of the distance between the time points. If so, one can specify `struct="CAR"` (for a continuous-time autoregressive structure), set `phi` to the autocorrelation (for two estimates one time-unit apart), and use argument `time` to specify the actual time points corresponding to the estimates. The correlation between two estimates, y_{it} and $y_{it'}$, in the i th cluster, with time points `timeit` and `timeit'`, is then given by $\phi^{|time_{it}-time_{it'}|}$. One can also specify a different value of `phi` for each cluster by passing a vector (of the same length as the number of clusters) to this argument.

One can also combine the compound symmetric and autoregressive structures if there are multiple time points and multiple effect size estimates available at these time points. One option is `struct="CS+CAR"`. In this case, one must specify the `time` argument and both `rho` and `phi`. The correlation between two estimates, y_{it} and $y_{it'}$, in the i th cluster, with time points `timeit` and `timeit'`, is then given by $\rho + (1 - \rho)\phi^{|time_{it}-time_{it'}|}$.

Alternatively, one can specify `struct="CS*CAR"`. In this case, one must specify both the `time` and `obs` arguments and both `rho` and `phi`. The correlation between two estimates, y_{ijt} and $y_{ijt'}$, with the same value for `obs` but different values for `time`, is then given by $\phi^{|time_{ijt}-time_{ijt'}|}$, the correlation between two estimates, y_{ijt} and $y_{ij't}$, with different values for `obs` but the same value for `time`, is then given by ρ , and the correlation between two estimates, y_{ijt} and $y_{ij't'}$, with different values for `obs` and different values for `time`, is then given by $\rho \times \phi^{|time_{ijt}-time_{ij't'}|}$.

Finally, if one actually knows the correlation (and hence the covariance) between each pair of estimates (or has an approximation thereof), one can also specify the entire variance-covariance matrix of the estimates (or more precisely, their sampling errors) via the `V` argument (in this case, arguments `struct`, `time`, `obs`, `rho`, and `phi` are ignored). Note that the `vcalc` function can be used to construct such a `V` matrix and provides even more flexibility for specifying various types of dependencies. See the ‘Examples’ below for an illustration of this.

Instead of using inverse-variance weighting (i.e., weighted/generalized least squares) to combine the estimates within clusters, one can set `weighted=FALSE` in which case the estimates are averaged within clusters without any weighting (although the correlations between estimates as specified are still taken into consideration).

Other variables (besides the estimates) will also be aggregated to the cluster level. By default, numeric/integer type variables are averaged, logicals are also averaged (yielding the proportion of TRUE values), and for all other types of variables (e.g., character variables or factors) the most frequent category/level is returned. One can also specify a list of three functions via the `fun` argument for aggregating variables belonging to these three types.

Argument `na.rm` controls how missing values should be handled. By default, any missing estimates are first removed before aggregating the non-missing values within each cluster. The same applies when aggregating the other variables. One can also specify a vector with two logicals for the `na.rm` argument to control how missing values should be handled when aggregating the estimates and when aggregating all other variables.

Value

An object of class `c("escalc", "data.frame")` that contains the (selected) variables aggregated to the cluster level.

The object is formatted and printed with the `print` function.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[escalc](#) for a function to create escalc objects.

Examples

```
### copy data into 'dat' and examine data
dat <- dat.konstantopoulos2011
head(dat, 11)

### aggregate estimates to the district level, assuming independent sampling
### errors for multiples studies/schools within the same district
agg <- aggregate(dat, cluster=district, struct="ID", addk=TRUE)
agg

### copy data into 'dat' and examine data
dat <- dat.assink2016
head(dat, 19)

### note: 'dat' is an 'escalc' object
class(dat)

### turn 'dat' into a regular data frame
dat <- as.data.frame(dat)
class(dat)

### turn data frame into an 'escalc' object
dat <- escalc(measure="SMD", yi=yi, vi=vi, data=dat)
class(dat)

### aggregate the estimates to the study level, assuming a CS structure for
### the sampling errors within studies with a correlation of 0.6
agg <- aggregate(dat, cluster=study, rho=0.6)
agg

### use vcalc() and then the V argument
V <- vcalc(vi, cluster=study, obs=esid, data=dat, rho=0.6)
agg <- aggregate(dat, cluster=study, V=V)
agg

### use a correlation of 0.7 for effect sizes corresponding to the same type of
### delinquent behavior and a correlation of 0.5 for effect sizes corresponding
### to different types of delinquent behavior
V <- vcalc(vi, cluster=study, type=deltype, obs=esid, data=dat, rho=c(0.7, 0.5))
```

```

agg <- aggregate(dat, cluster=study, V=V)
agg

### reshape 'dat.ishak2007' into long format
dat <- dat.ishak2007
dat <- reshape(dat.ishak2007, direction="long", idvar="study", v.names=c("yi","vi"),
               varying=list(c(2,4,6,8), c(3,5,7,9)))
dat <- dat[order(study, time),]
dat <- dat[!is.na(yi),]
rownames(dat) <- NULL
head(dat, 8)

### aggregate the estimates to the study level, assuming a CAR structure for
### the sampling errors within studies with an autocorrelation of 0.9
agg <- aggregate(dat, cluster=study, struct="CAR", time=time, phi=0.9)
head(agg, 5)

```

anova.rma

Likelihood Ratio and Wald-Type Tests for 'rma' Objects

Description

For two (nested) models of class "rma.uni" or "rma.mv", the function provides a full versus reduced model comparison in terms of model fit statistics and a likelihood ratio test. When a single model is specified, a Wald-type test of one or more model coefficients or linear combinations thereof is carried out.

Usage

```

## S3 method for class 'rma'
anova(object, object2, btt, X, att, Z, rhs, adjust, digits, refit=FALSE, ...)

```

Arguments

<code>object</code>	an object of class "rma.uni" or "rma.mv".
<code>object2</code>	an (optional) object of class "rma.uni" or "rma.mv". Only relevant when conducting a model comparison and likelihood ratio test. See 'Details'.
<code>btt</code>	optional vector of indices (or list thereof) to specify which coefficients should be included in the Wald-type test. Can also be a string to grep for. See 'Details'.
<code>X</code>	optional numeric vector or matrix to specify one or more linear combinations of the coefficients in the model that should be tested. See 'Details'.
<code>att</code>	optional vector of indices (or list thereof) to specify which scale coefficients should be included in the Wald-type test. Can also be a string to grep for. See 'Details'. Only relevant for location-scale models (see rma.uni).
<code>Z</code>	optional numeric vector or matrix to specify one or more linear combinations of the scale coefficients in the model that should be tested. See 'Details'. Only relevant for location-scale models (see rma.uni).

rhs	optional scalar or vector of values for the right-hand side of the null hypothesis when testing a set of coefficients (via <code>btt</code> or <code>att</code>) or linear combinations thereof (via <code>X</code> or <code>Z</code>). If unspecified, this defaults to a vector of zeros of the appropriate length. See ‘Details’.
adjust	optional argument to specify (as a character string) a method for adjusting the p-values of Wald-type tests for multiple testing. See <code>p.adjust</code> for possible options. Can be abbreviated. Can also be a logical and if TRUE, then a Bonferroni correction is used.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
refit	logical to specify whether models fitted with REML estimation and differing in their fixed effects should be refitted with ML estimation when conducting a likelihood ratio test (the default is FALSE).
...	other arguments.

Details

The function can be used in three different ways:

1. When a single model is specified (via argument `object`), the function provides a Wald-type test of one or more model coefficients, that is,

$$H_0: \beta_{j \in \text{btt}} = 0,$$

where $\beta_{j \in \text{btt}}$ is the set of coefficients to be tested (by default whether the set of coefficients is significantly different from zero, but one can specify a different value under the null hypothesis via argument `rhs`).

In particular, for equal- or random-effects models (i.e., models without moderators), this is just the test of the single coefficient of the model (i.e., $H_0: \theta = 0$ or $H_0: \mu = 0$). For models including moderators, an omnibus test of all model coefficients is conducted that excludes the intercept (the first coefficient) if it is included in the model. If no intercept is included in the model, then the omnibus test includes all coefficients in the model including the first.

Alternatively, one can manually specify the indices of the coefficients to test via the `btt` (‘betas to test’) argument. For example, with `btt=c(3,4)`, only the third and fourth coefficients from the model are included in the test (if an intercept is included in the model, then it corresponds to the first coefficient in the model). Instead of specifying the coefficient numbers, one can specify a string for `btt`. In that case, `grep` will be used to search for all coefficient names that match the string (and hence, one can use regular expressions to fine-tune the search for matching strings). Using the `btt` argument, one can for example select all coefficients corresponding to a particular factor to test if the factor as a whole is significant. One can also specify a list of indices/strings, in which case tests of all list elements will be conducted. See ‘Examples’.

For location-scale models fitted with the `rma.uni` function, one can use the `att` argument in an analogous manner to specify the indices of the scale coefficients to test (i.e., $H_0: \alpha_{j \in \text{att}} = 0$, where $\alpha_{j \in \text{att}}$ is the set of coefficients to be tested).

- When a single model is specified (via argument `object`), one can use the `X` argument¹ to specify a linear combination of the coefficients in the model that should be tested using a Wald-type test, that is,

$$H_0: \tilde{x}\beta = 0,$$

where $\tilde{x}$ is a (row) vector of the same length as there are coefficients in the model (by default whether the linear combination is significantly different from zero, but one can specify a different value under the null hypothesis via argument `rhs`). One can also specify a matrix of linear combinations via the `X` argument to test

$$H_0: \tilde{X}\beta = 0,$$

where each row of $\tilde{X}$ defines a particular linear combination to be tested (if `rhs` is used, then it should either be a scalar or of the same length as the number of combinations to be tested). If the matrix is of full rank, an omnibus Wald-type test of all linear combinations is also provided. Linear combinations can also be obtained with the `predict` function, which provides corresponding confidence intervals. See also the `pairmat` function for constructing a matrix of pairwise contrasts for testing the levels of a categorical moderator against each other.

For location-scale models fitted with the `rma.uni` function, one can use the `Z` argument in an analogous manner to specify one or multiple linear combinations of the scale coefficients in the model that should be tested (i.e., $H_0: \tilde{Z}\alpha = 0$).

- When specifying two models for comparison (via arguments `object` and `object2`), the function provides a likelihood ratio test (LRT) comparing the two models. The two models must be based on the same set of data, must be of the same class, and should be nested for the LRT to make sense. Also, LRTs are not meaningful when using REML estimation and the two models differ in terms of their fixed effects (setting `refit=TRUE` automatically refits the two models using ML estimation). Also, the theory underlying LRTs is only really applicable when comparing models that were fitted with ML/REML estimation, so if some other estimation method was used to fit the two models, the results should be treated with caution.

¹ This argument used to be called `L`, but was renamed to `X` (but using `L` in place of `X` still works).

Value

An object of class `"anova.rma"`. When a single model is specified (without any further arguments or together with the `btt` or `att` argument), the object is a list containing the following components:

<code>QM</code>	test statistic of the Wald-type test of the model coefficients.
<code>QMdf</code>	corresponding degrees of freedom.
<code>QMp</code>	corresponding p-value.
<code>btt</code>	indices of the coefficients tested by the Wald-type test.
<code>k</code>	number of estimates included in the analysis.
<code>p</code>	number of coefficients in the model (including the intercept).
<code>m</code>	number of coefficients included in the Wald-type test.
<code>...</code>	some additional elements/values.

When `btt` or `att` was a list, then the object is a list of class `"list.anova.rma"`, where each element is an `"anova.rma"` object as described above.

When argument `X` is used, the object is a list containing the following components:

<code>QM</code>	test statistic of the omnibus Wald-type test of all linear combinations.
<code>QMdf</code>	corresponding degrees of freedom.
<code>QMp</code>	corresponding p-value.
<code>hyp</code>	description of the linear combinations tested.
<code>Xb</code>	values of the linear combinations.
<code>se</code>	standard errors of the linear combinations.
<code>zval</code>	test statistics of the linear combinations.
<code>pval</code>	corresponding p-values.

When two models are specified, the object is a list containing the following components:

<code>fit.stats.f</code>	log-likelihood, deviance, AIC, BIC, and AICc for the full model.
<code>fit.stats.r</code>	log-likelihood, deviance, AIC, BIC, and AICc for the reduced model.
<code>parms.f</code>	number of parameters in the full model.
<code>parms.r</code>	number of parameters in the reduced model.
<code>LRT</code>	likelihood ratio test statistic.
<code>pval</code>	corresponding p-value.
<code>QE.f</code>	test statistic of the test for (residual) heterogeneity from the full model.
<code>QE.r</code>	test statistic of the test for (residual) heterogeneity from the reduced model.
<code>tau2.f</code>	estimated τ^2 value from the full model. NA for <code>"rma.mv"</code> objects.
<code>tau2.r</code>	estimated τ^2 value from the reduced model. NA for <code>"rma.mv"</code> objects.
<code>R2</code>	amount (in percent) of the heterogeneity in the reduced model that is accounted for in the full model (NA for <code>"rma.mv"</code> objects). This can be regarded as a pseudo R^2 statistic (Raudenbush, 2009). Note that the value may not be very accurate unless k is large (López-López et al., 2014).
<code>...</code>	some additional elements/values.

The results are formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function.

Note

The function can also be used to conduct a likelihood ratio test (LRT) for the amount of (residual) heterogeneity in random- and mixed-effects models. The full model should then be fitted with either `method="ML"` or `method="REML"` and the reduced model with `method="EE"` (or with `tau2=0`). The p-value for the test is based on a chi-square distribution with 1 degree of freedom, but actually needs to be adjusted for the fact that the parameter (i.e., τ^2) falls on the boundary of the parameter space under the null hypothesis (see Viechtbauer, 2007, for more details).

LRTs for variance components in more complex models (as fitted with the `rma.mv` function) can also be conducted in this manner (see ‘Examples’).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Hardy, R. J., & Thompson, S. G. (1996). A likelihood approach to meta-analysis with random effects. *Statistics in Medicine*, **15**(6), 619–629. [https://doi.org/10.1002/\(sici\)1097-0258\(19960330\)15:6%3C619::ai](https://doi.org/10.1002/(sici)1097-0258(19960330)15:6%3C619::ai)
- Huizenga, H. M., Visser, I., & Dolan, C. V. (2011). Testing overall and moderator effects in random effects meta-regression. *British Journal of Mathematical and Statistical Psychology*, **64**(1), 1–19. <https://doi.org/10.1348/000711010X522687>
- López-López, J. A., Marín-Martínez, F., Sánchez-Meca, J., Van den Noortgate, W., & Viechtbauer, W. (2014). Estimation of the predictive power of the model in mixed-effects meta-regression: A simulation study. *British Journal of Mathematical and Statistical Psychology*, **67**(1), 30–48. <https://doi.org/10.1111/bmsp.12002>
- Raudenbush, S. W. (2009). Analyzing effect sizes: Random effects models. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 295–315). New York: Russell Sage Foundation.
- Viechtbauer, W. (2007). Hypothesis tests for population heterogeneity in meta-analysis. *British Journal of Mathematical and Statistical Psychology*, **60**(1), 29–60. <https://doi.org/10.1348/000711005X64042>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*. **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>

See Also

[rma.uni](#) and [rma.mv](#) for functions to fit models for which likelihood ratio and Wald-type tests can be conducted.

[print](#) for the print method and [as.data.frame](#) for the method to format the results as a data frame.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model
res1 <- rma(yi, vi, data=dat, method="ML")
res1

### fit mixed-effects model with two moderators (absolute latitude and publication year)
res2 <- rma(yi, vi, mods = ~ ablat + year, data=dat, method="ML")
res2

### Wald-type test of the two moderators
anova(res2)

### alternative way of specifying the same test
anova(res2, X=rbind(c(0,1,0), c(0,0,1)))
```

```

### corresponding likelihood ratio test
anova(res1, res2)

### Wald-type test of a linear combination
anova(res2, X=c(1,35,1970))

### use predict() to obtain the same linear combination (with its CI)
predict(res2, newmods=c(35,1970))

### Wald-type tests of several linear combinations
anova(res2, X=cbind(1,seq(0,60,by=10),1970))

### adjust for multiple testing with the Bonferroni method
anova(res2, X=cbind(1,seq(0,60,by=10),1970), adjust="bonf")

### mixed-effects model with three moderators
res3 <- rma(yi, vi, mods = ~ ablat + year + alloc, data=dat, method="ML")
res3

### Wald-type test of the 'alloc' factor
anova(res3, btt=4:5)

### instead of specifying the coefficient numbers, grep for "alloc"
anova(res3, btt="alloc")

### specify a list for the 'btt' argument
anova(res3, btt=list(2,3,4:5))

### adjust for multiple testing with the Bonferroni method
anova(res3, btt=list(2,3,4:5), adjust="bonf")

#####

### an example of doing LRTs of variance components in more complex models
dat <- dat.konstantopoulos2011
res <- rma.mv(yi, vi, random = ~ 1 | district/school, data=dat)

### likelihood ratio test of the district-level variance component
res0 <- rma.mv(yi, vi, random = ~ 1 | district/school, data=dat, sigma2=c(0,NA))
anova(res, res0)

### likelihood ratio test of the school-level variance component
res0 <- rma.mv(yi, vi, random = ~ 1 | district/school, data=dat, sigma2=c(NA,0))
anova(res, res0)

### likelihood ratio test of both variance components simultaneously
res0 <- rma.mv(yi, vi, data=dat)
anova(res, res0)

#####

### an example illustrating a workflow involving cluster-robust inference

```

```

dat <- dat.assink2016

### assume that the effect sizes within studies are correlated with rho=0.6
V <- vcalc(vi, cluster=study, obs=esid, data=dat, rho=0.6)

### fit multilevel model using this approximate V matrix
res <- rma.mv(yi, V, random = ~ 1 | study/esid, data=dat)
res

### likelihood ratio tests of the two variance components
res0 <- rma.mv(yi, V, random = ~ 1 | study/esid, data=dat, sigma2=c(0,NA))
anova(res, res0)
res0 <- rma.mv(yi, V, random = ~ 1 | study/esid, data=dat, sigma2=c(NA,0))
anova(res, res0)

### use cluster-robust methods for inferences about the fixed effects
sav <- robust(res, cluster=study, clubSandwich=TRUE)
sav

### examine if 'delttype' is a potential moderator
res <- rma.mv(yi, V, mods = ~ deltype, random = ~ 1 | study/esid, data=dat)
sav <- robust(res, cluster=study, clubSandwich=TRUE)
sav

### note: the (denominator) dfs for the omnibus F-test are very low, so the results
### of this test may not be trustworthy; consider using cluster wild bootstrapping
## Not run:
library(wildmeta)
Wald_test_cwb(res, constraints=constrain_zero(2:3), R=1000, seed=1234)

## End(Not run)

```

baujat

Baujat Plots for 'rma' Objects

Description

Function to create Baujat plots for objects of class "rma".

Usage

```

baujat(x, ...)

## S3 method for class 'rma'
baujat(x, xlim, ylim, xlab, ylab, cex, symbol="ids", grid=TRUE, progbar=FALSE, ...)

```

Arguments

x an object of class "rma".

<code>xlim</code>	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
<code>ylim</code>	y-axis limits. If unspecified, the function sets the y-axis limits to some sensible values.
<code>xlab</code>	title for the x-axis. If unspecified, the function sets an appropriate axis title.
<code>ylab</code>	title for the y-axis. If unspecified, the function sets an appropriate axis title.
<code>cex</code>	symbol/character expansion factor.
<code>symbol</code>	either an integer to specify the pch value (i.e., plotting symbol), or "slab" to plot the study labels, or "ids" (the default) to plot the study id numbers.
<code>grid</code>	logical to specify whether a grid should be added to the plot. Can also be a color name.
<code>progbars</code>	logical to specify whether a progress bar should be shown (the default is FALSE).
<code>...</code>	other arguments.

Details

The model specified via `x` must be a model fitted with either the `rma.uni`, `rma.mh`, or `rma.peto` functions.

Baujat et al. (2002) proposed a diagnostic plot to detect sources of heterogeneity in meta-analytic data. The plot shows the contribution of each study to the overall Q -test statistic for heterogeneity on the x-axis versus the influence of each study (defined as the standardized squared difference between the overall estimate based on an equal-effects model with and without the study included in the model fitting) on the y-axis. The same type of plot can be produced by first fitting an equal-effects model with either the `rma.uni` (using `method="EE"`), `rma.mh`, or `rma.peto` functions and then passing the fitted model object to the `baujat` function.

For models fitted with the `rma.uni` function (which may be random-effects or mixed-effects meta-regressions models), the idea underlying this type of plot can be generalized as described by Viechtbauer (2021): The x-axis then corresponds to the squared Pearson residual of a study, while the y-axis corresponds to the standardized squared difference between the predicted/fitted value for the study with and without the study included in the model fitting.

By default, the points plotted are the study id numbers, but one can also plot the study labels by setting `symbol="slab"` (if study labels are available within the model object) or one can specify a plotting symbol via the `symbol` argument that gets passed to `pch` (see [points](#) for possible options).

Value

A data frame with components:

<code>x</code>	the x-axis coordinates of the points that were plotted.
<code>y</code>	the y-axis coordinates of the points that were plotted.
<code>ids</code>	the study id numbers.
<code>slab</code>	the study labels.

Note that the data frame is returned invisibly.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Baujat, B., Mahe, C., Pignon, J.-P., & Hill, C. (2002). A graphical method for exploring heterogeneity in meta-analyses: Application to a meta-analysis of 65 trials. *Statistics in Medicine*, **21**(18), 2641–2652. <https://doi.org/10.1002/sim.1221>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/978131>

See Also

[rma.uni](#), [rma.mh](#) and [rma.peto](#) for functions to fit models for which Baujat plots can be created.
[influence](#) for other model diagnostics.

Examples

```
### copy data from Pignon et al. (2000) into 'dat'
dat <- dat.pignon2000

### calculate estimated log hazard ratios and sampling variances
dat$yi <- with(dat, OmE/V)
dat$vi <- with(dat, 1/V)

### meta-analysis based on all 65 trials
res <- rma(yi, vi, data=dat, method="EE", slab=trial)

### create Baujat plot
baujat(res)

### some variations of the plotting symbol
baujat(res, symbol=19)
baujat(res, symbol="slab")

### label only a selection of the more 'extreme' points
sav <- baujat(res, symbol=19, xlim=c(0,20))
sav <- sav[sav$x >= 10 | sav$y >= 0.10,]
text(sav$x, sav$y, sav$slab, pos=1, cex=0.8)
```

bldiag

Construct Block Diagonal Matrix

Description

Function to construct a block diagonal matrix from (a list of) matrices.

Usage

```
bldiag(..., order)
```

Arguments

<code>...</code>	individual matrices or a list of matrices.
<code>order</code>	optional argument to specify a variable based on which a square block diagonal matrix should be ordered.

Value

A (block diagonal) matrix.

Author(s)

Posted to R-help by Berton Gunter (2 Sep 2005) with some further adjustments by Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

See Also

[rma.mv](#) for the model fitting function that can take such a block diagonal matrix as input (for the `V` argument).

[blsplit](#) for a function that can split a block diagonal matrix into a list of sub-matrices.

Examples

```
### copy data into 'dat'
dat <- dat.berkey1998
dat

### construct list with the variance-covariance matrices of the effect sizes within the studies
V <- lapply(split(dat[c("v1i","v2i")], dat$trial), as.matrix)
V

### construct block diagonal matrix
V <- bldiag(V)
V

### if we split based on 'author', the list elements in V are in a different order than the data
V <- lapply(split(dat[c("v1i","v2i")], dat$author), as.matrix)
V

### can use 'order' argument to reorder the block-diagonal matrix into the correct order
V <- bldiag(V, order=dat$author)
V
```

blsplit	<i>Split Block Diagonal Matrix</i>
---------	------------------------------------

Description

Function to split a block diagonal matrix into a list of sub-matrices.

Usage

```
blsplit(x, cluster, fun, args, sort=FALSE)
```

Arguments

x	a block diagonal matrix.
cluster	vector to specify the clustering variable to use for splitting.
fun	optional argument to specify a function to apply to each sub-matrix.
args	optional argument to specify any additional argument(s) for the function specified via fun.
sort	logical to specify whether to sort the list by the unique cluster values (the default is FALSE).

Value

A list of one or more sub-matrices.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

See Also

[bldiag](#) for a function to create a block diagonal matrix based on sub-matrices.

[vcalc](#) for a function to construct a variance-covariance matrix of dependent estimates, which often has a block diagonal structure.

Examples

```
### copy data into 'dat'
dat <- dat.assink2016

### assume that the effect sizes within studies are correlated with rho=0.6
V <- vcalc(vi, cluster=study, obs=esid, data=dat, rho=0.6)

### split V matrix into list of sub-matrices
Vs <- blsplit(V, cluster=dat$study)
Vs[1:2]
lapply(Vs[1:2], cov2cor)
```

```
### illustrate the use of the fun and args arguments
blsplit(V, cluster=dat$study, cov2cor)[1:2]
blsplit(V, cluster=dat$study, round, 3)[1:2]
```

blup

Best Linear Unbiased Predictions for 'rma.uni' Objects

Description

Function to compute the best linear unbiased predictions (BLUPs) of the study-specific true effects for objects of class "rma.uni". Corresponding standard errors and prediction interval bounds are also provided.

Usage

```
blup(x, ...)

## S3 method for class 'rma.uni'
blup(x, level, digits, transf, targs, ...)
```

Arguments

x	an object of class "rma.uni".
level	numeric value between 0 and 100 to specify the prediction interval level (see here for details). If unspecified, the default is to take the value from the object.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
transf	optional argument to specify a function to transform the predicted values and interval bounds (e.g., transf=exp; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified under transf.
...	other arguments.

Value

An object of class "list.rma". The object is a list containing the following components:

pred	predicted values.
se	corresponding standard errors.
pi.lb	lower bound of the prediction intervals.
pi.ub	upper bound of the prediction intervals.
...	some additional elements/values.

The object is formatted and printed with the [print](#) function. To format the results as a data frame, one can use the [as.data.frame](#) function.

Note

The standard random effects model assumes $y_i|\theta_i \sim N(\theta_i, v_i)$ and $\theta_i \sim N(\mu, \tau^2)$. So θ_i denotes the true effect in the i th study and y_i the corresponding observed effect. While y_i is an estimate of θ_i , we can use information from the random-effects model to obtain better estimates of the θ_i values. These are the so-called best linear unbiased predictions (BLUPs) of the study-specific true effects, also sometimes called shrunken estimates, as they represent a compromise between the observed effect in the i th study (i.e., y_i) and the estimated average effect obtained from the random-effects model (i.e., $\hat{\mu}$). They can also be regarded as ‘empirical Bayes’ estimates under certain assumptions.

To be precise, it should be noted that the function actually computes empirical BLUPs (eBLUPs), since the predicted values are a function of the estimated value of τ^2 .

By default, a standard normal distribution is used to construct corresponding prediction intervals. When the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="adhoc"`, then a t -distribution with $k - p$ degrees of freedom is used.

Equal-effects models do not contain random study effects. The BLUPs for these models will therefore be equal to the fitted values, that is, those obtained with `fitted` and `predict`.

When using the `transf` argument, the transformation is applied to the predicted values and the corresponding interval bounds. The standard errors are then set equal to NA and are omitted from the printed output.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Kackar, R. N., & Harville, D. A. (1981). Unbiasedness of two-stage estimation and prediction procedures for mixed linear models. *Communications in Statistics, Theory and Methods*, **10**(13), 1249–1261. <https://doi.org/10.1080/03610928108828108>
- Raudenbush, S. W., & Bryk, A. S. (1985). Empirical Bayes meta-analysis. *Journal of Educational Statistics*, **10**(2), 75–98. <https://doi.org/10.3102/10769986010002075>
- Robinson, G. K. (1991). That BLUP is a good thing: The estimation of random effects. *Statistical Science*, **6**(1), 15–32. <https://doi.org/10.1214/ss/1177011926>
- van Aert, R. C. M., Schmid, C. H., Svensson, D., & Jackson, D. (2021). Study specific prediction intervals for random-effects meta-analysis: A tutorial. *Research Synthesis Methods*, **12**(4), 429–447. <https://doi.org/10.1002/jrsm.1490>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`rma.uni` for the function to fit models for which BLUPs can be computed.

`predict` and `fitted` for functions to compute the predicted/fitted values based only on the fixed effects and `ranef` for a function to compute the BLUPs based only on the random effects.

For conditional residuals (the deviations of the observed effect sizes from the BLUPs), see `rstandard` with `type="conditional"`.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### meta-analysis of the log risk ratios using a random-effects model
res <- rma(yi, vi, data=dat)

### BLUPs of the true risk ratios for each study
blup(res, transf=exp)

### illustrate shrinkage of BLUPs towards the (estimated) population average
res <- rma(yi, vi, data=dat)
blups <- blup(res)$pred
plot(NA, NA, xlim=c(0.8,2.4), ylim=c(-2,0.5), pch=19,
     xaxt="n", bty="n", xlab="", ylab="Log Risk Ratio")
segments(rep(1,13), dat$yi, rep(2,13), blups, col="darkgray")
points(rep(1,13), dat$yi, pch=19)
points(rep(2,13), blups, pch=19)
axis(side=1, at=c(1,2), labels=c("Observed\nValues", "BLUPs"), lwd=0)
segments(0, res$beta, 2.15, res$beta, lty="dotted")
text(2.3, res$beta, substitute(hat(mu)==muhat, list(muhat=round(res$beta[[1]], 2))), cex=1)
```

cmicalc

Calculate the Bias Correction Factor for SMDs and Related Measures

Description

Function to calculate the bias correction factor for standardized mean differences and related measures.

Usage

```
cmicalc(mi, method="exact")
```

Arguments

<code>mi</code>	numeric vector.
<code>method</code>	character string to specify if the exact ("exact") or approximate ("approx") formula should be used.

Details

Standardized mean differences (Cohen's d values) have a slight positive bias (i.e., they tend to be too large on average). Hedges (1981) derived the bias correction factor to make standardized mean differences unbiased (yielding what is commonly referred to as Hedges' g values).

The function computes the bias correction factor, using m_i as input. For standardized mean differences, this is equal to $m_i = n_{1i} + n_{2i} - 2$, where n_{1i} and n_{2i} are the number of subjects in the

two groups. The same bias correction factor can be used for various other related measures, such as standardized mean changes, except that m_i may need to be calculated differently.

By default (i.e., when method="exact"), the method uses the exact formula to calculate the bias correction factor (equation 6e in Hedges, 1981), that is,

$$c(m_i) = \frac{\Gamma(\frac{m_i}{2})}{\sqrt{\frac{m_i}{2}} \Gamma(\frac{m_i-1}{2})}.$$

When method="approx", the (very accurate) approximation

$$c(m_i) = 1 - \frac{3}{4m_i - 1}$$

is used instead (first equation on page 114 in Hedges, 1981).

Value

A numeric vector of length equal to the length of `mi`.

Note

The notation $c(m_i)$ is based on that used in Hedges (1981). Later publications and references often denote the bias correction factor J or $J(m_i)$.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics*, **6**(2), 107–128. <https://doi.org/10.3102/10769986006002107>

See Also

[escalc](#) for a function that makes use of this bias correction.

Examples

```
### copy dataset to dat
dat <- dat.furukawa2003

### calculate the bias correction factor for each study
dat$mi <- dat$Ne + dat$Nc - 2
dat$cmi <- cmicalc(dat$mi)
dat
```

coef.deltamethod	<i>Extract the Estimates and Variance-Covariance Matrix from 'deltamethod' Objects</i>
------------------	--

Description

Methods for objects of class "deltamethod".

Usage

```
## S3 method for class 'deltamethod'
coef(object, ...)
## S3 method for class 'deltamethod'
vcov(object, ...)
```

Arguments

object	an object of class "deltamethod".
...	other arguments.

Details

The coef function extracts the transformed estimates from objects of class "deltamethod". The vcov function extracts the corresponding variance-covariance matrix.

Value

Either a vector with the transformed estimates or a variance-covariance matrix.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[deltamethod](#) for the function to create deltamethod objects.

Description

Various extractor functions for objects of class "matreg".

Usage

```
## S3 method for class 'matreg'
coef(object, ...)
## S3 method for class 'matreg'
vcov(object, ...)
## S3 method for class 'matreg'
sigma(object, REML=TRUE, ...)

## S3 method for class 'matreg'
logLik(object, REML=FALSE, ...)
## S3 method for class 'matreg'
AIC(object, ..., k=2, correct=FALSE, REML=FALSE)
## S3 method for class 'matreg'
BIC(object, ..., REML=FALSE)

## S3 method for class 'matreg'
confint(object, parm, level, digits, ...)
## S3 method for class 'confint.matreg'
print(x, digits=x$digits, ...)
```

Arguments

object	an object of class "matreg".
REML	logical whether the returned value should be based on ML or REML estimation.
k	numeric value to specify the penalty per parameter. The default (k=2) is the classical AIC. See AIC for more details.
correct	logical to specify whether the regular (default) or corrected (i.e., AICc) should be extracted.

For confint():

parm	this argument is here for compatibility with the generic function confint , but is (currently) ignored.
level	numeric value between 0 and 100 to specify the confidence interval level (see here for details). If unspecified, the default is to take the value from the object.
digits	optional integer to specify the number of decimal places to which the results should be rounded. If unspecified, the default is to take the value from the object.
x	an object of class "confint.matreg".
...	other arguments.

Details

The `coef` function extracts the estimated (standardized) regression coefficients from objects of class "matreg". The `vcov` function extracts the corresponding variance-covariance matrix (note: the `se` function can also be used to extract the standard errors). The `confint` function extracts the confidence intervals.

Under the 'Regular *R* Matrix' case (see `matreg`), the `sigma` function extracts the square root of the estimated error variance (by default, based on the unbiased estimate of the error variance). The `logLik`, `AIC`, and `BIC` functions extract the corresponding values (note: for compatibility with the behavior for `lm` objects, these values are based by default on ML estimation).

Value

Depending on the function, either a vector, a matrix, or a scalar with the extracted value(s).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`matreg` for the function to create `matreg` objects and `predict.matreg` to compute predicted values based on `matreg` objects.

Examples

```
### fit a regression model with lm() to the 'mtcars' dataset
res <- lm(mpg ~ hp + wt + am, data=mtcars)
summary(res)
coef(res)
vcov(res)
se(res)
sigma(res)
confint(res)
logLik(res)
AIC(res)
BIC(res)

### covariance matrix of the dataset
S <- cov(mtcars)

### fit the same regression model using matreg()
res <- matreg(mpg ~ hp + wt + am, R=S, cov=TRUE,
              means=colMeans(mtcars), n=nrow(mtcars))
summary(res)
coef(res)
```

```
vcov(res)
se(res)
sigma(res)
confint(res)
logLik(res)
AIC(res)
BIC(res)
```

```
coef.permutest.rma.uni
```

Extract the Model Coefficient Table from 'permutest.rma.uni' Objects

Description

Function to extract the estimated model coefficients, corresponding standard errors, test statistics, p-values (based on the permutation tests), and confidence interval bounds from objects of class "permutest.rma.uni".

Usage

```
## S3 method for class 'permutest.rma.uni'
coef(object, ...)
```

Arguments

object	an object of class "permutest.rma.uni".
...	other arguments.

Value

A data frame with the following elements:

estimate	estimated model coefficient(s).
se	corresponding standard error(s).
zval	corresponding test statistic(s).
pval	p-value(s) based on the permutation test(s).
ci.lb	lower bound of the (permutation-based) confidence interval(s).
ci.ub	upper bound of the (permutation-based) confidence interval(s).

When the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="adhoc"`, then `zval` is called `tval` in the data frame that is returned by the function.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[permutest](#) for the function to conduct permutation tests and [rma.uni](#) for the function to fit models for which permutation tests can be conducted.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### carry out permutation test
## Not run:
set.seed(1234) # for reproducibility
sav <- permutest(res)
coef(sav)

## End(Not run)
```

coef.rma

Extract the Model Coefficients and Coefficient Table from 'rma' and 'summary.rma' Objects

Description

Function to extract the estimated model coefficients from objects of class "rma". For objects of class "summary.rma", the model coefficients, corresponding standard errors, test statistics, p-values, and confidence interval bounds are extracted.

Usage

```
## S3 method for class 'rma'
coef(object, ...)
## S3 method for class 'summary.rma'
coef(object, ...)
```

Arguments

object an object of class "rma" or "summary.rma".
 ... other arguments.

Value

Either a vector with the estimated model coefficient(s) or a data frame with the following elements:

estimate	estimated model coefficient(s).
se	corresponding standard error(s).
zval	corresponding test statistic(s).
pval	corresponding p-value(s).
ci.lb	corresponding lower bound of the confidence interval(s).
ci.ub	corresponding upper bound of the confidence interval(s).

When the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="adhoc"`, then `zval` is called `tval` in the data frame that is returned by the function.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which model coefficients/tables can be extracted.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### extract model coefficients
coef(res)

### extract model coefficient table
coef(summary(res))
```

Description

Functions to compute confidence intervals for the model coefficients, variance components, and other parameters in meta-analytic models.

Usage

```
## S3 method for class 'rma.uni'
confint(object, parm, level, fixed=FALSE, random=TRUE, type,
        digits, transf, targs, verbose=FALSE, control, ...)

## S3 method for class 'rma.mh'
confint(object, parm, level, digits, transf, targs,...)

## S3 method for class 'rma.peto'
confint(object, parm, level, digits, transf, targs, ...)

## S3 method for class 'rma.glmm'
confint(object, parm, level, digits, transf, targs, ...)

## S3 method for class 'rma.mv'
confint(object, parm, level, fixed=FALSE, sigma2, tau2, rho, gamma2, phi,
        digits, transf, targs, verbose=FALSE, control, ...)

## S3 method for class 'rma.uni.selmodel'
confint(object, parm, level, fixed=FALSE, tau2, delta,
        digits, transf, targs, verbose=FALSE, control, ...)

## S3 method for class 'rma.ls'
confint(object, parm, level, fixed=FALSE, alpha,
        digits, transf, targs, verbose=FALSE, control, ...)
```

Arguments

object	an object of class "rma.uni", "rma.mh", "rma.peto", "rma.mv", "rma.uni.selmodel", or "rma.ls". The method is not (yet) implemented for objects of class "rma.glmm".
parm	this argument is here for compatibility with the generic function <code>confint</code> , but is (currently) ignored.
fixed	logical to specify whether confidence intervals for the model coefficients should be returned.
random	logical to specify whether a confidence interval for the amount of (residual) heterogeneity should be returned.

type	optional character string to specify the method for computing the confidence interval for the amount of (residual) heterogeneity (either "QP", "GENQ", "PL", or "HT").
sigma2	integer to specify for which σ^2 parameter a confidence interval should be obtained.
tau2	integer to specify for which τ^2 parameter a confidence interval should be obtained.
rho	integer to specify for which ρ parameter the confidence interval should be obtained.
gamma2	integer to specify for which γ^2 parameter a confidence interval should be obtained.
phi	integer to specify for which ϕ parameter a confidence interval should be obtained.
delta	integer to specify for which δ parameter a confidence interval should be obtained.
alpha	integer to specify for which α parameter a confidence interval should be obtained.
level	numeric value between 0 and 100 to specify the confidence interval level (see here for details). If unspecified, the default is to take the value from the object.
digits	optional integer to specify the number of decimal places to which the results should be rounded. If unspecified, the default is to take the value from the object.
transf	optional argument to specify a function to transform the model coefficients and interval bounds (e.g., transf=exp; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified under transf.
verbose	logical to specify whether output should be generated on the progress of the iterative algorithms used to obtain the confidence intervals (the default is FALSE). See 'Details'.
control	list of control values for the iterative algorithms. If unspecified, default values are used. See 'Note'.
...	other arguments.

Details

Confidence intervals for the model coefficients can be obtained by setting fixed=TRUE and are simply the usual Wald-type intervals (which are also shown when printing the fitted object).

Other parameter(s) for which confidence intervals can be obtained depend on the model object:

- For objects of class "rma.uni" obtained with the [rma.uni](#) function, a confidence interval for the amount of (residual) heterogeneity (i.e., τ^2) can be obtained by setting random=TRUE (which is the default). The interval is obtained iteratively either via the Q-profile method or via the generalized Q-statistic method (Hartung and Knapp, 2005; Viechtbauer, 2007; Jackson, 2013; Jackson et al., 2014). The latter is automatically used when the model was fitted with

method="GENQ" or method="GENQM", the former is used in all other cases. Either method provides an exact confidence interval for τ^2 in random- and mixed-effects models. The square root of the interval bounds is also returned for easier interpretation. Confidence intervals for I^2 and H^2 are also provided (Higgins & Thompson, 2002). Since I^2 and H^2 are monotonic transformations of τ^2 (for details, see [print](#)), the confidence intervals for I^2 and H^2 are also exact. One can also set type="PL" to obtain a profile likelihood confidence interval for τ^2 (and corresponding CIs for I^2 and H^2), which would be more consistent with the use of ML/REML estimation, but is not exact (see 'Note'). For models without moderators (i.e., random-effects models), one can also set type="HT", in which case the 'test-based method' (method III in Higgins & Thompson, 2002) is used to construct confidence intervals for τ^2 , I^2 , and H^2 (see also Borenstein et al., 2009, chapter 16). However, note that this method tends to yield confidence intervals that are too narrow when the amount of heterogeneity is large.

- For objects of class "rma.mv" obtained with the [rma.mv](#) function, confidence intervals are obtained by default for all variance and correlation components of the model. Alternatively, one can use the sigma2, tau2, rho, gamma2, or phi arguments to specify for which variance/correlation parameter a confidence interval should be obtained. Only one of these arguments can be used at a time. A single integer is used to specify the number of the parameter. The function provides profile likelihood confidence intervals for these parameters. It is a good idea to examine the corresponding profile likelihood plots (via the [profile](#) function) to make sure that the bounds obtained are sensible.
- For selection model objects of class "rma.uni.selmodel" obtained with the [selmodel](#) function, confidence intervals are obtained by default for τ^2 (for models where this is an estimated parameter) and all selection model parameters. Alternatively, one can choose to obtain a confidence interval only for τ^2 by setting tau2=TRUE or for one of the selection model parameters by specifying its number via the delta argument. The function provides profile likelihood confidence intervals for these parameters. It is a good idea to examine the corresponding profile likelihood plots (via the [profile](#) function) to make sure that the bounds obtained are sensible.
- For location-scale model objects of class "rma.ls" obtained with the [rma.uni](#) function, confidence intervals are obtained by default for all scale parameters. Alternatively, one can choose to obtain a confidence interval for one of the scale parameters by specifying its number via the alpha argument. The function provides profile likelihood confidence intervals for these parameters. It is a good idea to examine the corresponding profile likelihood plots (via the [profile](#) function) to make sure that the bounds obtained are sensible.

The methods used to find confidence intervals for these parameters are iterative and require the use of the [uniroot](#) function. By default, the desired accuracy (tol) is set equal to `.Machine$double.eps^0.25` and the maximum number of iterations (maxiter) to 1000. These values can be adjusted with `control=list(tol=value, maxiter=value)`, but the defaults should be adequate for most purposes. If verbose=TRUE, output is generated on the progress of the iterative algorithms. This is especially useful when model fitting is slow, in which case finding the confidence interval bounds can also take considerable amounts of time.

When using the [uniroot](#) function, one must also set appropriate end points of the interval to be searched for the confidence interval bounds. The function sets some sensible defaults for the end points, but it may happen that the function is only able to determine that a bound is below/above a certain limit (this is indicated in the output accordingly with < or > signs). It can also happen that the model cannot be fitted or does not converge especially at the extremes of the interval to

be searched. This will result in missing (NA) bounds and corresponding warnings. It may then be necessary to adjust the end points manually (see ‘Note’).

Finally, it is also possible that the lower and upper confidence interval bounds for a variance component both fall below zero. Since both bounds then fall outside of the parameter space, the confidence interval then consists of the null/empty set. Alternatively, one could interpret this as a confidence interval with bounds $[0, 0]$ or as indicating ‘highly/overly homogeneous’ data.

Value

An object of class `"confint.rma"`. The object is a list with either one or two elements (named `fixed` and `random`) with the following elements:

<code>estimate</code>	estimate of the model coefficient, variance/correlation component, or selection model parameter.
<code>ci.lb</code>	lower bound of the confidence interval.
<code>ci.ub</code>	upper bound of the confidence interval.

When obtaining confidence intervals for multiple components, the object is a list of class `"list.confint.rma"`, where each element is a `"confint.rma"` object as described above.

The results are formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function.

Note

When computing a CI for τ^2 for objects of class `"rma.uni"`, the estimate of τ^2 will usually fall within the CI bounds provided by the Q-profile method. However, this is not guaranteed. Depending on the method used to estimate τ^2 and the width of the CI, it can happen that the CI does not actually contain the estimate. Using the empirical Bayes or Paule-Mandel estimator of τ^2 when fitting the model (i.e., using `method="EB"` or `method="PM"`) usually ensures that the estimate of τ^2 falls within the CI (for `method="PMM"`, this is guaranteed). When `method="GENQ"` was used to fit the model, the corresponding CI obtained via the generalized Q-statistic method also usually contains the estimate τ^2 (for `method="GENQM"`, this is guaranteed). When using ML/REML estimation, the profile likelihood CI (obtained when setting `type="PL"`) is guaranteed to contain the estimate of τ^2 .

When computing a CI for τ^2 for objects of class `"rma.uni"`, the end points of the interval to be searched for the CI bounds are $[0, 100]$ (or, for the upper bound, ten times the estimate of τ^2 , whichever is greater). The upper bound should be large enough for most cases, but can be adjusted with `control=list(tau2.max=value)`. One can also adjust the lower end point with `control=list(tau2.min=value)`. You should only play around with this value if you know what you are doing.

For objects of class `"rma.mv"`, the function provides profile likelihood CIs for the variance/correlation parameters in the model. For variance components, the lower end point of the interval to be searched is set to 0 and the upper end point to the larger of 10 and 100 times the value of the component. For correlations, the function sets the lower end point to a sensible default depending on the type of variance structure chosen, while the upper end point is set to 1. One can adjust the lower and/or upper end points with `control=list(vc.min=value, vc.max=value)`. Also, the function adjusts the lower/upper end points when the model does not converge at these extremes (the end points

are then moved closer to the estimated value of the component). The total number of tries for setting/adjusting the end points in this manner is determined via `control=list(eptries=value)`, with the default being 10 tries.

For objects of class `"rma.uni.selmodel"` or `"rma.ls"`, the function also sets some sensible defaults for the end points of the interval to be searched for the CI bounds (of the τ^2 , δ , and α parameter(s)). One can again adjust the end points and the number of retries (as described above) with `control=list(vc.min=value, vc.max=value, eptries=value)`.

The Q-profile and generalized Q-statistic methods are both exact under the assumptions of the random- and mixed-effects models (i.e., normally distributed observed and true effects and known sampling variances). In practice, these assumptions are usually only approximately true, turning CIs for τ^2 also into approximations. Profile likelihood CIs are not exact by construction and rely on the asymptotic behavior of the likelihood ratio statistic, so they may be inaccurate in small samples, but they are inherently consistent with the use of ML/REML estimation.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Borenstein, M., Hedges, L. V., Higgins, J. P. T., & Rothstein, H. (2009). *Introduction to meta-analysis*. Chichester, UK: Wiley.

Hardy, R. J., & Thompson, S. G. (1996). A likelihood approach to meta-analysis with random effects. *Statistics in Medicine*, **15**(6), 619–629. [https://doi.org/10.1002/\(sici\)1097-0258\(19960330\)15:6%3C619::ai](https://doi.org/10.1002/(sici)1097-0258(19960330)15:6%3C619::ai)

Hartung, J., & Knapp, G. (2005). On confidence intervals for the among-group variance in the one-way random effects model with unequal error variances. *Journal of Statistical Planning and Inference*, **127**(1-2), 157–177. <https://doi.org/10.1016/j.jspi.2003.09.032>

Higgins, J. P. T., & Thompson, S. G. (2002). Quantifying heterogeneity in a meta-analysis. *Statistics in Medicine*, **21**(11), 1539–1558. <https://doi.org/10.1002/sim.1186>

Jackson, D. (2013). Confidence intervals for the between-study variance in random effects meta-analysis using generalised Cochran heterogeneity statistics. *Research Synthesis Methods*, **4**(3), 220–229. <https://doi.org/10.1186/s12874-016-0219-y>

Jackson, D., Turner, R., Rhodes, K., & Viechtbauer, W. (2014). Methods for calculating confidence and credible intervals for the residual between-study variance in random effects meta-regression models. *BMC Medical Research Methodology*, **14**, 103. <https://doi.org/10.1186/1471-2288-14-103>

Viechtbauer, W. (2007). Confidence intervals for the amount of heterogeneity in meta-analysis. *Statistics in Medicine*, **26**(1), 37–52. <https://doi.org/10.1002/sim.2514>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*, **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>

See Also

`rma.uni`, `rma.mh`, `rma.peto`, `rma.glmm`, `rma.mv`, and `selmodel` for functions to fit models for which confidence intervals can be computed.

[profile](#) for functions to create profile likelihood plots corresponding to profile likelihood confidence intervals.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### meta-analysis of the log risk ratios using a random-effects model
res <- rma(yi, vi, data=dat, method="REML")

### confidence interval for the total amount of heterogeneity
confint(res)

### mixed-effects model with absolute latitude in the model
res <- rma(yi, vi, mods = ~ ablat, data=dat)

### confidence interval for the residual amount of heterogeneity
confint(res)

### multilevel random-effects model
res <- rma.mv(yi, vi, random = ~ 1 | district/school, data=dat.konstantopoulos2011)

### profile plots and confidence intervals for the variance components
## Not run:
par(mfrow=c(2,1))
profile(res, sigma2=1, steps=40, cline=TRUE)
sav <- confint(res, sigma2=1)
sav
abline(v=sav$random[1,2:3], lty="dotted")
profile(res, sigma2=2, steps=40, cline=TRUE)
sav <- confint(res, sigma2=2)
sav
abline(v=sav$random[1,2:3], lty="dotted")

## End(Not run)

### multivariate parameterization of the model
res <- rma.mv(yi, vi, random = ~ school | district, data=dat.konstantopoulos2011)

### profile plots and confidence intervals for the variance component and correlation
## Not run:
par(mfrow=c(2,1))
profile(res, tau2=1, steps=40, cline=TRUE)
sav <- confint(res, tau2=1)
sav
abline(v=sav$random[1,2:3], lty="dotted")
profile(res, rho=1, steps=40, cline=TRUE)
sav <- confint(res, rho=1)
sav
abline(v=sav$random[1,2:3], lty="dotted")

## End(Not run)
```

contrmat

*Construct Contrast Matrix for Two-Group Comparisons***Description**

Function to construct a matrix that indicates which two groups have been contrasted against each other in each row of a dataset.

Usage

```
contrmat(data, grp1, grp2, last, shorten=FALSE, minlen=2, check=TRUE, append=TRUE)
```

Arguments

data	a data frame in wide format.
grp1	either the name (given as a character string) or the position (given as a single number) of the first group variable in the data frame.
grp2	either the name (given as a character string) or the position (given as a single number) of the second group variable in the data frame.
last	optional character string to specify which group will be placed in the last column of the matrix (must be one of the groups in the group variables). If not given, the most frequently occurring second group is placed last.
shorten	logical to specify whether the variable names corresponding to the group names should be shortened (the default is FALSE).
minlen	integer to specify the minimum length of the shortened variable names (the default is 2).
check	logical to specify whether the variables names should be checked to ensure that they are syntactically valid variable names and if not, they are adjusted (by make.names) so that they are (the default is TRUE).
append	logical to specify whether the contrast matrix should be appended to the data frame specified via the data argument (the default is TRUE). If append=FALSE, only the contrast matrix is returned.

Details

The function can be used to construct a matrix that indicates which two groups have been contrasted against each other in each row of a data frame (with 1 for the first group, -1 for the second group, and 0 otherwise).

The grp1 and grp2 arguments are used to specify the group variables in the dataset (either as character strings or as numbers indicating the column positions of these variables in the dataset). Optional argument last is used to specify which group will be placed in the last column of the matrix.

If shorten=TRUE, the variable names corresponding to the group names are shortened (to at least minlen; the actual length might be longer to ensure uniqueness of the variable names).

The examples below illustrate the use of this function.

Value

A matrix with as many variables as there are groups.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[to.wide](#) for a function to create ‘wide’ format datasets.

[dat.senn2013](#), [dat.hasselblad1998](#), [dat.lopez2019](#) for illustrative examples.

Examples

```
### restructure to wide format
dat <- dat.senn2013
dat <- dat[c(1,4,3,2,5,6)]
dat <- to.wide(dat, study="study", grp="treatment", ref="placebo", grpvars=4:6)
dat

### add contrast matrix
dat <- contrmat(dat, grp1="treatment.1", grp2="treatment.2")
dat

### data in long format
dat <- dat.hasselblad1998
dat

### restructure to wide format
dat <- to.wide(dat, study="study", grp="trt", ref="no_contact", grpvars=6:7)
dat

### add contrast matrix
dat <- contrmat(dat, grp1="trt.1", grp2="trt.2", shorten=TRUE, minlen=3)
dat
```

Description

Function to reconstruct the cell frequencies of 2×2 tables based on other summary statistics.

Usage

```
conv.2x2(ori, ri, x2i, ni, n1i, n2i, sens, spec, ppv, npv,
         correct=TRUE, drop01=TRUE, data, include, var.names=c("ai", "bi", "ci", "di"),
         append=TRUE, replace="ifna", ...)
```

Arguments

<code>ori</code>	optional vector with the odds ratios corresponding to the tables.
<code>ri</code>	optional vector with the phi coefficients corresponding to the tables.
<code>x2i</code>	optional vector with the (signed) chi-square statistics corresponding to the tables.
<code>ni</code>	vector with the total sample sizes.
<code>n1i</code>	vector with the marginal counts for the outcome of interest on the first variable.
<code>n2i</code>	vector with the marginal counts for the outcome of interest on the second variable.
<code>sens</code>	optional vector with the sensitivities corresponding to the tables.
<code>spec</code>	optional vector with the specificities corresponding to the tables.
<code>ppv</code>	optional vector with the positive predictive values corresponding to the tables.
<code>npv</code>	optional vector with the negative predictive values corresponding to the tables.
<code>correct</code>	optional logical (or vector thereof) to specify whether chi-square statistics were computed using Yates's correction for continuity (the default is TRUE).
<code>drop01</code>	logical to specify whether studies where sens, spec, ppv, and/or npv is equal to 0 or 1 should be dropped (the default is TRUE).
<code>data</code>	optional data frame containing the variables given to the arguments above.
<code>include</code>	optional (logical or numeric) vector to specify the subset of studies for which the cell frequencies should be reconstructed.
<code>var.names</code>	character vector with four elements to specify the names of the variables for the reconstructed cell frequencies (the default is <code>c("ai", "bi", "ci", "di")</code>).
<code>append</code>	logical to specify whether the data frame provided via the <code>data</code> argument should be returned together with the reconstructed values (the default is TRUE).
<code>replace</code>	character string or logical to specify how values in <code>var.names</code> should be replaced (only relevant when using the <code>data</code> argument and if variables in <code>var.names</code> already exist in the data frame). See the 'Value' section for more details.
<code>...</code>	other arguments.

Details

For meta-analyses based on 2×2 table data, the problem often arises that some studies do not directly report the cell frequencies. The present function allows the reconstruction of such tables based on other summary statistics.

In particular, assume that the data of interest for a particular study are of the form:

	variable 2, outcome +	variable 2, outcome -	total
variable 1, outcome +	ai	bi	n1i
variable 1, outcome -	ci	di	
total	n2i		ni

where ai, bi, ci, and di denote the cell frequencies (i.e., the number of individuals falling into a particular category), n1i (i.e., ai+bi) and n2i (i.e., ai+ci) are the marginal totals for the outcome of interest on the first and second variable, respectively, and ni is the total sample size (i.e., ai+bi+ci+di) of the study.

For example, if variable 1 denotes two different groups (e.g., treated versus control) and variable 2 indicates whether a particular outcome of interest has occurred or not (e.g., death, complications, failure to improve under the treatment), then n1i denotes the number of individuals in the treatment group, but n2i is *not* the number of individuals in the control group, but the total number of individuals who experienced the outcome of interest on variable 2. **Note that the meaning of n2i is therefore different here compared to the `escal` function (where n2i denotes ci+di).**

If a study does not report the cell frequencies, but it reports the total sample size (which can be specified via the ni argument), the two marginal counts (which can be specified via the n1i and n2i arguments), and some other statistic corresponding to the table, then it may be possible to reconstruct the cell frequencies. The present function currently allows this for three different cases:

1. If the odds ratio

$$OR = \frac{a_i d_i}{b_i c_i}$$

is known, then the cell frequencies can be reconstructed (Bonett, 2007). Odds ratios can be specified via the `ori` argument.

2. If the phi coefficient

$$\phi = \frac{a_i d_i - b_i c_i}{\sqrt{n_{1i}(n_i - n_{1i})n_{2i}(n_i - n_{2i})}}$$

is known, then the cell frequencies can again be reconstructed (own derivation). Phi coefficients can be specified via the `ri` argument.

3. If the chi-square statistic from Pearson's chi-square test of independence is known (which can be specified via the `x2i` argument), then it can be used to recalculate the phi coefficient and hence again the cell frequencies can be reconstructed. However, the chi-square statistic does not carry information about the sign of the phi coefficient. Therefore, values specified via the `x2i` argument can be positive or negative, which allows the specification of the correct sign. Also, when using a chi-square statistic as input, it is assumed that it was computed using Yates's correction for continuity (unless `correct=FALSE`). If the chi-square statistic is not known, but its p-value, one can first back-calculate the chi-square statistic using `qchisq(<p-value>, df=1, lower.tail=FALSE)`.

Typically, the odds ratio, phi coefficient, or chi-square statistic (or its p-value) that can be extracted from a study will be rounded to a certain degree. The calculations underlying the function are exact only for unrounded values. Rounding can therefore introduce some discrepancies between the actual cell frequencies and the reconstructed ones.

If a marginal total is unknown, then external information needs to be used to 'guestimate' the number of individuals that experienced the outcome of interest on this variable. Depending on the

accuracy of such an estimate, the reconstructed cell frequencies will be more or less accurate and need to be treated with due caution.

The true marginal counts also put constraints on the possible values for the odds ratio, phi coefficient, and chi-square statistic. If a marginal count is replaced by a guestimate which is not compatible with the given statistic, one or more reconstructed cell frequencies may be negative. The function issues a warning if this happens and sets the cell frequencies to NA for such a study.

If only one of the two marginal counts is unknown but a 95% CI for the odds ratio is also available, then the **estimraw** package can also be used to reconstruct the corresponding cell frequencies (Di Pietrantonj, 2006; but see Veroniki et al., 2013, for some cautions).

Diagnostic Studies:

The present function can also be used to reconstruct 2×2 table data for diagnostic studies. Here, the table is assumed to be of the form:

	case	control
test+	ai	bi
test-	ci	di

where ai denotes the number of true positives, bi the number of false positives, ci the number of false negatives, and di the number of true negatives.

If the total sample size ($n_i = ai + bi + ci + di$) and the marginal totals are known (i.e., the number of positive tests, n_{1i} , and the number of cases, n_{2i}), then the diagnostic odds ratio (i.e., $ori = ai * di / (bi * ci)$) would be sufficient to reconstruct the table and the present function can be used as described above.

On the other hand, if the total sample size of the study (i.e., $n_i = ai + bi + ci + di$), the sensitivity (i.e., $sens = ai / (ai + ci)$), specificity (i.e., $spec = di / (bi + di)$), positive predictive value (i.e., $ppv = ai / (ai + bi)$), and negative predictive value (i.e., $npv = di / (ci + di)$) are known, then this is also sufficient information to recreate the table. Actually, only three of the four diagnostic accuracy measures are needed to reconstruct the table.

In practice, when such accuracy measures are reported, the values are typically rounded to some extent. This introduces inaccuracies into the reconstruction. The present function uses optimization methods to reconstruct the table counts so that the discrepancy between the reported measures and the reconstructed ones are minimized. This is not guaranteed to reconstruct the actual table exactly, but should usually yield a close match, especially if all four measures are available. In some rare cases, the reconstruction may also fail even if all four measures are reported. This often happens if at least one of the reported accuracy measures is equal to 0 or 1. By default (i.e., when `drop01=TRUE`), such studies are automatically dropped (i.e., the reconstructed cell frequencies are set to NA).

Value

If the data argument was not specified or `append=FALSE`, a data frame with four variables called `var.names` with the reconstructed cell frequencies.

If data was specified and `append=TRUE`, then the original data frame is returned. If `var.names[j]` (for $j \in 1, \dots, 4$) is a variable in data and `replace="ifna"` (or `replace=FALSE`), then only missing values in this variable are replaced with the estimated frequencies (where possible) and otherwise a new variable called `var.names[j]` is added to the data frame.

If `replace="all"` (or `replace=TRUE`), then all values in `var.names[j]` where a reconstructed cell frequency can be computed are replaced, even for cases where the value in `var.names[j]` is not missing. However, non-missing values are never replaced with missing values. To also allow this (i.e., to completely replace all values), use `replace="complete"`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Bonett, D. G. (2007). Transforming odds ratios into correlations for meta-analytic research. *American Psychologist*, **62**(3), 254–255. <https://doi.org/10.1037/0003-066x.62.3.254>
- Di Pietrantonj, C. (2006). Four-fold table cell frequencies imputation in meta analysis. *Statistics in Medicine*, **25**(13), 2299–2322. <https://doi.org/10.1002/sim.2287>
- Veroniki, A. A., Pavlides, M., Patsopoulos, N. A., & Salanti, G. (2013). Reconstructing 2 x 2 contingency tables from odds ratios using the Di Pietrantonj method: Difficulties, constraints and impact in meta-analysis results. *Research Synthesis Methods*, **4**(1), 78–94. <https://doi.org/10.1002/jrsm.1061>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[escalc](#) for a function to compute various effect size measures based on 2×2 table data.

Examples

```
#####

### demonstration that the reconstruction of the 2x2 table works
### (note: the values in rows 2, 3, and 4 correspond to those in row 1)
dat <- data.frame(ai=c(36,NA,NA,NA), bi=c(86,NA,NA,NA), ci=c(20,NA,NA,NA), di=c(98,NA,NA,NA),
                  oddsratio=NA, phi=NA, chisq=NA, ni=NA, n1i=NA, n2i=NA)
dat$oddsratio[2] <- round(exp(escalc(measure="OR", ai=ai, bi=bi, ci=ci, di=di, data=dat)$yi[1]), 2)
dat$phi[3] <- round(escalc(measure="PHI", ai=ai, bi=bi, ci=ci, di=di, data=dat)$yi[1], 2)
dat$chisq[4] <- round(chisq.test(matrix(c(t(dat[1,1:4])), nrow=2, byrow=TRUE))$statistic, 2)
dat$ni[2:4] <- with(dat, ai[1] + bi[1] + ci[1] + di[1])
dat$n1i[2:4] <- with(dat, ai[1] + bi[1])
dat$n2i[2:4] <- with(dat, ai[1] + ci[1])
dat

### reconstruct cell frequencies for rows 2, 3, and 4
dat <- conv.2x2(ri=phi, ori=oddsratio, x2i=chisq, ni=ni, n1i=n1i, n2i=n2i, data=dat)
dat

### same example but with cell frequencies that are 10 times as large
dat <- data.frame(ai=c(360,NA,NA,NA), bi=c(860,NA,NA,NA), ci=c(200,NA,NA,NA), di=c(980,NA,NA,NA),
                  oddsratio=NA, phi=NA, chisq=NA, ni=NA, n1i=NA, n2i=NA)
dat$oddsratio[2] <- round(exp(escalc(measure="OR", ai=ai, bi=bi, ci=ci, di=di, data=dat)$yi[1]), 2)
dat$phi[3] <- round(escalc(measure="PHI", ai=ai, bi=bi, ci=ci, di=di, data=dat)$yi[1], 2)
```

```

dat$chisq[4] <- round(chisq.test(matrix(c(t(dat[1,1:4])), nrow=2, byrow=TRUE))$statistic, 2)
dat$ni[2:4] <- with(dat, ai[1] + bi[1] + ci[1] + di[1])
dat$n1i[2:4] <- with(dat, ai[1] + bi[1])
dat$n2i[2:4] <- with(dat, ai[1] + ci[1])
dat <- conv.2x2(ri=phi, ori=oddsratio, x2i=chisq, ni=ni, n1i=n1i, n2i=n2i, data=dat)
dat # slight inaccuracy in row 3 due to rounding

### demonstrate what happens when a true marginal count is gestimated
escalc(measure="PHI", ai=176, bi=24, ci=72, di=128)
conv.2x2(ri=0.54, ni=400, n1i=200, n2i=248) # using the true marginal counts
conv.2x2(ri=0.54, ni=400, n1i=200, n2i=200) # marginal count for variable 2 is gestimated
conv.2x2(ri=0.54, ni=400, n1i=200, n2i=50) # marginal count for variable 2 is incompatible

### demonstrate that using the correct sign for the chi-square statistic is important
chisq <- round(chisq.test(matrix(c(40,60,60,40), nrow=2, byrow=TRUE))$statistic, 2)
conv.2x2(x2i=-chisq, ni=200, n1i=100, n2i=100) # correct reconstruction
conv.2x2(x2i=chisq, ni=200, n1i=100, n2i=100) # incorrect reconstruction

### demonstrate use of the 'correct' argument
tab <- matrix(c(28,14,12,18), nrow=2, byrow=TRUE)
chisq <- round(chisq.test(tab)$statistic, 2) # chi-square test with Yates' continuity correction
conv.2x2(x2i=chisq, ni=72, n1i=42, n2i=40) # correct reconstruction
chisq <- round(chisq.test(tab, correct=FALSE)$statistic, 2) # without Yates' continuity correction
conv.2x2(x2i=chisq, ni=72, n1i=42, n2i=40) # incorrect reconstruction
conv.2x2(x2i=chisq, ni=72, n1i=42, n2i=40, correct=FALSE) # correct reconstruction

### recalculate chi-square statistic based on p-value
pval <- round(chisq.test(tab)$p.value, 2)
chisq <- qchisq(pval, df=1, lower.tail=FALSE)
conv.2x2(x2i=chisq, ni=72, n1i=42, n2i=40)

#####

### reconstruct the 2x2 table counts for a diagnostic study
tab <- matrix(c(28,5,7,18), nrow=2, byrow=TRUE)
tab

### reconstruct from the diagnostic odds ratio and the marginals
dor <- tab[1,1] * tab[2,2] / (tab[1,2] * tab[2,1])
cases <- tab[1,1] + tab[2,1]
pos <- tab[1,1] + tab[1,2]
n <- sum(tab)
conv.2x2(ori=dor, n1i=pos, n2i=cases, ni=n)

### reconstruct from the diagnostic accuracy measures
sens <- tab[1,1] / sum(tab[,1])
spec <- tab[2,2] / sum(tab[,2])
ppv <- tab[1,1] / sum(tab[1,])
npv <- tab[2,2] / sum(tab[2,])
n <- sum(tab)
conv.2x2(sens=sens, spec=spec, ppv=ppv, npv=npv, ni=n)

### show that only three out of the four diagnostic statistics are needed

```

```

conv.2x2(sens=sens, spec=spec, ppv=ppv, ni=n)
conv.2x2(sens=sens, spec=spec, npv=npv, ni=n)
conv.2x2(sens=sens, ppv=ppv, npv=npv, ni=n)
conv.2x2(spec=spec, ppv=ppv, npv=npv, ni=n)

### reconstruct the 2x2 table counts from rounded statistics
dor <- round(dor, digits=2)
sens <- round(sens, digits=2)
spec <- round(spec, digits=2)
ppv <- round(ppv, digits=2)
npv <- round(npv, digits=2)
conv.2x2(ori=dor, n1i=pos, n2i=cases, ni=n)
conv.2x2(sens=sens, spec=spec, ppv=ppv, npv=npv, ni=n)

#####

```

conv.delta	<i>Transform Estimates and their Sampling Variances using the Delta Method</i>
------------	--

Description

Function to transform estimates and their sampling variances using the delta method.

Usage

```
conv.delta(yi, vi, ni, data, include, transf, var.names, append=TRUE, replace="ifna", ...)
```

Arguments

yi	vector with the effect size estimates.
vi	vector with the corresponding sampling variances.
ni	vector with the total sample sizes of the studies.
data	optional data frame containing the variables given to the arguments above.
include	optional (logical or numeric) vector to specify the subset of studies for which the transformation should be carried out.
transf	a function which should be used for the transformation.
var.names	character vector with two elements to specify the name of the variable for the transformed estimates and the name of the variable for the corresponding sampling variances (if data is an object of class "escalc", the var.names are taken from the object; otherwise the defaults are "yi" and "vi").
append	logical to specify whether the data frame provided via the data argument should be returned together with the estimated values (the default is TRUE).
replace	character string or logical to specify how values in var.names should be replaced (only relevant when using the data argument and if variables in var.names already exist in the data frame). See the 'Value' section for more details.
...	other arguments for the transformation function.

Details

The `escalc` function can be used to compute a wide variety of effect size measures. In some cases, it may be necessary to transform one type of measure to another. The present function provides a general method for doing so via the `delta method` (e.g., van der Vaart, 1998), which briefly works as follows.

Let y_i denote the effect size estimate for a particular study and v_i the corresponding sampling variance. Then $f(y_i)$ will be the transformed estimate, where $f(\cdot)$ is the function specified via the `transf` argument. The sampling variance of the transformed estimate is then computed with $v_i \times f'(y_i)^2$, where $f'(y_i)$ denotes the derivative of $f(\cdot)$ evaluated at y_i . The present function computes the derivative numerically using the `grad` function from the `numDeriv` package.

The first argument of the function specified via `transf` should be for the effect size estimates. The function can have additional arguments, which can be specified via the `...` argument. However, due to the manner in which these additional arguments are evaluated, they cannot have names that match one of the arguments of the `grad` function (an error will be issued if such a naming clash is detected).

Optionally, one can use the `ni` argument to supply the total sample sizes of the studies. This has no relevance for the calculations done by the present function, but some other functions may use this information (e.g., when drawing a funnel plot with the `funnel` function and one adjusts the `yaxis` argument to one of the options that puts the sample sizes or some transformation thereof on the y-axis).

Value

If the `data` argument was not specified or `append=FALSE`, a data frame of class `c("escalc", "data.frame")` with two variables called `var.names[1]` (by default `"yi"`) and `var.names[2]` (by default `"vi"`) with the transformed estimates and the corresponding sampling variances (computed as described above).

If `data` was specified and `append=TRUE`, then the original data frame is returned. If `var.names[1]` is a variable in `data` and `replace="ifna"` (or `replace=FALSE`), then only missing values in this variable are replaced with the transformed estimates (where possible) and otherwise a new variable called `var.names[1]` is added to the data frame. Similarly, if `var.names[2]` is a variable in `data` and `replace="ifna"` (or `replace=FALSE`), then only missing values in this variable are replaced with the sampling variances calculated as described above (where possible) and otherwise a new variable called `var.names[2]` is added to the data frame.

If `replace="all"` (or `replace=TRUE`), then all values in `var.names[1]` and `var.names[2]` are replaced, even for cases where the value in `var.names[1]` and `var.names[2]` is not missing. However, non-missing values are never replaced with missing values. To also allow this (i.e., to completely replace all values), use `replace="complete"`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

van der Vaart, A. W. (1998). *Asymptotic statistics*. Cambridge, UK: Cambridge University Press.

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`escalc` for a function to compute various effect size measures and `deltamethod` for a function to apply the multivariate delta method to a set of estimates.

Examples

```
#####

### the following examples illustrate that the use of the delta method (with numeric derivatives)
### yields essentially identical results as the analytic calculations that are done by escalc()

### compute logit transformed proportions and corresponding sampling variances for two studies
escalc(measure="PLO", xi=c(5,12), ni=c(40,80))

### compute raw proportions and corresponding sampling variances for the two studies
dat <- escalc(measure="PR", xi=c(5,12), ni=c(40,80))
dat

### apply the logit transformation (note: this yields the same values as above with measure="PLO")
conv.delta(dat$yi, dat$vi, transf=transf.logit)

### using the 'data' argument
conv.delta(yi, vi, data=dat, transf=transf.logit, var.names=c("yi.t", "vi.t"))

### or replace the existing 'yi' and 'vi' values
conv.delta(yi, vi, data=dat, transf=transf.logit, replace="all")

#####

### use escalc() with measure D2ORN which transforms standardized mean differences (computed
### from means and standard deviations) into the corresponding log odds ratios
escalc(measure="D2ORN", m1i=m1i, sd1i=sd1i, n1i=n1i,
      m2i=m2i, sd2i=sd2i, n2i=n2i, data=dat.normand1999)

### use escalc() to compute standardized mean differences (without the usual bias correction) and
### then apply the same transformation to the standardized mean differences
dat <- escalc(measure="SMD", m1i=m1i, sd1i=sd1i, n1i=n1i,
      m2i=m2i, sd2i=sd2i, n2i=n2i, data=dat.normand1999, correct=FALSE)
conv.delta(yi, vi, data=dat, transf=transf.dtolnor.norm, replace="all")

#####

### an example where the transformation function takes additional arguments

### use escalc() with measure RPB which transforms standardized mean differences (computed
### from means and standard deviations) into the corresponding point-biserial correlations
escalc(measure="RPB", m1i=m1i, sd1i=sd1i, n1i=n1i,
      m2i=m2i, sd2i=sd2i, n2i=n2i, data=dat.normand1999)
```

```

### use escalc() to compute standardized mean differences (without the usual bias correction) and
### then apply the same transformation to the standardized mean differences
dat <- escalc(measure="SMD", m1i=m1i, sd1i=sd1i, n1i=n1i,
              m2i=m2i, sd2i=sd2i, n2i=n2i, data=dat.normand1999, correct=FALSE)
conv.delta(yi, vi, data=dat, transf=transf.dtorpb, n1i=n1i, n2i=n2i, replace="all")

#####

### a more elaborate example showing how this function could be used in the data
### preparation steps for a meta-analysis of standardized mean differences (SMDs)

dat <- data.frame(study=1:6,
  m1i=c(2.03,NA,NA,NA,NA,NA), sd1i=c(0.95,NA,NA,NA,NA,NA), n1i=c(32,95,145,NA,NA,NA),
  m2i=c(1.25,NA,NA,NA,NA,NA), sd2i=c(1.04,NA,NA,NA,NA,NA), n2i=c(30,99,155,NA,NA,NA),
  tval=c(NA,2.12,NA,NA,NA,NA), dval=c(NA,NA,0.37,NA,NA,NA),
  ai=c(NA,NA,NA,26,NA,NA), bi=c(NA,NA,NA,58,NA,NA),
  ci=c(NA,NA,NA,11,NA,NA), di=c(NA,NA,NA,74,NA,NA),
  or=c(NA,NA,NA,NA,2.56,NA), lower=c(NA,NA,NA,NA,1.23,NA), upper=c(NA,NA,NA,NA,5.30,NA),
  corr=c(NA,NA,NA,NA,NA,0.32), ntot=c(NA,NA,NA,NA,NA,86))
dat

### study types:
### 1) reports means and SDs so that the SMD can be directly calculated
### 2) reports the t-statistic from an independent samples t-test (and group sizes)
### 3) reports the standardized mean difference directly (and group sizes)
### 4) dichotomized the continuous dependent variable and reports the resulting 2x2 table
### 5) dichotomized the continuous dependent variable and reports an odds ratio with 95% CI
### 6) treated the group variable continuously and reports a Pearson product-moment correlation

### use escalc() to directly compute the SMD and its variance for studies 1, 2, and 3
dat <- escalc(measure="SMD", m1i=m1i, sd1i=sd1i, n1i=n1i,
              m2i=m2i, sd2i=sd2i, n2i=n2i, ti=tval, di=dval, data=dat)
dat

### use escalc() with measure OR2DN to compute the SMD value for study 4
dat <- escalc(measure="OR2DN", ai=ai, bi=bi, ci=ci, di=di, data=dat, replace=FALSE)
dat

### use conv.wald() to convert the OR and CI into the log odds ratio and its variance for study 5
dat <- conv.wald(out=or, ci.lb=lower, ci.ub=upper, data=dat,
                transf=log, var.names=c("lnor","vlnor"))
dat

### use conv.delta() to transform the log odds ratio into the SMD value for study 5
dat <- conv.delta(lnor, vlnor, data=dat,
                transf=transf.lnortod.norm, var.names=c("yi","vi"))
dat

### remove the lnor and vlnor variables (no longer needed)
dat$lnor <- NULL
dat$vlnor <- NULL

```

```

### use escalc() with measure COR to compute the sampling variance of ri for study 6
dat <- escalc(measure="COR", ri=corr, ni=ntot, data=dat, var.names=c("ri","vri"))
dat

### use conv.delta() to transform the correlation into the SMD value for study 6
dat <- conv.delta(ri, vri, data=dat, transf=transf.rtod, var.names=c("yi","vi"))
dat

### remove the ri and vri variables (no longer needed)
dat$ri <- NULL
dat$vri <- NULL

### now variable 'yi' is complete with the SMD values for all studies
dat

### fit an equal-effects model to the SMD values
rma(yi, vi, data=dat, method="EE")

#####

### a more elaborate example showing how this function could be used in the data
### preparation steps for a meta-analysis of correlation coefficients

dat <- data.frame(study=1:6,
  ri=c(0.42,NA,NA,NA,NA,NA),
  tval=c(NA,2.85,NA,NA,NA,NA),
  phi=c(NA,NA,NA,0.27,NA,NA),
  ni=c(93,182,NA,112,NA,NA),
  ai=c(NA,NA,NA,NA,61,NA), bi=c(NA,NA,NA,NA,36,NA),
  ci=c(NA,NA,NA,NA,39,NA), di=c(NA,NA,NA,NA,57,NA),
  or=c(NA,NA,NA,NA,NA,1.86), lower=c(NA,NA,NA,NA,NA,1.12), upper=c(NA,NA,NA,NA,NA,3.10),
  m1i=c(NA,NA,54.1,NA,NA,NA), sd1i=c(NA,NA,5.79,NA,NA,NA), n1i=c(NA,NA,66,75,NA,NA),
  m2i=c(NA,NA,51.7,NA,NA,NA), sd2i=c(NA,NA,6.23,NA,NA,NA), n2i=c(NA,NA,65,88,NA,NA))
dat

### study types:
### 1) reports the correlation coefficient directly
### 2) reports the t-statistic from a t-test of H0: rho = 0
### 3) dichotomized one variable and reports means and SDs for the two corresponding groups
### 4) reports the phi coefficient, marginal counts, and total sample size
### 5) dichotomized both variables and reports the resulting 2x2 table
### 6) dichotomized both variables and reports an odds ratio with 95% CI

### use escalc() to directly compute the correlation and its variance for studies 1 and 2
dat <- escalc(measure="COR", ri=ri, ni=ni, ti=tval, data=dat)
dat

### use escalc() with measure RBIS to compute the biserial correlation for study 3
dat <- escalc(measure="RBIS", m1i=m1i, sd1i=sd1i, n1i=n1i,
  m2i=m2i, sd2i=sd2i, n2i=n2i, data=dat, replace=FALSE)
dat

### use conv.2x2() to reconstruct the 2x2 table for study 4

```

```

dat <- conv.2x2(ri=phi, ni=ni, nli=nli, n2i=n2i, data=dat)
dat

### use escalc() with measure RTET to compute the tetrachoric correlation for studies 4 and 5
dat <- escalc(measure="RTET", ai=ai, bi=bi, ci=ci, di=di, data=dat, replace=FALSE)
dat

### use conv.wald() to convert the OR and CI into the log odds ratio and its variance for study 6
dat <- conv.wald(out=or, ci.lb=lower, ci.ub=upper, data=dat,
                transf=log, var.names=c("lnor", "vlnor"))
dat

### use conv.delta() to estimate the tetrachoric correlation from the log odds ratio for study 6
dat <- conv.delta(lnor, vlnor, data=dat,
                transf=transf.lnortortet.pearson, var.names=c("yi", "vi"))
dat

### remove the lnor and vlnor variables (no longer needed)
dat$lnor <- NULL
dat$vlnor <- NULL

### now variable 'yi' is complete with the correlations for all studies
dat

### fit an equal-effects model to the correlations
rma(yi, vi, data=dat, method="EE")

#####

```

conv.fivenum

Estimate Means and Standard Deviations from Five-Number Summary Values

Description

Function to estimate means and standard deviations from five-number summary values.

Usage

```

conv.fivenum(min, q1, median, q3, max, n, data, include,
             method="default", dist="norm", transf=TRUE, test=TRUE,
             var.names=c("mean", "sd"), append=TRUE, replace="ifna", ...)

```

Arguments

min	vector with the minimum values.
q1	vector with the lower/first quartile values.
median	vector with the median values.
q3	vector with the upper/third quartile values.

max	vector with the maximum values.
n	vector with the sample sizes.
data	optional data frame containing the variables given to the arguments above.
include	optional (logical or numeric) vector to specify the subset of studies for which means and standard deviations should be estimated.
method	character string to specify the method to use. Either "default" (same as "luo/wan/shi" which is the current default), "qe", "bc", "mln", or "blue". Can be abbreviated. See 'Details'.
dist	character string to specify the assumed distribution for the underlying data (either "norm" for a normal distribution or "lnorm" for a log-normal distribution). Can also be a string vector if different distributions are assumed for different studies. Only relevant when method="default".
transf	logical to specify whether the estimated means and standard deviations of the log-transformed data should be back-transformed as described by Shi et al. (2020b) (the default is TRUE). Only relevant when dist="lnorm" and when method="default".
test	logical to specify whether a study should be excluded from the estimation if the test for skewness is significant (the default is TRUE, but whether this is applicable depends on the method; see 'Details').
var.names	character vector with two elements to specify the name of the variable for the estimated means and the name of the variable for the estimated standard deviations (the defaults are "mean" and "sd").
append	logical to specify whether the data frame provided via the data argument should be returned together with the estimated values (the default is TRUE).
replace	character string or logical to specify how values in var.names should be replaced (only relevant when using the data argument and if variables in var.names already exist in the data frame). See the 'Value' section for more details.
...	other arguments.

Details

Various effect size measures require means and standard deviations (SDs) as input (e.g., raw or standardized mean differences, ratios of means / response ratios; see [escalc](#) for further details). For some studies, authors may not report means and SDs, but other statistics, such as the so-called 'five-number summary', consisting of the minimum, lower/first quartile, median, upper/third quartile, and the maximum of the sample values (plus the sample sizes). Occasionally, only a subset of these values are reported.

The present function can be used to estimate means and standard deviations from five-number summary values based on various methods described in the literature (Bland, 2015; Cai et al. 2021; Hozo et al., 2005; Luo et al., 2016; McGrath et al., 2020; Shi et al., 2020a; Walter & Yao, 2007; Wan et al., 2014; Yang et al., 2022).

When method="default" (which is the same as "luo/wan/shi"), the following methods are used:

Case 1: Min, Median, Max:

In case only the minimum, median, and maximum is available for a study (plus the sample size), then the function uses the method by Luo et al. (2016), equation (7), to estimate the mean and the method by Wan et al. (2014), equation (9), to estimate the SD.

Case 2: Q1, Median, Q3:

In case only the lower/first quartile, median, and upper/third quartile is available for a study (plus the sample size), then the function uses the method by Luo et al. (2016), equation (11), to estimate the mean and the method by Wan et al. (2014), equation (16), to estimate the SD.

Case 3: Min, Q1, Median, Q3, Max:

In case the full five-number summary is available for a study (plus the sample size), then the function uses the method by Luo et al. (2016), equation (15), to estimate the mean and the method by Shi et al. (2020a), equation (10), to estimate the SD.

The median is not actually needed in the methods by Wan et al. (2014) and Shi et al. (2020a) and hence it is possible to estimate the SD even if the median is unavailable (this can be useful if a study reports the mean directly, but instead of the SD, it reports the minimum/maximum and/or first/third quartile values).

Note that the sample size must be at least 5 to apply these methods. Studies where the sample size is smaller are not included in the estimation. The function also checks that $\min \leq q1 \leq \text{median} \leq q3 \leq \max$ and throws an error if any studies are found where this is not the case.

Test for Skewness:

The methods described above were derived under the assumption that the data are normally distributed. Testing this assumption would require access to the raw data, but based on the three cases above, Shi et al. (2023) derived tests for skewness that only require the reported quantile values and the sample sizes. These tests are automatically carried out. When `test=TRUE` (which is the default), a study is automatically excluded from the estimation if the test is significant. If all studies should be included, set `test=FALSE`, but note that the accuracy of the methods will tend to be poorer when the data come from an apparently skewed (and hence non-normal) distribution.

Log-Normal Distribution:

When setting `dist="lnorm"`, the raw data are assumed to follow a log-normal distribution. In this case, the methods as described by Shi et al. (2020b) are used to estimate the mean and SD of the log transformed data for the three cases above. When `transf=TRUE` (the default), the estimated mean and SD of the log transformed data are back-transformed to the estimated mean and SD of the raw data (using the bias-corrected back-transformation as described by Shi et al., 2020b). Note that the test for skewness is also carried out when `dist="lnorm"`, but now testing if the log transformed data exhibit skewness.

Alternative Methods:

As an alternative to the methods above, one can make use of the methods implemented in the `estmeansd` package to estimate means and SDs based on the three cases above. Available are the quantile estimation method (`method="qe"`; using the `qe.mean.sd` function; McGrath et al., 2020), the Box-Cox method (`method="bc"`; using the `bc.mean.sd` function; McGrath et al., 2020), and the method for unknown non-normal distributions (`method="mln"`; using the `mln.mean.sd` function; Cai et al. 2021). The advantage of these methods is that they do not assume that the data

underlying the reported values are normally distributed (and hence the test argument is ignored), but they can only be used when the values are positive (except for the quantile estimation method, which can also be used when one or more of the values are negative, but in this case the method does assume that the data are normally distributed and hence the test for skewness is applied when `test=TRUE`). Note that all of these methods may struggle to provide sensible estimates when some of the values are equal to each other (which can happen when the data include a lot of ties and/or the reported values are rounded). Also, the Box-Cox method and the method for unknown non-normal distributions involve simulated data and hence results will slightly change on repeated runs. Setting the seed of the random number generator (with `set.seed`) ensures reproducibility. Finally, by setting `method="blue"`, one can make use of the `BLUE_s` function from the `metaBLUE` package to estimate means and SDs based on the three cases above (Yang et al., 2022). The method assumes that the underlying data are normally distributed (and hence the test for skewness is applied when `test=TRUE`).

Value

If the data argument was not specified or `append=FALSE`, a data frame with two variables called `var.names[1]` (by default "mean") and `var.names[2]` (by default "sd") with the estimated means and SDs.

If data was specified and `append=TRUE`, then the original data frame is returned. If `var.names[1]` is a variable in data and `replace="ifna"` (or `replace=FALSE`), then only missing values in this variable are replaced with the estimated means (where possible) and otherwise a new variable called `var.names[1]` is added to the data frame. Similarly, if `var.names[2]` is a variable in data and `replace="ifna"` (or `replace=FALSE`), then only missing values in this variable are replaced with the estimated SDs (where possible) and otherwise a new variable called `var.names[2]` is added to the data frame.

If `replace="all"` (or `replace=TRUE`), then all values in `var.names[1]` and `var.names[2]` where an estimated mean and SD can be computed are replaced, even for cases where the value in `var.names[1]` and `var.names[2]` is not missing. However, non-missing values are never replaced with missing values. To also allow this (i.e., to completely replace all values), use `replace="complete"`.

When missing values in `var.names[1]` are replaced, an attribute called "est" is added to the variable, which is a logical vector that is TRUE for values that were estimated. The same is done when missing values in `var.names[2]` are replaced.

Attributes called "tval", "crit", "sig", and "dist" are also added to `var.names[1]` corresponding to the test statistic and critical value for the test for skewness, whether the test was significant, and the assumed distribution (for the quantile estimation method, this is the distribution that provides the best fit to the given values).

Note

A word of caution: Under the given distributional assumptions, the estimated means and SDs are approximately unbiased and hence so are any effect sizes estimates computed based on them (assuming a measure is unbiased to begin with when computed with directly reported means and SDs). However, the estimated means and SDs are less precise (i.e., are more variable) than directly reported means and SDs (especially under case 1) and hence computing the sampling variance of a measure with equations that assume that directly reported means and SDs are available will tend to underestimate the actual sampling variance of the measure, giving too much weight to estimates computed based on estimated means and SDs (see also McGrath et al., 2023). It would therefore be

prudent to treat effect size estimates computed from estimated means and SDs with caution (e.g., by examining in a moderator analysis whether there are systematic differences between studies directly reporting means and SDs and those where the means and SDs needed to be estimated and/or as part of a sensitivity analysis). McGrath et al. (2023) also suggest to use bootstrapping to estimate the sampling variance of effect size estimates computed based on estimated means and SDs. See also the **metamedian** package for this purpose.

Also note that the development of methods for estimating means and SDs based on five-number summary values is an active area of research. Currently, when `method="default"`, then this is identical to `method="luo/wan/shi"`, but this might change in the future. For reproducibility, it is therefore recommended to explicitly set `method="luo/wan/shi"` (or one of the other methods) when running this function.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Bland, M. (2015). Estimating mean and standard deviation from the sample size, three quartiles, minimum, and maximum. *International Journal of Statistics in Medical Research*, **4**(1), 57–64. <https://doi.org/10.6000/1929-6029.2015.04.01.6>
- Cai, S., Zhou, J., & Pan, J. (2021). Estimating the sample mean and standard deviation from order statistics and sample size in meta-analysis. *Statistical Methods in Medical Research*, **30**(12), 2701–2719. <https://doi.org/10.1177/09622802211047348>
- Hozo, S. P., Djulbegovic, B. & Hozo, I. (2005). Estimating the mean and variance from the median, range, and the size of a sample. *BMC Medical Research Methodology*, **5**, 13. <https://doi.org/10.1186/1471-2288-5-13>
- Luo, D., Wan, X., Liu, J. & Tong, T. (2016). Optimally estimating the sample mean from the sample size, median, mid-range, and/or mid-quartile range. *Statistical Methods in Medical Research*, **27**(6), 1785–1805. <https://doi.org/10.1177/0962280216669183>
- McGrath, S., Zhao, X., Steele, R., Thombs, B. D., Benedetti, A., & the DEPRESSion Screening Data (DEPRESSD) Collaboration (2020). Estimating the sample mean and standard deviation from commonly reported quantiles in meta-analysis. *Statistical Methods in Medical Research*, **29**(9), 2520–2537. <https://doi.org/10.1177/0962280219889080>
- McGrath, S., Katzenschlager, S., Zimmer, A. J., Seitel, A., Steele, R., & Benedetti, A. (2023). Standard error estimation in meta-analysis of studies reporting medians. *Statistical Methods in Medical Research*, **32**(2), 373–388. <https://doi.org/10.1177/09622802221139233>
- Shi, J., Luo, D., Weng, H., Zeng, X.-T., Lin, L., Chu, H. & Tong, T. (2020a). Optimally estimating the sample standard deviation from the five-number summary. *Research Synthesis Methods*, **11**(5), 641–654. <https://doi.org/https://doi.org/10.1002/jrsm.1429>
- Shi, J., Tong, T., Wang, Y. & Genton, M. G. (2020b). Estimating the mean and variance from the five-number summary of a log-normal distribution. *Statistics and Its Interface*, **13**(4), 519–531. <https://doi.org/10.4310/sii.2020.v13.n4.a9>
- Shi, J., Luo, D., Wan, X., Liu, Y., Liu, J., Bian, Z. & Tong, T. (2023). Detecting the skewness of data from the five-number summary and its application in meta-analysis. *Statistical Methods in Medical Research*, **32**(7), 1338–1360. <https://doi.org/10.1177/09622802231172043>

- Walter, S. D. & Yao, X. (2007). Effect sizes can be calculated for studies reporting ranges for outcome variables in systematic reviews. *Journal of Clinical Epidemiology*, **60**(8), 849–852. <https://doi.org/10.1016/j.jclinepi.2007.05.008>
- Wan, X., Wang, W., Liu, J. & Tong, T. (2014). Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range. *BMC Medical Research Methodology*, **14**, 135. <https://doi.org/10.1186/1471-2288-14-135>
- Yang, X., Hutson, A. D., & Wang, D. (2022). A generalized BLUE approach for combining location and scale information in a meta-analysis. *Journal of Applied Statistics*, **49**(15), 3846–3867. <https://doi.org/10.1080/02664763.2021.1967890>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[escalc](#) for a function to compute various effect size measures based on means and standard deviations.

Examples

```
# example data frame
dat <- data.frame(case=c(1:3,NA), min=c(2,NA,2,NA), q1=c(NA,4,4,NA),
                  median=c(6,6,6,NA), q3=c(NA,10,10,NA), max=c(14,NA,14,NA),
                  mean=c(NA,NA,NA,7.0), sd=c(NA,NA,NA,4.2), n=c(20,20,20,20))

dat

# note that study 4 provides the mean and SD directly, while studies 1-3 provide five-number
# summary values or a subset thereof (corresponding to cases 1-3 above)

# estimate means/SDs (note: existing values in 'mean' and 'sd' are not touched)
dat <- conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat)
dat

# check attributes (none of the tests are significant, so means/SDs are estimated for studies 1-3)
dfround(data.frame(attributes(dat$mean)), digits=3)

# calculate the log transformed coefficient of variation and corresponding sampling variance
dat <- escalc(measure="CVLN", mi=mean, sdi=sd, ni=n, data=dat)
dat

# fit equal-effects model to the estimates
res <- rma(yi, vi, data=dat, method="EE")
res

# estimated coefficient of variation (with 95% CI)
predict(res, transf=exp, digits=2)

#####

# example data frame
dat <- data.frame(case=c(1:3,NA), min=c(2,NA,2,NA), q1=c(NA,4,4,NA),
                  median=c(6,6,6,NA), q3=c(NA,10,10,NA), max=c(14,NA,14,NA),
                  mean=c(NA,NA,NA,7.0), sd=c(NA,NA,NA,4.2), n=c(20,20,20,20))
```

```

dat

# try out different methods
conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat)
set.seed(1234)
conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat, method="qe")
conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat, method="bc")
conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat, method="mln")
conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat, method="blue")

#####

# example data frame
dat <- data.frame(case=c(1:3,NA), min=c(2,NA,2,NA), q1=c(NA,4,4,NA),
                  median=c(6,6,6,NA), q3=c(NA,10,14,NA), max=c(14,NA,20,NA),
                  mean=c(NA,NA,NA,7.0), sd=c(NA,NA,NA,4.2), n=c(20,20,20,20))

dat

# for study 3, the third quartile and maximum value suggest that the data have
# a right skewed distribution (they are much further away from the median than
# the minimum and first quartile)

# estimate means/SDs
dat <- conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat)
dat

# note that the mean and SD are not estimated for study 3; this is because the
# test for skewness is significant for this study
dfround(data.frame(attributes(dat$mean)), digits=3)

# estimate means/SDs, but assume that the data for study 3 come from a log-normal distribution
# and back-transform the estimated mean/SD of the log-transformed data back to the raw data
dat <- conv.fivenum(min=min, q1=q1, median=median, q3=q3, max=max, n=n, data=dat,
                  dist=c("norm","norm","lnorm","norm"), replace="all")
dat

# this works now because the test for skewness of the log-transformed data is not significant
dfround(data.frame(attributes(dat$mean)), digits=3)

```

conv.wald

Convert Wald-Type Confidence Intervals and Tests to Sampling Variances

Description

Function to convert Wald-type confidence intervals (CIs) and test statistics (or the corresponding p-values) to sampling variances.

Usage

```
conv.wald(out, ci.lb, ci.ub, zval, pval, n, data, include,
          level=95, transf, check=TRUE,
          var.names, append=TRUE, replace="ifna", ...)
```

Arguments

<code>out</code>	vector with the effect size estimates.
<code>ci.lb</code>	vector with the lower bounds of the corresponding Wald-type CIs.
<code>ci.ub</code>	vector with the upper bounds of the corresponding Wald-type CIs.
<code>zval</code>	vector with the Wald-type test statistics.
<code>pval</code>	vector with the p-values of the Wald-type tests.
<code>n</code>	vector with the total sample sizes of the studies.
<code>data</code>	optional data frame containing the variables given to the arguments above.
<code>include</code>	optional (logical or numeric) vector to specify the subset of studies for which the conversion should be carried out.
<code>level</code>	numeric value (or vector) to specify the confidence interval level(s) (the default is 95; see here for details).
<code>transf</code>	optional argument to specify a function to transform <code>out</code> , <code>ci.lb</code> , and <code>ci.ub</code> (e.g., <code>transf=log</code>). If unspecified, no transformation is used.
<code>check</code>	logical to specify whether the function should carry out a check to examine if the estimates fall (approximately) halfway between the CI bounds (the default is TRUE).
<code>var.names</code>	character vector with two elements to specify the name of the variable for the estimates and the name of the variable for the corresponding sampling variances (if <code>data</code> is an object of class "escalc", the <code>var.names</code> are taken from the object; otherwise the defaults are "yi" and "vi").
<code>append</code>	logical to specify whether the data frame provided via the <code>data</code> argument should be returned together with the estimated values (the default is TRUE).
<code>replace</code>	character string or logical to specify how values in <code>var.names</code> should be replaced (only relevant when using the <code>data</code> argument and if variables in <code>var.names</code> already exist in the data frame). See the 'Value' section for more details.
<code>...</code>	other arguments.

Details

The [escalc](#) function can be used to compute a wide variety of effect size measures. However, the inputs required to compute certain measures with this function may not be reported for all of the studies. Under certain circumstances, other information may be available that can be converted into the appropriate format needed for a meta-analysis. The purpose of the present function is to facilitate this process.

The function typically takes a data frame created with the [escalc](#) function as input via the `data` argument. This object should contain variables `yi` and `vi` (unless argument `var.names` was used to adjust these variable names when the "escalc" object was created) for the effect size estimates and

the corresponding sampling variances, respectively. For some studies, the values for these variables may be missing.

Converting Estimates and Confidence Intervals:

In some studies, the effect size estimate of interest may already be reported. If so, such values can be supplied via the `out` argument and are then substituted for missing `yi` values. At times, it may be necessary to transform the reported values (e.g., reported odds ratios to log odds ratios). Via argument `transf`, an appropriate transformation function can be specified (e.g., `transf=log`), in which case $y_i = f(\text{out})$ where $f(\cdot)$ is the function specified via `transf`.

Moreover, a confidence interval (CI) may have been reported together with the effect size estimate. The bounds of the CI can be supplied via arguments `ci.lb` and `ci.ub`, which are also transformed if a function is specified via `transf`. Assume that the bounds were obtained from a Wald-type CI of the form $y_i \pm z_{crit}\sqrt{v_i}$ (on the transformed scale if `transf` is specified), where v_i is the sampling variance corresponding to the estimate (so that $\sqrt{v_i}$ is the corresponding standard error) and z_{crit} is the appropriate critical value from a standard normal distribution (e.g., 1.96 for a 95% CI). Then

$$v_i = \left(\frac{\text{ci.ub} - \text{ci.lb}}{2 \times z_{crit}} \right)^2$$

is used to back-calculate the sampling variances of the (transformed) estimates and these values are then substituted for missing `vi` values in the dataset.

For example, consider the following dataset of three RCTs used as input for a meta-analysis of log odds ratios:

```
dat <- data.frame(study = 1:3,
                  cases.trt = c(23, NA, 4), n.trt = c(194, 183, 46),
                  cases.plc = c(38, NA, 7), n.plc = c(201, 188, 44),
                  oddsratio = c(NA, 0.64, NA), lower = c(NA, 0.33, NA), upper = c(NA, 1.22, NA))
dat <- escalc(measure="OR", ai=cases.trt, n1i=n.trt, ci=cases.plc, n2i=n.plc, data=dat)
dat
```

```
#  study cases.trt n.trt cases.plc n.plc oddsratio lower upper    yi    vi
# 1     1       23  194       38  201         NA     NA     NA -0.5500 0.0818
# 2     2      NA  183      NA  188        0.64  0.33  1.22     NA     NA
# 3     3       4   46       7   44         NA     NA     NA -0.6864 0.4437
```

where variable `yi` contains the log odds ratios and `vi` the corresponding sampling variances as computed from the counts and group sizes by `escalc()`.

Study 2 does not report the counts (or sufficient information to reconstruct them), but the odds ratio and a corresponding 95% confidence interval (CI) directly, as given by variables `oddsratio`, `lower`, and `upper`. The CI is a standard Wald-type CI that was computed on the log scale (and whose bounds were then exponentiated). Then the present function can be used as follows:

```
dat2 <- conv.wald(out=oddsratio, ci.lb=lower, ci.ub=upper, data=dat, transf=log)
dat2
```

```
#  study cases.trt n.trt cases.plc n.plc oddsratio lower upper    yi    vi
# 1     1       23  194       38  201         NA     NA     NA -0.5500 0.0818
# 2     2      NA  183      NA  188        0.64  0.33  1.22 -0.4463 0.1113
# 3     3       4   46       7   44         NA     NA     NA -0.6864 0.4437
```

Now variables y_i and v_i in the dataset are complete.

If the CI was not a 95% CI, then one can specify the appropriate level via the `level` argument. This can also be an entire vector in case different studies used different levels.

By default (i.e., when `check=TRUE`), the function carries out a rough check to examine if the estimate falls (approximately) halfway between the CI bounds (on the transformed scale) for each study for which the conversion was carried out. A warning is issued if there are studies where this is not the case. This may indicate that a particular CI was not a Wald-type CI or was computed on a different scale (in which case the back-calculation above would be inappropriate), but can also arise due to rounding of the reported values (in which case the back-calculation would still be appropriate, albeit possibly a bit inaccurate). Care should be taken when using such back-calculated values in a meta-analysis.

When the CI bounds are log transformed (as in the example above) and rounded, then using the lower bound in the back-calculation can be more inaccurate than using the upper CI bound and the estimate for the back-calculation. When the lower CI bounds are not specified or are missing for a study, then the function automatically uses the formula

$$v_i = \left(\frac{ci.ub - y_i}{z_{crit}} \right)^2$$

for the back-calculation (after first applying the transformation specified via `transf`). In the example above, we could therefore do:

```
dat3 <- conv.wald(out=oddsratio, ci.ub=upper, data=dat, transf=log)
dat3
```

#	study	cases.trt	n.trt	cases.plc	n.plc	oddsratio	lower	upper	y_i	v_i
# 1	1	23	194	38	201	NA	NA	NA	-0.5500	0.0818
# 2	2	NA	183	NA	188	0.64	0.33	1.22	-0.4463	0.1083
# 3	3	4	46	7	44	NA	NA	NA	-0.6864	0.4437

In this case, the check that the estimate falls halfway between the CI bounds is of course skipped.

Converting Test Statistics and P-Values:

Similarly, study authors may report the test statistic and/or p-value from a Wald-type test of the form $zval = y_i / \sqrt{v_i}$ (on the transformed scale if `transf` is specified), with the corresponding two-sided p-value given by $pval = 2(1 - \Phi(|zval|))$, where $\Phi(\cdot)$ denotes the cumulative distribution function of a standard normal distribution (i.e., `pnorm`). Test statistics and/or corresponding p-values of this form can be supplied via arguments `zval` and `pval`.

A given p-value can be back-transformed into the corresponding test statistic (if it is not already available) with $zval = \Phi^{-1}(1 - pval/2)$, where $\Phi^{-1}(\cdot)$ denotes the quantile function (i.e., the inverse of the cumulative distribution function) of a standard normal distribution (i.e., `qnorm`). Then

$$v_i = \left(\frac{y_i}{zval} \right)^2$$

is used to back-calculate a missing v_i value in the dataset.

Note that the conversion of a p-value to the corresponding test statistic (which is then converted into sampling variance) as shown above assumes that the exact p-value is reported. If authors only report that the p-value fell below a certain threshold (e.g., $p < 0.01$ or if authors only state that the test was significant – which typically implies $p < 0.05$), then a common approach is to use

the value of the cutoff reported (e.g., if $p < 0.01$ is reported, then assume $p = 0.01$), which is conservative (since the actual p-value was below that assumed value by some unknown amount). The conversion will therefore tend to be much less accurate.

Using the earlier example, suppose that only the odds ratio and the corresponding two-sided p-value from a Wald-type test (whether the log odds ratio differs significantly from zero) is reported for study 2.

```
dat <- data.frame(study = 1:3,
                  cases.trt = c(23, NA, 4), n.trt = c(194, 183, 46),
                  cases.plc = c(38, NA, 7), n.plc = c(201, 188, 44),
                  oddsratio = c(NA, 0.64, NA), pval = c(NA, 0.17, NA))
dat <- escalc(measure="OR", ai=cases.trt, n1i=n.trt, ci=cases.plc, n2i=n.plc, data=dat)
dat
```

	study	cases.trt	n.trt	cases.plc	n.plc	oddsratio	pval	yi	vi
1	1	23	194	38	201	NA	NA	-0.5500	0.0818
2	2	NA	183	NA	188	0.64	0.17	NA	NA
3	3	4	46	7	44	NA	NA	-0.6864	0.4437

Then the function can be used as follows:

```
dat4 <- conv.wald(out=oddsratio, pval=pval, data=dat, transf=log)
dat4
```

#	study	cases.trt	n.trt	cases.plc	n.plc	oddsratio	pval	yi	vi
# 1	1	23	194	38	201	NA	NA	-0.5500	0.0818
# 2	2	NA	183	NA	188	0.64	0.17	-0.4463	0.1058
# 3	3	4	46	7	44	NA	NA	-0.6864	0.4437

Note that the back-calculated sampling variance for study 2 is not identical in these two examples, because the CI bounds and p-value are rounded to two decimal places, which introduces some inaccuracies. Also, if both (ci.lb, ci.ub) and either zval or pval is available for a study, then the back-calculation of v_i via the confidence interval is preferred.

Optionally, one can use the n argument to supply the total sample sizes of the studies. This has no relevance for the calculations done by the present function, but some other functions may use this information (e.g., when drawing a funnel plot with the [funnel](#) function and one adjusts the yaxis argument to one of the options that puts the sample sizes or some transformation thereof on the y-axis).

Value

If the data argument was not specified or append=FALSE, a data frame of class c("escalc", "data.frame") with two variables called var.names[1] (by default "yi") and var.names[2] (by default "vi") with the (transformed) estimates and the corresponding sampling variances (computed as described above).

If data was specified and append=TRUE, then the original data frame is returned. If var.names[1] is a variable in data and replace="ifna" (or replace=FALSE), then only missing values in this variable are replaced with the (possibly transformed) estimates from out (where possible) and otherwise a new variable called var.names[1] is added to the data frame. Similarly, if var.names[2]

is a variable in data and `replace="ifna"` (or `replace=FALSE`), then only missing values in this variable are replaced with the sampling variances back-calculated as described above (where possible) and otherwise a new variable called `var.names[2]` is added to the data frame.

If `replace="all"` (or `replace=TRUE`), then all values in `var.names[1]` and `var.names[2]` are replaced, even for cases where the value in `var.names[1]` and `var.names[2]` is not missing. However, non-missing values are never replaced with missing values. To also allow this (i.e., to completely replace all values), use `replace="complete"`.

Note

A word of caution: Except for the check on the CI bounds, there is no possibility to determine if the back-calculations done by the function are appropriate in a given context. They are only appropriate when the CI bounds and tests statistics (or p-values) arose from Wald-type CIs / tests as described above. Using the same back-calculations for other purposes is likely to yield nonsensical values.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[escalc](#) for a function to compute various effect size measures.

Examples

```
### a very simple example
dat <- data.frame(or=c(1.37,1.89), or.lb=c(1.03,1.60), or.ub=c(1.82,2.23))
dat

### convert the odds ratios and CIs into log odds ratios with corresponding sampling variances
dat <- conv.wald(out=or, ci.lb=or.lb, ci.ub=or.ub, data=dat, transf=log)
dat

#####

### a more elaborate example based on the BCG vaccine dataset
dat <- dat.bcg[,c(2:7)]
dat

### with complete data, we can use escalc() in the usual way
dat1 <- escalc(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat)
dat1

### random-effects model fitted to these data
res1 <- rma(yi, vi, data=dat1)
res1
```

```

### now suppose that the 2x2 table data are not reported in all studies, but that the
### following dataset could be assembled based on information reported in the studies
dat2 <- data.frame(summary(dat1))
dat2[c("yi", "ci.lb", "ci.ub")] <- data.frame(summary(dat1, transf=exp))[c("yi", "ci.lb", "ci.ub")]
names(dat2)[which(names(dat2) == "yi")] <- "or"
dat2[,c("or", "ci.lb", "ci.ub", "pval")] <- round(dat2[,c("or", "ci.lb", "ci.ub", "pval")], digits=2)
dat2$vi <- dat2$sei <- dat2$zi <- NULL
dat2$ntot <- with(dat2, tpos + tneg + cpos + cneg)
dat2[c(1,12),c(3:6,9:10)] <- NA
dat2[c(4,9), c(3:6,8)] <- NA
dat2[c(2:3,5:8,10:11,13),c(7:10)] <- NA
dat2$ntot[!is.na(dat2$tpos)] <- NA
dat2

### in studies 1 and 12, authors reported only the odds ratio and the corresponding p-value
### in studies 4 and 9, authors reported only the odds ratio and the corresponding 95% CI

### use the escalc() function first to compute log odds ratios and corresponding sampling
### variances for the studies for which we have 2x2 table data
dat2 <- escalc(measure="OR", ai=tpos, bi=tneg, ci=cpus, di=cneg, data=dat2)
dat2

### fill in the missing log odds ratios and sampling variances using conv.wald()
dat2 <- conv.wald(out=or, ci.lb=ci.lb, ci.ub=ci.ub, pval=pval, n=ntot, data=dat2, transf=log)
dat2

### random-effects model fitted to these data
res2 <- rma(yi, vi, data=dat2)
res2

### any differences between res1 and res2 are a result of or, ci.lb, ci.ub, and pval being
### rounded in dat2 to two decimal places; without rounding, the results would be identical

```

cumul

Cumulative Meta-Analysis for 'rma' Objects

Description

Function to carry out a 'cumulative meta-analysis', by repeatedly fitting the specified model adding one estimate at a time.

Usage

```

cumul(x, ...)

## S3 method for class 'rma.uni'
cumul(x, order, digits, transf, targs, collapse=FALSE, progbar=FALSE, ...)
## S3 method for class 'rma.mh'
cumul(x, order, digits, transf, targs, collapse=FALSE, progbar=FALSE, ...)

```

```
## S3 method for class 'rma.peto'
cumul(x, order, digits, transf, targs, collapse=FALSE, progbar=FALSE, ...)
```

Arguments

<code>x</code>	an object of class "rma.uni", "rma.mh", or "rma.peto".
<code>order</code>	optional argument to specify a variable based on which the estimates will be ordered for the cumulative meta-analysis.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
<code>transf</code>	optional argument to specify a function to transform the model coefficients and interval bounds (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
<code>targs</code>	optional arguments needed by the function specified under <code>transf</code> .
<code>collapse</code>	logical to specify whether estimates with the same value of the order variable should be added simultaneously (the default is FALSE).
<code>progbar</code>	logical to specify whether a progress bar should be shown (the default is FALSE).
<code>...</code>	other arguments.

Details

A cumulative meta-analysis describes the accumulation of evidence (e.g., about the effectiveness of a particular treatment or intervention) as the effect sizes estimates are added to the analysis in (typically) chronological order (Chalmers & Lau, 1993; Lau et al., 1995). Therefore, given k estimates, we conduct k meta-analyses, first only including the first estimate, then including the first and second estimate, then estimates 1 through 3, and so on, until all k estimates are included.

For "rma.uni" objects, the model specified via `x` must be a model without moderators (i.e., either an equal- or a random-effects model).

If argument `order` is not specified, the estimates are added according to their order in the original dataset.

When a variable is specified for `order`, the variable is assumed to be of the same length as the original dataset that was used in the model fitting (and if the `data` argument was used in the original model fit, then the variable will be searched for within this data frame first). Any subsetting and removal of rows with missing values that was applied during the model fitting is also automatically applied to the variable specified via the `order` argument.

By default, estimates are added one at a time. However, if a variable is specified for the `order` argument and `collapse=TRUE`, then estimates with the same value of the order variable are added simultaneously.

Value

An object of class `c("list.rma", "cumul.rma")`. The object is a list containing the following components:

<code>k</code>	number of estimates included in the analysis.
----------------	---

<code>estimate</code>	the pooled estimates.
<code>se</code>	corresponding standard errors.
<code>zval</code>	corresponding test statistics.
<code>pval</code>	corresponding p-values.
<code>ci.lb</code>	lower bounds of the confidence intervals.
<code>ci.ub</code>	upper bounds of the confidence intervals.
<code>Q</code>	test statistics for the test of heterogeneity.
<code>Qp</code>	corresponding p-values.
<code>tau2</code>	estimated amount of heterogeneity (only for random-effects models).
<code>I2</code>	values of I^2 .
<code>H2</code>	values of H^2 .
<code>order</code>	values of the order variable (if specified).
<code>...</code>	other arguments.

When the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="ad hoc"`, then `zval` is called `tval` in the object that is returned by the function.

The object is formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function. A forest plot showing the results from the cumulative meta-analysis can be obtained with `forest`. Alternatively, `plot` can also be used to visualize the results.

Note

When using the `transf` option, the transformation is applied to the pooled estimates and the corresponding interval bounds. The standard errors are then set equal to NA and are omitted from the printed output.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Chalmers, T. C., & Lau, J. (1993). Meta-analytic stimulus for changes in clinical trials. *Statistical Methods in Medical Research*, **2**(2), 161–172. <https://doi.org/10.1177/096228029300200204>
- Lau, J., Schmid, C. H., & Chalmers, T. C. (1995). Cumulative meta-analysis of clinical trials builds evidence for exemplary medical care. *Journal of Clinical Epidemiology*, **48**(1), 45–57. [https://doi.org/10.1016/0895-4356\(94\)00106-z](https://doi.org/10.1016/0895-4356(94)00106-z)
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`forest` for a function to draw cumulative forest plots and `plot` for a different visualization of the cumulative results.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
             data=dat.bcg, slab=paste0(author, ", ", year))

### fit random-effects model
res <- rma(yi, vi, data=dat, digits=3)

### cumulative meta-analysis (in the order of publication year)
cumul(res, order=year)
cumul(res, order=year, transf=exp)

### add studies with the same publication year simultaneously
cumul(res, order=year, transf=exp, collapse=TRUE)

### meta-analysis of the (log) risk ratios using the Mantel-Haenszel method
res <- rma.mh(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
             data=dat.bcg, slab=paste0(author, ", ", year), digits=3)

### cumulative meta-analysis
cumul(res, order=year)
cumul(res, order=year, transf=exp)

### add studies with the same publication year simultaneously
cumul(res, order=year, transf=exp, collapse=TRUE)

### meta-analysis of the (log) odds ratios using Peto's method
res <- rma.peto(ai=tpos, bi=tneg, ci=cpos, di=cneg,
              data=dat.bcg, slab=paste0(author, ", ", year), digits=3)

### cumulative meta-analysis
cumul(res, order=year)
cumul(res, order=year, transf=exp)

### add studies with the same publication year simultaneously
cumul(res, order=year, transf=exp, collapse=TRUE)

### make the first log risk ratio missing and fit the model without study 2;
### then the variable specified via 'order' should still be of the same length
### as the original dataset; subsetting and removal of studies with missing
### values is automatically done by the cumul() function
dat$yi[1] <- NA
res <- rma(yi, vi, data=dat, subset=-2, digits=3)
cumul(res, transf=exp, order=year)
```

deltamethod

Apply the (Multivariate) Delta Method

Description

Function to apply the (multivariate) delta method to a set of estimates.

Usage

```
deltamethod(x, vcov, fun, order=1, level, H0=0, digits)
```

Arguments

<code>x</code>	either a vector of estimates or a model object from which model coefficients can be extracted via <code>coef(x)</code> .
<code>vcov</code>	when <code>x</code> is a vector of estimates, the corresponding variance-covariance matrix (ignored when <code>x</code> is a model object, in which case <code>vcov(x)</code> is used to extract the variance-covariance matrix).
<code>fun</code>	a function to apply to the estimates.
<code>order</code>	numeric value equal to 1, 2, or 3 to determine whether the first-, second-, or third-order delta method should be used (the default is 1).
<code>level</code>	numeric value between 0 and 100 to specify the confidence interval level (see here for details). If unspecified, this either defaults to 95 or, if possible, the corresponding value from the model object.
<code>H0</code>	numeric value to specify the value under the null hypothesis for the Wald-type test(s) (the default is 0). Can also be a vector.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded.

Details

Let $\hat{\theta}$ denote a vector of p estimates which can be specified via the `x` argument and let Σ denote the corresponding $p \times p$ variance-covariance matrix, which can be specified via the `vcov` argument. If `x` is not a vector with estimates, then the function assumes that `x` is a model object and will try to use `coef(x)` and `vcov(x)` to extract the model coefficients and the corresponding variance-covariance matrix (in this case, the `vcov` argument is ignored).

Let $f(\cdot)$ be a function, specified via the `fun` argument, with p inputs/arguments (or with a single argument that is assumed to be a vector of length p), which returns a numeric (and atomic) vector of q transformed estimates. Then the function computes $f(\hat{\theta})$ and the corresponding variance-covariance matrix of the transformed estimates using the **multivariate delta method** (e.g., van der Vaart, 1998) with

$$\text{Var}[f(\hat{\theta})] = \nabla f(\hat{\theta})' \cdot \Sigma \cdot \nabla f(\hat{\theta})$$

where $\nabla f(\hat{\theta})$ denotes the gradient of $f(\cdot)$ evaluated at $\hat{\theta}$. The function computes the gradient numerically using the **derivative** function from the **calculus** package.

When `order=2`, then the second-order delta method is used. For this, the variance-covariance matrix of the transformed estimates is approximated by

$$\text{Var}[f(\hat{\theta})] = \nabla f(\hat{\theta})' \cdot \Sigma \cdot \nabla f(\hat{\theta}) + \frac{1}{2} \text{trace}(H \cdot \Sigma \cdot H \cdot \Sigma)$$

where H denotes the Hessian matrix evaluated at $\hat{\theta}$.

When `order=3`, then the third-order delta method is used. For this, the variance-covariance matrix of the transformed estimates is approximated by

$$\text{Var}[f(\hat{\theta})] = \text{Cov}(L, L) + \text{Cov}(Q, Q) + \text{Cov}(L, C) + \text{Cov}(C, L) + \text{Cov}(C, C)$$

where L , Q , and C denote the linear, quadratic, and cubic terms in the Taylor expansion of $f(\cdot)$.

The function also computes Wald-type tests and confidence intervals for the q transformed estimates. The `level` argument can be used to control the confidence interval level.

Value

An object of class "deltamethod". The object is a list containing the following components:

<code>tab</code>	a data frame with the transformed estimates, standard errors, test statistics, p-values, and lower/upper confidence interval bounds.
<code>vcov</code>	the variance-covariance matrix of the transformed estimates.
<code>...</code>	some additional elements/values.

The results are formatted and printed with the `print` function. Extractor functions include `coef` and `vcov`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- van der Vaart, A. W. (1998). *Asymptotic statistics*. Cambridge, UK: Cambridge University Press.
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`conv.delta` for a function to apply the (univariate) delta method to a set of estimates and their sampling variances.

Examples

```
#####

### copy data into 'dat'
dat <- dat.craft2003

### construct dataset and var-cov matrix of the correlations
tmp <- rcalc(ri ~ var1 + var2 | study, ni=ni, data=dat)
V <- tmp$V
dat <- tmp$dat

### turn var1.var2 into a factor with the desired order of levels
dat$var1.var2 <- factor(dat$var1.var2,
  levels=c("acog.perf", "asom.perf", "conf.perf", "acog.asom", "acog.conf", "asom.conf"))

### multivariate random-effects model
res <- rma.mv(yi, V, mods = ~ 0 + var1.var2, random = ~ var1.var2 | study, struct="UN", data=dat)
res
```

```

### restructure estimated mean correlations into a 4x4 matrix
R <- vec2mat(coef(res))
rownames(R) <- colnames(R) <- c("perf", "acog", "asom", "conf")
round(R, digits=3)

### check that order in vcov(res) corresponds to order in R
round(vcov(res), digits=4)

### fit regression model with 'perf' as outcome and 'acog', 'asom', and 'conf' as predictors
matreg(1, 2:4, R=R, V=vcov(res))

### same analysis but using the deltamethod() function
deltamethod(coef(res), vcov(res), fun=function(r1,r2,r3,r4,r5,r6) {
  R <- vec2mat(c(r1,r2,r3,r4,r5,r6))
  setNames(c(solve(R[-1,-1]) %*% R[2:4,1]), c("acog","asom","conf"))
})

### using a function that takes a vector as input
deltamethod(coef(res), vcov(res), fun=function(r) {
  R <- vec2mat(r)
  setNames(c(solve(R[-1,-1]) %*% R[2:4,1]), c("acog","asom","conf"))
})

#####

### construct dataset and var-cov matrix of the r-to-z transformed correlations
dat <- dat.craft2003
tmp <- rcalc(ri ~ var1 + var2 | study, ni=ni, data=dat, rtoz=TRUE)
V <- tmp$V
dat <- tmp$dat

### turn var1.var2 into a factor with the desired order of levels
dat$var1.var2 <- factor(dat$var1.var2,
  levels=c("acog.perf", "asom.perf", "conf.perf", "acog.asom", "acog.conf", "asom.conf"))

### multivariate random-effects model
res <- rma.mv(yi, V, mods = ~ 0 + var1.var2, random = ~ var1.var2 | study, struct="UN", data=dat)
res

### estimate the difference between r(acog,perf) and r(asom,perf)
deltamethod(res, fun=function(z1,z2,z3,z4,z5,z6) {
  transf.ztor(z1) - transf.ztor(z2)
})

### using a function that takes a vector as input
deltamethod(res, fun=function(z) {
  transf.ztor(z[1]) - transf.ztor(z[2])
})

#####

```

dfround

*Round Variables in a Data Frame***Description**

Function to round the numeric variables in a data frame.

Usage

```
dfround(x, digits, drop0=TRUE)
```

Arguments

<code>x</code>	a data frame.
<code>digits</code>	either a single integer or a numeric vector of the same length as there are columns in <code>x</code> .
<code>drop0</code>	logical (or a vector thereof) to specify whether trailing zeros after the decimal mark should be removed (the default is TRUE).

Details

A simple convenience function to round the numeric variables in a data frame, possibly to different numbers of digits. Hence, `digits` can either be a single integer (which will then be used to round all numeric variables to the specified number of digits) or a numeric vector (of the same length as there are columns in `x`) to specify the number of digits to which each variable should be rounded.

Non-numeric variables are skipped. If `digits` is a vector, some arbitrary value (or NA) can be specified for those variables.

Note: When `drop0=FALSE`, then `formatC` is used to format the numbers, which turns them into character variables.

Value

Returns the data frame with variables rounded as specified.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

Examples

```
dat <- dat.bcg
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat)
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)
coef(summary(res))
dfround(coef(summary(res)), digits=c(2,3,2,3,2,2))
```

Description

Function to create a reference grid for use with the [emmeans](#) function from the package of the same name.

Usage

```
emmprep(x, verbose=FALSE, ...)
```

Arguments

<code>x</code>	an object of class "rma".
<code>verbose</code>	logical to specify whether information on some (extracted) settings should be printed when creating the reference grid (the default is FALSE).
<code>...</code>	other arguments that will be passed on to the qdrgr function.

Details

The [emmeans](#) package is a popular package that facilitates the computation of 'estimated marginal means'. The function is a wrapper around the [qdrgr](#) function from the [emmeans](#) package to make "rma" objects compatible with the latter. Unless one needs to pass additional arguments to the [qdrgr](#) function, one simply applies this function to the "rma" object and then the [emmeans](#) function (or one of the other functions that can be applied to "emmGrid" objects) to the resulting object to obtain the desired estimated marginal means.

Value

An "emmGrid" object as created by the [qdrgr](#) function from the [emmeans](#) package.

The resulting object will typically be used in combination with the [emmeans](#) function.

Note

When creating the reference grid, the function extracts the degrees of freedom for tests/confidence intervals from the model object (if the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="adhoc"`; otherwise the degrees of freedom are infinity). In some cases, there is not just a single value for the degrees of freedom, but an entire vector (e.g., for models fitted with [rma.mv](#)). In this case, the smallest value will be used (as a conservative option). One can set a different/custom value for the degrees of freedom with `emmprep(..., df=value)`.

When the model object contains information about the effect size measure used in the analysis (which should be the case if the effect size estimates were computed with [escalc](#) or if the measure argument was set when fitting the model), then information about the appropriate back-transformation (if available) is stored as part of the returned object. If so, the back-transformation is automatically applied when calling [emmeans](#) with `type="response"`.

The function also tries to extract the estimated value of τ^2 (or more precisely, its square root) from the model object (when the model is a random/mixed-effects model). This value is only needed when computing prediction intervals (i.e., when `interval="predict"` in `predict.emmGrid`) or when applying the bias adjustment in the back-transformation (i.e., when `bias.adjust=TRUE` in `summary.emmGrid`). For some models (e.g., those fitted with `rma.mv`), it is not possible to automatically extract the estimate. In this case, one can manually set the value with `emmprep(..., sigma=value)` (note: the argument is called `sigma`, following the conventions of `summary.emmGrid` and one must supply the square root of the τ^2 estimate).

By default, the reference grid is created based on the data used for fitting the original model (which is typically the sensible thing to do). One can specify a different dataset with `emmprep(..., data=obj)`, where `obj` must be a data frame that contains the same variables as used in the original model fitted.

If the model fitted involved redundant predictors that were dropped from the model (due to ‘rank deficiencies’), then the function cannot be used. In this case, one should remove any redundancies in the model fitted before using this function.

Author(s)

Wolfgang Viechtbauer (<wv@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit meta-regression model with absolute latitude as predictor
res <- rma(yi, vi, mods = ~ ablat, data=dat)
res

### create reference grid
sav <- emmprep(res, verbose=TRUE)

### estimated marginal mean (back-transformed to the risk ratio scale)
if (require(emmeans))
  emmeans(sav, specs="1", type="response")

### same as the predicted effect at the mean absolute latitude
predict(res, newmods = mean(model.matrix(res, asdf=TRUE)$ablat), transf=exp, digits=3)

### fit meta-regression model with allocation factor
res <- rma(yi, vi, mods = ~ alloc, data=dat)
res

### create reference grid
sav <- emmprep(res)
```

```

### estimated marginal mean using proportional cell weighting
if (require(emmeans))
  emmeans(sav, specs="1", type="response", weights="proportional")

### estimated marginal mean using equal cell weighting (this is actually the default)
if (require(emmeans))
  emmeans(sav, specs="1", type="response", weights="equal")

### same as the predicted effect using cell proportions as observed in the data
### or using equal proportions for the three groups
predict(res, newmods = colMeans(model.matrix(res))[-1], transf=exp, digits=3)
predict(res, newmods = c(1/3,1/3), transf=exp, digits=3)

### fit meta-regression model with absolute latitude and allocation as predictors
res <- rma(yi, vi, mods = ~ ablat + alloc, data=dat)
res

### create reference grid
sav <- emmprep(res)

### estimated marginal mean using equal cell weighting
if (require(emmeans))
  emmeans(sav, specs="1", type="response")

### same as the predicted effect at the mean absolute latitude and using equal proportions
### for the allocation factor
predict(res, newmods = c(mean(model.matrix(res, asdf=TRUE)$ablat),1/3,1/3), transf=exp, digits=3)

### create reference grid with ablat set equal to 10, 30, and 50 degrees
sav <- emmprep(res, at=list(ablat=c(10,30,50)))

### estimated marginal means at the three ablat values
if (require(emmeans))
  emmeans(sav, specs="1", by="ablat", type="response")

### same as the predicted effect at the chosen absolute latitude values and using equal
### proportions for the allocation factor
predict(res, newmods = cbind(c(10,30,50),1/3,1/3), transf=exp, digits=3)

#####

### copy data into 'dat' and examine data
dat <- dat.mcdaniel1994
head(dat)

### calculate r-to-z transformed correlations and corresponding sampling variances
dat <- escalc(measure="ZCOR", ri=ri, ni=ni, data=dat)

### mixed-effects model with interview type as factor
res <- rma(yi, vi, mods = ~ factor(type), data=dat, test="knha")
res

```

```

### create reference grid
sav <- emmprep(res, verbose=TRUE)

### estimated marginal mean (back-transformed to the correlation scale)
if (require(emmeans))
  emmeans(sav, specs="1", type="response")

### same as the predicted correlation using equal cell proportions
predict(res, newmods = c(1/3,1/3), transf=transf.ztor, digits=3)

### estimated marginal means for the three interview types
if (require(emmeans))
  emmeans(sav, specs="type", type="response")

### same as the predicted correlations
predict(res, newmods = rbind(c(0,0), c(1,0), c(0,1)), transf=transf.ztor, digits=3)

### illustrate use of the 'df' and 'sigma' arguments
res <- rma.mv(yi, vi, mods = ~ factor(type), random = ~ 1 | study,
             data=dat, test="t", dfs="contain")
res

### create reference grid
sav <- emmprep(res, verbose=TRUE, df=154, sigma=0.1681)

### estimated marginal mean (back-transformed to the correlation scale)
if (require(emmeans))
  emmeans(sav, specs="1", type="response")

```

escalc

Calculate Effect Sizes and Outcome Measures

Description

Function to calculate various effect sizes or outcome measures (and the corresponding sampling variances) that are commonly used in meta-analyses.

Usage

```

escalc(measure, ai, bi, ci, di, n1i, n2i, x1i, x2i, t1i, t2i,
       m1i, m2i, sd1i, sd2i, xi, mi, ri, ti, fi, pi,
       sdi, r2i, mini, maxi, ni, yi, vi, sei,
       data, slab, flip, subset, include,
       add=1/2, to="only0", drop00=FALSE,
       vtype="LS", correct=TRUE, cutoff,
       var.names=c("yi", "vi"), add.measure=FALSE,
       append=TRUE, replace=TRUE, digits, ...)

```

Arguments

measure a character string to specify which effect size or outcome measure should be calculated (e.g., "SMD", "ZCOR", "OR"). See 'Details' for possible options and how the data needed to compute the selected effect size or outcome measure should then be specified (i.e., which of the following arguments need to be used).

These arguments pertain to data input:

ai	vector with the 2×2 table frequencies (upper left cell).
bi	vector with the 2×2 table frequencies (upper right cell).
ci	vector with the 2×2 table frequencies (lower left cell).
di	vector with the 2×2 table frequencies (lower right cell).
n1i	vector with the group sizes or row totals (first group/row).
n2i	vector with the group sizes or row totals (second group/row).
x1i	vector with the number of events (first group).
x2i	vector with the number of events (second group).
t1i	vector with the total person-times (first group).
t2i	vector with the total person-times (second group).
m1i	vector with the means (first group or time point).
m2i	vector with the means (second group or time point).
sd1i	vector with the standard deviations (first group or time point).
sd2i	vector with the standard deviations (second group or time point).
xi	vector with the frequencies of the event of interest.
mi	vector with the frequencies of the complement of the event of interest or the group means.
ri	vector with the raw correlation coefficients.
ti	vector with the total person-times or t-test statistics.
fi	vector with the F-test statistics.
pi	vector with the (signed) p-values.
sdi	vector with the standard deviations.
r2i	vector with the R^2 values.
mini	vector with the minimum values.
maxi	vector with the maximum values.
ni	vector with the sample/group sizes.
yi	vector with the effect size estimates.
vi	vector with the corresponding sampling variances.
sei	vector with the corresponding standard errors.
data	optional data frame containing the variables given to the arguments above.
slab	optional vector with labels for the studies.

flip	optional logical to indicate whether to flip the sign of the estimates. Can also be a vector. Can also be a numeric vector to specify a multiplier.
subset	optional (logical or numeric) vector to specify the subset of studies that will be included in the data frame returned by the function.
include	optional (logical or numeric) vector to specify the subset of studies for which the measure should be calculated. See the ‘Value’ section for more details.

These arguments pertain to handling of zero cells/counts/frequencies:

add	a non-negative number to specify the amount to add to zero cells, counts, or frequencies. See ‘Details’ and ‘Note’.
to	a character string to specify when the values under add should be added (either "all", "only0", "if0all", or "none"). See ‘Details’.
drop00	logical to specify whether studies with no cases/events (or only cases) in both groups should be dropped when calculating the estimates. See ‘Details’.

These arguments pertain to the computations:

vtype	a character string to specify the type of sampling variances to calculate. Can also be a vector. See ‘Details’.
correct	logical to specify whether a bias correction should be applied to the estimates (the default is TRUE).
cutoff	optional numeric value to adjust the cutoff for flagging large estimates. See ‘Details’.

These arguments pertain to the formatting of the returned data frame:

var.names	character vector with two elements to specify the name of the variable for the estimates and the name of the variable for the corresponding sampling variances (the defaults are "yi" and "vi").
add.measure	logical to specify whether a variable should be added to the data frame (with default name "measure") that indicates the type of measure computed. When using this option, var.names can have a third element to change this variable name.
append	logical to specify whether the data frame provided via the data argument should be returned together with the estimates and corresponding sampling variances (the default is TRUE).
replace	logical to specify whether existing values for yi and vi in the data frame should be replaced. Only relevant when append=TRUE and the data frame already contains the yi and vi variables. If replace=TRUE (the default), all of the existing values will be overwritten. If replace=FALSE, only NA values will be replaced. See the ‘Value’ section for more details.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is 4. Note that the values are stored without rounding in the returned object. See also here for further details on how to control the number of digits in the output.
...	other arguments.

Details

Before a meta-analysis can be conducted, the relevant results from each study must be quantified in such a way that the resulting values can be further aggregated and compared. Depending on (a) the goals of the meta-analysis, (b) the design and types of studies included, and (c) the information provided therein, one of the various effect sizes or ‘outcome measures’ described below may be appropriate for the meta-analysis and can be computed with the `escalc` function.

The `measure` argument is a character string to specify the measure that should be calculated (see below for the various options), arguments `ai` through `ni` are then used to specify the information needed to calculate the various measures (depending on the chosen measure, different arguments need to be specified), and `data` can be used to specify a data frame containing the variables given to the previous arguments. The `add`, `to`, and `drop00` arguments may be needed when dealing with frequency or count data that needs special handling when some of the frequencies or counts are equal to zero (see below for details). Finally, the `vtype` argument is used to specify how the sampling variances should be computed (again, see below for details).

To provide a structure to the measures that can be calculated with the `escalc` function, we can distinguish between measures that are used to:

- (1) contrast two independent (either experimentally created or naturally occurring) groups,
- (2) describe the direction and strength of the association between two variables,
- (3) summarize some characteristic or attribute of individual groups, or
- (4) quantify change within a single group or the difference between two matched/paired samples.

Furthermore, where appropriate, we can further distinguish between measures that are applicable when the characteristic, response, or dependent variable assessed within the individual studies is:

- (a) a quantitative variable (e.g., amount of depression as assessed by a rating scale),
- (b) a dichotomous (binary) variable (e.g., remission versus no remission),
- (c) a count of events per time unit (e.g., number of migraines per year), or
- (d) a mix of the types above.

Below, these number and letter codes are used (also in combination) to make it easier to quickly find a measure suitable for a particular meta-analysis (e.g., search for (1b) to find measures that describe the difference between two groups with respect to a dichotomous variable or (2a) for measures that quantify the association between two quantitative variables).

(1) Measures for Two-Group Comparisons:

In many meta-analyses, the goal is to synthesize the results from studies that compare or contrast two groups. The groups may be experimentally defined (e.g., a treatment and a control group created via random assignment) or may occur naturally (e.g., men and women, employees working under high- versus low-stress conditions, people/animals/plants exposed to some environmental risk factor versus those not exposed, patients versus controls).

(1a) Measures for Quantitative Variables:

When the response or dependent variable assessed within the individual studies is measured on a quantitative scale, it is customary to report certain summary statistics, such as the mean and standard deviation of the observations within the two groups (in case medians, min/max values, and quartiles are reported, see `conv.fivenum` for a function that can be used to estimate means

and standard deviations from such statistics). The data layout for a study comparing two groups with respect to such a variable is then of the form:

	mean	standard deviation	group size
group 1	m1i	sd1i	n1i
group 2	m2i	sd2i	n2i

where m1i and m2i are the observed means of the two groups, sd1i and sd2i are the observed standard deviations, and n1i and n2i denote the number of individuals in each group.

Measures for Differences in Central Tendency

Often, interest is focused on differences between the two groups with respect to their central tendency. The raw mean difference, the standardized mean difference, and the (log transformed) ratio of means (also called the log ‘response ratio’) are useful measures when meta-analyzing studies of this type.

The options for the measure argument are then:

- "MD" for the *raw mean difference* (e.g., Borenstein, 2009),
- "SMD" for the *standardized mean difference* (Hedges, 1981),
- "SMDH" for the *standardized mean difference* with heteroscedastic population variances in the two groups (Bonett, 2008, 2009),
- "SMD1" for the *standardized mean difference* where the mean difference is divided by the standard deviation of the second group (and "SMD1H" for the same but with heteroscedastic population variances),
- "ROM" for the *log transformed ratio of means* (Hedges et al., 1999; Lajeunesse, 2011),
- "POMPMD" for the *difference of POMP means*.

The raw mean difference is simply (m1i – m2i), while the standardized mean difference is given by (m1i – m2i)/sdi. For measure="SMD", $sdi = \sqrt{\frac{(n1i-1)sd1i^2 + (n2i-1)sd2i^2}{n1i + n2i - 2}}$ is the pooled standard deviation of the two groups (assuming homoscedasticity of the population variances).

For measure="SMDH", $sdi = \sqrt{\frac{sd1i^2 + sd2i^2}{2}}$ is the square root of the average variance (allowing for heteroscedastic population variances). Finally, for measure="SMD1" and measure="SMD1H", sdi = sd2i (note: for measure="SMD1", only sd2i needs to be specified and sd1i is ignored).

For measure="SMD", the positive bias in the standardized mean difference (i.e., in a Cohen’s d value) is automatically corrected for within the function, yielding Hedges’ g (Hedges, 1981). Similarly, the analogous bias correction is applied for measure="SMDH" (Bonett, 2009), measure="SMD1" (Hedges, 1981), and measure="SMD1H". With correct=FALSE, these bias corrections can be switched off.

For measure="ROM", the log is taken of the ratio of means (i.e., log(m1i/m2i)), which makes this measure symmetric around 0 and results in a sampling distribution that is closer to normality. Hence, this measure cannot be computed when m1i and m2i have opposite signs (in fact, this measure is only meant to be used for ratio scale measurements, where both means should be positive anyway). A bias correction is also applied to this measure (Lajeunesse, 2015, equation 8) unless correct=FALSE.

For measure="POMPMD", arguments mini and maxi must be used to supply the theoretical (not observed!) minimum and maximum values of the measurements. These values are then used to convert the means of the two groups into percent of maximum possible (POMP) means (Cohen et al. 1999) and these are then further treated as described for measure="MD".

For measure="SMD", if the means and standard deviations are unknown for some studies, various other inputs can be used to recover the standardized mean differences. In the case that the

standardized mean differences (Cohen's d values) are directly available (e.g., they are reported in some studies), then these can be specified via argument `di`. If the point-biserial correlations (between the group dummy variable and the quantitative response/dependent variable) are known, these can be specified via argument `ri`. If the t -statistics from an independent samples (Student's) t -test are available, these can be specified via argument `ti`. Note that the sign of these inputs is then taken to be the sign of the standardized mean differences. If only the two-sided p -values corresponding to the t -tests are known, one can specify those values via argument `pi` (which are then transformed into the t -statistics and then further into the standardized mean differences). However, since a two-sided p -value does not carry information about the sign of the test statistic (and hence neither about the standardized mean difference), the sign of the p -values (which can be negative) is used as the sign of the standardized mean differences (e.g., `escalc(measure="SMD", pi=-0.018, n1i=20, n2i=20)` yields a negative standardized mean difference of -0.7664). See [here](#) for a more detailed illustration of using the `di`, `ti`, and `pi` arguments.

For `measure="MD"`, one can choose between `vtype="LS"` (the default) and `vtype="H0"`. The former computes the sampling variances without assuming homoscedasticity (i.e., that the true variances of the measurements are the same in group 1 and group 2 within each study), while the latter assumes homoscedasticity (equations 12.5 and 12.3 in Borenstein, 2009, respectively). For `measure="SMD"`, one can choose between `vtype="LS"` (the default) for the usual large-sample approximation to compute the sampling variances (equation 8 in Hedges, 1982), `vtype="LS2"` to compute the sampling variances as described in Borenstein (2009; equation 12.17), `vtype="LS3"` to compute the sampling variances as done in Revman (equation 14 in Lin & Aloe, 2020), `vtype="UB"` to compute unbiased estimates of the sampling variances (equation 9 in Hedges, 1983), `vtype="AV"` to compute the sampling variances with the usual large-sample approximation but plugging the sample-size weighted average of the Hedges' g values into the equation, and `vtype="H0"` to compute the sampling variances under the null hypothesis (that the true standardized mean differences are equal to zero). The same choices also apply to `measure="SMD1"`. For `measure="ROM"`, one can choose between `vtype="LS"` (the default) for the usual large-sample approximation to compute the sampling variances (equation 1 in Hedges et al., 1999), `vtype="H0"` to compute the sampling variances assuming homoscedasticity (the unnumbered equation after equation 1 in Hedges et al., 1999), `vtype="LS2"` to compute the sampling variances based on the second-order Taylor expansion (equation 9 in Lajeunesse, 2015), `vtype="AV"` to compute the sampling variances assuming homoscedasticity of the coefficient of variation within each group across studies, `vtype="AVH0"` to compute the sampling variances assuming homoscedasticity of the coefficient of variation for both groups across studies (see Nakagawa et al., 2023, for details on the latter two options and why they can be advantageous).

Datasets corresponding to data of this type are provided in [dat.normand1999](#), [dat.curtis1998](#), and [dat.gibson2002](#).

Measures for Variability Differences

Interest may also be focused on differences between the two groups with respect to their variability. For this, the (log transformed) ratio of the standard deviations (also called the 'variability ratio') can be used (Nakagawa et al., 2015). To also account for differences in mean levels, the (log transformed) ratio between the coefficients of variation from the two groups (also called the 'coefficient of variation ratio') can be a useful measure (Nakagawa et al., 2015). The options for the `measure` argument are:

- "`VR`" for the *log transformed variability ratio*,
- "`CVR`" for the *log transformed coefficient of variation ratio*.

Measure "VR" is computed with $\log(sd1i/sd2i)$ and hence one only needs to specify $sd1i$, $sd2i$, $n1i$, and $n2i$ (i.e., arguments $m1i$ and $m2i$ are irrelevant). Measure "CVR" is computed with $\log((sd1i/m1i)/(sd2i/m2i))$. Note that a bias correction is applied for both of these measures (Senior et al., 2020, equations 5 and 6) unless `correct=FALSE`. When `vtype="LS"` (the default), then the sampling variances are computed with equations 11 and 13 from Senior et al. (2020) for "VR" and "CVR", respectively. When `vtype="LS2"`, then equations 15 and 16 (with a slight correction, multiplying the third and sixth term by 1/2) based on the second-order Taylor expansions are used instead.

Measures for Stochastic Superiority

Another way to quantify the difference between two groups is in terms of the 'common language effect size' (CLES) (McGraw & Wong, 1992). This measure provides an estimate of $P(X > Y)$, that is, the probability that a randomly chosen person from the first group has a larger value on the response variable than a randomly chosen person from the second group (or in case X and Y values can be tied, we define the measure as $P(X > Y) + \frac{1}{2}P(X = Y)$). This measure is identical to the area under the curve (AUC) under the receiver operating characteristic (ROC) curve (e.g., for a diagnostic test or more broadly for a binary classifier) and the 'concordance probability' (or c-statistic) and is directly related to the U statistic from the Mann-Whitney U test (i.e., $CLES = U/(n_1 \times n_2)$).

If the CLES/AUC values with corresponding sampling variances (or standard errors) are known, they can be directly meta-analyzed for example using the `rma.uni` function. However, in practice, one is likely to encounter studies that only report CLES/AUC values and the group sizes. In this case, one can specify these values via the `ai`, `n1i`, and `n2i` arguments and set `measure="CLES"` (or equivalently `measure="AUC"`). If `vtype="LS"` (the default), the sampling variances are then computed based on Newcombe (2006) (method 4), but using $(n1i-1)(n2i-1)$ in the denominator as suggested by Cho et al. (2019). If `vtype="LS2"`, the sampling variances are computed based on Hanley and McNeil (1982; equations 1 and 2), again using $(n1i-1)(n2i-1)$ in the denominator (and in the unlikely case that the proportion of tied values is known, this can be specified via argument `mi`, in which case the adjustment as described by Cho et al. (2019) is also applied).

Under the assumption that the data within the two groups are normally distributed (the so-called binormal model), one can also estimate the CLES/AUC values from the means and standard deviations of the two groups. For this, one sets `measure="CLESN"` (or equivalently `measure="AUCN"`) and specifies the means via arguments `m1i` and `m2i`, the standard deviations via arguments `sd1i` and `sd2i`, and the group sizes via arguments `n1i` and `n2i`. If `vtype="LS"` (the default), the sampling variances are then computed based on the large-sample approximation derived via the delta method (equation 3 (with a correction, since the plus sign in front of the last term in braces should be a multiplication sign) and equation 4 in Goddard & Hinberg, 1990, but using $n1i-1$ and $n2i-1$ in the denominators). Computing the CLES/AUC values and corresponding sampling variances does not assume homoscedasticity of the variances in the two groups. However, when `vtype="H0"`, then homoscedasticity is assumed (this will also affect the calculation of the CLES/AUC values themselves). As for `measure="SMD"`, one can also specify standardized mean differences via argument `di`, t-statistics from an independent samples t-test via argument `ti`, and/or signed two-sided p-values corresponding to the t-tests via argument `pi`, which all can be converted into CLES/AUC values (note that this automatically assumes homoscedasticity). One can also directly specify binormal model CLES/AUC values via argument `ai` (but unless the corresponding `sd1i` and `sd2i` values are also specified, the sampling variances are then computed under the assumption of homoscedasticity).

(1b) Measures for Dichotomous Variables:

In various fields of research (such as the health and medical sciences), the response variable measured within the individual studies is often dichotomous (binary), so that the data from a study comparing two different groups can be expressed in terms of a 2×2 table, such as:

	outcome 1	outcome 2	total
group 1	ai	bi	n1i
group 2	ci	di	n2i

where ai, bi, ci, and di denote the cell frequencies (i.e., the number of individuals falling into a particular category) and n1i and n2i are the row totals (i.e., the group sizes).

For example, in a set of randomized clinical trials, group 1 and group 2 may refer to the treatment and placebo/control group, respectively, with outcome 1 denoting some event of interest (e.g., death, complications, failure to improve under the treatment) and outcome 2 its complement. Similarly, in a set of cohort studies, group 1 and group 2 may denote those who engage in and those who do not engage in a potentially harmful behavior (e.g., smoking), with outcome 1 denoting the development of a particular disease (e.g., lung cancer) during the follow-up period. Finally, in a set of case-control studies, group 1 and group 2 may refer to those with the disease (i.e., cases) and those free of the disease (i.e., controls), with outcome 1 denoting, for example, exposure to some environmental risk factor in the past and outcome 2 non-exposure. Note that in all of these examples, the stratified sampling scheme fixes the row totals (i.e., the group sizes) by design.

A meta-analysis of studies reporting results in terms of 2×2 tables can be based on one of several different measures, including the risk ratio (also called the relative risk), the odds ratio, the risk difference, and the arcsine square root transformed risk difference (e.g., Fleiss & Berlin, 2009, Rücker et al., 2009). For any of these measures, one needs to specify the cell frequencies via the ai, bi, ci, and di arguments (or alternatively, one can use the ai, ci, n1i, and n2i arguments).

The options for the measure argument are then:

- "RR" for the *log risk ratio*,
- "OR" for the *log odds ratio*,
- "RD" for the *risk difference*,
- "AS" for the *arcsine square root transformed risk difference* (Rücker et al., 2009),
- "PETO" for the *log odds ratio* estimated with Peto's method (Yusuf et al., 1985).

Let $p1i = ai/n1i$ and $p2i = ci/n2i$ denote the proportion of individuals with outcome 1 in group 1 and group 2, respectively. Then the log risk ratio is computed with $\log(p1i/p2i)$, the log odds ratio with $\log\left(\left(\frac{p1i}{1-p1i}\right)/\left(\frac{p2i}{1-p2i}\right)\right)$, the risk difference with $p1i - p2i$, and the arcsine square root transformed risk difference with $\text{asin}(\sqrt{p1i}) - \text{asin}(\sqrt{p2i})$. See Yusuf et al. (1985) for the computation of the log odds ratio when measure="PETO". Note that the log is taken of the risk ratio and the odds ratio, which makes these measures symmetric around 0 and results in corresponding sampling distributions that are closer to normality. Also, when multiplied by 2, the arcsine square root transformed risk difference is identical to Cohen's h (Cohen, 1988). For all of these measures, a positive value indicates that the proportion of individuals with outcome 1 is larger in group 1 compared to group 2.

Cell entries with a zero count can be problematic, especially for the risk ratio and the odds ratio. Adding a small constant to the cells of the 2×2 tables is a common solution to this problem. When to="only0" (the default), the value of add (the default is 1/2; but see 'Note') is added to each cell of those 2×2 tables with at least one cell equal to 0. When to="all",

the value of `add` is added to each cell of all 2×2 tables. When `to="if0all"`, the value of `add` is added to each cell of all 2×2 tables, but only when there is at least one 2×2 table with a zero cell. Setting `to="none"` or `add=0` has the same effect: No adjustment to the observed table frequencies is made. Depending on the measure and the data, this may lead to division by zero (when this occurs, the resulting value is recoded to NA). Also, studies where `ai=ci=0` or `bi=di=0` may be considered to be uninformative about the size of the effect and dropping such studies has sometimes been recommended (Higgins et al., 2019). This can be done by setting `drop00=TRUE`. The values for such studies will then be set to NA (i.e., missing).

Datasets corresponding to data of this type are provided in [dat.bcg](#), [dat.collins1985a](#), [dat.collins1985b](#), [dat.egger2001](#), [dat.hine1989](#), [dat.laopaiboon2015](#), [dat.lee2004](#), [dat.li2007](#), [dat.linde2005](#), [dat.nielweise2007](#), and [dat.yusuf1985](#).

If the 2×2 table is not available (or cannot be reconstructed, for example with the [conv.2x2](#) function) for a study, but the odds ratio and the corresponding confidence interval is reported, one can easily transform these values into the corresponding log odds ratio and sampling variance (and combine such a study with those that do report 2×2 table data). See the [conv.wald](#) function and [here](#) for an illustration/discussion of this.

(1c) Measures for Event Counts:

In medical and epidemiological studies comparing two different groups (e.g., treated versus untreated patients, exposed versus unexposed individuals), results are sometimes reported in terms of event counts (i.e., the number of events, such as strokes or myocardial infarctions) over a certain period of time. Data of this type are also referred to as ‘person-time data’. Assume that the studies report data in the form:

	number of events	total person-time
group 1	x_{1i}	t_{1i}
group 2	x_{2i}	t_{2i}

where x_{1i} and x_{2i} denote the number of events in the first and the second group, respectively, and t_{1i} and t_{2i} the corresponding total person-times at risk. Often, the person-time is measured in years, so that t_{1i} and t_{2i} denote the total number of follow-up years in the two groups.

This form of data is fundamentally different from what was described in the previous section, since the total follow-up time may differ even for groups of the same size and the individuals studied may experience the event of interest multiple times. Hence, different measures than the ones described in the previous section need to be considered when data are reported in this format. These include the incidence rate ratio, the incidence rate difference, and the square root transformed incidence rate difference (Bagos & Nikolopoulos, 2009; Rothman et al., 2008). For any of these measures, one needs to specify the total number of events via the x_{1i} and x_{2i} arguments and the corresponding total person-time values via the t_{1i} and t_{2i} arguments.

The options for the measure argument are then:

- "IRR" for the *log incidence rate ratio*,
- "IRD" for the *incidence rate difference*,
- "IRSD" for the *square root transformed incidence rate difference*.

Let $ir_{1i} = x_{1i}/t_{1i}$ and $ir_{2i} = x_{2i}/t_{2i}$ denote the observed incidence rates in each group. Then the log incidence rate ratio is computed with $\log(ir_{1i}/ir_{2i})$, the incidence rate difference with $ir_{1i} - ir_{2i}$, and the square root transformed incidence rate difference with $\sqrt{ir_{1i}} - \sqrt{ir_{2i}}$. Note that the log is taken of the incidence rate ratio, which makes this measure symmetric around 0 and results in a sampling distribution that is closer to normality.

Studies with zero events in one or both groups can be problematic, especially for the incidence rate ratio. Adding a small constant to the number of events is a common solution to this problem. When `to="only0"` (the default), the value of `add` (the default is 1/2; but see 'Note') is added to `x1i` and `x2i` only in the studies that have zero events in one or both groups. When `to="all"`, the value of `add` is added to `x1i` and `x2i` in all studies. When `to="if0all"`, the value of `add` is added to `x1i` and `x2i` in all studies, but only when there is at least one study with zero events in one or both groups. Setting `to="none"` or `add=0` has the same effect: No adjustment to the observed number of events is made. Depending on the measure and the data, this may lead to division by zero (when this occurs, the resulting value is recoded to NA). Like for 2×2 table data, studies where `x1i=x2i=0` may be considered to be uninformative about the size of the effect and dropping such studies has sometimes been recommended. This can be done by setting `drop00=TRUE`. The values for such studies will then be set to NA. Datasets corresponding to data of this type are provided in [dat.hart1999](#) and [dat.nielweise2008](#).

(1d) Transforming SMDs to ORs and Vice-Versa:

In some meta-analyses, one may encounter studies that contrast two groups with respect to a quantitative response variable (case 1a above) and other studies that contrast the same two groups with respect to a dichotomous variable (case 2b above). If both types of studies are to be combined in the same analysis, one needs to compute the same measure across all studies.

For this, one may need to transform standardized mean differences into log odds ratios (e.g., Cox & Snell, 1989; Chinn, 2000; Hasselblad & Hedges, 1995; Sánchez-Meca et al., 2003). Here, the data need to be specified as described under (1a) and the options for the measure argument are then:

- "D2ORN" for the *transformed standardized mean difference* assuming normal distributions,
- "D2ORL" for the *transformed standardized mean difference* assuming logistic distributions.

Both of these transformations provide an estimate of the log odds ratio, the first assuming that the responses within the two groups are normally distributed, while the second assumes that the responses follow logistic distributions.

Alternatively, assuming that the dichotomous outcome in a 2×2 table is actually a dichotomized version of the responses on an underlying quantitative scale, it is also possible to estimate the standardized mean difference based on 2×2 table data, using either the probit transformed risk difference or a transformation of the odds ratio (e.g., Cox & Snell, 1989; Chinn, 2000; Hasselblad & Hedges, 1995; Sánchez-Meca et al., 2003). Here, the data need to be specified as described under (1b) and the options for the measure argument are then:

- "PBIT" for the *probit transformed risk difference*,
- "OR2DN" for the *transformed odds ratio* assuming normal distributions,
- "OR2DL" for the *transformed odds ratio* assuming logistic distributions.

All of these transformations provide an estimate of the standardized mean difference, the first two assuming that the responses on the underlying quantitative scale are normally distributed, while the third assumes that the responses follow logistic distributions.

A dataset illustrating the combined analysis of standardized mean differences and probit transformed risk differences is provided in [dat.gibson2002](#).

(2) Measures for Variable Association:

Meta-analyses are often used to synthesize studies that examine the direction and strength of the association between two variables measured concurrently and/or without manipulation by experimenters. In this section, a variety of measures will be discussed that may be suitable for a meta-analysis with this purpose. We can distinguish between measures that are applicable when both

variables are measured on quantitative scales, when both variables measured are dichotomous, and when the two variables are of mixed types.

(2a) Measures for Two Quantitative Variables:

The (Pearson or product-moment) correlation coefficient quantifies the direction and strength of the (linear) relationship between two quantitative variables and is therefore frequently used in meta-analyses. Two alternative measures are a bias-corrected version of the correlation coefficient and Fisher's *r*-to-*z* transformed correlation coefficient.

For these measures, one needs to specify *ri*, the vector with the raw correlation coefficients, and *ni*, the corresponding sample sizes. The options for the measure argument are then:

- "COR" for the *raw correlation coefficient*,
- "UCOR" for the *raw correlation coefficient* corrected for its slight negative bias (based on equation 2.3 in Olkin & Pratt, 1958),
- "ZCOR" for *Fisher's r-to-z transformed correlation coefficient* (Fisher, 1921).

If the correlation coefficient is unknown for some studies, but the *t*-statistics (i.e., $t_i = r_i \sqrt{n_i - 2} / \sqrt{1 - r_i^2}$) are available for those studies (for the standard test of $H_0: \rho_i = 0$), one can specify those values via argument *ti*, which are then transformed into the corresponding correlation coefficients within the function (the sign of the *t*-statistics is then taken to be the sign of the correlations). If only the two-sided *p*-values corresponding to the *t*-tests are known, one can specify those values via argument *pi*. However, since a two-sided *p*-value does not carry information about the sign of the test statistic (and hence neither about the correlation), the sign of the *p*-values (which can be negative) is used as the sign of the correlation coefficients (e.g., `escalc(measure="COR", pi=-0.07, ni=30)` yields a negative correlation of -0.3354).

For `measure="COR"` and `measure="UCOR"`, one can choose between `vtype="LS"` (the default) for the usual large-sample approximation to compute the sampling variances (i.e., plugging the (biased-corrected) correlation coefficients into equation 12.27 in Borenstein, 2009) and `vtype="AV"` to compute the sampling variances with the usual large-sample approximation but plugging the sample-size weighted average of the (bias-corrected) correlation coefficients into the equation. For `measure="COR"`, one can also choose `vtype="H0"` to compute the sampling variances under the null hypothesis (that the true correlations are equal to zero). For `measure="UCOR"`, one can also choose `vtype="UB"` to compute unbiased estimates of the sampling variances (see Hedges, 1989, but using the exact equation instead of the approximation). Datasets corresponding to data of this type are provided in [dat.mcdaniel1994](#) and [dat.molloy2014](#). For meta-analyses involving multiple (dependent) correlation coefficients extracted from the same sample, see also the `rcalc` function.

(2b) Measures for Two Dichotomous Variables:

When the goal of a meta-analysis is to examine the relationship between two dichotomous variables, the data for each study can again be presented in the form of a 2×2 table, except that there may not be a clear distinction between the grouping variable and the outcome variable. Moreover, the table may be a result of cross-sectional (i.e., multinomial) sampling, where none of the table margins (except the total sample size) are fixed by the study design.

In particular, assume that the data of interest for a particular study are of the form:

	variable 2, outcome +	variable 2, outcome -	total
variable 1, outcome +	ai	bi	n1i
variable 1, outcome -	ci	di	n2i

where *ai*, *bi*, *ci*, and *di* denote the cell frequencies (i.e., the number of individuals falling into

a particular category) and $n1i$ and $n2i$ are the row totals.

The phi coefficient and the odds ratio are commonly used measures of association for 2×2 table data (e.g., Fleiss & Berlin, 2009). The latter is particularly advantageous, as it is directly comparable to values obtained from stratified sampling (as described earlier). Yule's Q and Yule's Y (Yule, 1912) are additional measures of association for 2×2 table data (although they are not typically used in meta-analyses). Finally, assuming that the two dichotomous variables are actually dichotomized versions of the responses on two underlying quantitative scales (and assuming that the two variables follow a bivariate normal distribution), it is also possible to estimate the correlation between the two quantitative variables using the tetrachoric correlation coefficient (Pearson, 1900; Kirk, 1973).

For any of these measures, one needs to specify the cell frequencies via the a_i , b_i , c_i , and d_i arguments (or alternatively, one can use the a_i , c_i , $n1i$, and $n2i$ arguments). The options for the measure argument are then:

- "OR" for the *log odds ratio*,
- "PHI" for the *phi coefficient*,
- "YUQ" for *Yule's Q* (Yule, 1912),
- "YUY" for *Yule's Y* (Yule, 1912),
- "RTET" for the *tetrachoric correlation coefficient*.

There are also measures "ZPHI" and "ZTET" for applying Fisher's r-to-z transformation to these measures. This may be useful when combining these with other types of correlation coefficients that were r-to-z transformed. However, note that the r-to-z transformation is *not* a variance-stabilizing transformation for these measures.

Tables with one or more zero counts are handled as described earlier. For `measure="PHI"`, one must indicate via `vtype="ST"` or `vtype="CS"` whether the data for the studies were obtained using stratified or cross-sectional (i.e., multinomial) sampling, respectively (it is also possible to specify an entire vector for the `vtype` argument in case the sampling scheme differed for the various studies). Note that `vtype="LS"` is treated like `vtype="CS"`.

A dataset corresponding to data of this type is provided in `dat.bourassa1996`.

(2d) Measures for Mixed Variable Types:

We can also consider measures that can be used to describe the relationship between two variables, where one variable is dichotomous and the other variable measures some quantitative characteristic. In that case, it is likely that study authors again report summary statistics, such as the mean and standard deviation of the measurements within the two groups (defined by the dichotomous variable). Based on this information, one can compute the point-biserial correlation coefficient (Tate, 1954) as a measure of association between the two variables. If the dichotomous variable is actually a dichotomized version of the responses on an underlying quantitative scale (and assuming that the two variables follow a bivariate normal distribution), it is also possible to estimate the correlation between the two variables using the biserial correlation coefficient (Pearson, 1909; Soper, 1914; Jacobs & Viechtbauer, 2017).

Here, one again needs to specify $m1i$ and $m2i$ for the observed means of the two groups, $sd1i$ and $sd2i$ for the observed standard deviations, and $n1i$ and $n2i$ for the number of individuals in each group. The options for the measure argument are then:

- "RPB" for the *point-biserial correlation coefficient*,
- "RBIS" for the *biserial correlation coefficient*.

There are also measures "ZPB" and "ZBIS" for applying Fisher's r-to-z transformation to these measures. This may be useful when combining these with other types of correlation coefficients

that were r-to-z transformed. However, note that the r-to-z transformation is *not* a variance-stabilizing transformation for these measures.

If the means and standard deviations are unknown for some studies, one can also use arguments *di*, *ri*, *ti*, or *pi* to specify standardized mean differences (Cohen's *d* values), point-biserial correlations, t-statistics from an independent samples t-test, or signed p-values for the t-test, respectively, as described earlier under (1a) (together with the group sizes, these are sufficient statistics for computing the (point-)biserial correlation coefficients).

For *measure*="RPB", one must indicate via *vtype*="ST" or *vtype*="CS" whether the data for the studies were obtained using stratified or cross-sectional (i.e., multinomial) sampling, respectively (it is also possible to specify an entire vector for the *vtype* argument in case the sampling scheme differed for the various studies). Note that *vtype*="LS" is treated like *vtype*="ST".

(3) Measures for Individual Groups:

In this section, measures will be described which may be useful when the goal of a meta-analysis is to synthesize studies that characterize some property of individual groups. We will again distinguish between measures that are applicable when the characteristic assessed is a quantitative variable, a dichotomous variable, or when the characteristic represents an event count.

(3a) Measures for Quantitative Variables:

The goal of a meta-analysis may be to characterize individual groups, where the response, characteristic, or dependent variable assessed in the individual studies is measured on some quantitative scale. In the simplest case, the raw mean for the quantitative variable is reported for each group, which then becomes the estimate of interest for the meta-analysis. Here, one needs to specify *mi*, *sdi*, and *ni* for the observed means, the observed standard deviations, and the sample sizes, respectively. One can also compute the 'single-group standardized mean', where the mean is divided by the standard deviation (when first subtracting some fixed constant from each mean, then this is the 'single-group standardized mean difference'). Also, one can transform means into percent of possible maximum (POMP) means (Cohen et al., 1999), which requires that the *mini* and *maxi* arguments are used to supply the theoretical (not observed!) minimum and maximum values of the measurements.

If focus is solely on the variability of the measurements, then the log transformed standard deviation is a useful measure (Nakagawa et al., 2015; Raudenbush & Bryk, 1987). For this measure, one only needs to specify *sdi* and *ni*. For ratio scale measurements, the log transformed mean or the log transformed coefficient of variation may also be of interest (Nakagawa et al., 2015).

The options for the *measure* argument are:

- "MN" for the *raw mean*,
- "SMN" for the *single-group standardized mean (difference)*,
- "POMPMN" for the *POMP mean*,
- "MNLN" for the *log transformed mean*,
- "SDLN" for the *log transformed standard deviation*,
- "CVLN" for the *log transformed coefficient of variation*.

Note that *sdi* is used to specify the standard deviations of the observed values of the response, characteristic, or dependent variable and not the standard errors of the means. Also, the sampling variances for *measure*="CVLN" are computed as given by equation 27 in Nakagawa et al. (2015), but without the ' $-2\rho \dots$ ' term, since for normally distributed data (which we assume here) the mean and variance (and transformations thereof) are independent.

(3b) Measures for Dichotomous Variables:

A meta-analysis may also be conducted to aggregate studies that provide data about individual groups with respect to a dichotomous dependent variable. Here, one needs to specify x_i and n_i , denoting the number of individuals experiencing the event of interest and the total number of individuals within each study, respectively. Instead of specifying n_i , one can use m_i to specify the number of individuals that do not experience the event of interest (i.e., $m_i = n_i - x_i$). The options for the measure argument are then:

- "PR" for the *raw proportion*,
- "PLN" for the *log transformed proportion*,
- "PLO" for the *logit transformed proportion* (i.e., log odds),
- "PRZ" for the *probit transformed proportion*,
- "PAS" for the *arcsine square root transformed proportion* (i.e., the angular transformation),
- "PFT" for the *Freeman-Tukey double arcsine transformed proportion* (Freeman & Tukey, 1950).

However, for reasons discussed in Schwarzer et al. (2019) and Röver and Friede (2022), the use of double arcsine transformed proportions for a meta-analysis is not recommended.

Zero cell entries can be problematic for certain measures. When `to="only0"` (the default), the value of `add` (the default is 1/2; but see 'Note') is added to x_i and m_i only for studies where x_i or m_i is equal to 0. When `to="all"`, the value of `add` is added to x_i and m_i in all studies. When `to="if0all"`, the value of `add` is added in all studies, but only when there is at least one study with a zero value for x_i or m_i . Setting `to="none"` or `add=0` has the same effect: No adjustment to the observed values is made. Depending on the measure and the data, this may lead to division by zero (when this occurs, the resulting value is recoded to NA).

Datasets corresponding to data of this type are provided in [dat.pritz1997](#), [dat.debruin2009](#), [dat.hannum2020](#), and [dat.crisafuli2020](#).

(3c) Measures for Event Counts:

Various measures can be used to characterize individual groups when the dependent variable assessed is an event count. Here, one needs to specify x_i and t_i , denoting the number of events that occurred and the total person-times at risk, respectively. The options for the measure argument are then:

- "IR" for the *raw incidence rate*,
- "IRLN" for the *log transformed incidence rate*,
- "IRS" for the *square root transformed incidence rate*,
- "IRFT" for the *Freeman-Tukey transformed incidence rate* (Freeman & Tukey, 1950).

Measures "IR" and "IRLN" can also be used when meta-analyzing standardized incidence ratios (SIRs), where the observed number of events is divided by the expected number of events. In this case, arguments x_i and t_i are used to specify the observed and expected number of events in the studies. Since SIRs are not symmetric around 1, it is usually more appropriate to meta-analyze the log transformed SIRs (i.e., using measure "IRLN"), which are symmetric around 0.

Studies with zero events can be problematic, especially for the log transformed incidence rate. Adding a small constant to the number of events is a common solution to this problem. When `to="only0"` (the default), the value of `add` (the default is 1/2; but see 'Note') is added to x_i only in the studies that have zero events. When `to="all"`, the value of `add` is added to x_i in all studies. When `to="if0all"`, the value of `add` is added to x_i in all studies, but only when there is at least one study with zero events. Setting `to="none"` or `add=0` has the same effect: No adjustment to the observed number of events is made. Depending on the measure and the data, this may lead to division by zero (when this occurs, the resulting value is recoded to NA).

(4) Measures for Change or Matched Pairs:

The purpose of a meta-analysis may be to assess the amount of change within individual groups (e.g., before and after a treatment or under two different treatments) or when dealing with matched pairs designs.

(4a) Measures for Quantitative Variables:

When the response or dependent variable assessed in the individual studies is measured on some quantitative scale, the raw mean change, standardized versions thereof, the common language effect size (area under the curve), or the (log transformed) ratio of means (log response ratio) can be used as measures (Becker, 1988; Gibbons et al., 1993; Lajeunesse, 2011; Morris, 2000). Here, one needs to specify $m1i$ and $m2i$, the observed means at the two measurement occasions, $sd1i$ and $sd2i$ for the corresponding observed standard deviations, ri for the correlation between the measurements at the two measurement occasions, and ni for the sample size. The options for the measure argument are then:

- "MC" for the *raw mean change*,
- "SMCC" for the *standardized mean change* using change score standardization (Gibbons et al., 1993),
- "SMCR" for the *standardized mean change* using raw score standardization (Becker, 1988),
- "SMCRH" for the *standardized mean change* using raw score standardization with heteroscedastic population variances at the two measurement occasions (Bonett, 2008),
- "SMCRP" for the *standardized mean change* using raw score standardization with pooled standard deviations (Cousineau, 2020),
- "SMCRPH" for the *standardized mean change* using raw score standardization with pooled standard deviations and heteroscedastic population variances at the two measurement occasions (Bonett, 2008),
- "CLESCN" (or "AUCCN") for the *common language effect size* (area under the curve) based on a bivariate normal model for dependent samples,
- "ROMC" for the *log transformed ratio of means* (Lajeunesse, 2011),
- "POMPMC" for the *POMP mean change*.

The raw mean change is simply $m1i - m2i$, while the standardized mean change is given by $(m1i - m2i)/sdi$. For `measure="SMCC"`, $sdi = \sqrt{sd1i^2 + sd2i^2 - 2 \times ri \times sd1i \times sd2i}$ is the standard deviation of the change scores, for `measure="SMCR"` and `measure="SMCRH"`, $sdi = sd1i$, and for `measure="SMCRP"` and `measure="SMCRPH"`, $sdi = \sqrt{\frac{sd1i^2 + sd2i^2}{2}}$ is the square root of the average variance. See also Morris and DeShon (2002) for a thorough discussion of the difference between the "SMCC" and "SMCR" change score measures. The log transformed ratio of means is simply $\log(m1i/m2i)$. Bias corrections are applied to measures "SMCC", "SMCR", "SMCRH", "SMCRP", "SMCRPH", and "ROMC" unless `correct=FALSE`. All of these measures are also applicable for matched pairs designs (subscripts 1 and 2 then simply denote the first and second group that are formed by the matching).

In practice, one often has a mix of information available from the individual studies to compute these measures. In particular, if $m1i$ and $m2i$ are unknown, but the raw mean change is directly reported in a particular study, then one can set $m1i$ to that value and $m2i$ to 0 (making sure that the raw mean change was computed as $m1i - m2i$ within that study and not the other way around). Also, for measures "MC" and "SMCC", if $sd1i$, $sd2i$, and ri are unknown, but the standard deviation of the change scores is directly reported, then one can set $sd1i$ to that value and both $sd2i$ and ri to 0. For `measure="SMCR"`, argument $sd2i$ is actually not needed, as the standardization

is only based on `sd1i` (Becker, 1988; Morris, 2000), which is usually the pre-test standard deviation (if the post-test standard deviation should be used, then set `sd1i` to that). Finally, for `measure="SMCC"`, one can also directly specify standardized mean change values via argument `di` or the t-statistics from a paired samples t-test or the corresponding two-sided p-values via argument `ti` or `pi`, respectively (which are then transformed into the corresponding standardized mean change values within the function). The sign of the p-values (which can be negative) is used as the sign of the standardized mean change values (e.g., `escalc(measure="SMCC", pi=-0.018, ni=50)` yields a negative standardized mean change value of -0.3408).

For `measure="POMPMC"`, arguments `mini` and `maxi` must be used to supply the theoretical (not observed!) minimum and maximum values of the measurements. These values are then used to convert the means into percent of maximum possible (POMP) means (Cohen et al. 1999) and these are then further treated as described for `measure="MC"`.

Finally, interest may also be focused on differences in the variability of the measurements at the two measurement occasions (or between the two matched groups). For this, the (log transformed) ratio of the standard deviations (also called the ‘variability ratio’) can be used (Senior et al., 2020). To also account for differences in mean levels, the (log transformed) ratio between the coefficients of variation from the two groups (also called the ‘coefficient of variation ratio’) can be a useful measure (Senior et al., 2020). The options for the `measure` argument are:

- "VRC" for the *log transformed variability ratio*,
- "CVRC" for the *log transformed coefficient of variation ratio*.

Note that a bias correction is applied to `measure="VRC"` unless `correct=FALSE`. When `vtype="LS"` (the default), then the sampling variances are computed with equations 21 and 23 from Senior et al. (2020) for "VRC" and "CVRC", respectively. When `vtype="LS2"`, then the second-order Taylor expansions are used instead (but not equations 22 and 24 from the same paper, as these are incorrect; the correct equations were rederived).

(4b) Measures for Dichotomous Variables:

The data for a study examining change in a dichotomous variable gives rise to a paired 2×2 table, which is of the form:

	trt 2 outcome 1	trt 2 outcome 2
trt 1 outcome 1	ai	bi
trt 1 outcome 2	ci	di

where `ai`, `bi`, `ci`, and `di` denote the cell frequencies. Note that ‘trt1’ and ‘trt2’ may be applied to a single group of subjects or to matched pairs of subjects. Also, ‘trt1’ and ‘trt2’ might refer to two different time points (e.g., before and after a treatment). In any case, the data from such a study can be rearranged into a marginal table of the form:

	outcome 1	outcome 2
trt 1	ai+bi	ci+di
trt 2	ai+ci	bi+di

which is of the same form as a 2×2 table that would arise in a study comparing/contrasting two independent groups.

The options for the `measure` argument that will compute estimates based on the marginal table are:

- "MPRR" for the matched pairs *marginal log risk ratio*,

- "MPOR" for the matched pairs *marginal log odds ratio*,
- "MPRD" for the matched pairs *marginal risk difference*.

See Becker and Balagtas (1993), Curtin et al. (2002), Elbourne et al. (2002), Fagerland et al. (2014), May and Johnson (1997), Newcombe (1998), Stedman et al. (2011), and Zou (2007) for discussions of these measures.

The options for the measure argument that will compute estimates based on the paired table are:

- "MPORC" for the *conditional log odds ratio*,
- "MPPETO" for the *conditional log odds ratio* estimated with Peto's method.

See Curtin et al. (2002) and Zou (2007) for discussions of these measures.

If only marginal tables are available, then another possibility is to compute the marginal log odds ratios based on these table directly. However, for the correct computation of the sampling variances, the correlations (phi coefficients) from the paired tables must be known (or 'guestimated'). To use this approach, set measure="MPORM" and use argument ri to specify the correlation coefficients. Instead of specifying ri, one can use argument pi to specify the proportions (or 'guestimates' thereof) of individuals (or pairs) that experienced the outcome of interest (i.e., 'outcome1' in the paired 2×2 table) under both treatments (i.e., $\pi_i = a_i / (a_i + b_i + c_i + d_i)$). Based on these proportions, the correlation coefficients are then back-calculated and used to compute the correct sampling variances. Note that the values in the marginal tables put constraints on the possible values for ri and pi. If a specified value for ri or pi is not feasible under a given table, the corresponding sampling variance will be NA.

(5) Other Measures for Meta-Analyses:

Other measures are sometimes used for meta-analyses that do not directly fall into the categories above. These are described in this section.

Cronbach's alpha and Transformations Thereof:

Meta-analytic methods can also be used to aggregate Cronbach's alpha values from multiple studies. This is usually referred to as a 'reliability generalization meta-analysis' (Vacha-Haase, 1998). Here, one needs to specify ai, mi, and ni for the observed alpha values, the number of items/replications/parts of the measurement instrument, and the sample sizes, respectively. One can either directly analyze the raw Cronbach's alpha values or transformations thereof (Bonett, 2002, 2010; Hakstian & Whalen, 1976). The options for the measure argument are then:

- "ARAW" for *raw alpha* values,
- "AHW" for *transformed alpha values* (Hakstian & Whalen, 1976),
- "ABT" for *transformed alpha values* (Bonett, 2002).

Note that the transformations implemented here are slightly different from the ones described by Hakstian and Whalen (1976) and Bonett (2002). In particular, for "AHW", the transformation $1 - (1 - a_i)^{1/3}$ is used, while for "ABT", the transformation $-\log(1 - a_i)$ is used. This ensures that the transformed values are monotonically increasing functions of ai.

A dataset corresponding to data of this type is provided in [dat.bonett2010](#).

Partial and Semi-Partial Correlations:

Aloe and Becker (2012), Aloe and Thompson (2013), and Aloe (2014) describe the use of partial and semi-partial correlation coefficients for meta-analyzing the results from regression models (when the focus is on a common regression coefficient of interest across studies). To compute these measures, one needs to specify ti for the test statistics (i.e., t-tests) of the 'focal' regression coefficient of interest, ni for the sample sizes of the studies, mi for the total number of predictors in the regression models (counting the focal predictor of interest, but not the intercept term), and

$r2i$ for the R^2 values of the regression models (the latter is only needed when `measure="SPCOR"` or `measure="ZSPCOR"`). The options for the `measure` argument are then:

- "PCOR" for the *partial correlation coefficient*,
- "ZPCOR" for *Fisher's r-to-z transformed partial correlation coefficient*,
- "SPCOR" for the *semi-partial correlation coefficient*,
- "ZSPCOR" for *Fisher's r-to-z transformed semi-partial correlation coefficient*.

Note that the signs of the (semi-)partial correlation coefficients is determined based on the signs of the values specified via the `ti` argument. Also, while the Fisher transformation can be applied to both measures, it is only a variance-stabilizing transformation for partial correlation coefficients.

If the test statistic (i.e., t-test) of the regression coefficient of interest is unknown for some studies, but the two-sided p-values corresponding to the t-tests are known, one can specify those values via argument `pi`. However, since a two-sided p-value does not carry information about the sign of the test statistic (and hence neither about the correlation), the sign of the p-values (which can be negative) is used as the sign of the correlation coefficients (e.g., `escalc(measure="PCOR", pi=-0.07, mi=5, ni=30)` yields a negative partial correlation of -0.3610).

In the rare case that the (semi-)partial correlations are known for some of the studies, then these can be directly specified via the `ri` argument. This can be useful, for example, when η_p^2 (i.e., partial eta squared) is known for the regression coefficient of interest, since the square root thereof is identical to the absolute value of the partial correlation (although the correct sign then still needs to be reconstructed based on other information).

A dataset corresponding to data of this type is provided in [dat.aloe2013](#).

Coefficients of Determination:

One can in principle also meta-analyze coefficients of determination (i.e., R^2 values / R-squared values) obtained from a series of linear regression models (however, see the caveat mentioned below). For this, one needs to specify $r2i$ for the R^2 values of the regression models, ni for the sample sizes of the studies, and mi for the number of predictors in the regression models (not counting the intercept term). The options for the `measure` argument are then:

- "R2" for the *raw coefficient of determination* with predictor values treated as random,
- "ZR2" for the corresponding *r-to-z transformed coefficient of determination*,
- "R2F" for the *raw coefficient of determination* with predictor values treated as fixed,
- "ZR2F" for the corresponding *r-to-z transformed coefficient of determination*.

If the R^2 values are unknown for some studies, but the F-statistics (for the omnibus test of the regression coefficients) are available, one can specify those values via argument `fi`, which are then transformed into the corresponding R^2 values within the function. If only the p-values corresponding to the F-tests are known, one can specify those values via argument `pi` (which are then transformed into the F-statistics and then further into the R^2 values).

For `measure="R2"`, one can choose to compute the sampling variances with `vtype="LS"` (the default) for the large-sample approximation given by equation 27.88 in Kendall and Stuart (1979), `vtype="LS2"` for the large-sample approximation given by equation 27.87, or `vtype="AV"` and `vtype="AV2"` which use the same approximations but plugging the sample-size weighted average of the R^2 values into the equations. For `measure="ZR2"`, the variance-stabilizing transformation $\frac{1}{2} \log \left(\frac{1+\sqrt{r2i}}{1-\sqrt{r2i}} \right)$ is used (see Olkin & Finn, 1995, but with the additional $\frac{1}{2}$ factor), which uses $1/ni$ as the large-sample approximation to the sampling variances. The equations used for these measures were derived under the assumption that the values of the outcome variable and the predictors were sampled from a multivariate normal distribution within each study

(sometimes called ‘random-X regression’) and that the sample sizes of the studies are large. Moreover, the equations assume that the true R^2 values are non-zero.

For the case where the predictor values are treated as fixed (sometimes called ‘fixed-X regression’), one can use measures “R2F” and “ZR2F”. Here, the sampling variances of the R^2 values are computed based on the known relationship between the non-central F-distribution and its non-centrality parameter (which in turn is a function of the true R^2). However, note that the r-to-z transformation is *not* a variance-stabilizing transformation for this case.

Given that observed R^2 values cannot be negative, there is no possibility for values to cancel each other out and hence it is guaranteed that the pooled estimate is positive. Hence, a meta-analysis of R^2 values cannot be used to test if the pooled estimate is different from zero (it is by construction as long as the number of studies is sufficiently large).

Relative Excess Heterozygosity:

Ziegler et al. (2011) describe the use of meta-analytic methods to examine deviations from the Hardy-Weinberg equilibrium across multiple studies. The relative excess heterozygosity (REH) is the proposed measure for such a meta-analysis, which can be computed by setting `measure="REH"`. Here, one needs to specify `ai` for the number of individuals with homozygous dominant alleles, `bi` for the number of individuals with heterozygous alleles, and `ci` for the number of individuals with homozygous recessives alleles.

Note that the log is taken of the REH values, which makes this measure symmetric around 0 and results in a sampling distribution that is closer to normality.

A dataset corresponding to data of this type is provided in `dat.frank2008`.

(6) Converting a Data Frame to an ‘escalc’ Object:

The function can also be used to convert a regular data frame to an ‘escalc’ object. One simply sets the `measure` argument to one of the options described above (or to `measure="GEN"` for a set of generic estimates not further specified) and passes the estimates via the `yi` argument and the corresponding sampling variances via the `vi` argument (or the standard errors via the `sei` argument) to the function.

Flagging Large Estimates:

For some of the measures above, the function runs a check to examine if one or more of the computed estimates are unusually large. This may happen due to data extraction or reporting errors and might warrant double-checking the values used as input to the function.

At the moment, this check is conducted for standardized mean differences (i.e., `measure="SMD"`) and its variants. A common mistake when computing such measures is that standard errors of the means are used in place of standard deviations. Recall that a standardized mean difference (before the bias correction) is given by $d = (m1i - m2i)/sdi$, where $sdi = \sqrt{\frac{(n1i-1)sd1i^2 + (n2i-1)sd2i^2}{n1i + n2i - 2}}$ is the pooled standard deviation and `sd1i` and `sd2i` are the standard deviations of the two groups (see section (1a) above). When using the standard errors of the means in place of the standard deviations, then `sdi` will be too small (since the standard error of a mean is equal to the standard deviation divided by the square root of the sample size) and the computed d value will be too large. An excessively large d value might be an indication that such a mistake was made. Since this invalidates all meta-analytic results computed based on such a value, the function checks if any of the calculated standardized mean differences are larger than 2 in absolute value. This cutoff is somewhat arbitrary and can be adjusted via the `cutoff` argument. The check can be skipped by setting `cutoff=NA`. Also, to avoid annoyance due to false positives, the check is only conducted once for a given set of estimates within a session.

Other Arguments:

Argument `slab` can be used to specify (study) labels for the estimates. These labels are passed on to other functions and used as needed (e.g., for labeling the estimates in a forest plot). Note that missing values in the labels are not allowed.

The `flip` argument, when set to `TRUE`, can be used to flip the sign of the estimates. This can also be a logical vector to indicate for which studies the sign should be flipped. The argument can also be a numeric vector to specify a multiplier for the estimates (the corresponding sampling variances are adjusted accordingly).

The `subset` argument can be used to select the studies that will be included in the data frame returned by the function. On the other hand, the `include` argument simply selects for which studies the measure will be computed (if it shouldn't be computed for all of them).

Value

An object of class `c("escalc", "data.frame")`. The object is a data frame containing the following components:

<code>yi</code>	vector with the estimates.
<code>vi</code>	vector with the corresponding sampling variances.

If a data frame was specified via the `data` argument and `append=TRUE`, then variables `yi` and `vi` are appended to this data frame. Note that the `var.names` argument actually specifies the names of these two variables ("`yi`" and "`vi`" are the defaults).

If the data frame already contains two variables with names as specified by the `var.names` argument, the values for these two variables will be overwritten when `replace=TRUE` (which is the default). By setting `replace=FALSE`, only values that are `NA` will be replaced.

The object is formatted and printed with the `print` function. The `summary` function can be used to obtain confidence intervals around the estimates. See `methods.escalc` for some additional method functions for "`escalc`" objects.

With the `aggregate` function, one can aggregate multiple estimates belonging to the same study (or some other clustering variable) into a single combined estimate.

Note

The variable names specified under `var.names` should be syntactically valid variable names. If necessary, they are adjusted so that they are.

Although the default value for `add` is `1/2`, for certain measures the use of such a bias correction makes little sense and for these measures, the function internally sets `add=0`. This applies to the following measures: "`AS`", "`PHI`", "`ZPHI`", "`RTET`", "`ZTET`", "`IRSD`", "`PAS`", "`PFT`", "`IRS`", and "`IRFT`". One can still force the use of the bias correction by explicitly setting the `add` argument to some non-zero value when calling the function.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Aloe, A. M. (2014). An empirical investigation of partial effect sizes in meta-analysis of correlational data. *Journal of General Psychology*, **141**(1), 47–64. <https://doi.org/10.1080/00221309.2013.853021>
- Aloe, A. M., & Becker, B. J. (2012). An effect size for regression predictors in meta-analysis. *Journal of Educational and Behavioral Statistics*, **37**(2), 278–297. <https://doi.org/10.3102/1076998610396901>
- Aloe, A. M., & Thompson, C. G. (2013). The synthesis of partial effect sizes. *Journal of the Society for Social Work and Research*, **4**(4), 390–405. <https://doi.org/10.5243/jsswr.2013.24>
- Bagos, P. G., & Nikolopoulos, G. K. (2009). Mixed-effects Poisson regression models for meta-analysis of follow-up studies with constant or varying durations. *The International Journal of Biostatistics*, **5**(1). <https://doi.org/10.2202/1557-4679.1168>
- Becker, B. J. (1988). Synthesizing standardized mean-change measures. *British Journal of Mathematical and Statistical Psychology*, **41**(2), 257–278. <https://doi.org/10.1111/j.2044-8317.1988.tb00901.x>
- Becker, M. P., & Balagtas, C. C. (1993). Marginal modeling of binary cross-over data. *Biometrics*, **49**(4), 997–1009. <https://doi.org/10.2307/2532242>
- Bonett, D. G. (2002). Sample size requirements for testing and estimating coefficient alpha. *Journal of Educational and Behavioral Statistics*, **27**(4), 335–340. <https://doi.org/10.3102/10769986027004335>
- Bonett, D. G. (2008). Confidence intervals for standardized linear contrasts of means. *Psychological Methods*, **13**(2), 99–109. <https://doi.org/10.1037/1082-989X.13.2.99>
- Bonett, D. G. (2009). Meta-analytic interval estimation for standardized and unstandardized mean differences. *Psychological Methods*, **14**(3), 225–238. <https://doi.org/10.1037/a0016619>
- Bonett, D. G. (2010). Varying coefficient meta-analytic methods for alpha reliability. *Psychological Methods*, **15**(4), 368–385. <https://doi.org/10.1037/a0020142>
- Borenstein, M. (2009). Effect sizes for continuous data. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 221–235). New York: Russell Sage Foundation.
- Chinn, S. (2000). A simple method for converting an odds ratio to effect size for use in meta-analysis. *Statistics in Medicine*, **19**(22), 3127–3131. [https://doi.org/10.1002/1097-0258\(20001130\)19:22<3127::ai](https://doi.org/10.1002/1097-0258(20001130)19:22<3127::ai)
- Cho, H., Matthews, G. J., & Harel, O. (2019). Confidence intervals for the area under the receiver operating characteristic curve in the presence of ignorable missing data. *International Statistical Review*, **87**(1), 152–177. <https://doi.org/10.1111/insr.12277>
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Cohen, P., Cohen, J., Aiken, L. S., & West, S. G. (1999). The problem of units and the circumstance for POMP. *Multivariate Behavioral Research*, **34**(3), 315–346. https://doi.org/10.1207/S15327906MBR3403_2
- Cousineau, D. (2020). Approximating the distribution of Cohen's d_p in within-subject designs. *The Quantitative Methods for Psychology*, **16**(4), 418–421. <https://doi.org/10.20982/tqmp.16.4.p418>
- Cox, D. R., & Snell, E. J. (1989). *Analysis of binary data* (2nd ed.). London: Chapman & Hall.
- Curtin, F., Elbourne, D., & Altman, D. G. (2002). Meta-analysis combining parallel and cross-over clinical trials. II: Binary outcomes. *Statistics in Medicine*, **21**(15), 2145–2159. <https://doi.org/10.1002/sim.1206>
- Elbourne, D. R., Altman, D. G., Higgins, J. P. T., Curtin, F., Worthington, H. V., & Vail, A. (2002). Meta-analyses involving cross-over trials: Methodological issues. *International Journal of Epidemiology*, **31**(1), 140–149. <https://doi.org/10.1093/ije/31.1.140>

- Fagerland, M. W., Lydersen, S., & Laake, P. (2014). Recommended tests and confidence intervals for paired binomial proportions. *Statistics in Medicine*, **33**(16), 2850–2875. <https://doi.org/10.1002/sim.6148>
- Fisher, R. A. (1921). On the “probable error” of a coefficient of correlation deduced from a small sample. *Metron*, **1**, 1–32. <https://hdl.handle.net/2440/15169>
- Fleiss, J. L., & Berlin, J. (2009). Effect sizes for dichotomous data. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 237–253). New York: Russell Sage Foundation.
- Freeman, M. F., & Tukey, J. W. (1950). Transformations related to the angular and the square root. *Annals of Mathematical Statistics*, **21**(4), 607–611. <https://doi.org/10.1214/aoms/1177729756>
- Gibbons, R. D., Hedeker, D. R., & Davis, J. M. (1993). Estimation of effect size from a series of experiments involving paired comparisons. *Journal of Educational Statistics*, **18**(3), 271–279. <https://doi.org/10.3102/10769986018003271>
- Goddard, M. J., & Hinberg, I. (1990). Receiver operator characteristic (ROC) curves and non-normal data: An empirical study. *Statistics in Medicine*, **9**(3), 325–337. <https://doi.org/10.1002/sim.4780090315>
- Hakstian, A. R., & Whalen, T. E. (1976). A k-sample significance test for independent alpha coefficients. *Psychometrika*, **41**(2), 219–231. <https://doi.org/10.1007/BF02291840>
- Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, **143**(1), 29–36. <https://doi.org/10.1148/radiology.143.1.7063747>
- Hasselblad, V., & Hedges, L. V. (1995). Meta-analysis of screening and diagnostic tests. *Psychological Bulletin*, **117**(1), 167–178. <https://doi.org/10.1037/0033-2909.117.1.167>
- Hedges, L. V. (1981). Distribution theory for Glass’s estimator of effect size and related estimators. *Journal of Educational Statistics*, **6**(2), 107–128. <https://doi.org/10.3102/10769986006002107>
- Hedges, L. V. (1982). Estimation of effect size from a series of independent experiments. *Psychological Bulletin*, **92**(2), 490–499. <https://doi.org/10.1037/0033-2909.92.2.490>
- Hedges, L. V. (1983). A random effects model for effect sizes. *Psychological Bulletin*, **93**(2), 388–395. <https://doi.org/10.1037/0033-2909.93.2.388>
- Hedges, L. V. (1989). An unbiased correction for sampling error in validity generalization studies. *Journal of Applied Psychology*, **74**(3), 469–477. <https://doi.org/10.1037/0021-9010.74.3.469>
- Hedges, L. V., Gurevitch, J., & Curtis, P. S. (1999). The meta-analysis of response ratios in experimental ecology. *Ecology*, **80**(4), 1150–1156. [https://doi.org/10.1890/0012-9658\(1999\)080\[1150:TMAORR\]2.0.CO;2](https://doi.org/10.1890/0012-9658(1999)080[1150:TMAORR]2.0.CO;2)
- Higgins, J. P. T., Thomas, J., Chandler, J., Cumpston, M., Li, T., Page, M. J., & Welch, V. A. (Eds.) (2019). *Cochrane handbook for systematic reviews of interventions* (2nd ed.). Chichester, UK: Wiley. <https://training.cochrane.org/handbook>
- Jacobs, P., & Viechtbauer, W. (2017). Estimation of the biserial correlation and its sampling variance for use in meta-analysis. *Research Synthesis Methods*, **8**(2), 161–180. <https://doi.org/10.1002/jrsm.1218>
- Kendall, M., & Stuart, A. (1979). *Kendall’s advanced theory of statistics, Vol. 2: Inference and relationship* (4th ed.). New York: Macmillan.
- Kirk, D. B. (1973). On the numerical approximation of the bivariate normal (tetrachoric) correlation coefficient. *Psychometrika*, **38**(2), 259–268. <https://doi.org/10.1007/BF02291118>
- Lajeunesse, M. J. (2011). On the meta-analysis of response ratios for studies with correlated and multi-group designs. *Ecology*, **92**(11), 2049–2055. <https://doi.org/10.1890/11-0423.1>
- Lajeunesse, M. J. (2015). Bias and correction for the log response ratio in ecological meta-analysis. *Ecology*, **96**(8), 2056–2063. <https://doi.org/10.1890/14-2402.1>

- Lin, L., & Aloe, A. M. (2020). Evaluation of various estimators for standardized mean difference in meta-analysis. *Statistics in Medicine*, **40**(2), 403–426. <https://doi.org/10.1002/sim.8781>
- May, W. L., & Johnson, W. D. (1997). Confidence intervals for differences in correlated binary proportions. *Statistics in Medicine*, **16**(18), 2127–2136. [https://doi.org/10.1002/\(SICI\)1097-0258\(19970930\)16:18<2127::AID-SIM581>3.0.CO;2-1](https://doi.org/10.1002/(SICI)1097-0258(19970930)16:18<2127::AID-SIM581>3.0.CO;2-1)
- McGraw, K. O., & Wong, S. P. (1992). A common language effect size statistic. *Psychological Bulletin*, **111**(2), 361–365. <https://doi.org/10.1037/0033-2909.111.2.361>
- Morris, S. B. (2000). Distribution of the standardized mean change effect size for meta-analysis on repeated measures. *British Journal of Mathematical and Statistical Psychology*, **53**(1), 17–29. <https://doi.org/10.1348/000711000159150>
- Morris, S. B., & DeShon, R. P. (2002). Combining effect size estimates in meta-analysis with repeated measures and independent-groups designs. *Psychological Methods*, **7**(1), 105–125. <https://doi.org/10.1037/1082-989X.7.1.105>
- Nakagawa, S., Poulin, R., Mengersen, K., Reinhold, K., Engqvist, L., Lagisz, M., & Senior, A. M. (2015). Meta-analysis of variation: Ecological and evolutionary applications and beyond. *Methods in Ecology and Evolution*, **6**(2), 143–152. <https://doi.org/10.1111/2041-210X.12309>
- Nakagawa, S., Noble, D. W. A., Lagisz, M., Spake, R., Viechtbauer, W., & Senior, A. M. (2023). A robust and readily implementable method for the meta-analysis of response ratios with and without missing standard deviations. *Ecology Letters*, **26**(2), 232–244. <https://doi.org/10.1111/ele.14144>
- Newcombe, R. G. (1998). Improved confidence intervals for the difference between binomial proportions based on paired data. *Statistics in Medicine*, **17**(22), 2635–2650. [https://doi.org/10.1002/\(SICI\)1097-0258\(199811\)17:22<2635::AID-SIM776>3.0.CO;2-1](https://doi.org/10.1002/(SICI)1097-0258(199811)17:22<2635::AID-SIM776>3.0.CO;2-1)
- Newcombe, R. G. (2006). Confidence intervals for an effect size measure based on the Mann-Whitney statistic. Part 2: Asymptotic methods and evaluation. *Statistics in Medicine*, **25**(4), 559–573. <https://doi.org/10.1002/sim.2324>
- Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin*, **118**(1), 155–164. <https://doi.org/10.1037/0033-2909.118.1.155>
- Olkin, I., & Pratt, J. W. (1958). Unbiased estimation of certain correlation coefficients. *Annals of Mathematical Statistics*, **29**(1), 201–211. <https://doi.org/10.1214/aoms/1177706717>
- Pearson, K. (1900). Mathematical contributions to the theory of evolution. VII. On the correlation of characters not quantitatively measurable. *Philosophical Transactions of the Royal Society of London, Series A*, **195**, 1–47. <https://doi.org/10.1098/rsta.1900.0022>
- Pearson, K. (1909). On a new method of determining correlation between a measured character A, and a character B, of which only the percentage of cases wherein B exceeds (or falls short of) a given intensity is recorded for each grade of A. *Biometrika*, **7**(1/2), 96–105. <https://doi.org/10.1093/biomet/7.1-2.96>
- Raudenbush, S. W., & Bryk, A. S. (1987). Examining correlates of diversity. *Journal of Educational Statistics*, **12**(3), 241–269. <https://doi.org/10.3102/10769986012003241>
- Rothman, K. J., Greenland, S., & Lash, T. L. (2008). *Modern epidemiology* (3rd ed.). Philadelphia: Lippincott Williams & Wilkins.
- Röver, C., & Friede, T. (2022). Double arcsine transform not appropriate for meta-analysis. *Research Synthesis Methods*, **13**(5), 645–648. <https://doi.org/10.1002/jrsm.1591>
- Rücker, G., Schwarzer, G., Carpenter, J., & Olkin, I. (2009). Why add anything to nothing? The arcsine difference as a measure of treatment effect in meta-analysis with zero cells. *Statistics in Medicine*, **28**(5), 721–738. <https://doi.org/10.1002/sim.3511>
- Sánchez-Meca, J., Marín-Martínez, F., & Chacón-Moscoso, S. (2003). Effect-size indices for dichotomized outcomes in meta-analysis. *Psychological Methods*, **8**(4), 448–467. <https://doi.org/10.1037/1082-989X.8.4.448>

- Schwarzer, G., Chemaitelly, H., Abu-Raddad, L. J., & Rücker, G. (2019). Seriously misleading results using inverse of Freeman-Tukey double arcsine transformation in meta-analysis of single proportions. *Research Synthesis Methods*, **10**(3), 476–483. <https://doi.org/10.1002/jrsm.1348>
- Senior, A. M., Viechtbauer, W., & Nakagawa, S. (2020). Revisiting and expanding the meta-analysis of variation: The log coefficient of variation ratio. *Research Synthesis Methods*, **11**(4), 553–567. <https://doi.org/10.1002/jrsm.1423>
- Soper, H. E. (1914). On the probable error of the bi-serial expression for the correlation coefficient. *Biometrika*, **10**(2/3), 384–390. <https://doi.org/10.1093/biomet/10.2-3.384>
- Stedman, M. R., Curtin, F., Elbourne, D. R., Kesselheim, A. S., & Brookhart, M. A. (2011). Meta-analyses involving cross-over trials: Methodological issues. *International Journal of Epidemiology*, **40**(6), 1732–1734. <https://doi.org/10.1093/ije/dyp345>
- Tate, R. F. (1954). Correlation between a discrete and a continuous variable: Point-biserial correlation. *Annals of Mathematical Statistics*, **25**(3), 603–607. <https://doi.org/10.1214/aoms/1177728730>
- Vacha-Haase, T. (1998). Reliability generalization: Exploring variance in measurement error affecting score reliability across studies. *Educational and Psychological Measurement*, **58**(1), 6–20. <https://doi.org/10.1177/0013164498058001002>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Yule, G. U. (1912). On the methods of measuring association between two attributes. *Journal of the Royal Statistical Society*, **75**(6), 579–652. <https://doi.org/10.2307/2340126>
- Yusuf, S., Peto, R., Lewis, J., Collins, R., & Sleight, P. (1985). Beta blockade during and after myocardial infarction: An overview of the randomized trials. *Progress in Cardiovascular Disease*, **27**(5), 335–371. [https://doi.org/10.1016/s0033-0620\(85\)80003-7](https://doi.org/10.1016/s0033-0620(85)80003-7)
- Ziegler, A., Steen, K. V. & Wellek, S. (2011). Investigating Hardy-Weinberg equilibrium in case-control or cohort studies or meta-analysis. *Breast Cancer Research and Treatment*, **128**(1), 197–201. <https://doi.org/10.1007/s10549-010-1295-z>
- Zou, G. Y. (2007). One relative risk versus two odds ratios: Implications for meta-analyses involving paired and unpaired binary data. *Clinical Trials*, **4**(1), 25–31. <https://doi.org/10.1177/1740774506075667>

See Also

[print](#) and [summary](#) for the print and summary methods.

[conv.2x2](#) for a function to reconstruct the cell frequencies of 2×2 tables based on other summary statistics.

[conv.fivenum](#) for a function to convert five-number summary values to means and standard deviations (needed to compute various effect size measures, such as raw or standardized mean differences and ratios of means / response ratios).

[conv.wald](#) for a function to convert Wald-type confidence intervals and test statistics to sampling variances.

[conv.delta](#) for a function to transform estimates and their sampling variances using the delta method.

[vcalc](#) for a function to construct or approximate the variance-covariance matrix of dependent estimates.

`rcalc` for a function to construct the variance-covariance matrix of dependent correlation coefficients.

`rma.uni` and `rma.mv` for model fitting functions that can take the calculated estimates (and the corresponding sampling variances) as input.

`rma.mh`, `rma.peto`, and `rma.glmm` for model fitting functions that take similar inputs.

Examples

```
#####

### data from the meta-analysis by Coliditz et al. (1994) on the efficacy of
### BCG vaccine in the prevention of tuberculosis dat.bcg
dat.bcg

### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
dat

### suppose that for a particular study, yi and vi are known (i.e., have
### already been calculated) but the 2x2 table counts are not known; with
### replace=FALSE, the yi and vi values for that study are not replaced
dat[1:12,10:11] <- NA
dat[13,4:7] <- NA
dat
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat, replace=FALSE)
dat

### illustrate difference between 'subset' and 'include' arguments
escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, subset=1:6)
escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, include=1:6)

### illustrate the 'var.names' argument
escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, var.names=c("lnrr", "var"))

### illustrate the 'slab' argument
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
              data=dat.bcg, slab=paste0(author, ", ", year))
dat

### note: the output looks the same but the study labels are stored as an attribute with the
### effect size estimates (together with the total sample size of the studies and the chosen
### effect size measure)
dat$yi

### this information can then be used by other functions; for example in a forest plot
forest(dat$yi, dat$vi)

#####

### convert a regular data frame to an 'escalc' object
### dataset from Lipsey & Wilson (2001), Table 7.1, page 130
dat <- data.frame(id = c(100, 308, 1596, 2479, 9021, 9028, 161, 172, 537, 7049),
```

```

      yi = c(-0.33, 0.32, 0.39, 0.31, 0.17, 0.64, -0.33, 0.15, -0.02, 0.00),
      vi = c(0.084, 0.035, 0.017, 0.034, 0.072, 0.117, 0.102, 0.093, 0.012, 0.067),
      random = c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1),
      intensity = c(7, 3, 7, 5, 7, 7, 4, 4, 5, 6))

dat
dat <- escalc(measure="SMD", yi=yi, vi=vi, data=dat, slab=paste("Study ID:", id), digits=3)
dat

#####

```

fitstats

*Fit Statistics and Information Criteria for 'rma' Objects***Description**

Functions to extract the log-likelihood, deviance, AIC, BIC, and AICc values from objects of class "rma".

Usage

```

fitstats(object, ...)

## S3 method for class 'rma'
fitstats(object, ..., REML)

## S3 method for class 'rma'
logLik(object, REML, ...)
## S3 method for class 'rma'
deviance(object, REML, ...)

## S3 method for class 'rma'
AIC(object, ..., k=2, correct=FALSE)
## S3 method for class 'rma'
BIC(object, ...)

## S3 method for class 'rma'
df.residual(object, ...)
## S3 method for class 'rma'
nobs(object, all=FALSE, ...)

```

Arguments

object	an object of class "rma".
...	optionally more fitted model objects (only for fitstats(), AIC(), and BIC()).
REML	logical to specify whether the regular or restricted likelihood function should be used to obtain the fit statistics and information criteria. Defaults to the method of estimation used (i.e., TRUE if object was fitted with method="REML" and FALSE otherwise).

<code>k</code>	numeric value to specify the penalty per parameter. The default (<code>k=2</code>) is the classical AIC. See AIC for more details.
<code>correct</code>	logical to specify whether the regular (default) or corrected (i.e., AICc) should be extracted.
<code>all</code>	logical to indicate whether various ways to count the number of observations should be returned (only for <code>nobs()</code>).

Value

For `fitstats`, a data frame with the (restricted) log-likelihood, deviance, AIC, BIC, and AICc values for each model passed to the function.

For `logLik`, an object of class "logLik", providing the (restricted) log-likelihood of the model evaluated at the estimated coefficient(s).

For `deviance`, a numeric value with the corresponding deviance.

For AIC and BIC, either a numeric value with the corresponding AIC, AICc, or BIC or a data frame with rows corresponding to the models and columns representing the number of parameters in the model (`df`) and the AIC, AICc, or BIC.

For `df.residual`, the residual degrees-of-freedom.

For `nobs`, the number of observations (effect size estimates) or a vector based on various ways to count the number of observations (when `all=TRUE`).

Note

Variance components in the model (e.g., τ^2 in random/mixed-effects models fitted with [rma.uni](#)) are counted as additional parameters in the calculation of the AIC, BIC, and AICc. Also, the fixed effects are counted as parameters in the calculation of the AIC, BIC, and AICc even when using REML estimation.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which fit statistics and information criteria can be extracted.

[anova](#) for a function to conduct likelihood ratio tests.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### random-effects model
res1 <- rma(yi, vi, data=dat, method="ML")

### mixed-effects model with absolute latitude and publication year as moderators
res2 <- rma(yi, vi, mods = ~ ablat + year, data=dat, method="ML")

### compare fit statistics
fitstats(res1, res2)

### log-likelihoods
logLik(res1)
logLik(res2)

### deviances
deviance(res1)
deviance(res2)

### AIC, AICc, and BIC values
AIC(res1, res2)
AIC(res1, res2, correct=TRUE)
BIC(res1, res2)
```

fitted.rma

Fitted Values for 'rma' Objects

Description

Function to compute the fitted values for objects of class "rma".

Usage

```
## S3 method for class 'rma'
fitted(object, ...)
```

Arguments

```
object      an object of class "rma".
...         other arguments.
```

Value

A vector with the fitted values.

Note

The `predict` function also provides standard errors and confidence intervals for the fitted values. Best linear unbiased predictions (BLUPs) that combine the fitted values based on the fixed effects and the estimated contributions of the random effects can be obtained with `blup` (only for objects of class `"rma.uni"`).

For objects not involving moderators, the fitted values are all identical to the estimated value of the model intercept.

Author(s)

Wolfgang Viechtbauer (<wvbm@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`predict` for a function to compute predicted values and `blup` for a function to compute BLUPs that combine the fitted values and predicted random effects.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### compute the fitted values
fitted(res)
```

forest

Forest Plots

Description

Function to create forest plots.

Usage

```
forest(x, ...)
```

Arguments

`x` either an object of class `"rma"`, a vector with estimates, or an object of class `"cumul.rma"`. See ‘Details’.

`...` other arguments.

Details

Currently, methods exist for three types of situations.

In the first case, object `x` is a fitted model object coming from the `rma.uni`, `rma.mh`, or `rma.peto` functions. The corresponding method is then `forest.rma`.

Alternatively, object `x` can be a vector with estimates. The corresponding method is then `forest.default`.

Finally, object `x` can be an object coming from the `cumul` function. The corresponding method is then `forest.cumul.rma`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Lewis, S., & Clarke, M. (2001). Forest plots: Trying to see the wood and the trees. *British Medical Journal*, **322**(7300), 1479–1480. <https://doi.org/10.1136/bmj.322.7300.1479>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`forest.rma`, `forest.default`, and `forest.cumul.rma` for the specific method functions.

forest.cumul.rma	<i>Forest Plots (Method for 'cumul.rma' Objects)</i>
------------------	--

Description

Function to create forest plots for objects of class "cumul.rma".

Usage

```
## S3 method for class 'cumul.rma'
forest(x, annotate=TRUE, header=TRUE,
       xlim, alim, olim, ylim, at, steps=5,
       refline=0, digits=2L, width,
       xlab, ilab, ilab.lab, ilab.xpos, ilab.pos,
       transf, atransf, targs, rows,
       efac=1, pch, psize, col, colci, shade, colshade,
       lty, fonts, cex, cex.lab, cex.axis, ...)
```

Arguments

<code>x</code>	an object of class "cumul.rma" obtained with <code>cumul</code> .
<code>annotate</code>	logical to specify whether annotations should be added to the plot (the default is TRUE).
<code>header</code>	logical to specify whether column headings should be added to the plot (the default is TRUE). Can also be a character vector to specify the left and right headings (or only the left one).
<code>xlim</code>	horizontal limits of the plot region. If unspecified, the function sets the horizontal plot limits to some sensible values.
<code>alim</code>	the x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
<code>olim</code>	argument to specify observation limits. If unspecified, no limits are used.
<code>ylim</code>	the y-axis limits of the plot. If unspecified, the function sets the y-axis limits to some sensible values. Can also be a single value to set the lower bound (while the upper bound is still set automatically).
<code>at</code>	position of the x-axis tick marks and corresponding labels. If unspecified, the function sets the tick mark positions/labels to some sensible values.
<code>steps</code>	the number of tick marks for the x-axis (the default is 5). Ignored when the positions are specified via the <code>at</code> argument.
<code>refline</code>	numeric value to specify the location of the vertical 'reference' line (the default is 0). The line can be suppressed by setting this argument to NA. Can also be a vector to add multiple lines.
<code>digits</code>	integer to specify the number of decimal places to which the annotations and tick mark labels of the x-axis should be rounded (the default is 2L). Can also be a vector of two integers, the first to specify the number of decimal places for the annotations, the second for the x-axis labels. When specifying an integer (e.g., 2L), trailing zeros after the decimal mark are dropped for the x-axis labels. When specifying a numeric value (e.g., 2), trailing zeros are retained.
<code>width</code>	optional integer to manually adjust the width of the columns for the annotations (either a single integer or a vector of the same length as the number of annotation columns).
<code>xlab</code>	title for the x-axis. If unspecified, the function sets an appropriate axis title. Can also be a vector of three/two values (to also/only add labels at the end points of the x-axis limits).
<code>ilab</code>	optional vector, matrix, or data frame providing additional information about the studies that should be added to the plot.
<code>ilab.lab</code>	optional character vector with (column) labels for the variable(s) given via <code>ilab</code> .
<code>ilab.xpos</code>	optional numeric vector to specify the horizontal position(s) of the variable(s) given via <code>ilab</code> .
<code>ilab.pos</code>	integer(s) (either 1, 2, 3, or 4) to specify the alignment of the variable(s) given via <code>ilab</code> (2 means right, 4 means left aligned). If unspecified, the default is to center the values.

transf	optional argument to specify a function to transform the estimates and confidence interval bounds (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
atransf	optional argument to specify a function to transform the x-axis labels and annotations (e.g., <code>atransf=exp</code> ; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified via <code>transf</code> or <code>atransf</code> .
rows	optional vector to specify the rows (or more generally, the positions) for plotting the estimates. Can also be a single value to specify the row of the first estimate (the remaining estimates are then plotted below this starting row).
efac	vertical expansion factor for confidence interval limits and arrows. The default value of 1 should usually work fine. Can also be a vector of two numbers, the first for CI limits, the second for arrows.
pch	plotting symbol to use for the estimates. By default, a filled square is used. See points for other options. Can also be a vector of values.
psize	numeric value to specify the point sizes for the estimates (the default is 1). Can also be a vector of values.
col	optional character string to specify the color of the estimates. Can also be a vector.
colci	optional character string to specify the color of the CI lines. Can also be a vector.
shade	optional character string or a (logical or numeric) vector for shading rows of the plot.
colshade	optional argument to specify the color for the shading.
lty	optional argument to specify the line type for the confidence intervals. If unspecified, the function sets this to "solid" by default.
fonts	optional character string to specify the font for the study labels, annotations, and the extra information (if specified via <code>ilab</code>). If unspecified, the default font is used.
cex	optional character and symbol expansion factor. If unspecified, the function sets this to a sensible value.
cex.lab	optional expansion factor for the x-axis title. If unspecified, the function sets this to a sensible value.
cex.axis	optional expansion factor for the x-axis labels. If unspecified, the function sets this to a sensible value.
...	other arguments.

Details

The [cumul](#) function can be used to conduct a cumulative meta-analysis. Passing an object created by this function to `forest()` creates a cumulative forest plot that shows the pooled estimate with corresponding confidence interval bounds as one estimate at a time is added to the analysis.

See [forest.default](#) and [forest.rma](#) for further details on the purpose of the various arguments.

Note

The function sets some sensible values for the optional arguments, but it may be necessary to adjust these in certain circumstances.

The function actually returns some information about the chosen values invisibly. Printing this information is useful as a starting point to customize the plot.

If the number of studies is quite large, the labels, annotations, and symbols may become quite small and impossible to read. Stretching the plot window vertically may then provide a more readable figure (one should call the function again after adjusting the window size, so that the label/symbol sizes can be properly adjusted). Also, the `cex`, `cex.lab`, and `cex.axis` arguments are then useful to adjust the symbol and text sizes.

If the estimates are bounded (e.g., correlations are bounded between -1 and +1, proportions are bounded between 0 and 1), one can use the `olim` argument to enforce those limits (i.e., the pooled estimates and confidence interval bounds cannot exceed those limits then).

The `lty` argument can also be a vector of two elements, the first for specifying the line type of the individual CIs ("solid" by default), the second for the line type of the horizontal line that is automatically added to the plot ("solid" by default; set to "blank" to remove it).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Chalmers, T. C., & Lau, J. (1993). Meta-analytic stimulus for changes in clinical trials. *Statistical Methods in Medical Research*, **2**(2), 161–172. <https://doi.org/10.1177/096228029300200204>
- Lau, J., Schmid, C. H., & Chalmers, T. C. (1995). Cumulative meta-analysis of clinical trials builds evidence for exemplary medical care. *Journal of Clinical Epidemiology*, **48**(1), 45–57. [https://doi.org/10.1016/0895-4356\(94\)00106-z](https://doi.org/10.1016/0895-4356(94)00106-z)
- Lewis, S., & Clarke, M. (2001). Forest plots: Trying to see the wood and the trees. *British Medical Journal*, **322**(7300), 1479–1480. <https://doi.org/10.1136/bmj.322.7300.1479>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[forest](#) for an overview of the various forest functions.

[cumul](#) for the function to create `cumul.rma` objects.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
             data=dat.bcg, slab=paste(author, year, sep=", "))

### fit random-effects model
res <- rma(yi, vi, data=dat)
```

```

### draw cumulative forest plots
x <- cumul(res, order=year)
forest(x)
forest(x, xlim=c(-4,2.5), alim=c(-2,1), steps=7)

### meta-analysis of the (log) risk ratios using the Mantel-Haenszel method
res <- rma.mh(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg,
             slab=paste(author, year, sep=", "))

### draw cumulative forest plot
x <- cumul(res, order=year)
forest(x, xlim=c(-4,2.5), alim=c(-2,1), steps=7)

```

forest.default	<i>Forest Plots (Default Method)</i>
----------------	--------------------------------------

Description

Function to create forest plots for a given set of data.

Usage

```

## Default S3 method:
forest(x, vi, sei, ci.lb, ci.ub,
       annotate=TRUE, showweights=FALSE, header=TRUE,
       xlim, alim, olim, ylim, at, steps=5,
       level=95, refline=0, digits=2L, width,
       xlab, slab, ilab, ilab.lab, ilab.xpos, ilab.pos,
       order, subset, transf, atransf, targs, rows,
       efac=1, pch, psize, plim=c(0.5,1.5), col, colci,
       shade, colshade, lty, fonts, cex, cex.lab, cex.axis, ...)

```

Arguments

<code>x</code>	vector of length k with the estimates.
<code>vi</code>	vector of length k with the corresponding sampling variances.
<code>sei</code>	vector of length k with the corresponding standard errors (note: only one of the two, <code>vi</code> or <code>sei</code> , needs to be specified).
<code>ci.lb</code>	vector of length k with the corresponding lower confidence interval bounds. Not needed if <code>vi</code> or <code>sei</code> is specified. See ‘Details’.
<code>ci.ub</code>	vector of length k with the corresponding upper confidence interval bounds. Not needed if <code>vi</code> or <code>sei</code> is specified. See ‘Details’.
<code>annotate</code>	logical to specify whether annotations should be added to the plot (the default is TRUE).
<code>showweights</code>	logical to specify whether the annotations should also include the inverse variance weights (the default is FALSE).

header	logical to specify whether column headings should be added to the plot (the default is TRUE). Can also be a character vector to specify the left and right headings (or only the left one).
xlim	horizontal limits of the plot region. If unspecified, the function sets the horizontal plot limits to some sensible values.
alim	the x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
olim	argument to specify observation limits. If unspecified, no limits are used.
ylim	the y-axis limits of the plot. If unspecified, the function sets the y-axis limits to some sensible values. Can also be a single value to set the lower bound (while the upper bound is still set automatically).
at	position of the x-axis tick marks and corresponding labels. If unspecified, the function sets the tick mark positions/labels to some sensible values.
steps	the number of tick marks for the x-axis (the default is 5). Ignored when the positions are specified via the at argument.
level	numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see here for details).
refline	numeric value to specify the location of the vertical 'reference' line (the default is 0). The line can be suppressed by setting this argument to NA. Can also be a vector to add multiple lines.
digits	integer to specify the number of decimal places to which the annotations and tick mark labels of the x-axis should be rounded (the default is 2L). Can also be a vector of two integers, the first to specify the number of decimal places for the annotations, the second for the x-axis labels (when showweights=TRUE, can also specify a third value for the weights). When specifying an integer (e.g., 2L), trailing zeros after the decimal mark are dropped for the x-axis labels. When specifying a numeric value (e.g., 2), trailing zeros are retained.
width	optional integer to manually adjust the width of the columns for the annotations (either a single integer or a vector of the same length as the number of annotation columns).
xlab	title for the x-axis. If unspecified, the function sets an appropriate axis title. Can also be a vector of three/two values (to also/only add labels at the end points of the x-axis limits).
slab	optional vector with labels for the k studies. If unspecified, the function tries to extract study labels from x and otherwise simple labels are created within the function. To suppress labels, set this argument to NA.
ilab	optional vector, matrix, or data frame providing additional information about the studies that should be added to the plot.
ilab.lab	optional character vector with (column) labels for the variable(s) given via ilab.
ilab.xpos	optional numeric vector to specify the horizontal position(s) of the variable(s) given via ilab.
ilab.pos	integer(s) (either 1, 2, 3, or 4) to specify the alignment of the variable(s) given via ilab (2 means right, 4 means left aligned). If unspecified, the default is to center the values.

order	optional character string to specify how the studies should be ordered. Can also be a variable based on which the studies will be ordered. See ‘Details’.
subset	optional (logical or numeric) vector to specify the subset of studies that should be included in the plot.
transf	optional argument to specify a function to transform the estimates and corresponding confidence interval bounds (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
atransf	optional argument to specify a function to transform the x-axis labels and annotations (e.g., <code>atransf=exp</code> ; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified via <code>transf</code> or <code>atransf</code> .
rows	optional vector to specify the rows (or more generally, the positions) for plotting the estimates. Can also be a single value to specify the row of the first estimate (the remaining estimates are then plotted below this starting row).
efac	vertical expansion factor for confidence interval limits and arrows. The default value of 1 should usually work fine. Can also be a vector of two numbers, the first for CI limits, the second for arrows.
pch	plotting symbol to use for the estimates. By default, a filled square is used. See points for other options. Can also be a vector of values.
psize	optional numeric value to specify the point sizes for the estimates. If unspecified, the point sizes are a function of the precision of the estimates. Can also be a vector of values.
plim	numeric vector of length 2 to scale the point sizes (ignored when <code>psize</code> is specified). See ‘Details’.
col	optional character string to specify the color of the estimates. Can also be a vector.
colci	optional character string to specify the color of the CI lines. Can also be a vector.
shade	optional character string or a (logical or numeric) vector for shading rows of the plot. See ‘Details’.
colshade	optional argument to specify the color for the shading.
lty	optional argument to specify the line type for the confidence intervals. If unspecified, the function sets this to "solid" by default.
fonts	optional character string to specify the font for the study labels, annotations, and the extra information (if specified via <code>ilab</code>). If unspecified, the default font is used.
cex	optional character and symbol expansion factor. If unspecified, the function sets this to a sensible value.
cex.lab	optional expansion factor for the x-axis title. If unspecified, the function sets this to a sensible value.
cex.axis	optional expansion factor for the x-axis labels. If unspecified, the function sets this to a sensible value.
...	other arguments.

Details

The plot shows the effect size estimates (by default as filled squares) with corresponding 1level% confidence intervals (as horizontal lines extending from the estimates). To use the function, one should specify the estimates (via the `x` argument) together with the corresponding sampling variances (via the `vi` argument) or with the corresponding standard errors (via the `sei` argument). The confidence intervals are computed with $y_i \pm z_{crit} \sqrt{v_i}$, where y_i denotes the estimate in the i th study, v_i the corresponding sampling variance (and hence $\sqrt{v_i}$ is the corresponding standard error), and z_{crit} is the appropriate critical value from a standard normal distribution (e.g., 1.96 for a 95% CI). Alternatively, one can directly specify the confidence interval bounds via the `ci.lb` and `ci.ub` arguments.

Applying a Transformation:

With the `transf` argument, the estimates and corresponding confidence interval bounds can be transformed with some suitable function. For example, when plotting log odds ratios, one could use `transf=exp` to obtain a forest plot showing the odds ratios. Note that when the transformation is non-linear (as is the case for `transf=exp`), the interval bounds will be asymmetric (which is visually not so appealing). Alternatively, one can use the `atransf` argument to transform the x-axis labels and annotations. For example, when using `atransf=exp`, the x-axis will correspond to a log scale. See [transf](#) for some other useful transformation functions in the context of a meta-analysis. See below for examples.

Ordering of Studies:

By default, the studies are ordered from top to bottom (i.e., the first study in the dataset will be placed in row k , the second study in row $k - 1$, and so on, until the last study, which is placed in the first row). The studies can be reordered with the `order` argument:

- `order="est"`: the studies are ordered by the estimates,
- `order="prec"`: the studies are ordered by their sampling variances.

Alternatively, it is also possible to set `order` equal to a variable based on which the studies will be ordered (see ‘Examples’). One can also use the `rows` argument to specify the rows (or more generally, the positions) for plotting the estimates.

Adding Additional Information to the Plot:

Additional columns with information about the studies can be added to the plot via the `ilab` argument. This can either be a single variable or an entire matrix / data frame (with as many rows as there are studies in the forest plot). The `ilab.xpos` argument can be used to specify the horizontal position of the variables specified via `ilab`. The `ilab.pos` argument can be used to specify how the variables should be aligned. The `ilab.lab` argument can be used to add headers to the columns.

Pooled estimates can be added to the plot as polygons with the [addpoly](#) function. See the documentation for that function for examples.

Adjusting the Point Sizes:

By default (i.e., when `psize` is not specified), the point sizes are a function of the precision (i.e., inverse standard errors) of the estimates. This way, more precise estimates are visually more prominent in the plot. By making the point sizes a function of the inverse standard errors of the estimates, their areas are proportional to the inverse sampling variances, which corresponds to the weights they would receive in an equal-effects model. However, the point sizes are rescaled so

that the smallest point size is `plim[1]` and the largest point size is `plim[2]`. As a result, their relative sizes (i.e., areas) no longer exactly correspond to their relative weights in such a model. If exactly relative point sizes are desired, one can set `plim[2]` to NA, in which case the points are rescaled so that the smallest point size corresponds to `plim[1]` and all other points are scaled accordingly. As a result, the largest point may be very large. Alternatively, one can set `plim[1]` to NA, in which case the points are rescaled so that the largest point size corresponds to `plim[2]` and all other points are scaled accordingly. As a result, the smallest point may be very small and essentially indistinguishable from the confidence interval line. To avoid the latter, one can also set `plim[3]`, which enforces a minimal point size.

Shading Rows:

With the `shade` argument, one can shade rows of the plot. The argument can be set to one of the following character strings: `"zebra"` (same as `shade=TRUE`) or `"zebra2"` to use zebra-style shading (starting either at the first or second study) or to `"all"` in which case all rows are shaded. Alternatively, the argument can be set to a logical or numeric vector to specify which rows should be shaded. The `colshade` argument can be used to set the color of shaded rows.

Note

The function sets some sensible values for the optional arguments, but it may be necessary to adjust these in certain circumstances.

The function actually returns some information about the chosen values invisibly. Printing this information is useful as a starting point to customize the plot.

If the number of studies is quite large, the labels, annotations, and symbols may become quite small and impossible to read. Stretching the plot window vertically may then provide a more readable figure (one should call the function again after adjusting the window size, so that the label/symbol sizes can be properly adjusted). Also, the `cex`, `cex.lab`, and `cex.axis` arguments are then useful to adjust the symbol and text sizes.

If the estimates are bounded (e.g., correlations are bounded between -1 and +1, proportions are bounded between 0 and 1), one can use the `olim` argument to enforce those limits (i.e., the estimates and confidence interval bounds cannot exceed those limits then).

The `lty` argument can also be a vector of two elements, the first for specifying the line type of the individual CIs (`"solid"` by default), the second for the line type of the horizontal line that is automatically added to the plot (`"solid"` by default; set to `"blank"` to remove it).

Additional Optional Arguments

There are some additional optional arguments that can be passed to the function via `...` (hence, they cannot be abbreviated):

top single numeric value to specify the amount of space (in terms of number of rows) to leave empty at the top of the plot (e.g., for adding headers). The default is 3.

annosym vector of length 3 to select the left bracket, separation, and right bracket symbols for the annotations. The default is `c("[", " ", " ")`. Can also include a 4th element to adjust the look of the minus symbol, for example to use a proper minus sign (`—`) instead of a hyphen-minus (`-`). Can also include a 5th element that should be a space-like symbol (e.g., an `'en space'`) that is used in place of numbers (only relevant when trying to line up numbers exactly). For example, `annosym=c("[", " ", " ", "\u2212", "\u2002")` would use

a proper minus sign and an ‘en space’ for the annotations. The decimal point character can be adjusted via the `OutDec` argument of the `options` function before creating the plot (e.g., `options(OutDec=",")`).

tabfig single numeric value (either a 1, 2, or 3) to set `annosym` automatically to a vector that will exactly align the numbers in the annotations when using a font that provides ‘tabular figures’. Value 1 corresponds to using “\u2212” (a minus) and “\u2002” (an ‘en space’) in `annosym` as shown above. Value 2 corresponds to “\u2013” (an ‘en dash’) and “\u2002” (an ‘en space’). Value 3 corresponds to “\u2212” (a minus) and “\u2007” (a ‘figure space’). The appropriate value for this argument depends on the font used. For example, for fonts Calibri and Carlito, 1 or 2 should work; for fonts Source Sans 3 and Palatino Linotype, 1, 2, and 3 should all work; for Computer/Latin Modern and Segoe UI, 2 should work; for Lato, Roboto, and Open Sans (and maybe Arial), 3 should work. Other fonts may work as well, but this is untested.

textpos numeric vector of length 2 to specify the placement of the study labels and the annotations. The default is to use the horizontal limits of the plot region, i.e., the study labels to the right of `xlim[1]` and the annotations to the left of `xlim[2]`.

rowadj numeric vector of length 3 to vertically adjust the position of the study labels, the annotations, and the extra information (if specified via `ilab`). This is useful for fine-tuning the position of text added with different positional alignments (i.e., argument `pos` in the `text` function).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Lewis, S., & Clarke, M. (2001). Forest plots: Trying to see the wood and the trees. *British Medical Journal*, **322**(7300), 1479–1480. <https://doi.org/10.1136/bmj.322.7300.1479>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`forest` for an overview of the various forest functions and especially `forest.rma` for a function to draw forest plots including a pooled estimate polygon.

`addpoly` for a function to add polygons to forest plots.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
             data=dat.bcg, slab=paste(author, year, sep=", "))

### default forest plot of the observed log risk ratios
forest(dat$yi, dat$vi)

### directly specify the CI bounds
out <- summary(dat)
forest(dat$yi, ci.lb=out$ci.lb, ci.ub=out$ci.ub)
```

```

### the with() function can be used to avoid having to retype dat$... over and over
with(dat, forest(yi, vi))

### forest plot of the observed risk ratios (transform the estimates)
with(dat, forest(yi, vi, transf=exp, alim=c(0,2), steps=5,
                xlim=c(-2.5,4), reline=1))

### forest plot of the observed risk ratios (transform the x-axis)
with(dat, forest(yi, vi, atranf=exp, at=log(c(0.05,0.25,1,4,20)),
                xlim=c(-10,8)))

### make all points the same size
with(dat, forest(yi, vi, atranf=exp, at=log(c(0.05,0.25,1,4,20)),
                xlim=c(-10,8), psize=1))

### and remove the vertical lines at the end of the CI bounds
with(dat, forest(yi, vi, atranf=exp, at=log(c(0.05,0.25,1,4,20)),
                xlim=c(-10,8), psize=1, efac=0))

### forest plot of the observed risk ratios with studies ordered by the RRs
with(dat, forest(yi, vi, atranf=exp, at=log(c(0.05,0.25,1,4,20)),
                xlim=c(-10,8), order="obs"))

### forest plot of the observed risk ratios with studies ordered by absolute latitude
with(dat, forest(yi, vi, atranf=exp, at=log(c(0.05,0.25,1,4,20)),
                xlim=c(-10,8), order=ablat))

### see also examples for the forest.rma function

```

forest.rma

Forest Plots (Method for 'rma' Objects)

Description

Function to create forest plots for objects of class "rma".

Usage

```

## S3 method for class 'rma'
forest(x, annotate=TRUE, addfit=TRUE,
       addpred=FALSE, predstyle="line", preddist,
       showweights=FALSE, header=TRUE,
       xlim, alim, olim, ylim, predlim, at, steps=5,
       level=x$level, reline=0, digits=2L, width,
       xlab, slab, mlab, ilab, ilab.lab, ilab.xpos, ilab.pos,
       order, transf, atranf, targ, rows,
       efac=1, pch, psize, plim=c(0.5,1.5), colout, colci, col, border,
       shade, colshade, lty, fonts, cex, cex.lab, cex.axis, ...)

```

Arguments

<code>x</code>	an object of class "rma".
<code>annotate</code>	logical to specify whether annotations should be added to the plot (the default is TRUE).
<code>addfit</code>	logical to specify whether the pooled estimate (for models without moderators) or fitted values (for models with moderators) should be added to the plot (the default is TRUE). See 'Details'.
<code>addpred</code>	logical to specify whether the prediction interval should be added to the plot (the default is FALSE). See 'Details'.
<code>predstyle</code>	character string to specify the style of the prediction interval (either "line" (the default), "polygon", "bar", "shade", or "dist"). Can be abbreviated. Specifying this argument automatically flips <code>addpred=FALSE</code> to <code>addpred=TRUE</code> .
<code>preddist</code>	optional list of two elements to manually specify the predictive distribution.
<code>showweights</code>	logical to specify whether the annotations should also include the weights given to the estimates during the model fitting (the default is FALSE). See 'Details'.
<code>header</code>	logical to specify whether column headings should be added to the plot (the default is TRUE). Can also be a character vector to specify the left and right headings (or only the left one).
<code>xlim</code>	horizontal limits of the plot region. If unspecified, the function sets the horizontal plot limits to some sensible values.
<code>alim</code>	the x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
<code>olim</code>	optional argument to specify observation limits. If unspecified, no limits are used.
<code>ylim</code>	the y-axis limits of the plot. If unspecified, the function sets the y-axis limits to some sensible values. Can also be a single value to set the lower bound (while the upper bound is still set automatically).
<code>predlim</code>	optional argument to specify the limits of the predictive distribution when <code>predstyle="dist"</code> .
<code>at</code>	position of the x-axis tick marks and corresponding labels. If unspecified, the function sets the tick mark positions/labels to some sensible values.
<code>steps</code>	the number of tick marks for the x-axis (the default is 5). Ignored when the positions are specified via the <code>at</code> argument.
<code>level</code>	numeric value between 0 and 100 to specify the confidence (and prediction) interval level (see here for details). The default is to take the value from the object.
<code>refline</code>	numeric value to specify the location of the vertical 'reference' line (the default is 0). The line can be suppressed by setting this argument to NA. Can also be a vector to add multiple lines.
<code>digits</code>	integer to specify the number of decimal places to which the annotations and tick mark labels of the x-axis should be rounded (the default is 2L). Can also be a vector of two integers, the first to specify the number of decimal places for the annotations, the second for the x-axis labels (when <code>showweights=TRUE</code> , can also specify a third value for the weights). When specifying an integer (e.g.,

	2L), trailing zeros after the decimal mark are dropped for the x-axis labels. When specifying a numeric value (e.g., 2), trailing zeros are retained.
width	optional integer to manually adjust the width of the columns for the annotations (either a single integer or a vector of the same length as the number of annotation columns).
xlab	title for the x-axis. If unspecified, the function sets an appropriate axis title. Can also be a vector of three/two values (to also/only add labels at the end points of the x-axis limits).
slab	optional vector with labels for the k studies. If unspecified, the function tries to extract study labels from x or simple labels are created within the function. To suppress labels, set this argument to NA.
mlab	optional character string giving a label to the pooled estimate. If unspecified, the function sets a default label.
ilab	optional vector, matrix, or data frame providing additional information about the studies that should be added to the plot.
ilab.lab	optional character vector with (column) labels for the variable(s) given via ilab.
ilab.xpos	optional numeric vector to specify the horizontal position(s) of the variable(s) given via ilab.
ilab.pos	integer(s) (either 1, 2, 3, or 4) to specify the alignment of the variable(s) given via ilab (2 means right, 4 means left aligned). If unspecified, the default is to center the values.
order	optional character string to specify how the studies should be ordered. Can also be a variable based on which the studies will be ordered. See ‘Details’.
transf	optional argument to specify a function to transform the estimates and confidence/prediction interval bounds (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
atransf	optional argument to specify a function to transform the x-axis labels and annotations (e.g., <code>atransf=exp</code> ; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified via <code>transf</code> or <code>atransf</code> .
rows	optional vector to specify the rows (or more generally, the positions) for plotting the estimates. Can also be a single value to specify the row of the first estimate (the remaining estimates are then plotted below this starting row).
efac	vertical expansion factor for confidence interval limits, arrows, and the polygon. The default value of 1 should usually work fine. Can also be a vector of two numbers, the first for CI limits and arrows, the second for the polygon. Can also be a vector of three numbers, the first for CI limits, the second for arrows, the third for the polygon. Can also include a fourth element to adjust the height of the prediction interval/distribution when <code>predstyle</code> is not "line".
pch	plotting symbol to use for the estimates. By default, a filled square is used. See points for other options. Can also be a vector of values.
psize	optional numeric value to specify the point sizes for the estimates. If unspecified, the point sizes are a function of the model weights. Can also be a vector of values.

<code>plim</code>	numeric vector of length 2 to scale the point sizes (ignored when <code>psize</code> is specified). See ‘Details’.
<code>colout</code>	optional character string to specify the color of the estimates. Can also be a vector.
<code>colci</code>	optional character string to specify the color of the CI lines. Can also be a vector.
<code>col</code>	optional character string to specify the color of the polygon.
<code>border</code>	optional character string to specify the border color of the polygon.
<code>shade</code>	optional character string or a (logical or numeric) vector for shading rows of the plot. See ‘Details’.
<code>colshade</code>	optional argument to specify the color for the shading.
<code>lty</code>	optional argument to specify the line type for the confidence intervals. If unspecified, the function sets this to "solid" by default.
<code>fonts</code>	optional character string to specify the font for the study labels, annotations, and the extra information (if specified via <code>ilab</code>). If unspecified, the default font is used.
<code>cex</code>	optional character and symbol expansion factor. If unspecified, the function sets this to a sensible value.
<code>cex.lab</code>	optional expansion factor for the x-axis title. If unspecified, the function sets this to a sensible value.
<code>cex.axis</code>	optional expansion factor for the x-axis labels. If unspecified, the function sets this to a sensible value.
<code>...</code>	other arguments.

Details

The plot shows the effect size estimates (by default as filled squares) with corresponding `level%` confidence intervals (as horizontal lines extending from the observed values). The confidence intervals are computed with $y_i \pm z_{crit} \sqrt{v_i}$, where y_i denotes the estimate in the i th study, v_i the corresponding sampling variance (and hence $\sqrt{v_i}$ is the corresponding standard error), and z_{crit} is the appropriate critical value from a standard normal distribution (e.g., 1.96 for a 95% CI).

Equal- and Random-Effects Models:

For an equal- and a random-effects model (i.e., for models without moderators), a four-sided polygon, sometimes called a summary ‘diamond’, is added to the bottom of the forest plot, showing the pooled estimate based on the model (with the center of the polygon corresponding to the estimate and the left/right edges indicating the confidence interval limits). The `col` and `border` arguments can be used to adjust the (border) color of the polygon. Drawing of the polygon can be suppressed by setting `addfit=FALSE`.

Prediction Interval for Random-Effects Models:

For random-effects models and if `addpred=TRUE`, a dotted line is added to the polygon which indicates the bounds of the prediction interval (Riley et al., 2011). For random-effects models of class `"rma.mv"` (see [rma.mv](#)) with multiple τ^2 values, the `addpred` argument can be used to specify for which level of the inner factor the prediction interval should be provided (since the intervals differ depending on the τ^2 value). If the model also contains multiple γ^2 values, the

addpred argument should then be of length 2 to specify the levels of both inner factors. See also [predict](#), which is used to compute these interval bounds.

Instead of showing the prediction interval as a dotted line (which corresponds to `predstyle="line"`), one can choose a different style via the `predstyle` argument:

- `predstyle="polygon"`: the prediction interval is shown as an additional polygon below the polygon for the pooled estimate,
- `predstyle="bar"`: the prediction interval is shown as a bar below the polygon for the pooled estimate,
- `predstyle="shade"`: the bar is shaded in color intensity in accordance with the density of the predictive distribution,
- `predstyle="dist"`: the entire predictive distribution is shown and the regions beyond the prediction interval bounds are shaded in gray; the region below or above zero (depending on whether the pooled estimate is positive or negative) is also shaded in a lighter shade of gray.

In all of these cases, the prediction interval bounds are then also provided as part of the annotations. For `predstyle="dist"`, one can adjust the range of values for which the predictive distribution is shown via the `predlim` argument. Note that the shaded regions may not be visible depending on the location/shape of the distribution.

Internally, [predict](#) is used to obtain the prediction interval / predictive distribution. However, one can also specify the predictive distribution manually via argument `preddist` (this can be useful if the distribution was estimated via some other method). The list should contain two elements, the first containing the x-values and the second the corresponding densities. The examples below illustrate the use of these arguments.

When using `preddist`, the bounds of the prediction interval are by default obtained numerically by constructing the corresponding empirical cumulative distribution function (the range of x-values at which the densities are given should therefore be wide enough to span the entire distribution, so that tail areas can be accurately determined). However, if `preddist` contains elements `pi.lb` and `pi.ub` (and optionally element `level` for the prediction interval level), then these are taken as the prediction interval bounds.

Meta-Regression Models:

For meta-regression models (i.e., models involving moderators), the fitted value for each study is added as a polygon to the plot. By default, the width of the polygons corresponds to the confidence interval limits for the fitted values. By setting `addpred=TRUE`, the width reflects the prediction interval limits. Again, the `col` and `border` arguments can be used to adjust the (border) color of the polygons. These polygons can be suppressed by setting `addfit=FALSE`.

Applying a Transformation:

With the `transf` argument, the estimates and confidence/prediction interval bounds can be transformed with some suitable function. For example, when plotting log odds ratios, one could use `transf=exp` to obtain a forest plot showing the odds ratios. Note that when the transformation is non-linear (as is the case for `transf=exp`), the interval bounds will be asymmetric (which is visually not so appealing). Alternatively, one can use the `atransf` argument to transform the x-axis labels and annotations. For example, when using `atransf=exp`, the x-axis will correspond to a log scale. See [transf](#) for some other useful transformation functions in the context of a meta-analysis. See below for examples.

Ordering of Studies:

By default, the studies are ordered from top to bottom (i.e., the first study in the dataset will be placed in row k , the second study in row $k - 1$, and so on, until the last study, which is placed in the first row). The studies can be reordered with the `order` argument:

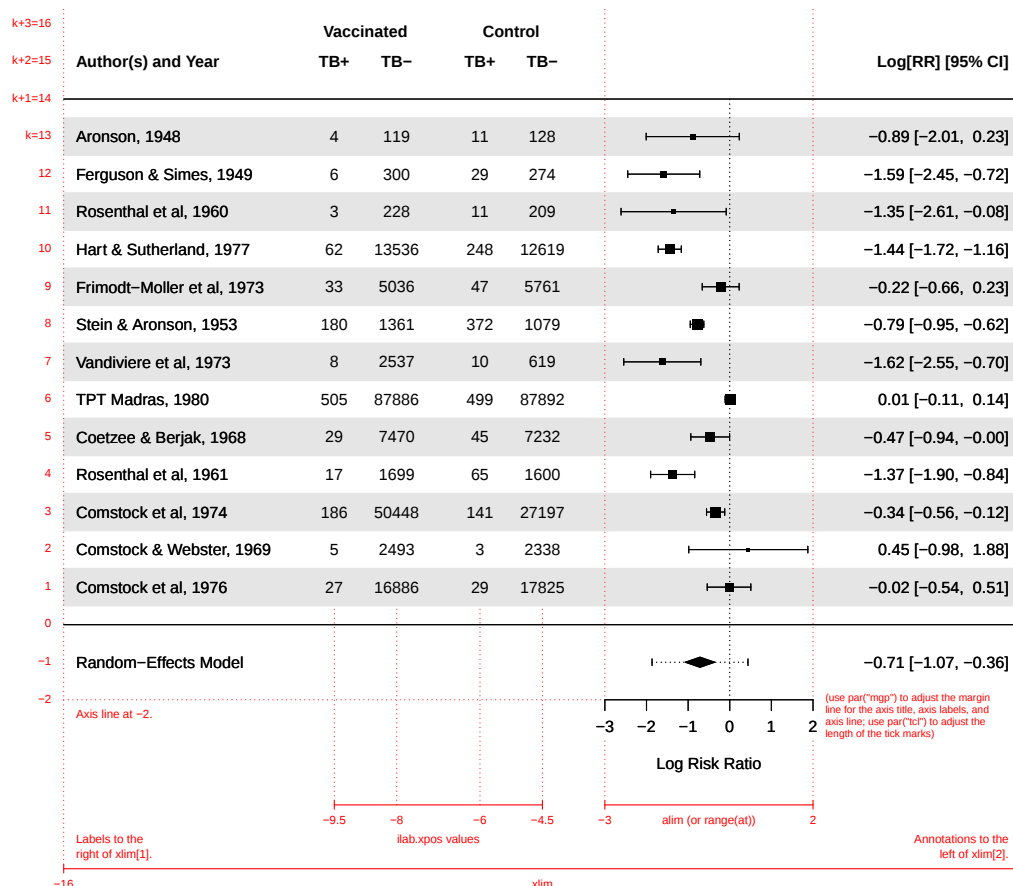
- `order="est"`: the studies are ordered by the estimates,
- `order="fit"`: the studies are ordered by the fitted values,
- `order="prec"`: the studies are ordered by their sampling variances,
- `order="resid"`: the studies are ordered by the size of their residuals,
- `order="rstandard"`: the studies are ordered by the size of their standardized residuals,
- `order="abs.resid"`: the studies are ordered by the size of their absolute residuals,
- `order="abs.rstandard"`: the studies are ordered by the size of their absolute standardized residuals.

Alternatively, it is also possible to set `order` equal to a variable based on which the studies will be ordered. One can also use the `rows` argument to specify the rows (or more generally, the positions) for plotting the estimates.

Adding Additional Information to the Plot:

Additional columns with information about the studies can be added to the plot via the `ilab` argument. This can either be a single variable or an entire matrix / data frame (with as many rows as there are studies in the forest plot). The `ilab.xpos` argument can be used to specify the horizontal position of the variables specified via `ilab`. The `ilab.pos` argument can be used to specify how the variables should be aligned. The `ilab.lab` argument can be used to add headers to the columns.

The figure below illustrates how the elements in a forest plot are arranged and the meaning of the some of the arguments such as `xlim`, `alim`, `at`, `ilab`, `ilab.xpos`, and `ilab.lab`.



The figure corresponds to the following code:

```
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
              slab=paste(author, year, sep=", "), data=dat.bcg)
res <- rma(yi, vi, data=dat)
forest(res, addpred=TRUE, xlim=c(-16,7), at=seq(-3,2,by=1), shade=TRUE,
       ilab=cbind(tpos, tneg, cpos, cneg), ilab.xpos=c(-9.5, -8, -6, -4.5),
       ilab.lab=c("TB+", "TB-", "TB+", "TB-"), cex=0.75, header="Author(s) and Year")
text(c(-8.75, -5.25), res$k+2.8, c("Vaccinated", "Control"), cex=0.75, font=2)
```

Additional pooled estimates can be added to the plot as polygons with the `addpoly` function. See the documentation for that function for examples.

When `showweights=TRUE`, the annotations will include information about the weights given to the estimates during the model fitting. For simple models (such as those fitted with the `rma.uni` function), these weights correspond to the 'inverse-variance weights' (but are given in percent). For models fitted with the `rma.mv` function, the weights are based on the diagonal of the weight matrix. Note that the weighting structure is typically more complex in such models (i.e., the weight matrix is usually not just a diagonal matrix) and the weights shown therefore do not reflect this complexity. See `weights` for more details (for the special case that `x` is an intercept-only "rma.mv" model, one can also set `showweights="rowsum"` to show the 'row-sum weights').

Adjusting the Point Sizes:

By default (i.e., when `psize` is not specified), the point sizes are a function of the square root of the model weights. This way, their areas are proportional to the weights. However, the point sizes are rescaled so that the smallest point size is `plim[1]` and the largest point size is `plim[2]`. As a result, their relative sizes (i.e., areas) no longer exactly correspond to their relative weights. If exactly relative point sizes are desired, one can set `plim[2]` to NA, in which case the points are rescaled so that the smallest point size corresponds to `plim[1]` and all other points are scaled accordingly. As a result, the largest point may be very large. Alternatively, one can set `plim[1]` to NA, in which case the points are rescaled so that the largest point size corresponds to `plim[2]` and all other points are scaled accordingly. As a result, the smallest point may be very small and essentially indistinguishable from the confidence interval line. To avoid the latter, one can also set `plim[3]`, which enforces a minimal point size.

Shading Rows:

With the `shade` argument, one can shade rows of the plot. The argument can be set to one of the following character strings: "zebra" (same as `shade=TRUE`) or "zebra2" to use zebra-style shading (starting either at the first or second study) or to "all" in which case all rows are shaded. Alternatively, the argument can be set to a logical or numeric vector to specify which rows should be shaded. The `colshade` argument can be used to set the color of shaded rows.

Note

The function sets some sensible values for the optional arguments, but it may be necessary to adjust these in certain circumstances.

The function actually returns some information about the chosen values invisibly. Printing this information is useful as a starting point to customize the plot (see 'Examples').

For arguments `slab` and `ilab` and when specifying vectors for arguments `pch`, `psize`, `order`, `colout` and/or `colci` (and when `shade` is a logical vector), the variables specified are assumed to be of the same length as the data originally passed to the model fitting function (and if the data argument was used in the original model fit, then the variables will be searched for within this data frame first). Any subsetting and removal of studies with missing values is automatically applied to the variables specified via these arguments.

If the number of studies is quite large, the labels, annotations, and symbols may become quite small and impossible to read. Stretching the plot window vertically may then provide a more readable figure (one should call the function again after adjusting the window size, so that the label/symbol sizes can be properly adjusted). Also, the `cex`, `cex.lab`, and `cex.axis` arguments are then useful to adjust the symbol and text sizes.

If the estimates are bounded (e.g., correlations are bounded between -1 and +1, proportions are bounded between 0 and 1), one can use the `olim` argument to enforce those limits (i.e., the estimates and confidence/prediction interval bounds cannot exceed those limits then).

The models without moderators, the `col` argument can also be a vector of two elements, the first for the color of the polygon, the second for the color of the line for the prediction interval. For `predstyle="polygon"` and `predstyle="bar"`, `col[2]` can be used to adjust the polygon/bar color and `border[2]` the border color. For `predstyle="shade"`, `col` can be a vector of up to three elements, where `col[2]` and `col[3]` specify the colors for the center and the ends of the shading region. For `predstyle="dist"`, `col` can be a vector of up to four elements, `col[2]` for the tail regions, `col[3]` for the color above/below zero, `col[4]` for the opposite side (transparent by default), and `border[2]` for the color of the lines. Setting a color to NA makes it transparent.

The `lty` argument can also be a vector of up to three elements, the first for specifying the line type of the individual CIs ("solid" by default), the second for the line type of the prediction interval ("dotted" by default), the third for the line type of the horizontal lines that are automatically added to the plot ("solid" by default; set to "blank" to remove them).

Additional Optional Arguments

There are some additional optional arguments that can be passed to the function via `...` (hence, they cannot be abbreviated):

top single numeric value to specify the amount of space (in terms of number of rows) to leave empty at the top of the plot (e.g., for adding headers). The default is 3.

annosym vector of length 3 to select the left bracket, separation, and right bracket symbols for the annotations. The default is `c("[", " ", " ")`. Can also include a 4th element to adjust the look of the minus symbol, for example to use a proper minus sign (—) instead of a hyphen-minus (-). Can also include a 5th element that should be a space-like symbol (e.g., an ‘en space’) that is used in place of numbers (only relevant when trying to line up numbers exactly). For example, `annosym=c("[", " ", " ", "\u2212", "\u2002")` would use a proper minus sign and an ‘en space’ for the annotations. The decimal point character can be adjusted via the `OutDec` argument of the `options` function before creating the plot (e.g., `options(OutDec=",")`).

tabfig single numeric value (either a 1, 2, or 3) to set `annosym` automatically to a vector that will exactly align the numbers in the annotations when using a font that provides ‘tabular figures’. Value 1 corresponds to using `"\u2212"` (a minus) and `"\u2002"` (an ‘en space’) in `annosym` as shown above. Value 2 corresponds to `"\u2013"` (an ‘en dash’) and `"\u2002"` (an ‘en space’). Value 3 corresponds to `"\u2212"` (a minus) and `"\u2007"` (a ‘figure space’). The appropriate value for this argument depends on the font used. For example, for fonts Calibri and Carlito, 1 or 2 should work; for fonts Source Sans 3 and Palatino Linotype, 1, 2, and 3 should all work; for Computer/Latin Modern and Segoe UI, 2 should work; for Lato, Roboto, and Open Sans (and maybe Arial), 3 should work. Other fonts may work as well, but this is untested.

textpos numeric vector of length 2 to specify the placement of the study labels and the annotations. The default is to use the horizontal limits of the plot region, i.e., the study labels to the right of `xlim[1]` and the annotations to the left of `xlim[2]`.

rowadj numeric vector of length 3 to vertically adjust the position of the study labels, the annotations, and the extra information (if specified via `ilab`). This is useful for fine-tuning the position of text added with different positional alignments (i.e., argument `pos` in the `text` function).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Lewis, S., & Clarke, M. (2001). Forest plots: Trying to see the wood and the trees. *British Medical Journal*, **322**(7300), 1479–1480. <https://doi.org/10.1136/bmj.322.7300.1479>
- Riley, R. D., Higgins, J. P. T., & Deeks, J. J. (2011). Interpretation of random effects meta-analyses. *British Medical Journal*, **342**, d549. <https://doi.org/10.1136/bmj.d549>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`forest` for an overview of the various forest functions and `forest.default` for a function to draw forest plots without a polygon.

`rma.uni`, `rma.mh`, `rma.peto`, `rma.glmm`, and `rma.mv` for functions to fit models for which forest plots can be drawn.

`addpoly` for a function to add polygons to forest plots.

Examples

```
### meta-analysis of the log risk ratios using a random-effects model
res <- rma(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg,
          slab=paste(author, year, sep=" ", "))

### default forest plot of the log risk ratios and pooled estimate
forest(res)

### pooled estimate in row -1; studies in rows k=13 through 1; horizontal
### lines in rows 0 and k+1; two extra lines of space at the top for headings,
### and other annotations; headings in line k+2
op <- par(xpd=TRUE)
text(x=-8.1, y=-1:16, -1:16, pos=4, cex=0.6, col="red")
par(op)

### can also inspect defaults chosen
defaults <- forest(res)
defaults

### several forest plots illustrating the use of various arguments
forest(res)
forest(res, alim=c(-3,3))
forest(res, alim=c(-3,3), order="prec")
forest(res, alim=c(-3,3), order="obs")
forest(res, alim=c(-3,3), order="ablat")

### various ways to show the prediction interval
forest(res, addpred=TRUE)
forest(res, predstyle="line") # same
forest(res, predstyle="polygon")
forest(res, predstyle="polygon", col=c("black","white"))
forest(res, predstyle="bar")
forest(res, predstyle="shade")
forest(res, predstyle="dist")

### specify the predictive distribution via the 'preddist' argument
pred <- predict(res)
dens <- list(x=seq(-3, 3, length.out=10000))
dens$y <- dnorm(dens$x, mean=coef(res), sd=pred$pi.se)
forest(res, predstyle="dist", preddist=dens)
```

```

### adjust xlim values to see how that changes the plot
defaults <- forest(res)
defaults$xlim # this shows what xlim values were chosen by default
par("usr")[1:2] # or use par("usr") to get the same values
forest(res, xlim=c(-12,16))
forest(res, xlim=c(-18,10))
forest(res, xlim=c(-6,4))

### illustrate the transf argument (note the asymmetric CI bounds)
forest(res, transf=exp, at=log(c(0.05,0.25,1,4,20)), refline=1)

### illustrate the atranf argument (note that the CIs now look symmetric)
forest(res, atranf=exp, at=log(c(0.05,0.25,1,4,20)), xlim=c(-8,7))

### showweights argument
forest(res, atranf=exp, at=log(c(0.05,0.25,1,4,20)), xlim=c(-8,8),
       order="prec", showweights=TRUE)

### illustrate shade argument
forest(res, shade="zebra")      # string
forest(res, shade=year >= 1970) # logical vector
forest(res, shade=c(1,5,10))   # numeric vector

### forest plot with extra annotations
### note: may need to widen the plotting device to avoid overlapping text
forest(res, atranf=exp, at=log(c(0.05, 0.25, 1, 4)), xlim=c(-16,6),
       ilab=cbind(tpos, tneg, cpos, cneg), ilab.lab=c("TB+", "TB-", "TB+", "TB-"),
       ilab.xpos=c(-9.5,-8,-6,-4.5), cex=0.85, header="Author(s) and Year")
text(c(-8.75,-5.25), res$k+2.8, c("Vaccinated", "Control"), cex=0.85, font=2)

### mixed-effects model with absolute latitude as moderator
res <- rma(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, mods = ~ ablat,
          data=dat.bcg, slab=paste(author, year, sep=" "))

### forest plot with observed and fitted values
forest(res, xlim=c(-9,5), at=log(c(0.05,0.25,1,4)), order="fit",
       ilab=ablat, ilab.xpos=-4.5, ilab.lab="Latitude", atranf=exp,
       header="Author(s) and Year")

### meta-analysis of the log risk ratios using a random-effects model
res <- rma(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg,
          slab=paste(author, year, sep=" "))

### for more complicated plots, the ylim and rows arguments may be useful
forest(res)
forest(res, ylim=c(-2, 16)) # the default
forest(res, ylim=c(-2, 20)) # extra space in plot
forest(res, ylim=c(-2, 20), rows=c(17:15, 12:6, 3:1)) # set positions

### forest plot with subgrouping of studies
### note: may need to widen plotting device to avoid overlapping text
tmp <- forest(res, xlim=c(-16, 6), at=log(c(0.05, 0.25, 1, 4)), atranf=exp,

```

```

        ilab=cbind(tpos, tneg, cpos, cneg), ilab.lab=c("TB+", "TB-", "TB+", "TB-"),
        ilab.xpos=c(-9.5, -8, -6, -4.5), cex=0.85, ylim=c(-2, 21),
        order=alloc, rows=c(1:2, 5:11, 14:17),
        header="Author(s) and Year", shade=c(3, 12, 18))
op <- par(cex=tmp$cex)
text(c(-8.75, -5.25), tmp$ylim[2]-0.2, c("Vaccinated", "Control"), font=2)
text(-16, c(18, 12, 3), c("Systematic Allocation", "Random Allocation",
        "Alternate Allocation"), font=4, pos=4)
par(op)

### see also the addpoly.rma function for an example where summaries
### for the three subgroups are added to such a forest plot

### illustrate the efac argument
forest(res)
forest(res, efac=c(0, 1, 1)) # no vertical lines at the end of the CIs
forest(res, alim=c(-3, 1), efac=c(0, 1, 1)) # still shows arrows when CIs extend beyond 'alim'

### illustrate use of the olim argument with a meta-analysis of raw proportions
### (data from Pritz, 1997); without olim=c(0, 1), some of the CIs would have upper
### bounds larger than 1
dat <- escalc(measure="PR", xi=xi, ni=ni, data=dat.pritz1997)
res <- rma(yi, vi, data=dat, slab=paste0(study, " ", authors))
forest(res, xlim=c(-0.8, 1.6), alim=c(0, 1), psize=1, refline=coef(res), olim=c(0, 1))

### an example of a forest plot where the data have a multilevel structure and
### we want to reflect this by grouping together estimates from the same cluster
dat <- dat.konstantopoulos2011
res <- rma.mv(yi, vi, random = ~ 1 | district/school, data=dat,
        slab=paste0("District ", district, ", School: ", school))
dd <- c(0, diff(dat$district))
dd[dd > 0] <- 1
rows <- (1:res$k) + cumsum(dd)
op <- par(tck=-0.01, mgp = c(1.6, 0.2, 0), mar=c(3, 8, 1, 6))
forest(res, cex=0.5, rows=rows, ylim=c(-2, max(rows)+3))
abline(h = rows[c(1, diff(rows)) == 2] - 1, lty="dotted")
par(op)

### another approach where clusters are shaded in a zebra style
forest(res, cex=0.6, shade=as.numeric(factor(dat$district)) %% 2 != 0)

```

Description

Functions to format various types of outputs.

Usage

```

fmt(p, digits=4, pname="", equal=FALSE, sep=FALSE, add0=FALSE, quote=FALSE)
fmt2(p, cutoff=c(0.001,0.06), pname="p", sep=TRUE, add0=FALSE, quote=FALSE)
fmt(x, digits=4, flag="", quote=FALSE, ...)
fmtt(val, tname, df, df1, df2, pval, digits=4,
      pname="p-val", format=1, sep=TRUE, quote=FALSE, call=FALSE, ...)

```

Arguments

Arguments for fmt and fmt2:

<code>p</code>	vector of p-values to be formatted.
<code>digits</code>	integer to specify the number of decimal places to which the values should be rounded. For <code>fmtt</code> , can be a vector of length 2, to specify the number of digits for the test statistic and the p-value, respectively.
<code>pname</code>	string to add as a prefix to the p-value (e.g., something like "p-val" or "p").
<code>equal</code>	logical to specify whether an equal symbol should be shown before the p-value (when it is larger than the rounding cutoff).
<code>sep</code>	logical to specify whether a space should be added between <code>pname</code> , the equal/lesser symbol, and the p-value.
<code>add0</code>	logical to specify whether a 0 should be shown before the decimal point (for <code>fmt</code> , this only applies when the p-value is below the rounding cutoff).
<code>quote</code>	logical to specify whether formatted strings should be quoted when printed.
<code>cutoff</code>	numeric vector giving the cutoff values.

Arguments specific for fmtx:

<code>x</code>	vector of numeric values to be formatted.
<code>flag</code>	a character string giving a format modifier as defined for formatC .

Arguments specific for fmtt:

<code>val</code>	test statistic value to be formatted.
<code>tname</code>	character string for the name of the test statistic.
<code>df</code>	optional value for the degrees of freedom of the test statistic.
<code>df1</code>	optional value for the numerator degrees of freedom of the test statistic.
<code>df2</code>	optional value for the denominator degrees of freedom of the test statistic.
<code>pval</code>	the p-value corresponding to the test statistic.
<code>format</code>	either 1 or 2 to denote whether the degrees of freedom should be given before the test statistic (in parentheses) or after the test statistic.
<code>call</code>	logical to specify whether the formatted test result should be returned as a call or not.
<code>...</code>	other arguments.

Details

The `fntp` function takes one or multiple p-values as input and rounds them to the chosen number of digits. For p-values that are smaller than $10^{-(\text{digits})}$ (e.g., 0.0001 for `digits=4`), the value is shown to fall below this bound (e.g., <0.0001). One can further customize the way the output of the values is formatted via the `pname`, `equal`, `sep`, `add0`, and `quote` arguments.

The `fntp2` function is an alternative function to format p-values, which yields output that essentially matches APA style guidelines. Values that fall below the first cutoff are printed as such (e.g., a p-value of 0.00002 would be printed as $p < 0.001$), values that fall in between the first and second cutoff are printed as exact p-values with the number of digits determined by the first cutoff (e.g., a p-value of 0.01723 would be printed as $p = 0.017$), and values falling above the second cutoff are printed as exact p-values with the number of digits determined by the second cutoff (e.g., a p-value of 0.08432 would be printed as $p = 0.08$). Note that the second cutoff is by default 0.06 to show that p-values in the range of 0.051 and 0.054 are above 0.05.

The `fmtx` function takes one or multiple numeric values as input and rounds them to the chosen number of digits, without using scientific notation and without dropping trailing zeros (using `formatC`).

The `fmtt` function takes a single test statistic value as input (and, if applicable, its degrees of freedom via argument `df` or its numerator and denominator degrees of freedom via arguments `df1` and `df2`) and the corresponding p-value and formats it for printing. Two different formats are available (chosen via the `format` argument), one giving the degrees of freedom before the test statistic (in parentheses) and one after the test statistic.

Value

A character vector with the formatted values. By default (i.e., when `quote=FALSE`), formatted strings are not quoted when printed.

Note

The option in `fmtt` to return the formatted test result as a call can be useful when adding the output to a plot with `text` and one would like to use `plotmath` formatting for `tname`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Examples

```
# examples for fntp()
fntp(c(0.0002, 0.00008), quote=TRUE, digits=4)
fntp(c(0.0002, 0.00008), quote=TRUE, digits=4, equal=TRUE)
fntp(c(0.0002, 0.00008), quote=TRUE, digits=4, equal=TRUE, sep=TRUE)
fntp(c(0.0002, 0.00008), quote=TRUE, digits=4, equal=TRUE, sep=TRUE, add0=TRUE)
```

```
# example for fmp2()
fmp2(c(0.0005, 0.001, 0.002, 0.0423, 0.0543, 0.0578, 0.0623, 0.5329), quote=TRUE)

# examples for fmtx()
fmtx(c(1.0002, 2.00008, 3.00004), digits=4)
fmtx(c(-1, 1), digits=4)
fmtx(c(-1, 1), digits=4, flag=" ")

# examples for fmtt()
fmtt(2.45, "z", pval=0.01429, digits=2)
fmtt(3.45, "z", pval=0.00056, digits=2)
fmtt(2.45, "t", df=23, pval=0.02232, digits=2)
fmtt(3.45, "t", df=23, pval=0.00218, digits=2)
fmtt(3.45, "t", df=23, pval=0.00218, digits=2, format=2)
fmtt(46.23, "Q", df=29, pval=0.0226, digits=2)
fmtt(46.23, "Q", df=29, pval=0.0226, digits=2, format=2)
fmtt(8.75, "F", df1=2, df2=35, pval=0.00083, digits=c(2,3))
fmtt(8.75, "F", df1=2, df2=35, pval=0.00083, digits=c(2,3), format=2, pname="p")
fmtt(8.75, "F", df1=2, df2=35, pval=0.00083, digits=c(2,3), format=2, pname="p", sep=FALSE)
```

formula.rma

Extract the Model Formula from 'rma' Objects

Description

Function to extract the model formula from objects of class "rma".

Usage

```
## S3 method for class 'rma'
formula(x, type="mods", ...)
```

Arguments

x	an object of class "rma".
type	the formula which should be returned; either "mods" (default), "yi" (in case argument yi was used to specify a formula), or "scale" (only for location-scale models).
...	other arguments.

Value

The requested formula.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which a model formula can be extracted.

Examples

```
### copy BCG vaccine data into 'dat'
dat <- dat.bcg

### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat,
             slab=paste(author, ", ", year, sep=""))

### mixed-effects meta-regression model
res <- rma(yi, vi, mods = ~ ablat + alloc, data=dat)
formula(res, type="mods")

### specify moderators via 'yi' argument
res <- rma(yi ~ ablat + alloc, vi, data=dat)
formula(res, type="yi")
```

 fsn

Fail-Safe N Analysis (File Drawer Analysis)

Description

Function to compute the fail-safe N (also called a file drawer analysis).

Usage

```
fsn(x, vi, sei, subset, data, type, alpha=0.05, target,
    method, exact=FALSE, verbose=FALSE, digits, ...)
```

Arguments

<code>x</code>	a vector with the effect size estimates or an object of class "rma".
<code>vi</code>	vector with the corresponding sampling variances. Ignored if <code>x</code> is an object of class "rma".
<code>sei</code>	vector with the corresponding standard errors (note: only one of the two, <code>vi</code> or <code>sei</code> , needs to be specified).
<code>subset</code>	optional (logical or numeric) vector to specify the subset of studies that should be used for the calculation. Ignored if <code>x</code> is an object of class "rma".

data	optional data frame containing the variables given to the arguments above.
type	optional character string to specify the type of method to use for the calculation of the fail-safe N. Possible options are "Rosenthal" (the default when x is a vector with estimates), "Orwin", "Rosenberg", or "General" (the default when x is an object of class "rma"). Can be abbreviated. See 'Details'.
alpha	target alpha level for the Rosenthal, Rosenberg, and General methods (the default is 0.05).
target	target pooled estimate for the Orwin and General methods.
method	optional character string to specify the model fitting method for type="General" (if unspecified, either "REML" by default or the method that was used in fitting the "rma" model). See rma.uni for options.
exact	logical to specify whether the general method should be based on exact (but slower) or approximate (but faster) calculations.
verbose	logical to specify whether output should be generated on the progress of the calculations for type="General" (the default is FALSE).
digits	optional integer to specify the number of decimal places to which the printed results should be rounded.
...	other arguments.

Details

The function can be used to calculate the 'fail-safe N', that is, the minimum number of studies averaging null results that would have to be added to a given set of k studies to change the conclusion of a meta-analysis. If this number is small (in relation to the actual number of studies), then this indicates that the results based on the observed studies are not robust to publication bias (of the form assumed by the method, that is, where a set of studies averaging null results is missing). The method is also called a 'file drawer analysis' as it assumes that there is a set of studies averaging null results hiding in file drawers, which can overturn the findings from a meta-analysis. There are various types of methods that are all based on the same principle, which are described in more detail further below. Note that *the fail-safe N is not an estimate of the number of missing studies*, only how many studies must be hiding in file drawers for the findings to be overturned.

Via argument x, one can either pass a vector with the effect size estimates to the function (and then the corresponding sampling variances via vi or the standard errors via sei) or an object of class "rma". When passing a model object to argument x, the model must be a model without moderators (i.e., either an equal- or a random-effects model).

Rosenthal Method:

The Rosenthal method (type="Rosenthal") calculates the minimum number of studies averaging null results that would have to be added to a given set of studies to reduce the (one-tailed) combined significance level (i.e., p-value) to a particular alpha level, which can be specified via the alpha argument (0.05 by default). The calculation is based on Stouffer's method for combining p-values and is described in Rosenthal (1979). Note that the method is primarily of interest for historical reasons, but the other methods described below are more closely aligned with the way meta-analyses are typically conducted in practice.

Orwin Method:

The Orwin method (`type="Orwin"`) calculates the minimum number of studies averaging null results that would have to be added to a given set of studies to reduce the pooled estimate to a target value (as specified via the `target` argument). The method is described in Orwin (1983). When `vi` (or `sei`) is not specified, the method is based on the unweighted average of the estimates; otherwise, the method uses the inverse-variance weighted average. If the `target` argument is not specified, then the target value will be equal to the (unweighted) average of the estimates divided by 2 (which is entirely arbitrary and will always lead to a fail-safe N number that is equal to k). One should really set `target` to a value that reflects a pooled estimate that would be considered to be practically irrelevant. Note that if `target` has the opposite sign as the actually observed average, then its sign is automatically flipped.

Rosenberg Method:

The Rosenberg method (`type="Rosenberg"`) calculates the minimum number of studies averaging null results that would have to be added to a given set of studies to reduce the significance level (i.e., p -value) of the pooled estimate (as estimated based on an equal-effects model) to a particular alpha level, which can be specified via the `alpha` argument (0.05 by default). The method is described in Rosenberg (2005). Note that the p -value is calculated based on a standard normal distribution (instead of a t -distribution, as suggested by Rosenberg, 2005), but the difference is typically negligible.

General Method:

This method is a generalization of the methods by Orwin and Rosenberg (see Viechtbauer, 2024). By default (i.e., when `target` is not specified), it calculates the minimum number of studies averaging null results that would have to be added to a given set of studies to reduce the significance level (i.e., p -value) of the pooled estimate (as estimated based on a chosen model) to a particular alpha level, which can be specified via the `alpha` argument (0.05 by default). The type of model that is used in the calculation is chosen via the `method` argument. If this is unspecified, then a random-effects model is automatically used (using `method="REML"`) or the method that was used in fitting the `"rma"` model (see [rma.uni](#) for options). Therefore, when setting `method="EE"`, then an equal-effects model is used, which yields (essentially) identical results as Rosenberg's method. If `target` is specified, then the method calculates the minimum number of studies averaging null results that would have to be added to a given set of studies to reduce the pooled estimate (as estimated based on a chosen model) to a target value (as specified via the `target` argument). As described above, the type of model that is used in the calculation is chosen via the `method` argument. When setting `method="EE"`, then an equal-effects model is used, which yields (essentially) identical results as Orwin's method with inverse-variance weights.

The method uses an iterative algorithm for calculating the fail-safe N , which can be computationally expensive especially when N is large. By default, the method uses approximate (but faster) calculations, but when setting `exact=TRUE`, the method uses exact (but slower) calculations. The difference between the two is typically negligible. If N is larger than 10^7 , then the calculated number is given as $>1e+07$.

Value

An object of class `"fsn"`. The object is a list containing the following components (some of which may be NA if they are not applicable to the chosen method):

`type` the type of method used.

<code>fsum</code>	the calculated fail-safe N.
<code>est</code>	the pooled estimate based on the observed studies.
<code>tau2</code>	the estimated amount of heterogeneity based on the observed studies.
<code>pval</code>	the p-value of the observed results.
<code>alpha</code>	the specified target alpha level.
<code>target</code>	the target pooled estimate.
<code>est.fsn</code>	the pooled estimate when combining the observed studies with those in the file drawer.
<code>tau2</code>	the estimated amount of heterogeneity when combining the observed studies with those in the file drawer.
<code>pval</code>	the p-value when combining the observed studies with those in the file drawer.
<code>...</code>	some additional elements/values.

The results are formatted and printed with the `print` function.

Note

If the significance level of the pooled estimate is already above the specified alpha level or if the pooled estimate is already below the target value for the given set of studies, then the fail-safe N value is zero.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Rosenthal, R. (1979). The "file drawer problem" and tolerance for null results. *Psychological Bulletin*, **86**(3), 638–641. <https://doi.org/10.1037/0033-2909.86.3.638>
- Orwin, R. G. (1983). A fail-safe N for effect size in meta-analysis. *Journal of Educational Statistics*, **8**(2), 157–159. <https://doi.org/10.3102/10769986008002157>
- Rosenberg, M. S. (2005). The file-drawer problem revisited: A general weighted method for calculating fail-safe numbers in meta-analysis. *Evolution*, **59**(2), 464–468. <https://doi.org/10.1111/j.0014-3820.2005.tb0>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W. (2024). A fail-safe N computation based on the random-effects model. *Annual Meeting of the Society for Research Synthesis Methodology*, Amsterdam, The Netherlands. https://www.wvbauer.com/lib/exe/fetch.php/talks:2024_viechtbauer_srsm_fail_safe_n.pdf

See Also

`regtest` for the regression test, `ranktest` for the rank correlation test, `trimfill` for the trim and fill method, `tes` for the test of excess significance, and `selmodel` for selection models.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit equal-effects model
rma(yi, vi, data=dat, method="EE")

### fail-safe N computations
fsn(yi, vi, data=dat)
fsn(yi, data=dat, type="Orwin", target=log(0.95)) # target corresponds to a 5% risk reduction
fsn(yi, vi, data=dat, type="Orwin", target=log(0.95)) # Orwin's method with 1/vi weights
fsn(yi, vi, data=dat, type="General", target=log(0.95), method="EE") # like Orwin's method
fsn(yi, vi, data=dat, type="Rosenberg")
fsn(yi, vi, data=dat, type="General", method="EE") # like Rosenberg's method
fsn(yi, vi, data=dat, type="General") # based on a random-effects model
fsn(yi, vi, data=dat, type="General", target=log(0.95)) # based on a random-effects model

### fit a random-effects model and use fsn() on the model object
res <- rma(yi, vi, data=dat)
fsn(res)
fsn(res, target=log(0.95))
```

funnel

Funnel Plots

Description

Function to create funnel plots.

Usage

```
funnel(x, ...)
```

```
## S3 method for class 'rma'
funnel(x, yaxis="sei",
       xlim, ylim, xlab, ylab, slab,
       steps=5, at, atransf, targs, digits, level=x$level,
       addtau2=FALSE, type="rstandard",
       back, shade, hlines,
       refline, lty=3, pch, pch.fill, col, bg,
       label=FALSE, offset=0.4, legend=FALSE, ...)
```

```
## Default S3 method:
funnel(x, vi, sei, ni, subset, yaxis="sei",
       xlim, ylim, xlab, ylab, slab,
       steps=5, at, atransf, targs, digits, level=95,
       back, shade, hlines,
       refline=0, lty=3, pch, col, bg,
       label=FALSE, offset=0.4, legend=FALSE, ...)
```

Arguments

<code>x</code>	an object of class "rma" or a vector with the effect size estimates.
<code>vi</code>	vector with the corresponding sampling variances (needed if <code>x</code> is a vector with estimates).
<code>sei</code>	vector with the corresponding standard errors (note: only one of the two, <code>vi</code> or <code>sei</code> , needs to be specified).
<code>ni</code>	vector with the corresponding sample sizes. Only relevant when passing a vector via <code>x</code> .
<code>subset</code>	optional (logical or numeric) vector to specify the subset of studies that should be included in the plot. Only relevant when passing a vector via <code>x</code> .
<code>yaxis</code>	either "sei", "vi", "seinv", "vinv", "ni", "ninv", "sqrtni", "sqrtninv", "lni", or "wi" to specify what values should be placed on the y-axis. See 'Details'.
<code>xlim</code>	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
<code>ylim</code>	y-axis limits. If unspecified, the function sets the y-axis limits to some sensible values.
<code>xlab</code>	title for the x-axis. If unspecified, the function sets an appropriate axis title.
<code>ylab</code>	title for the y-axis. If unspecified, the function sets an appropriate axis title.
<code>slab</code>	optional vector with labels for the k studies. If unspecified, the function tries to extract study labels from <code>x</code> .
<code>steps</code>	the number of tick marks for the y-axis (the default is 5).
<code>at</code>	position of the x-axis tick marks and corresponding labels. If unspecified, the function sets the tick mark positions/labels to some sensible values.
<code>atransf</code>	optional argument to specify a function to transform the x-axis labels (e.g., <code>atransf=exp</code> ; see also transf). If unspecified, no transformation is used.
<code>targs</code>	optional arguments needed by the function specified via <code>atransf</code> .
<code>digits</code>	optional integer to specify the number of decimal places to which the tick mark labels of the x- and y-axis should be rounded. Can also be a vector of two integers, the first to specify the number of decimal places for the x-axis, the second for the y-axis labels (e.g., <code>digits=c(2,3)</code>). If unspecified, the function tries to set the argument to some sensible values.
<code>level</code>	numeric value between 0 and 100 to specify the level of the pseudo confidence interval region (see here for details). For "rma" objects, the default is to take the value from the object. May also be a vector of values to obtain multiple regions (for contour-enhanced funnel plots). See 'Examples'.
<code>addtau2</code>	logical to specify whether the amount of heterogeneity should be accounted for when drawing the pseudo confidence interval region (the default is FALSE). Ignored when <code>x</code> is a meta-regression model and residuals are plotted. See 'Details'.
<code>type</code>	either "rstandard" (default) or "rstudent" to specify whether the usual or deleted residuals should be used in creating the funnel plot when <code>x</code> is a meta-regression model. See 'Details'.

back	optional character string to specify the color of the plotting region background.
shade	optional character string to specify the color of the pseudo confidence interval region. When level is a vector of values, different shading colors can be specified for each region.
hlines	optional character string to specify the color of the horizontal reference lines.
refline	numeric value to specify the location of the vertical ‘reference’ line and where the pseudo confidence interval should be centered. If unspecified, the reference line is drawn at the pooled estimate from an equal- or random-effects model and at zero for meta-regression models (in which case the residuals are plotted) or when directly passing a set of estimates to the function.
lty	line type for the pseudo confidence interval region and the reference line. The default is to draw dotted lines (see par for other options). Can also be a vector to specify the two line types separately.
pch	plotting symbol to use for the estimates. By default, a filled circle is used. Can also be a vector of values. See points for other options.
pch.fill	plotting symbol to use for estimates filled in by the trim and fill method. By default, an open circle is used. Only relevant when plotting an object created by the trimfill function.
col	optional character string to specify the (border) color of the points. Can also be a vector.
bg	optional character string to specify the background color of open plot symbols. Can also be a vector.
label	argument to control the labeling of the points (the default is FALSE). See ‘Details’.
offset	argument to control the distance between the points and the corresponding labels.
legend	logical to specify whether a legend should be added to the plot (the default is FALSE). See ‘Details’.
...	other arguments.

Details

For equal- and random-effects models (i.e., models not involving moderators), the plot shows the effect size estimates on the x-axis against the corresponding standard errors (i.e., the square root of the sampling variances) on the y-axis. A vertical line indicates the pooled estimate based on the model (or at the value specified via the `refline` argument). A pseudo confidence interval region is drawn around this value with bounds equal to $\pm 1.96SE$ (assuming `level=95`), where `SE` is the standard error value from the y-axis. If `addtau2=TRUE` (only for models of class `"rma.uni"`), then the bounds of the pseudo confidence interval region are equal to $\pm 1.96\sqrt{SE^2 + \hat{\tau}^2}$, where $\hat{\tau}^2$ is the amount of heterogeneity as estimated by the model.

The `level` argument can be an entire vector to highlight multiple pseudo confidence interval regions in the plot. The `shade` argument can be used to specify the corresponding colors (if unspecified, default colors are chosen). This way, one draw contour-enhanced funnel plots (Peters et al., 2008).

For (mixed-effects) meta-regression models (i.e., models involving moderators), the plot shows the residuals on the x-axis against their corresponding standard errors. Either the usual or deleted

residuals can be used for that purpose (set via the `type` argument). See [residuals](#) for more details on the different types of residuals.

With the `atransf` argument, the labels on the x-axis can be transformed with some suitable function. For example, when plotting log odds ratios, one could use `transf=exp` to obtain a funnel plot with the values on the x-axis corresponding to the odds ratios. See also [transf](#) for some other useful transformation functions in the context of a meta-analysis.

Instead of placing the standard errors on the y-axis, several other options are available by setting the `yaxis` argument to:

- `yaxis="vi"` for the sampling variances,
- `yaxis="seinv"` for the inverse of the standard errors,
- `yaxis="vinv"` for the inverse of the sampling variances,
- `yaxis="ni"` for the sample sizes,
- `yaxis="ninv"` for the inverse of the sample sizes,
- `yaxis="sqrtni"` for the square root of the sample sizes,
- `yaxis="sqrtninv"` for the inverse square root of the sample sizes,
- `yaxis="lni"` for the log of the sample sizes,
- `yaxis="wi"` for the weights.

However, only when `yaxis="sei"` (the default) will the pseudo confidence region have the expected (upside-down) funnel shape with straight lines. Also, when placing (a function of) the sample sizes or the weights on the y-axis, then the pseudo confidence region cannot be drawn. See Sterne and Egger (2001) for more details on the choice of the y-axis.

If the object passed to the function comes from the [trimfill](#) function, the studies that are filled in by the trim and fill method are also added to the funnel plot. The symbol to use for plotting the filled in studies can be specified via the `pch.fill` argument. Arguments `col` and `bg` can then be of length 2 to specify the (border) color and background color of the observed and filled in studies.

One can also directly pass a vector with estimates (via `x`) and the corresponding sampling variances (via `vi`), standard errors (via `sei`), and/or sample sizes (via `ni`) to the function. By default, the vertical reference line is then drawn at zero.

The arguments `back`, `shade`, and `hlines` can be set to `NULL` to suppress the shading and the horizontal reference line. One can also suppress the funnel by setting `refline` to `NULL`. Using `refline2`, one can also add a second reference line with a funnel to the plot (see ‘Examples’).

With the `label` argument, one can control whether points in the plot will be labeled. If `label="all"` (or `label=TRUE`), all points in the plot will be labeled. If `label="out"`, points falling outside of the pseudo confidence region will be labeled. Finally, one can also set this argument to a numeric value (between 1 and k) to specify how many of the most extreme points should be labeled (e.g., with `label=1` only the most extreme point are labeled, while with `label=3`, the most extreme, and the second and third most extreme points are labeled). With the `offset` argument, one can adjust the distance between the labels and the corresponding points.

By setting the `legend` argument to `TRUE`, a legend is added to the plot. One can also use a keyword for this argument to specify the position of the legend (e.g., `legend="topright"`; see [legend](#) for options). Finally, this argument can also be a list, with elements `x`, `y`, `inset`, `bty`, and `bg`, which are passed on to the corresponding arguments of the [legend](#) function for even more control

(elements not specified are set to defaults). The list can also include elements `studies` (a logical to specify whether to include ‘Studies’ in the legend; default is TRUE) and `show` (either `"pvals"` to show the p-values corresponding to the shade regions, `"cis"` to show the confidence interval levels corresponding to the shade regions, or NA to show neither; default is `"pvals"`).

Value

A data frame with components:

<code>x</code>	the x-axis coordinates of the points that were plotted.
<code>y</code>	the y-axis coordinates of the points that were plotted.
<code>slab</code>	the study labels.

Note that the data frame is returned invisibly.

Note

Placing (a function of) the sample sizes on the y-axis (i.e., using `yaxis="ni"`, `yaxis="ninv"`, `yaxis="sqrtni"`, `yaxis="sqrtninv"`, or `yaxis="lni"`) is only possible when information about the sample sizes is actually stored within the object passed to the `funnel` function. That should automatically be the case when the estimates were computed with the `escalc` function or when the estimates were computed within the model fitting function. On the other hand, this will not be the case when `rma.uni` was used together with the `yi` and `vi` arguments and the `yi` and `vi` values were *not* computed with `escalc`. In that case, it is still possible to pass information about the sample sizes to the `rma.uni` function (e.g., use `rma.uni(yi, vi, ni=ni, data=dat)`, where data frame `dat` includes a variable called `ni` with the sample sizes).

When using unweighted estimation, using `yaxis="wi"` will place all points on a horizontal line. When directly passing a vector with the estimates to the function, `yaxis="wi"` is equivalent to `yaxis="vinv"`, except that the weights are expressed in percent.

For argument `slab` and when specifying vectors for arguments `pch`, `col`, and/or `bg` and when `x` is an object of class `"rma"`, the variables specified are assumed to be of the same length as the data passed to the model fitting function (and if the data argument was used in the original model fit, then the variables will be searched for within this data frame first). Any subsetting and removal of studies with missing values is automatically applied to the variables specified via these arguments.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Light, R. J., & Pillemer, D. B. (1984). *Summing up: The science of reviewing research*. Cambridge, MA: Harvard University Press.

Peters, J. L., Sutton, A. J., Jones, D. R., Abrams, K. R., & Rushton, L. (2008). Contour-enhanced meta-analysis funnel plots help distinguish publication bias from other causes of asymmetry. *Journal of Clinical Epidemiology*, **61**(10), 991–996. <https://doi.org/10.1016/j.jclinepi.2007.11.010>

Sterne, J. A. C., & Egger, M. (2001). Funnel plots for detecting bias in meta-analysis: Guidelines on choice of axis. *Journal of Clinical Epidemiology*, **54**(10), 1046–1055. [https://doi.org/10.1016/s0895-4356\(01\)00377-](https://doi.org/10.1016/s0895-4356(01)00377-)

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which funnel plots can be drawn.

[trimfill](#) for the trim and fill method, [regtest](#) for the regression test, and [ranktest](#) for the rank correlation test.

Examples

```
### copy BCG vaccine data into 'dat'
dat <- dat.bcg

### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat)

### fit random-effects model
res <- rma(yi, vi, data=dat, slab=paste(author, year, sep=", "))

### draw a standard funnel plot
funnel(res)

### show risk ratio values on x-axis (log scale)
funnel(res, attransf=exp)

### label points outside of the pseudo confidence interval region
funnel(res, attransf=exp, label="out")

### passing log risk ratios and sampling variances directly to the function
### note: same plot, except that the reference line is centered at zero
funnel(dat$yi, dat$vi)

### the with() function can be used to avoid having to retype dat$... over and over
with(dat, funnel(yi, vi))

### can accomplish the same thing by setting refline=0
funnel(res, refline=0)

### adjust the position of the x-axis labels, number of digits, and y-axis limits
funnel(res, attransf=exp, at=log(c(0.125, 0.25, 0.5, 1, 2)), digits=3L, ylim=c(0,0.8))

### contour-enhanced funnel plot centered at 0 (see Peters et al., 2008)
funnel(res, level=c(90, 95, 99), shade=c("white", "gray55", "gray75"), refline=0, legend=TRUE)

### default shades are chosen if the argument is not specified
funnel(res, level=c(90, 95, 99), refline=0, legend=TRUE)

### same, but show risk ratio values on the x-axis and some further adjustments
funnel(res, level=c(90, 95, 99), digits=3L, ylim=c(0,0.8),
       attransf=exp, at=log(c(0.125, 0.25, 0.5, 1, 2, 4, 8)), refline=0, legend=TRUE)
```

```

### same, but show confidence interval levels in the legend
funnel(res, level=c(90, 95, 99), digits=3L, ylim=c(0,0.8),
       atransf=exp, at=log(c(0.125, 0.25, 0.5, 1, 2, 4, 8)), refline=0, legend=list(show="cis"))

### illustrate the use of vectors for 'pch' and 'col'
res <- rma(yi, vi, data=dat, subset=2:10)
funnel(res, pch=ifelse(yi > -1, 19, 21), col=ifelse(sqrt(vi) > 0.3, "red", "blue"))

### can add a second funnel via (undocumented) argument refline2
funnel(res, atransf=exp, at=log(c(0.125, 0.25, 0.5, 1, 2, 4)), digits=3L, ylim=c(0,0.8), refline2=0)

### mixed-effects model with absolute latitude in the model
res <- rma(yi, vi, mods = ~ ablat, data=dat)

### funnel plot of the residuals
funnel(res)

### simulate a large meta-analytic dataset (correlations with rho = 0.2)
### with no heterogeneity or publication bias; then try out different
### versions of the funnel plot

gencor <- function(rhoi, ni) {
  x1 <- rnorm(ni, mean=0, sd=1)
  x2 <- rnorm(ni, mean=0, sd=1)
  x3 <- rhoi*x1 + sqrt(1-rhoi^2)*x2
  cor(x1, x3)
}

set.seed(1234)
k <- 200 # number of studies to simulate
ni <- round(rchisq(k, df=2) * 20 + 20) # simulate sample sizes (skewed distribution)
ri <- mapply(gencor, rep(0.2,k), ni) # simulate correlations

res <- rma(measure="ZCOR", ri=ri, ni=ni, method="EE") # use r-to-z transformed correlations

funnel(res, yaxis="sei")
funnel(res, yaxis="vi")
funnel(res, yaxis="seinv")
funnel(res, yaxis="vinv")
funnel(res, yaxis="ni")
funnel(res, yaxis="ninv")
funnel(res, yaxis="sqrtni")
funnel(res, yaxis="sqrtninv")
funnel(res, yaxis="lni")
funnel(res, yaxis="wi")

```

Description

Function to create GOSH plots for objects of class "rma".

Usage

```
gosh(x, ...)

## S3 method for class 'rma'
gosh(x, subsets, progbar=TRUE, parallel="no", ncpus=1, cl, ...)
```

Arguments

<code>x</code>	an object of class "rma".
<code>subsets</code>	optional integer to specify the number of subsets.
<code>progbar</code>	logical to specify whether a progress bar should be shown (the default is TRUE).
<code>parallel</code>	character string to specify whether parallel processing should be used (the default is "no"). For parallel processing, set to either "snow" or "multicore". See 'Note'.
<code>ncpus</code>	integer to specify the number of processes to use in the parallel processing.
<code>cl</code>	optional cluster to use if <code>parallel="snow"</code> . If unspecified, a cluster on the local machine is created for the duration of the call.
<code>...</code>	other arguments.

Details

The model specified via `x` must be a model fitted with either the [rma.uni](#), [rma.mh](#), or [rma.peto](#) functions.

Olkin et al. (2012) proposed the GOSH (graphical display of study heterogeneity) plot, which is based on examining the results of an equal-effects model in all possible subsets of size $1, \dots, k$ of the k studies included in a meta-analysis. In a homogeneous set of studies, the pooled estimates obtained this way should form a roughly symmetric, contiguous, and unimodal distribution. On the other hand, when the distribution is multimodal, then this suggests the presence of heterogeneity, possibly due to outliers and/or distinct subgroups of studies. Plotting the pooled estimates against some measure of heterogeneity (e.g., I^2 , H^2 , or the Q -statistic) can also help to reveal subclusters, which are indicative of heterogeneity. The same type of plot can be produced by first fitting an equal-effects model with either the [rma.uni](#) (using `method="EE"`), [rma.mh](#), or [rma.peto](#) functions and then passing the fitted model object to the `gosh` function and then plotting the results.

For models fitted with the [rma.uni](#) function (which may be random-effects or mixed-effects meta-regressions models), the idea underlying this type of plot can be generalized (Viechtbauer, 2021) by examining the distribution of all model coefficients, plotting them against each other, and against some measure of (residual) heterogeneity (including the estimate of τ^2 or its square root).

Note that for models without moderators, application of the method requires fitting a total of $2^k - 1$ models, which could be an excessively large number when k is large. For example, for $k = 10$, there are only 1023 possible subsets, but for $k = 20$, this number already grows to 1,048,575. For even larger k , it may become computationally infeasible to consider all possible subsets. Instead, we can then examine (a sufficiently large number of) random subsets.

By default, if the number of possible subsets is $\leq 10^6$, the function will consider all possible subsets and otherwise 10^6 random subsets. One can use the `subsets` argument to specify a different number of subsets to consider. If `subsets` is specified and it is actually larger than the number of possible subsets, then the function automatically only considers the possible subsets and does not use random subsets.

When `x` is an equal-effects model or a random-effects model fitted using `method="DL"`, provisions have been made to speed up the model fitting to the various subsets. For random-effects models using some other estimator of τ^2 (especially an iterative one like `method="REML"`), the computations will be considerably slower.

Value

An object of class `"gosh.rma"`. The object is a list containing the following components:

<code>res</code>	a data frame with the results for each subset (including various heterogeneity statistics and the model coefficient(s)).
<code>incl</code>	a matrix indicating which studies were included in which subset.
<code>...</code>	some additional elements/values.

The results can be printed with the `print` function and plotted with the `plot` function.

Note

On machines with multiple cores, one can try to speed things up by delegating the model fitting to separate worker processes, that is, by setting `parallel="snow"` or `parallel="multicore"` and `ncpus` to some value larger than 1. Parallel processing makes use of the `parallel` package, using the `makePSOCKcluster` and `parLapply` functions when `parallel="snow"` or using `mclapply` when `parallel="multicore"` (the latter only works on Unix/Linux-alikes).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Olkin, I., Dahabreh, I. J., & Trikalinos, T. A. (2012). GOSH - a graphical display of study heterogeneity. *Research Synthesis Methods*, **3**(3), 214–223. <https://doi.org/10.1002/jrsm.1053>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/9781315>

See Also

[rma.uni](#), [rma.mh](#), and [rma.peto](#) for functions to fit models for which GOSH plots can be drawn.
[influence](#) for other model diagnostics.

Examples

```
### calculate log odds ratios and corresponding sampling variances
dat <- escalc(measure="OR", ai=ai, n1i=n1i, ci=ci, n2i=n2i, data=dat.egger2001)

### meta-analysis of all trials including ISIS-4 using an equal-effects model
res <- rma(yi, vi, data=dat, method="EE")

### fit FE model to all possible subsets (65535 models)
## Not run:
sav <- gosh(res, progbar=FALSE)
sav

### create GOSH plot
### red points for subsets that include and blue points
### for subsets that exclude study 16 (the ISIS-4 trial)
plot(sav, out=16, breaks=100)

## End(Not run)
```

 hc

Meta-Analysis based on the Method by Henmi and Copas (2010)

Description

Function to obtain a pooled estimate and corresponding confidence interval under a random-effects model using the method described by Henmi and Copas (2010).

Usage

```
hc(object, ...)

## S3 method for class 'rma.uni'
hc(object, digits, transf, targs, control, ...)
```

Arguments

object	an object of class "rma.uni".
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
transf	optional argument to specify a function to transform the estimate and the corresponding interval bounds (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified under <code>transf</code> .
control	list of control values for the iterative algorithm. If unspecified, default values are used. See 'Note'.
...	other arguments.

Details

The model specified via `object` must be a model without moderators (i.e., either an equal- or a random-effects model).

When using the usual methods for fitting a random-effects model (i.e., weighted estimation with inverse-variance weights), the weights assigned to smaller and larger studies become more uniform as the amount of heterogeneity increases. As a consequence, the pooled estimate can become increasingly biased under certain forms of publication bias (where smaller studies on one side of the funnel plot are missing). The method by Henmi and Copas (2010) counteracts this problem by providing a pooled estimate that is based on inverse-variance weights as used under an equal-effects model, which are not affected by the amount of heterogeneity. The amount of heterogeneity is still estimated (with the DerSimonian-Laird estimator) and incorporated into the standard error of the pooled estimate and the corresponding confidence interval.

Currently, there is only a method for handling objects of class `"rma.uni"` with the `hc` function. It therefore provides a method for conducting a sensitivity analysis after the model has been fitted with the `rma.uni` function.

Value

An object of class `"hc.rma.uni"`. The object is a list containing the following components:

<code>beta</code>	the pooled estimate.
<code>se</code>	corresponding standard error.
<code>ci.lb</code>	lower bound of the confidence interval.
<code>ci.ub</code>	upper bound of the confidence interval.
<code>...</code>	some additional elements/values.

The results are formatted and printed with the `print` function.

Note

The method makes use of the `uniroot` function. By default, the desired accuracy is set equal to `.Machine$double.eps^0.25` and the maximum number of iterations to 1000. The desired accuracy (`tol`) and the maximum number of iterations (`maxiter`) can be adjusted with the `control` argument (i.e., `control=list(tol=value, maxiter=value)`).

Author(s)

Original code by Henmi and Copas (2010). Corrected for typos by Michael Dewey (<lists@dewey.myzen.co.uk>). Incorporated into the package with some small adjustments for consistency with the other functions in the package by Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Henmi, M., & Copas, J. B. (2010). Confidence intervals for random effects meta-analysis and robustness to publication bias. *Statistics in Medicine*, **29**(29), 2969–2983. <https://doi.org/10.1002/sim.4029>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#) for the function to fit `rma.uni` models.

Examples

```
### calculate log odds ratios and corresponding sampling variances
dat <- escalc(measure="OR", ai=ai, nli=nli, ci=ci, n2i=n2i, data=dat.lee2004)
dat

### meta-analysis based on log odds ratios
res <- rma(yi, vi, data=dat)
res

### funnel plot as in Henmi and Copas (2010)
funnel(res, yaxis="seinv", refline=0, xlim=c(-3,3), ylim=c(0.5,3.5),
       steps=7, digits=1, back="white")

### use method by Henmi and Copas (2010) as a sensitivity analysis
hc(res)

### back-transform results to odds ratio scale
hc(res, transf=exp)
```

 hettest

Test for Heteroscedastic Heterogeneity

Description

Function to test if the heterogeneity in the true effects is heteroscedastic.

Usage

```
hettest(x, vi, sei, subset, data, method="REML",
       test="score", boot=TRUE, progbar=TRUE, digits, control, ...)

## S3 method for class 'hettest'
print(x, digits=x$digits, ...)
```

Arguments

<code>x</code>	a vector with the effect size estimates or an object of class <code>"rma"</code> .
<code>vi</code>	vector with the corresponding sampling variances. Ignored if <code>x</code> is an object of class <code>"rma"</code> .
<code>sei</code>	vector with the corresponding standard errors (note: only one of the two, <code>vi</code> or <code>sei</code> , needs to be specified).
<code>subset</code>	optional (logical or numeric) vector to specify the subset of studies that should be used for the calculation. Ignored if <code>x</code> is an object of class <code>"rma"</code> .

<code>data</code>	optional data frame containing the variables given to the arguments above.
<code>method</code>	character string to specify whether the test should be based on maximum likelihood ("ML") or restricted maximum likelihood ("REML") estimation. The default is "REML". Ignored if <code>x</code> is an object of class "rma".
<code>test</code>	character string (case insensitive) to specify which test to run. See 'Details'.
<code>boot</code>	logical to specify whether bootstrapping should be used (the default is TRUE). Can also be a number to specify the number of bootstrap samples to take.
<code>progbar</code>	logical to specify whether a progress bar should be shown when using bootstrapping (the default is TRUE).
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded.
<code>control</code>	optional list of control values. See 'Note'.
<code>...</code>	other arguments.

Details

The standard random-effects model (that can be fitted with the [rma.uni](#) function) assumes that the heterogeneity in the true effects is homoscedastic. This function can be used to test this assumption.

Via argument `x`, one can either pass a vector with the effect size estimates to the function (and then the corresponding sampling variances via `vi` or the standard errors via `sei`) or an object of class "rma". When passing a model object to argument `x`, the model must be a standard random-effects model that was fitted with either maximum likelihood (ML) or restricted maximum likelihood (REML) estimation.

When `x` is a vector with the effect size estimates, then argument `method` can be used to select whether the test should be based on ML or REML estimation. The latter is the default. On the other hand, if `x` is a model object, then the estimation method that was used for fitting the model determines on which estimation method the test is based.

The test to run is determined by the `test` argument and can either be:

- `test="lrt"` for a likelihood ratio test,
- `test="wald"` for a Wald-type test,
- `test="score"` for a score test.

The likelihood ratio test compares the fit of the standard random-effects model with a saturated random-effects model where each study has a study-specific τ_i^2 value. The Wald-type test directly tests all $k - 1$ contrasts between the estimated τ_i^2 values. Finally, the score test evaluates whether allowing study-specific τ_i^2 values would improve model fit based on the score vector and the information matrix evaluated under the standard random-effects model, without requiring estimation of the saturated model. This tends to make this test more stable, yielding better control of the Type I error rate, and therefore is the default option.

However, the assumed null distribution for all of these tests (a chi-square distribution with $k - 1$ degrees of freedom) is not correct. Therefore, when `boot=TRUE` (the default), the function uses parametric bootstrapping to obtain a p-value for a given test. By default, 1000 bootstrap samples are taken, but one can also specify the number of samples to take via the `boot` argument. When `progbar=TRUE` (the default), a progress bar is shown to indicate the progress of the bootstrapping.

One can also use the function to carry out several goodness-of-fit tests based on the Pearson residuals obtained from the standard random-effects model (see also the [residuals](#) functions, which can compute such residuals). If the assumptions underlying the model are correct (including the assumption that the heterogeneity is homoscedastic), then the Pearson residuals are asymptotically standard normal. Accordingly, one can use the Kolmogorov-Smirnov (KS) test or the Anderson-Darling (AD) test to test if this assumption holds. Alternatively, one can use these test to test if the squared Pearson residuals are chi-square distributed with one degree of freedom. Therefore, additional options for the test argument are:

- `test="ksn"` to test if the Pearson residuals are standard normal using the KS test,
- `test="ksx2"` to test if the squared Pearson residuals are chi-squared distributed with one degree of freedom using the KS test,
- `test="adn"` to test if the Pearson residuals are standard normal using the AD test,
- `test="adx2"` to test if the squared Pearson residuals are chi-squared distributed with one degree of freedom using the AD test.

In the present context, we cannot rely on known properties of these tests to obtain the corresponding p-values. Therefore, bootstrapping is always used for these tests to examine how extreme the observed statistics of the KS and AD tests are under their corresponding bootstrap distributions.

Value

An object of class "hettest". The object is a list containing the following components:

<code>statistic</code>	the test statistic.
<code>df</code>	the degrees of freedom.
<code>pval</code>	the p-value.
<code>tau2i</code>	the estimated τ_i^2 values from the saturated random-effects model.
<code>se.tau2i</code>	the corresponding standard errors (only when running a Wald-type test).
<code>statistic.boot</code>	the bootstrap values of the test statistic (only when <code>boot=TRUE</code>).
<code>...</code>	some additional elements/values.

The results are formatted and printed with the `print` function.

Note

The saturated random-effects model is fitted with an iterative algorithm, alternating between the computation of the τ_i^2 estimates and the estimate of μ . By default, the initial τ_i^2 values are set equal to the τ^2 estimate from the standard random-effects model. This process continues for a maximum of 1000 iterations until the change in all of the estimated τ_i^2 values is smaller than 10^{-8} from one iteration to the next. The starting τ_i^2 values, threshold, and maximum number of iterations can be adjusted via the `control` argument by setting `control=list(tau2i.init=values, threshold=value, maxiter=value)`.

When bootstrapping, the iterative algorithm must be run for each bootstrap sample. In rare cases, the algorithm may not converge. The output shows for how many bootstrap samples the calculations could be completed (this should usually be '1000 / 1000').

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

Examples

```
### copy data from Bangert-Drowns et al. (2004) into 'dat'
dat <- dat.bangertdrowns2004

### fit a standard random-effects model
res <- rma(yi, vi, data=dat)
res

## Not run:
### test for heteroscedastic heterogeneity
set.seed(1234)
hettest(res)

## End(Not run)
```

influence.rma.mv

Model Diagnostics for 'rma.mv' Objects

Description

Functions to compute various outlier and influential study diagnostics (some of which indicate the influence of deleting one study at a time on the model fit or the fitted/residual values) for objects of class "rma.mv".

Usage

```
## S3 method for class 'rma.mv'
cooks.distance(model, progbar=FALSE, cluster,
               reestimate=TRUE, parallel="no", ncpus=1, cl, ...)

## S3 method for class 'rma.mv'
dfbetas(model, progbar=FALSE, cluster,
         reestimate=TRUE, parallel="no", ncpus=1, cl, ...)

## S3 method for class 'rma.mv'
hatvalues(model, type="diagonal", ...)
```

Arguments

model	an object of class "rma.mv".
progbar	logical to specify whether a progress bar should be shown (the default is FALSE).
cluster	optional vector to specify a clustering variable to use for computing the Cook's distances and DFBETAS values. If unspecified, these measures are computed based on the individual estimates.

<code>reestimate</code>	logical to specify whether variance/correlation components should be re-estimated after deletion of the i th case (the default is TRUE).
<code>parallel</code>	character string to specify whether parallel processing should be used (the default is "no"). For parallel processing, set to either "snow" or "multicore". See 'Note'.
<code>ncpus</code>	integer to specify the number of processes to use in the parallel processing.
<code>cl</code>	optional cluster to use if <code>parallel="snow"</code> . If unspecified, a cluster on the local machine is created for the duration of the call.
<code>type</code>	character string to specify whether only the diagonal of the hat matrix ("diagonal") or the entire hat matrix ("matrix") should be returned.
<code>...</code>	other arguments.

Details

The term 'case' below refers to a particular row from the dataset used in the model fitting (when argument `cluster` is not specified) or each level of the variable specified via `cluster`.

Cook's distance for the i th case can be interpreted as the Mahalanobis distance between the entire set of predicted values once with the i th case included and once with the i th case excluded from the model fitting.

The DFBETAS value(s) essentially indicate(s) how many standard deviations the estimated coefficient(s) change(s) after excluding the i th case from the model fitting.

Value

The `cooks.distance` function returns a vector. The `dfbetas` function returns a data frame. The `hatvalues` function returns either a vector with the diagonal elements of the hat matrix or the entire hat matrix.

Note

The variable specified via `cluster` is assumed to be of the same length as the data originally passed to the `rma.mv` function (and if the data argument was used in the original model fit, then the variable will be searched for within this data frame first). Any subsetting and removal of studies with missing values that was applied during the model fitting is also automatically applied to the variable specified via the `cluster` argument.

Leave-one-out diagnostics are calculated by refitting the model k times (where k denotes the number of cases). Depending on how large k is, it may take a few moments to finish the calculations. For complex models fitted with `rma.mv`, this can become computationally expensive.

On machines with multiple cores, one can try to speed things up by delegating the model fitting to separate worker processes, that is, by setting `parallel="snow"` or `parallel="multicore"` and `ncpus` to some value larger than 1. Parallel processing makes use of the `parallel` package, using the `makePSOCKcluster` and `parLapply` functions when `parallel="snow"` or using `mclapply` when `parallel="multicore"` (the latter only works on Unix/Linux-alikes).

Alternatively (or in addition to using parallel processing), one can also set `reestimate=FALSE`, in which case any variance/correlation components in the model are not re-estimated after deleting the i th case from the dataset. Doing so only yields an approximation to the Cook's distances and

DFBETAS values that ignores the influence of the i th case on the variance/correlation components, but is considerably faster (and often yields similar results).

It may not be possible to fit the model after deletion of the i th case from the dataset. This will result in NA values for that case.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics*. New York: Wiley.

Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression*. London: Chapman and Hall.

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/978131>

Viechtbauer, W., & Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research Synthesis Methods*, **1**(2), 112–125. <https://doi.org/10.1002/jrsm.11>

See Also

[rstudent](#) for externally standardized residuals and [weights](#) for model fitting weights.

Examples

```
### copy data from Konstantopoulos (2011) into 'dat'
dat <- dat.konstantopoulos2011

### multilevel random-effects model
res <- rma.mv(yi, vi, random = ~ 1 | district/school, data=dat)
print(res, digits=3)

### compute Cook's distance for each estimate
x <- cooks.distance(res)
x
plot(x, type="o", pch=19, xlab="Observed Outcome", ylab="Cook's Distance", las=1)

### compute Cook's distance for each district
x <- cooks.distance(res, cluster=district)
x
plot(x, type="o", pch=19, xlab="District", ylab="Cook's Distance", xaxt="n", las=1)
axis(side=1, at=seq_along(x), labels=as.numeric(names(x)))

### hat values
hatvalues(res)
```

influence.rma.uni *Model Diagnostics for 'rma.uni' Objects*

Description

Functions to compute various outlier and influential study diagnostics (some of which indicate the influence of deleting one study at a time on the model fit or the fitted/residual values) for objects of class "rma.uni". For the corresponding documentation for "rma.mv" objects, see [influence](#).

Usage

```
## S3 method for class 'rma.uni'
influence(model, digits, progbar=FALSE, ...)

## S3 method for class 'infl.rma.uni'
print(x, digits=x$digits, infonly=FALSE, ...)

## S3 method for class 'rma.uni'
cooks.distance(model, progbar=FALSE, ...)
## S3 method for class 'rma.uni'
dfbetas(model, progbar=FALSE, ...)
## S3 method for class 'rma.uni'
hatvalues(model, type="diagonal", ...)
```

Arguments

model	an object of class "rma.uni".
x	an object of class "infl.rma.uni" (for print).
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
progbar	logical to specify whether a progress bar should be shown (the default is FALSE).
infonly	logical to specify whether only the influential cases should be printed (the default is FALSE).
type	character string to specify whether only the diagonal of the hat matrix ("diagonal") or the entire hat matrix ("matrix") should be returned.
...	other arguments.

Details

The term 'case' below refers to a particular row from the dataset used in the model fitting (which is typically synonymous with 'study').

The influence function calculates the following leave-one-out diagnostics for each case:

- externally standardized residual,

- DFFITS value,
- Cook's distance,
- covariance ratio,
- the leave-one-out amount of (residual) heterogeneity,
- the leave-one-out test statistic of the test for (residual) heterogeneity,
- DFBETAS value(s).

The diagonal elements of the hat matrix and the weights (in %) given to the estimates during the model fitting are also provided (except for their scaling, the hat values and weights are the same for models without moderators, but will differ when moderators are included).

For details on externally standardized residuals, see [rstudent](#).

The DFFITS value essentially indicates by how many standard deviations the predicted value for the i th case changes after excluding the i th case from the model fitting.

Cook's distance can be interpreted as the Mahalanobis distance between the entire set of predicted values once with the i th case included and once with the i th case excluded from the model fitting.

The covariance ratio is defined as the determinant of the variance-covariance matrix of the parameter estimates based on the dataset with the i th case removed divided by the determinant of the variance-covariance matrix of the parameter estimates based on the complete dataset. A value below 1 therefore indicates that removal of the i th case yields more precise estimates of the model coefficients.

The leave-one-out amount of (residual) heterogeneity is the estimated value of τ^2 based on the dataset with the i th case removed. This is always equal to 0 for equal-effects models.

Similarly, the leave-one-out test statistic of the test for (residual) heterogeneity is the value of the test statistic of the test for (residual) heterogeneity calculated based on the dataset with the i th case removed.

Finally, the DFBETAS value(s) essentially indicate(s) how many standard deviations the estimated coefficient(s) change(s) after excluding the i th case from the model fitting.

A case may be considered to be 'influential' if at least one of the following is true:

- The absolute DFFITS value is larger than $3 \times \sqrt{p/(k-p)}$, where p is the number of model coefficients and k the number of cases.
- The lower tail area of a chi-square distribution with p degrees of freedom cut off by the Cook's distance is larger than 50%.
- The hat value is larger than $3 \times (p/k)$.
- Any DFBETAS value is larger than 1.

Cases which are considered influential with respect to any of these measures are marked with an asterisk. Note that the chosen cut-offs are (somewhat) arbitrary. Substantively informed judgment should always be used when examining the influence of each case on the results.

Value

An object of class "infl.rma.uni", which is a list containing the following components:

inf	an element of class "list.rma" with the externally standardized residuals, DF-FITS values, Cook's distances, covariance ratios, leave-one-out τ^2 estimates, leave-one-out (residual) heterogeneity test statistics, hat values, weights, and an indicator whether a case is influential.
dfbs	an element of class "list.rma" with the DFBETAS values.
...	some additional elements/values.

The results are printed with `print` and plotted with `plot`. To format the results as a data frame, one can use the `as.data.frame` function.

Note

Leave-one-out diagnostics are calculated by refitting the model k times. Depending on how large k is, it may take a few moments to finish the calculations. There are shortcuts for calculating at least some of these values without refitting the model each time, but these are currently not implemented (and may not exist for all of the leave-one-out diagnostics calculated by the function).

It may not be possible to fit the model after deletion of the i th case from the dataset. This will result in NA values for that case.

Certain relationships between the leave-one-out diagnostics and the (internally or externally) standardized residuals (Belsley, Kuh, & Welsch, 1980; Cook & Weisberg, 1982) no longer hold for meta-analytic models. Maybe there are other relationships. These remain to be determined.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics*. New York: Wiley.
- Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression*. London: Chapman and Hall.
- Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. San Diego, CA: Academic Press.
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/978131>
- Viechtbauer, W., & Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research Synthesis Methods*, **1**(2), 112–125. <https://doi.org/10.1002/jrsm.11>

See Also

`plot` for a method to plot the outlier and influential case diagnostics.

`rstudent` for externally standardized residuals and `weights` for model fitting weights.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### compute the diagnostics
inf <- influence(res)
inf

### plot the values
plot(inf)

### compute Cook's distances, DFBETAS values, and hat values
cooks.distance(res)
dfbetas(res)
hatvalues(res)
```

labbe

L'Abbe Plots for 'rma' Objects

Description

Function to create L'Abbé plots for objects of class "rma".

Usage

```
labbe(x, ...)

## S3 method for class 'rma'
labbe(x, xlim, ylim, lim, xlab, ylab, flip=FALSE,
      ci=FALSE, pi=FALSE, grid=FALSE, legend=FALSE,
      add=x$add, to=x$to, transf, targ,
      pch=21, psize, plim=c(0.5,3.5),
      col, bg, lty, ...)
```

Arguments

x	an object of class "rma".
xlim	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
ylim	y-axis limits. If unspecified, the function sets the y-axis limits to some sensible values.
lim	axis limits. If specified, this is used for both xlim and ylim.
xlab	title for the x-axis. If unspecified, the function sets an appropriate axis title.
ylab	title for the y-axis. If unspecified, the function sets an appropriate axis title.

<code>flip</code>	logical to specify whether the groups to plot on the x- and y-axis should be flipped (the default is FALSE).
<code>ci</code>	logical to specify whether the confidence interval region should be shown in the plot (the default is FALSE). Can also be a color name.
<code>pi</code>	logical to specify whether the prediction interval region should be shown in the plot (the default is FALSE). Can also be a color name.
<code>grid</code>	logical to specify whether a grid should be added to the plot (the default is FALSE). Can also be a color name.
<code>legend</code>	logical to specify whether a legend should be added to the plot (the default is FALSE). See 'Details'.
<code>add</code>	See the documentation of the escalc function for more details.
<code>to</code>	See the documentation of the escalc function for more details.
<code>transf</code>	optional argument to specify a function to transform the estimates (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
<code>targs</code>	optional arguments needed by the function specified under <code>transf</code> .
<code>pch</code>	plotting symbol to use for the estimates. By default, an open circle is used. Can also be a vector of values. See points for other options.
<code>psize</code>	optional numeric vector to specify the point sizes for the estimates. If unspecified, the point sizes are a function of the precision of the estimates. Can also be a vector of values.
<code>plim</code>	numeric vector of length 2 to scale the point sizes (ignored when <code>psize</code> is specified). See 'Details'.
<code>col</code>	optional character string to specify the (border) color of the points. Can also be a vector.
<code>bg</code>	optional character string to specify the background color of open plot symbols. Can also be a vector. Set to NA to make the plotting symbols transparent.
<code>lty</code>	optional argument to specify the line type for the diagonal reference line and the line that indicates the pooled estimate based on the fitted model. If unspecified, the function sets this to <code>c("solid", "dashed")</code> by default (use "blank" to suppress a line).
<code>...</code>	other arguments.

Details

The model specified via `x` must be a model without moderators (i.e., either an equal- or a random-effects model) fitted with either the [rma.uni](#), [rma.mh](#), [rma.peto](#), or [rma.glmm](#) functions. Moreover, the model must have been fitted with `measure` set equal to "RD" (for risk differences), "RR" (for risk ratios), "OR" (for odds ratios), "AS" (for arcsine square root transformed risk differences), "IRR" (for incidence rate ratios), "IRD" (for incidence rate differences), or "IRSD" (for square root transformed incidence rate differences).

The function calculates the arm-level estimates for the two groups (e.g., treatment and control) and plots them against each other. In particular, the function plots the raw proportions of the two groups against each other when analyzing risk differences, the log of the proportions when analyzing (log)

risk ratios, the log odds when analyzing (log) odds ratios, the arcsine square root transformed proportions when analyzing arcsine square root transformed risk differences, the raw incidence rates when analyzing incidence rate differences, the log of the incidence rates when analyzing (log) incidence rate ratios, and the square root transformed incidence rates when analyzing square root transformed incidence rate differences. The `transf` argument can be used to transform these values (e.g., `transf=exp` to transform the log of the proportions back to raw proportions; see also [transf](#)).

As described under the documentation for the [escalc](#) function, zero cells can lead to problems when calculating particular effect size estimates. Adding a small constant to the cells of the 2×2 tables is a common solution to this problem. By default, the functions adopts the same method for handling zero cells as was used when fitting the model.

By default (i.e., when `psize` is not specified), the point sizes are a function of the precision (i.e., inverse standard errors) of the estimates. This way, more precise estimates are visually more prominent in the plot. By making the point sizes a function of the inverse standard errors of the estimates, their areas are proportional to the inverse sampling variances, which corresponds to the weights they would receive in an equal-effects model. However, the point sizes are rescaled so that the smallest point size is `plim[1]` and the largest point size is `plim[2]`. As a result, their relative sizes (i.e., areas) no longer exactly correspond to their relative weights in such a model. If exactly relative point sizes are desired, one can set `plim[2]` to `NA`, in which case the points are rescaled so that the smallest point size corresponds to `plim[1]` and all other points are scaled accordingly. As a result, the largest point may be very large. Alternatively, one can set `plim[1]` to `NA`, in which case the points are rescaled so that the largest point size corresponds to `plim[2]` and all other points are scaled accordingly. As a result, the smallest point may be very small. To avoid the latter, one can also set `plim[3]`, which enforces a minimal point size.

The solid line corresponds to identical estimates in the two groups (i.e., the absence of a difference between the two groups). The dashed line indicates the pooled estimate based on the fitted model. If `ci=TRUE`, then the darker shaded region indicates the corresponding confidence interval. If `pi=TRUE`, then the lighter shaded region indicates the corresponding prediction interval.

By setting the `legend` argument to `TRUE`, a legend is added to the plot. One can also use a keyword for this argument to specify the position of the legend (e.g., `legend="topleft"`; see [legend](#) for options). Finally, this argument can also be a list, with elements `x`, `y`, `inset`, `cex`, and `pt.cex`, which are passed on to the corresponding arguments of the [legend](#) function for even more control (elements not specified are set to defaults).

Value

A data frame with components:

<code>x</code>	the x-axis coordinates of the points that were plotted.
<code>y</code>	the y-axis coordinates of the points that were plotted.
<code>cex</code>	the point sizes.
<code>pch</code>	the plotting symbols.
<code>col</code>	the point colors.
<code>bg</code>	the background colors.
<code>ids</code>	the study id numbers.
<code>slab</code>	the study labels.

Note that the data frame is returned invisibly.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Jiménez, F. J., Guallar, E., & Martín-Moreno, J. M. (1997). A graphical display useful for meta-analysis. *European Journal of Public Health*, **7**(1), 101–105. <https://doi.org/10.1093/eurpub/8.1.92>
- L'Abbé, K. A., Detsky, A. S., & O'Rourke, K. (1987). Meta-analysis in clinical research. *Annals of Internal Medicine*, **107**(2), 224–233. <https://doi.org/10.7326/0003-4819-107-2-224>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), and [rma.glmm](#) for functions to fit models for which L'Abbé plots can be drawn.

Examples

```
### meta-analysis of log odds ratios using a random-effects model
dat <- dat.damico2009
res <- rma(measure="OR", ai=xt, n1i=nt, ci=xc, n2i=nc, data=dat)
res

### default plot with log odds on the x- and y-axis
labbe(res)

### plot with odds values on the x- and y-axis and some customization
labbe(res, ci=TRUE, pi=TRUE, grid=TRUE, legend=TRUE, bty="l",
      transf=exp, xlab="Odds (Control Group)", ylab="Odds (Treatment Group)")

### plot with risk values on the x- and y-axis and some customization
labbe(res, ci=TRUE, pi=TRUE, grid=TRUE, legend=TRUE, bty="l",
      transf=plogis, lim=c(0,1), xlab="Risk (Control Group)",
      ylab="Risk (Treatment Group)")
```

Description

Functions to carry out a 'leave-one-out analysis', by repeatedly fitting the specified model leaving out one study (or cluster level) at a time.

Usage

```

leave1out(x, ...)

## S3 method for class 'rma.uni'
leave1out(x, cluster, digits, transf, targs, progbar=FALSE, ...)
## S3 method for class 'rma.mh'
leave1out(x, cluster, digits, transf, targs, progbar=FALSE, ...)
## S3 method for class 'rma.peto'
leave1out(x, cluster, digits, transf, targs, progbar=FALSE, ...)

```

Arguments

<code>x</code>	an object of class "rma.uni", "rma.mh", or "rma.peto".
<code>cluster</code>	optional vector to specify a clustering variable.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
<code>transf</code>	optional argument to specify a function to transform the estimates and interval bounds (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
<code>targs</code>	optional arguments needed by the function specified under <code>transf</code> .
<code>progbar</code>	logical to specify whether a progress bar should be shown (the default is FALSE).
<code>...</code>	other arguments.

Details

In a leave-one-out analysis, the same model is repeatedly fitted, leaving out one study at a time. By doing so, we can assess how much the results (e.g., the pooled estimate and the amount of heterogeneity) are influenced by each individual study. It is also possible to specify a cluster variable, in which case each cluster level is left out in turn.

Note that for "rma.uni" objects, the model specified via `x` must be a model without moderators (i.e., either an equal- or a random-effects model).

Value

An object of class "list.rma". The object is a list containing the following components:

<code>estimate</code>	pooled estimates.
<code>se</code>	corresponding standard errors.
<code>zval</code>	corresponding test statistics.
<code>pval</code>	corresponding p-values.
<code>ci.lb</code>	lower bounds of the confidence intervals.
<code>ci.ub</code>	upper bounds of the confidence intervals.
<code>Q</code>	test statistics for the test of heterogeneity.
<code>Qp</code>	corresponding p-values.

tau2	estimated amounts of heterogeneity (only for random-effects models).
I2	values of I^2 .
H2	values of H^2 .

When the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="ad hoc"`, then `zval` is called `tval` in the object that is returned by the function.

The object is formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function.

Note

When using the `transf` option, the transformation is applied to the estimated coefficients and the corresponding interval bounds. The standard errors are then set equal to NA and are omitted from the printed output.

The variable specified via `cluster` is assumed to be of the same length as the data originally passed to the model fitting function (and if the `data` argument was used in the original model fit, then the variable will be searched for within this data frame first). Any subsetting and removal of studies with missing values that was applied during the model fitting is also automatically applied to the variable specified via the `cluster` argument.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/9781315>

Viechtbauer, W., & Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research Synthesis Methods*, **1**(2), 112–125. <https://doi.org/10.1002/jrsm.11>

See Also

`rma.uni`, `rma.mh`, and `rma.peto` for functions to fit models for which leave-one-out diagnostics can be computed.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### random-effects model
res <- rma(yi, vi, data=dat)

### leave-one-out analysis
leave1out(res)
leave1out(res, transf=exp)
```

```

### leave-one-out analysis with a cluster variable
leave1out(res, cluster=alloc)

### meta-analysis of the (log) risk ratios using the Mantel-Haenszel method
res <- rma.mh(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### leave-one-out analysis
leave1out(res)
leave1out(res, transf=exp)

### meta-analysis of the (log) odds ratios using Peto's method
res <- rma.peto(ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### leave-one-out analysis
leave1out(res)
leave1out(res, transf=exp)

```

llplot

*Plot of Likelihood Functions for Individual Studies***Description**

Function to plot the likelihood functions of the true effects given the study data.

Usage

```

llplot(measure, yi, vi, sei, ai, bi, ci, di, n1i, n2i, data, subset, drop00=TRUE,
       xvals=1000, xlim, ylim, xlab, ylab, scale=TRUE,
       lty, lwd, col, level=99.99, refline=0, ...)

```

Arguments

measure	a character string to specify for which effect size or outcome measure the likelihood functions should be drawn. See ‘Details’ for possible options and how the data should then be specified.
yi	vector with the effect size estimates.
vi	vector with the corresponding sampling variances.
sei	vector with the corresponding standard errors.
ai	vector to specify the 2×2 table frequencies (upper left cell).
bi	vector to specify the 2×2 table frequencies (upper right cell).
ci	vector to specify the 2×2 table frequencies (lower left cell).
di	vector to specify the 2×2 table frequencies (lower right cell).
n1i	vector to specify the group sizes or row totals (first group/row).
n2i	vector to specify the group sizes or row totals (second group/row).

<code>data</code>	optional data frame containing the variables given to the arguments above.
<code>subset</code>	optional (logical or numeric) vector to specify the subset of studies that should be included in the plot.
<code>drop00</code>	logical to specify whether studies with no cases (or only cases) in both groups should be dropped. See ‘Details’.
<code>xvals</code>	integer to specify for how many distinct values the likelihood should be evaluated.
<code>xlim</code>	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
<code>ylim</code>	y-axis limits. If unspecified, the function sets the y-axis limits to some sensible values.
<code>xlab</code>	title for the x-axis. If unspecified, the function sets an appropriate axis title.
<code>ylab</code>	title for the y-axis. If unspecified, the function sets an appropriate axis title.
<code>scale</code>	logical to specify whether the likelihood values should be scaled, so that the total area under each curve is (approximately) equal to 1.
<code>lty</code>	the line types (either a single value or a vector of length k). If unspecified, the function sets the line types according to some characteristics of the likelihood function. See ‘Details’.
<code>lwd</code>	the line widths (either a single value or a vector of length k). If unspecified, the function sets the widths according to the sampling variances (so that the line is thicker for more precise studies and vice-versa).
<code>col</code>	the line colors (either a single value or a vector of length k). If unspecified, the function uses various shades of gray according to the sampling variances (so that darker shades are used for more precise studies and vice-versa).
<code>level</code>	numeric value between 0 and 100 to specify the plotting limits for each likelihood function in terms of the confidence interval (the default is 99.99).
<code>refline</code>	numeric value to specify the location of the vertical ‘reference’ line (the default is 0). The line can be suppressed by setting this argument to NA.
<code>...</code>	other arguments.

Details

At the moment, the function only accepts `measure="GEN"` or `measure="OR"`.

For `measure="GEN"`, one must specify arguments `yi` for the effect size estimates and `vi` for the corresponding sampling variances (instead of specifying `vi`, one can specify the standard errors via the `sei` argument). The function then plots the likelihood function of the true effect based on a normal sampling distribution with the observed estimate as given by `yi` and variance as given by `vi` for each study.

For `measure="OR"`, one must specify arguments `ai`, `bi`, `ci`, and `di`, which denote the cell frequencies of the 2×2 tables. Alternatively, one can specify `ai`, `ci`, `n1i`, and `n2i`. See [escalc](#) function for more details. The function then plots the likelihood function of the true log odds ratio based on the non-central hypergeometric distribution for each 2×2 table. Since studies with no cases (or only cases) in both groups have a flat likelihood and are not informative about the odds ratio, they are dropped by default (i.e., `drop00=TRUE`) and are hence not drawn (if `drop00=FALSE`, these likelihoods are indicated by dotted lines). For studies that have a single zero count, the MLE of the odds ratio is infinite and these likelihoods are indicated by dashed lines.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

van Houwelingen, H. C., Zwinderman, K. H., & Stijnen, T. (1993). A bivariate approach to meta-analysis. *Statistics in Medicine*, **12**(24), 2273–2284. <https://doi.org/10.1002/sim.4780122405>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#) and [rma.glmm](#) for model fitting functions that are based on corresponding likelihood functions.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### draw likelihood functions
llplot(measure="GEN", yi=yi, vi=vi, data=dat, lwd=1, refline=NA, xlim=c(-3,2))

### create plot (Figure 2 in van Houwelingen, Zwinderman, & Stijnen, 1993)
llplot(measure="OR", ai=b.xci, nli=nci, ci=b.xti, n2i=nti, data=dat.collins1985a,
       lwd=1, refline=NA, xlim=c(-4,4), drop00=FALSE)
```

matreg

Fit Regression Models based on a Correlation and Covariance Matrix

Description

Function to fit regression models based on a correlation and covariance matrix.

Usage

```
matreg(y, x, R, n, V, cov=FALSE, means, ztor=FALSE,
       nearpd=FALSE, level=95, digits, ...)
```

Arguments

y	model formula or the index (given as a number) or name (given as a character string) of the outcome variable.
x	indices (given as a numeric vector) or names (given as a character vector) of the predictor variables. Ignored when y is a formula.
R	correlation or covariance matrix (or only the lower triangular part including the diagonal).

n	sample size based on which the elements in the correlation/covariance matrix were computed.
V	variance-covariance matrix of the lower triangular elements of the correlation/covariance matrix. Either V or n should be specified, not both. See ‘Details’.
cov	logical to specify whether R is a covariance matrix (the default is FALSE).
means	optional vector to specify the means of the variables (only relevant when cov=TRUE).
ztor	logical to specify whether R is a matrix of r-to-z transformed correlations and hence should be back-transformed to raw correlations (the default is FALSE). See ‘Details’.
nearpd	logical to specify whether the nearPD function from the Matrix package should be used when the $R_{x,x}$ matrix cannot be inverted. See ‘Note’.
level	numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see here for details).
digits	optional integer to specify the number of decimal places to which the printed results should be rounded.
...	other arguments.

Details

Let R be a $p \times p$ correlation or covariance matrix. Let y denote the row/column of the outcome variable and x the row(s)/column(s) of the predictor variable(s) in this matrix. Let m denote the length of x (i.e., the number of predictors). Let $R_{x,x}$ and $R_{x,y}$ denote the corresponding submatrices of R . Then

$$b = R_{x,x}^{-1} R_{x,y}$$

yields the standardized or raw regression coefficients (depending on whether R is a correlation or covariance matrix, respectively) when regressing the outcome variable on the predictor variable(s).

The y and x variables can be specified as character vectors (assuming that the matrix specified via R has corresponding row/column names) or as indices. One can also specify a model formula via argument y, giving the name of the outcome variable on the left-hand side and the name(s) of the predictor(s) on the right-hand side.

Regular R Matrix:

The R matrix may be computed based on a single sample of n subjects. In this case, one should specify the sample size via argument n. The variance-covariance matrix of the (standardized) regression coefficients is then given by $\text{Var}[b] = \hat{\sigma}^2 \times R_{x,x}^{-1}$, where $\hat{\sigma}^2 = (1 - b' R_{x,y})/\text{df}$ is the estimated error variance and df denotes the residual degrees of freedom (which are $n - m - 1$ when R is a covariance matrix and $n - m$ when R is a correlation matrix). The standard errors are then given by the square root of the diagonal elements of $\text{Var}[b]$. Test statistics (in this case, t-statistics) and the corresponding p-values can then be computed as in a regular regression analysis. When R is a covariance matrix, one should set cov=TRUE and specify the means of the p variables in the R matrix via argument means to obtain raw regression coefficients including the intercept and corresponding standard errors (when means is not specified, then the intercept estimate will be NA).

Meta-Analytic R Matrix:

Alternatively, R may be the result of a meta-analysis of correlation coefficients. In this case, the elements in R are pooled correlation coefficients and the variance-covariance matrix of these pooled coefficients should be specified via argument V . The order of elements in V should correspond to the order of elements in the lower triangular part of R column-wise. For example, if R is a 4×4 matrix of the form:

$$\begin{bmatrix} 1 & & & \\ r_{21} & 1 & & \\ r_{31} & r_{32} & 1 & \\ r_{41} & r_{42} & r_{43} & 1 \end{bmatrix}$$

then the elements are r_{21} , r_{31} , r_{41} , r_{32} , r_{42} , and r_{43} and hence V should be a 6×6 variance-covariance matrix of these elements in this order.

The standardized regression coefficients are still computed as described above, but the variance-covariance matrix of the standardized regression coefficients (i.e., $\text{Var}[b]$) is then computed as a function of V as described in Becker (1992) using the multivariate delta method. The standard errors are then again given by the square root of the diagonal elements of $\text{Var}[b]$. Test statistics (in this case, z-statistics) and the corresponding p-values can then be computed in the usual manner.

In case R is the result of a meta-analysis of Fisher r-to-z transformed correlation coefficients (and hence V is then the corresponding variance-covariance matrix of these pooled transformed coefficients), one should set argument `ztor=TRUE`, so that the appropriate back-transformation is then applied to R (and V) within the function before the standardized regression coefficients will be computed.

Finally, R may be a covariance matrix based on a meta-analysis (e.g., the estimated variance-covariance matrix of the random effects in a multivariate model). In this case, one should set `cov=TRUE` and V should again be the variance-covariance matrix of the elements in R , but now including the diagonal. Hence, if R is a 4×4 matrix of the form:

$$\begin{bmatrix} \tau_1^2 & & & \\ \tau_{21} & \tau_2^2 & & \\ \tau_{31} & \tau_{32} & \tau_3^2 & \\ \tau_{41} & \tau_{42} & \tau_{43} & \tau_4^2 \end{bmatrix}$$

then the elements are τ_1^2 , τ_{21} , τ_{31} , τ_{41} , τ_2^2 , τ_{32} , τ_{42} , τ_3^2 , τ_{43} , and τ_4^2 , and hence V should be a 10×10 variance-covariance matrix of these elements in this order. Argument `means` can then again be used to specify the means of the variables.

Value

An object of class "matreg". The object is a list containing the following components:

<code>tab</code>	a data frame with the estimated (standardized) regression coefficients, standard errors, test statistics, degrees of freedom (only for t-tests), p-values, and lower/upper confidence interval bounds.
<code>vb</code>	the variance-covariance matrix of the estimated model coefficients.
<code>...</code>	some additional elements/values.

The results are formatted and printed with the `print` function. Extractor functions include `coef`, `vcov`, `se`, `sigma`, `confint`, `logLik`, `deviance`, `AIC`, and `BIC` (some of these only work under the 'Regular R Matrix' case).

Note

Only the lower triangular part of R (and V if it is specified) is used in the computations.

If $R_{x,x}$ is not invertible, an error will be issued. In this case, one can set argument `nearpd=TRUE`, in which case the `nearPD` function from the **Matrix** package will be used to find the nearest positive semi-definite matrix, which should be invertible. The results should be treated with caution when this is done.

When R is a covariance matrix with V and means specified, the means are treated as known constants when estimating the standard error of the intercept.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Becker, B. J. (1992). Using results from replicated studies to estimate linear models. *Journal of Educational Statistics*, **17**(4), 341–362. <https://doi.org/10.3102/10769986017004341>

Becker, B. J. (1995). Corrections to "Using results from replicated studies to estimate linear models". *Journal of Educational and Behavioral Statistics*, **20**(1), 100–102. <https://doi.org/10.3102/10769986020001100>

Becker, B. J., & Aloe, A. (2019). Model-based meta-analysis and related approaches. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (3rd ed., pp. 339–363). New York: Russell Sage Foundation.

See Also

`rma.mv` for a function to meta-analyze multiple correlation coefficients that can be used to construct an R matrix.

`rcalc` for a function to construct the variance-covariance matrix of dependent correlation coefficients.

Examples

```
#####

### first an example unrelated to meta-analysis, simply demonstrating that
### one can obtain the same results from lm() and matreg()

### fit a regression model with lm() to the 'mtcars' dataset
res <- lm(mpg ~ hp + wt + am, data=mtcars)
summary(res)

### covariance matrix of the dataset
S <- cov(mtcars)

### fit the same regression model using matreg()
res <- matreg(mpg ~ hp + wt + am, R=S, cov=TRUE,
              means=colMeans(mtcars), n=nrow(mtcars))
summary(res)
```

```

### specify the outcome and predictors as character vectors
res <- matreg(y="mpg", x=c("hp","wt","am"), R=S, cov=TRUE,
             means=colMeans(mtcars), n=nrow(mtcars))
summary(res)

### specify the outcome and predictors as indices
res <- matreg(y=1, x=c(4,6,9), R=S, cov=TRUE,
             means=colMeans(mtcars), n=nrow(mtcars))
summary(res)

### copy the 'mtcars' dataset to 'dat' and standardize all variables
dat <- mtcars
dat[] <- scale(dat)

### fit a regression model with lm() to obtain standardized regression coefficients ('betas')
res <- lm(mpg ~ 0 + hp + wt + am, data=dat)
summary(res)

### correlation matrix of the dataset
R <- cor(mtcars)

### fit the same regression model using matreg()
res <- matreg(y="mpg", x=c("hp","wt","am"), R=R, n=nrow(mtcars))
summary(res)

### note: the standard errors of the betas should not be used to construct CIs
### as they assume that the null hypothesis ( $H_0: \beta_j = 0$ ) is true

### construct the var-cov matrix of correlations in R
V <- rcalc(R, ni=nrow(mtcars))$V

### fit the same regression model using matreg() but now supply V
res <- matreg(y="mpg", x=c("hp","wt","am"), R=R, V=V)
summary(res)

### the standard errors computed in this way can now be used to construct
### CIs for the betas (here, the difference is relatively small)

#####

### copy data into 'dat'
dat <- dat.craft2003

### construct dataset and var-cov matrix of the correlations
tmp <- rcalc(ri ~ var1 + var2 | study, ni=ni, data=dat)
V <- tmp$V
dat <- tmp$dat

### turn var1.var2 into a factor with the desired order of levels
dat$var1.var2 <- factor(dat$var1.var2,
  levels=c("acog.perf", "asom.perf", "conf.perf", "acog.asom", "acog.conf", "asom.conf"))

### multivariate random-effects model

```

```

res <- rma.mv(yi, V, mods = ~ 0 + var1.var2, random = ~ var1.var2 | study, struct="UN", data=dat)
res

### restructure estimated mean correlations into a 4x4 matrix
R <- vec2mat(coef(res))
rownames(R) <- colnames(R) <- c("perf", "acog", "asom", "conf")
round(R, digits=3)

### check that order in vcov(res) corresponds to order in R
round(vcov(res), digits=4)

### fit regression model with 'perf' as outcome and 'acog', 'asom', and 'conf' as predictors
matreg(1, 2:4, R=R, V=vcov(res))

### can also specify variable names
matreg("perf", c("acog", "asom", "conf"), R=R, V=vcov(res))

## Not run:
### repeat the above but with r-to-z transformed correlations
dat <- dat.craft2003
tmp <- rcalc(ri ~ var1 + var2 | study, ni=ni, data=dat, rtoz=TRUE)
V <- tmp$V
dat <- tmp$dat
dat$var1.var2 <- factor(dat$var1.var2,
  levels=c("acog.perf", "asom.perf", "conf.perf", "acog.asom", "acog.conf", "asom.conf"))
res <- rma.mv(yi, V, mods = ~ 0 + var1.var2, random = ~ var1.var2 | study, struct="UN", data=dat)
R <- vec2mat(coef(res))
rownames(R) <- colnames(R) <- c("perf", "acog", "asom", "conf")
matreg(1, 2:4, R=R, V=vcov(res), ztor=TRUE)

## End(Not run)

#####

### a different example based on van Houwelingen et al. (2002)

### create dataset in long format
dat.long <- to.long(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
  data=dat.colditz1994, append=FALSE)
dat.long <- escalc(measure="PLO", xi=out1, mi=out2, data=dat.long)
dat.long$group <- factor(dat.long$group, levels=c(2,1), labels=c("con", "exp"))
dat.long

### fit bivariate model
res <- rma.mv(yi, vi, mods = ~ 0 + group, random = ~ group | study, struct="UN",
  data=dat.long, method="ML")
res

### regression of log(odds)_exp on log(odds)_con
matreg(y=2, x=1, R=res$G, cov=TRUE, means=coef(res), n=res$g.levels.comb.k)

### but the SE of the 'con' coefficient is not computed correctly, since we treat res$G above as if
### it was a var-cov matrix computed from raw data based on res$g.levels.comb.k (= 13) data points

```

```
### fit bivariate model and get the var-cov matrix of the estimates in res$G
res <- rma.mv(yi, vi, mods = ~ 0 + group, random = ~ group | study, struct="UN",
             data=dat.long, method="ML", cvvc="varcov", control=list(nearpd=TRUE))

### now use res$vcv as the var-cov matrix of the estimates in res$G
matreg(y=2, x=1, R=res$G, cov=TRUE, means=coef(res), V=res$vcv)

#####
```

metafor.news

Read News File of the Metafor Package

Description

Function to read the ‘NEWS’ file of the [metafor-package](#).

Usage

```
metafor.news()
```

Details

The function is simply a wrapper for `news(package="metafor")` which parses and displays the ‘NEWS’ file of the package.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Examples

```
## Not run:
metafor.news()

## End(Not run)
```

methods.vcovmat	<i>Methods for 'vcovmat' Objects</i>
-----------------	--------------------------------------

Description

Methods for objects of class "vcovmat".

Usage

```
## S3 method for class 'vcovmat'
print(x, digits=4, tol, zero=".", na="", ...)

## S3 method for class 'vcovmat'
x[i, j, ...]
```

Arguments

x	an object of class "vcovmat".
digits	integer to specify the number of decimal places to which the printed results should be rounded (the default is 4).
tol	numeric value giving the tolerance for values that will be considered to be zeros.
zero	character string to represent zero values.
na	character string to represent missing values.
i, j	indices to select rows/columns.
...	other arguments.

Details

When printing a "vcovmat" object, values equal to zero are printed by default as a period. This makes it easier to see the structure of the variance-covariance matrix, which often has a block-diagonal structure. Values sufficiently close to zero are also treated as zero. This is controlled by the `tol` argument, which, if unspecified, is set by default to $10 \times .Machine$double.eps$.

Note

To turn a matrix `x` of class "vcovmat" into a regular matrix, just use `unclass(x)`.

Author(s)

The print method is based on the code from [print.table](#) with some minor tweaks.

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[vcalc](#) and [rcalc](#) for functions that create vcovmat objects.

mfopt

Getting and Setting Package Options

Description

Functions for getting and setting **metafor** package options.

Usage

```
getmfopt(x, default=NULL)
setmfopt(...)
```

Arguments

x	The name of an option. If unspecified, all options are returned.
default	value to return if the option name does not exist.
...	one or more option names and the corresponding values to which they should be set.

Details

The **metafor** package stores some of its options as a list element called "metafor" in the system options (see [options](#)). Hence, `getmfopt()` is the same as `getOption("metafor")`. One can also set x to the name of an option to return. With `setmfopt()`, one can set one or more options to their desired values.

Currently, the following options are supported:

`check` logical to specify whether a version check should be carried out when loading the package (the default is TRUE). See [here](#) for details. Obviously, this option must be set before loading the package (e.g., with `options(metafor=list(check=FALSE))`).

`silent` logical to specify whether a startup message should be issued when loading the package (the default is FALSE). Obviously, this option must be set before loading the package (e.g., with `options(metafor=list(silent=TRUE))`). Note that messages about required packages that are automatically loaded are not suppressed by this. To fully suppress all startup messages, load the package with [suppressPackageStartupMessages](#).

`space` logical to specify whether an empty line should be added before and after the output (the default is TRUE). See [here](#) for details.

`digits` a named vector to specify how various aspects of the output should be rounded (unset by default). See [here](#) for details.

`style` a list whose elements specify the styles for various parts of the output when the **crayon** package is loaded and a terminal is used that supports 'ANSI' color/highlight codes (unset by default). See [here](#) for details. Can also be a logical and set to FALSE to switch off output styling when the crayon package is loaded.

theme character string to specify how plots created by the package should be themed. The default is "default", which means that the default foreground and background colors of plotting devices are used. Alternative options are "light" and "dark", which forces plots to be drawn with a light or dark background, respectively. See [here](#) for further details. RStudio users can also set this to "auto", in which case plotting colors are chosen depending on the RStudio theme used (for some themes, using "auto2" might be visually more appealing). One can also use `setmfopt(theme="custom", fg=<color>, bg=<color>)` to set the foreground and background colors to custom choices (depending on the colors chosen, using "custom2" might be visually more appealing).

Value

Either a vector with the value for the chosen option or a list with all options.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Examples

```
getmfopt()
getmfopt(space)
setmfopt(space=FALSE)
getmfopt()
setmfopt(space=TRUE)
getmfopt()
```

Description

Books and articles about meta-analysis often describe and discuss the difference between the so-called ‘fixed-effects model’ and the ‘random-effects model’ (e.g., Cooper et al., 2009). The former term is (mostly) avoided throughout the documentation of the **metafor** package. The term ‘equal-effects model’ is used instead, since it more concretely describes the main assumption underlying this model (i.e., that the underlying true effects are homogeneous, or in other words, that they are all equal to each other). The terms ‘common-effect(s) model’ or ‘homogenous-effect(s) model’ have also sometimes been used in the literature to describe this model and are equally descriptive.

Moreover, the term ‘fixed-effects model’ creates a bit of a conundrum. When authors use this term, they are really typically referring to the equal-effects model. There is however another type of model, the ‘real’ fixed-effects model, that is different from the equal-effects model, but now we would need to invent (unnecessarily) a different term to refer to this model. Some have done

so or tried to make a distinction between the ‘fixed-effect model’ (without the s!) and the ‘fixed-effects model’, but this subtle difference in terminology is easily overlooked/missed. Using the term ‘equal-effects model’ avoids this confusion and is more informative.

However, the question then remains what the real fixed-effects model is all about. The purpose of this page is to describe this model and to contrast it with the well-known random-effects model.

Details

Fixed-Effects Model:

Assume we have a set of $i = 1, \dots, k$ independent studies and let y_i denote the effect size estimate in the i th study. Let θ_i denote the corresponding (unknown) true effect, such that

$$y_i \mid \theta_i \sim N(\theta_i, v_i).$$

In other words, the estimates are assumed to be unbiased and normally distributed estimates of the corresponding true effects with sampling variances equal to v_i . The v_i values are assumed to be known.

The fixed-effects model is simply given by

$$y_i = \theta_i + \varepsilon_i,$$

where the θ_i values are the (fixed) true effects of the k studies. Therefore, the model ‘conditions’ on the true effects and provides a *conditional inference* about the k studies included in the meta-analysis.

When using weighted estimation (the default in [rma.uni](#) when `method="FE"`), this implies that the fitted model provides an estimate of

$$\bar{\theta}_w = \frac{\sum_{i=1}^k w_i \theta_i}{\sum_{i=1}^k w_i},$$

that is, the *weighted average* of the true effects in the k studies, with weights equal to $w_i = 1/v_i$. As an example, consider the meta-analysis by Bangert-Drowns et al. (2004) on the effectiveness of writing-to-learn interventions on academic achievement. The dataset ([dat.bangertdrowns2004](#)) includes the observed standardized mean differences (variable `yi`) and the corresponding sampling variances (variable `vi`) of 48 studies that have examined such an intervention. We can fit a fixed-effects model to these data with:

```
# copy data into 'dat'
dat <- dat.bangertdrowns2004

# fit a fixed-effects model
res <- rma(yi, vi, data=dat, method="FE")
res

# Fixed-Effects Model (k = 48)
#
# I^2 (total heterogeneity / total variability): 56.12%
# H^2 (total variability / sampling variability): 2.28
#
```

```
# Test for Heterogeneity:
# Q(df = 47) = 107.1061, p-val < .0001
#
# Model Results:
#
# estimate      se      zval      pval      ci.lb      ci.ub
# 0.1656 0.0269 6.1499 <.0001 0.1128 0.2184
```

The Q-test suggests that the underlying true standardized mean differences are heterogeneous ($Q(df = 47) = 107.11, p < .0001$). Therefore, if we believe this to be true, then the value shown under estimate is an estimate of the inverse-variance weighted average of the true standardized mean differences of these 48 studies (i.e., $\hat{\theta}_w = 0.17$).

One can also employ an unweighted estimation method (by setting `weighted=FALSE` in [rma.uni](#)), which provides an estimate of the *unweighted average* of the true effects in the k studies, that is, an estimate of

$$\bar{\theta}_u = \frac{\sum_{i=1}^k \theta_i}{k}.$$

Returning to the example, we then find:

```
# fit a fixed-effects model using unweighted estimation
res <- rma(yi, vi, data=dat, method="FE", weighted=FALSE)
res

# Fixed-Effects Model (k = 48)
#
# I^2 (total heterogeneity / total variability): 56.12%
# H^2 (total variability / sampling variability): 2.28
#
# Test for Heterogeneity:
# Q(df = 47) = 107.1061, p-val < .0001
#
# Model Results:
#
# estimate      se      zval      pval      ci.lb      ci.ub
# 0.2598 0.0380 6.8366 <.0001 0.1853 0.3343
```

Therefore, the value shown under estimate is now an estimate of the unweighted average of the true standardized mean differences of these 48 studies (i.e., $\hat{\theta}_u = 0.26$).

For weighted estimation, one could also choose to estimate $\bar{\theta}_w$, where the w_i values are user-defined weights (via argument `weights` in [rma.uni](#)). Hence, using inverse-variance weights or unit weights (as in unweighted estimation) are just special cases. It is up to the user to decide to what extent $\bar{\theta}_w$ is a meaningful parameter to estimate (regardless of the weights used).

For example, we could use the sample sizes of the studies as weights:

```
# fit a fixed-effects model using the sample sizes as weights
res <- rma(yi, vi, data=dat, method="FE", weights=ni)
res

# Fixed-Effects Model (k = 48)
```

```
#
# I^2 (total heterogeneity / total variability): 56.12%
# H^2 (total variability / sampling variability): 2.28
#
# Test for Heterogeneity:
# Q(df = 47) = 107.1061, p-val < .0001
#
# Model Results:
#
# estimate      se      zval      pval      ci.lb      ci.ub
# 0.1719 0.0269 6.3802 <.0001 0.1191 0.2248
```

We therefore obtain an estimate of the sample-size weighted average of the true standardized mean differences of these 48 studies (i.e., $\hat{\theta}_w = 0.17$). Since the sample sizes and the inverse sampling variances are highly correlated (`cor(dat$ni, 1/dat$vi)` yields 0.999), the results are almost identical to the ones we obtained earlier using inverse-variance weighting.

Random-Effects Model:

The random-effects model does not condition on the true effects. Instead, the k studies included in the meta-analysis are assumed to be a random sample from a larger population of studies. In rare cases, the studies included in a meta-analysis are actually sampled from a larger collection of studies. More typically, all efforts have been made to find and include all relevant studies providing evidence about the phenomenon of interest and hence the population of studies is a hypothetical population of an essentially infinite set of studies comprising all of the studies that have been conducted, that could have been conducted, or that may be conducted in the future. We assume that $\theta_i \sim N(\mu, \tau^2)$, that is, the true effects in the population of studies are normally distributed with μ denoting the average true effect and τ^2 the variance of the true effects in the population (τ^2 is therefore often referred to as the amount of ‘heterogeneity’ in the true effects). The random-effects model can also be written as

$$y_i = \mu + u_i + \varepsilon_i,$$

where $u_i \sim N(0, \tau^2)$ and $\varepsilon_i \sim N(0, v_i)$. The fitted model provides estimates of μ and τ^2 . Consequently, the random-effects model provides an *unconditional inference* about the average true effect in the population of studies (from which the k studies included in the meta-analysis are assumed to be a random sample).

Fitting a random-effects model to the example data yields:

```
# fit a random-effects model (note: method="REML" is the default)
res <- rma(yi, vi, data=dat)
res

# Random-Effects Model (k = 48; tau^2 estimator: REML)
#
# tau^2 (estimated amount of total heterogeneity): 0.0499 (SE = 0.0197)
# tau (square root of estimated tau^2 value):      0.2235
# I^2 (total heterogeneity / total variability): 58.37%
# H^2 (total variability / sampling variability): 2.40
#
# Test for Heterogeneity:
```

```
# Q(df = 47) = 107.1061, p-val < .0001
#
# Model Results:
#
# estimate      se      zval      pval      ci.lb      ci.ub
# 0.2219 0.0460 4.8209 <.0001 0.1317 0.3122
```

The value shown under estimate is now an estimate of the average true standardized mean difference of studies in the population of studies from which the 48 studies included in this dataset have come (i.e., $\hat{\mu} = 0.22$).

When using weighted estimation in the context of a random-effects model, the model is fitted with weights equal to $w_i = 1/(\tau^2 + v_i)$, with τ^2 replaced by its estimate (the default in rma.uni when method is set to one of the possible choices for estimating τ^2). One can also choose unweighted estimation in the context of the random-effects model (weighted=FALSE) or specify user-defined weights (via weights), although the parameter that is estimated (i.e., μ) remains the same regardless of the estimation method and weights used (as opposed to the fixed-effect model, where the parameter estimated is different for weighted versus unweighted estimation or when using different weights than the standard inverse-variance weights). Since weighted estimation with inverse-variance weights is most efficient, it is usually to be preferred for random-effects models (while in the fixed-effect model case, we must carefully consider whether $\bar{\theta}_w$ or $\bar{\theta}_u$ is the more meaningful parameter to estimate).

Conditional versus Unconditional Inferences:

Contrary to what is often stated in the literature, it is important to realize that the fixed-effects model does *not* assume that the true effects are homogeneous (i.e., that θ_i is equal to some common value θ in all k studies). In other words, the fixed-effects model provides perfectly valid inferences under heterogeneity, as long as one is restricting these inferences to the set of studies included in the meta-analysis and one realizes that the model does not provide an estimate of θ or μ , but of $\bar{\theta}_w$ or $\bar{\theta}_u$ (depending on the estimation method used).

However, such inferences are conditional on the included studies. It is therefore not permissible to generalize those inferences beyond the set of studies included in a meta-analysis (or doing so requires ‘extra-statistical’ arguments). In contrast, a random-effects model provides unconditional inferences and therefore allows a generalization beyond the set of included studies, although the population of studies to which we can generalize is typically only vaguely defined (since the included studies are not a proper random sample from a specified sampling frame). Instead, we simply must assume that the included studies are a representative sample of *some* population and it is to that population to which we are generalizing.

Leaving aside this issue, the above implies that there is nothing wrong with fitting both the fixed- and random-effects models to the same data, since these models address inherently different questions (i.e., what was the average effect in the studies that have been conducted and are included in this meta-analysis versus what is the average effect in the larger population of studies?).

Equal-Effects Model:

In the special case that the true effects are actually homogeneous (the equal-effects case), the distinction between the fixed- and random-effects models disappears, since homogeneity implies that $\mu = \bar{\theta}_w = \bar{\theta}_u \equiv \theta$. Therefore, if one believes that the true effects are homogeneous, then one can fit an equal-effects model (using weighted estimation), since this will provide the most efficient estimate of θ (note that if the true effects are really homogeneous but we fit a random-

effects model, it can happen that the estimate of τ^2 is actually larger than 0, which then leads to a loss of efficiency).

However, since there is no infallible method to test whether the true effects are really homogeneous or not, a researcher should decide on the type of inference desired before examining the data and choose the model accordingly.

Note that fitting an equal-effects model (with `method="EE"`) yields the exact same output as fitting a fixed-effects model, since the equations used to fit these two models are identical. However, the interpretation of the results is different. If we fit an equal-effects model, we make the assumption that the true effects are homogeneous and, if we believe this assumption to be justified, can interpret the estimate as an estimate of *the* true effect. On the other hand, if we reject the homogeneity assumption, then we should reject the model altogether. In contrast, if we fit a fixed-effects model, we do not assume homogeneity and instead interpret the estimate as an estimate of the (weighted) average true effect of the included studies.

For further discussions of the distinction between the equal-, fixed-, and random-effects models, see Laird and Mosteller (1990) and Hedges and Vevea (1998).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Cooper, H., Hedges, L. V., & Valentine, J. C. (Eds.) (2009). *The handbook of research synthesis and meta-analysis* (2nd ed.). New York: Russell Sage Foundation.

Hedges, L. V., & Vevea, J. L. (1998). Fixed- and random-effects models in meta-analysis. *Psychological Methods*, 3(4), 486–504. <https://doi.org/10.1037/1082-989X.3.4.486>

Laird, N. M., & Mosteller, F. (1990). Some statistical methods for combining experimental results. *International Journal of Technology Assessment in Health Care*, 6(1), 5–30. <https://doi.org/10.1017/S02664623000089>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, 36(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

misc-options

Miscellaneous Options and Features

Description

This page documents some miscellaneous options and features that do not fit very well elsewhere.

Details

Specifying the Confidence Level:

Several functions in the **metafor** package have a `level` argument for specifying the confidence level when calculating confidence (and prediction) intervals. The default is to use a 95% level throughout the package by convention. Note that values ≥ 1 are treated as coverage percentages, values between 0.5 and 1 as coverage proportions, and values below 0.5 as (two-sided) alpha values, so `level=95` is the same as `level=0.95` and `level=0.05` (but `level=0` is always treated as a 0% confidence level).

Controlling the Number of Digits in the Output:

Many functions in the **metafor** package have a `digits` argument, which can be used to control the number of digits that are displayed in the output when printing numeric values. For more control over the displayed output, one can set this argument to a named vector of the form:

```
digits=c(est=2, se=3, test=2, pval=3, ci=2, var=3, sevar=3, fit=3, het=3)
```

where the elements control the displayed number of digits for various aspects of the output, namely:

- `est` for estimates (e.g., effect size estimates, model coefficients, predicted values),
- `se` for standard errors,
- `test` for test statistics,
- `pval` for p-values,
- `ci` for confidence/prediction interval bounds,
- `var` for sampling variances and variance components,
- `sevar` for standard errors thereof,
- `fit` for fit statistics,
- `het` for heterogeneity statistics.

Instead of setting this argument in each function call, one can use `setmfopt(digits = ...)` to set the desired number of digits for the various elements (see `mfopt` for getting and setting package options). For example, `setmfopt(digits = c(est=2, se=3, test=2, pval=3, ci=2, var=3, sevar=3, fit=3, het=3))` could be a sensible choice when analyzing various types of standardized effect size measures.

Styled Output with the crayon Package:

The **crayon** package provides a way to create colored output. The **metafor** package is designed to automatically make use of this feature when the crayon package is installed (`install.packages("crayon")`) and loaded (`library(crayon)`). Note that this only works on terminals that support ‘ANSI’ color/highlight codes (e.g., not under RGui on Windows or R.app on macOS, but the RStudio console and all modern terminals should support this).

The default color style that is used is quite plain, but should work with a light or dark colored background. One can modify the color style with `setmfopt(style = ...)`, where `...` is a list whose elements specify the styles for various parts of the output (see below for some examples and the documentation of the crayon package for the syntax to specify styles). The following elements are recognized:

- `header` for the header of tables (underlined by default),
- `body1` for odd numbered rows in the body of tables,
- `body2` for even numbered rows in the body of tables,
- `na` for missing values in tables,
- `section` for section headers (bold by default),
- `text` for descriptive text in the output,
- `result` for the corresponding result(s),
- `stop` for errors (bold red by default),
- `warning` for warnings (yellow by default),
- `message` for messages (green by default),

- verbose for the text in verbose output (cyan by default),
- legend for legends (gray by default).

Elements not specified are styled according to their defaults. For example, one could use:

```
setmfont(style = list(header = combine_styles("gray20", "underline"),
                      body1  = make_style("gray40"),
                      body2  = make_style("gray40"),
                      na     = bold,
                      section = combine_styles("gray15", "bold"),
                      text   = make_style("gray50"),
                      result  = make_style("gray30"),
                      legend  = make_style("gray70")))
```

or

```
setmfont(style = list(header = combine_styles("gray80", "underline"),
                      body1  = make_style("gray60"),
                      body2  = make_style("gray60"),
                      na     = bold,
                      section = combine_styles("gray85", "bold"),
                      text   = make_style("gray50"),
                      result  = make_style("gray70"),
                      legend  = make_style("gray30")))
```

for a light or dark colored background, respectively. A slightly more colorful style could be:

```
setmfont(style = list(header = combine_styles("snow", make_style("royalblue4", bg=TRUE)),
                      body1  = combine_styles("gray10", make_style("gray95", bg=TRUE)),
                      body2  = combine_styles("gray10", make_style("gray85", bg=TRUE)),
                      na     = combine_styles("orange4", "bold"),
                      section = combine_styles("black", "bold", make_style("gray90", bg=TRUE)),
                      text   = make_style("gray40"),
                      result  = make_style("blue"),
                      legend  = make_style("gray70")))
```

or

```
setmfont(style = list(header = combine_styles("snow", make_style("royalblue4", bg=TRUE)),
                      body1  = combine_styles("gray90", make_style("gray10", bg=TRUE)),
                      body2  = combine_styles("gray90", make_style("gray15", bg=TRUE)),
                      na     = combine_styles("orange1", "bold"),
                      section = combine_styles("snow", "bold", make_style("gray10", bg=TRUE)),
                      text   = make_style("gray60"),
                      result  = make_style("steelblue1"),
                      legend  = make_style("gray30")))
```

for a light and dark colored background, respectively.

The following code snippet includes all output elements (except for an error) and can be used to test out a chosen color style:

```
# calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg,
              ci=cpos, di=cneg, data=dat.bcg)
dat$yi[1] <- NA # set one estimate to missing so we get a warning below
dat

# fit random-effects model
res <- rma(yi, vi, mods = ~ ablat, data=dat, verbose=3)
summary(res)
```

Note that support for 256 different colors and text formatting (such as underlined and bold text) differs across terminals.

To switch off output styling when the crayon package is loaded, use `setmfopt(style=FALSE)`.

Removing Empty Lines Before and After the Output:

When printing output, an empty line is usually added before and after the output. For more compact output, this can be suppressed with `setmfopt(space=FALSE)` (see [mfopt](#) for getting and setting package options). For example, running the following code:

```
# calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg,
              ci=cpos, di=cneg, data=dat.bcg)

# fit a random-effects model
res <- rma(yi, vi, data=dat)
res

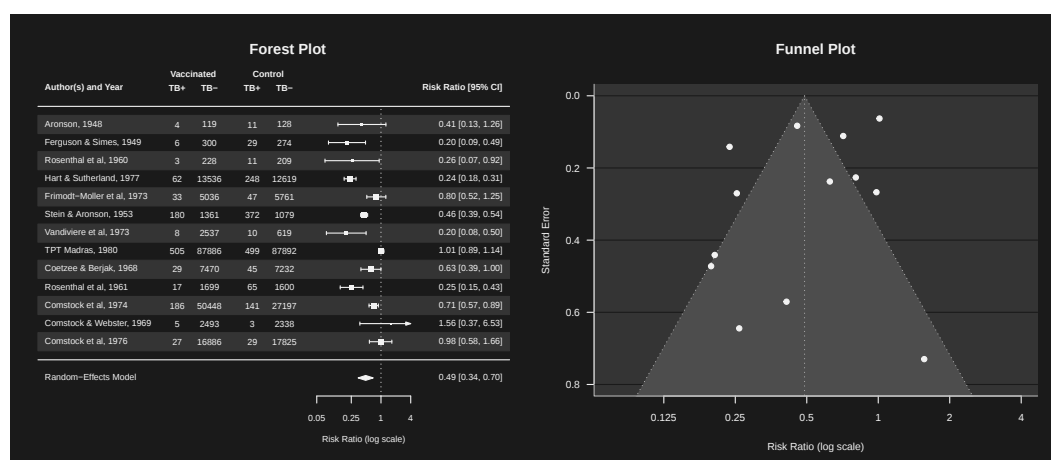
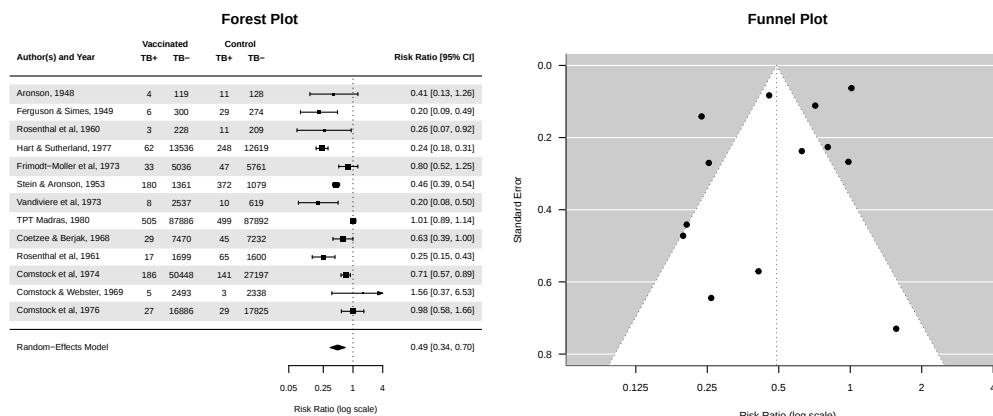
setmfopt(space=FALSE)
res
```

shows the difference.

Dark Mode for Plots:

By default, plots created in R have a white background and use black (and other darker colors) as the plotting color. Figures created with the **metafor** package also adhere to this standard. However, all plotting functions in the package are designed in such a way that switching to a dark background is easily possible. For this, one should set the canvas/figure background to a dark color (e.g., "black" or "gray10") and the foreground color to some bright color (e.g., "gray90", "gray95", or "white"). This can be easily accomplished with `setmfopt(theme="custom", fg="gray95", bg="gray10")` (see [mfopt](#) for getting and setting package options).

Figures that make use of additional colors for various plot elements will by default then use colors that are compatible with the chosen background. For example, the following two figures illustrate the difference between the two styles:



By setting `setmfont(theme="dark")`, all plots created by the package will automatically use a dark mode. RStudio users can also set `setmfont(theme="auto")`, in which case plotting colors are chosen depending on the RStudio theme used (for some themes, setting this to "auto2" might be visually more appealing).

Version Check:

When loading the **metafor** package in an [interactive](#) session, an automatic check is carried out to compare the version number of the installed package with the one available on [CRAN](#). If the installed version is older than the one available on CRAN, the user is notified that a new version is available. This check can be suppressed by setting the environment variable `METAFOR_VERSION_CHECK` to `FALSE` (e.g., with `Sys.setenv(METAFOR_VERSION_CHECK=FALSE)`) or with `options(metafor=list(check=FALSE))` before loading the package (see [mfont](#) for getting and setting package options).

By setting the environment variable to "devel" (e.g., with `Sys.setenv(METAFOR_VERSION_CHECK="devel")`) or with `options(metafor=list(check="devel"))`, the version check is run against the 'development version' of the package available on [GitHub](#).

Model Fitting / Processing Time:

The various model fitting functions (i.e., [rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), [rma.mv](#), and [selmodel](#)) and various other functions (e.g., [confint](#), [cumul](#), [leave1out](#), [profile](#), [rstudent](#))

automatically keep track of the model fitting / processing time. This information is stored as element `time` (in seconds) in the object that is returned. One can also use argument `time=TRUE` to nicely print this information. For example:

```
# fit multilevel mixed-effects meta-regression model and print the processing time
res <- rma.mv(yi, vi, mods = ~ condition,
             random = list(~ 1 | article/experiment/sample/id, ~ 1 | pairing),
             data=dat.mccurdy2020, sparse=TRUE, digits=3, time=TRUE)
```

```
# extract the processing time (should take somewhere around 10-20 seconds on a modern CPU)
res$time
```

Model Object Sizes:

The objects returned by model fitting functions like `rma.uni`, `rma.mh`, `rma.peto`, `rma.glmm`, and `rma.mv` contain information that is needed by some of the method functions that can be applied to such objects, but that can lead to objects that are relatively large in size. As an example, the model objects that are created as part of the example code for `dat.moura2021` are approximately 120MB in size. To reduce the object size, one can make use of the (undocumented) argument `outlist`. When setting `outlist="minimal"`, the resulting object contains only the minimal information needed to print the object (which results in an object that is around 13KB in size). Alternatively, one can set `outlist` to a string that specifies what objects that are created within the model fitting function should be returned (and under which name). For example, `outlist="coef=beta, vcov=vcov"` would indicate that only the model coefficient(s) (with name `coef`) and the corresponding variance-covariance matrix (with name `vcov`) should be returned (the resulting object then is only around 2KB in size). Note that this requires knowledge of how objects within the model fitting function are named, so inspection of the source code of a function will then be necessary. Also, there is no guarantee that method functions will still work when including only a subset of the information that is typically stored in model objects.

Load Balancing:

Several functions in the **metafor** package can make use of parallel processing (e.g., `profile`) to speed up intensive computations on machines with multiple cores. When using `parallel="snow"`, the default is to use the `parLapply` function from the **parallel** package for this purpose. In some cases (especially when the parallelized computations take up quite variable amounts of time to complete), using ‘load balancing’ may help to speed things up further (by using the `parLapplyLB` function). This can be enabled with `pbapply::pboptions(use_lb=TRUE)` before running the function that makes use of parallel processing. Whether this really does speed things up depends on many factors and is hard to predict.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Description

This page documents some recommended practices when working with the **metafor** package (and more generally when conducting meta-analyses).

Details

Restricted Maximum Likelihood Estimation:

When fitting models with the `rma.uni` and `rma.mv` functions, use of restricted maximum likelihood (REML) estimation is generally recommended. This is also the default setting (i.e., `method="REML"`). Various simulation studies have indicated that REML estimation tends to provide approximately unbiased estimates of the amount of heterogeneity (e.g., Langan et al., 2019; Veroniki et al., 2016; Viechtbauer, 2005), or more generally, of the variance components in more complex mixed-effects models (Harville, 1977).

For models fitted with the `rma.uni` function, the empirical Bayes / Paule-Mandel estimators (i.e., `method="EB"` / `method="PM"`), which can actually be shown to be identical to each other despite their different derivations (Viechtbauer et al., 2015), also have some favorable properties. However, these estimators do not generalize in a straightforward manner to more complex models, such as those that can be fitted with the `rma.mv` function.

Improved Inference Methods:

When fitting models with the `rma.uni` function, tests of individual model coefficients and the corresponding confidence intervals are by default (i.e., when `test="z"`) based on a standard normal distribution, while the omnibus test is based on a chi-square distribution. These inference methods may not perform nominally (i.e., the Type I error rate of tests and the coverage rate of confidence intervals may deviate from the chosen level), especially when the number of studies, k , is low. Therefore, it is highly recommended to use the method by Hartung (1999), Sidik and Jonkman (2002), and Knapp and Hartung (2003) (the Knapp-Hartung method; also referred to as the Hartung-Knapp-Sidik-Jonkman method) by setting `test="knha"` (or equivalently, `test="hksj"`). Then tests of individual coefficients and confidence intervals are based on a t-distribution with $k - p$ degrees of freedom, while the omnibus test then uses an F-distribution with m and $k - p$ degrees of freedom (with m denoting the number of coefficients tested and p the total number of model coefficients). Various simulation studies have shown that this method works very well in providing tests and confidence intervals with close to nominal performance (e.g., Sánchez-Meca & Marín-Martínez, 2008; Viechtbauer et al., 2015).

Alternatively, one can also conduct permutation tests using the `permtest` function. These also perform very well (and are, in a certain sense, ‘exact’ tests), but are computationally expensive.

For models fitted with the `rma.mv` and `rma.glmm` functions, the Knapp-Hartung method and permutation tests are not available. Instead, one can set `test="t"` to also use t- and F-distributions for making inferences (although this does not involve the adjustment to the standard errors of the estimated model coefficients that is made as part of the Knapp-Hartung method). For `rma.mv`, one should also set `dfs="contain"`, which uses an improved method for approximating the degrees of freedom of the t- and F-distributions.

Note that `test="z"` is the default for the `rma.uni`, `rma.mv`, and the `rma.glmm` functions. While the improved inference methods described above should ideally be the default, changing this now would break backwards compatibility.

General Workflow for Meta-Analyses Involving Complex Dependency Structures:

Many meta-analyses involve effect size estimates that cannot be assumed to be independent, because some estimates were computed based on the same sample of subjects (or at least a partially overlapping set). In this case, one should compute the covariances for any pair of estimates that involve (fully or partially) overlapping subjects. Doing so is difficult, but we can often construct an approximate variance-covariance matrix (say V) of such dependent estimates. This can be done with the `vcalc` function (and/or see the `rcalc` function when dealing specifically with dependent correlation coefficients). We can then fit a multivariate/multilevel model to the estimates with the `rma.mv` function, using V as the approximate var-cov matrix of the estimates and adding fixed and random effects to the model as deemed necessary. However, since V is often only a rough approximation (and since the random effects structure may not fully capture all dependencies in the underlying true effects), we can then apply cluster-robust inference methods (also known as robust variance estimation) to the model. This can be done with the `robust` function, which also interfaces with the improved inference methods implemented in the `clubSandwich` package to obtain the cluster-robust tests and confidence intervals.¹ Finally, we can compute predicted effects (with corresponding confidence intervals) and test sets of coefficients or linear combinations thereof using the `predict` and `anova` functions. See Pustejovsky and Tipton (2022) for a paper describing such a workflow for various cases.

To summarize, the general workflow therefore will often consist of these steps:

```
# construct/approximate the var-cov matrix of dependent estimates
V <- vcalc(...)

# fit a multivariate/multilevel model with appropriate fixed/random effects
res <- rma.mv(yi, V, mods = ~ ..., random = ~ ...)

# apply cluster-robust inference methods (robust variance estimation)
# note: use the improved methods from the clubSandwich package
sav <- robust(res, cluster = ..., clubSandwich = TRUE)
sav

# compute predicted effects (with corresponding CIs) as needed
predict(sav, ...)

# test sets of coefficients / linear combinations as needed
anova(sav, ...)
```

How `vcalc` and `rma.mv` should be used (and the clustering variable specified for `robust`) will depend on the specifics of the application.

See [dat.assink2016](#), [dat.knapp2017](#), and [dat.tannersmith2016](#) for some examples illustrating this workflow.

Profile Likelihood Plots to Check Parameter Identifiability:

When fitting complex models, it is not guaranteed that all parameters of the model are identifiable (i.e., that there is a unique set of values for the parameters that maximizes the (restricted)

likelihood function). For models fitted with the `rma.mv` function, this pertains especially to the variance/correlation components of the model (i.e., what is specified via the `random` argument). Therefore, it is strongly advised in general to do post model fitting checks to make sure that the likelihood surface around the ML/REML estimates is not flat for some combination of the parameter estimates (which would imply that the estimates are essentially arbitrary). For example, one can plot the (restricted) log-likelihood as a function of each variance/correlation component in the model to make sure that each profile plot shows a clear peak at the corresponding ML/REML estimate. The `profile` function can be used for this purpose. See also Raue et al. (2009) for some further discussion of parameter identifiability and the use of profile likelihoods to check for this.

The `profile` function should also be used after fitting location-scale models (Viechtbauer & López-López, 2022) with the `rma.uni` function and after fitting selection models with the `selmodel` function.

¹ In small meta-analyses, the (denominator) degrees of freedom for the approximate t- and F-tests provided by the cluster-robust inference methods might be very low, in which case the tests may not be trustworthy and overly conservative (Joshi et al., 2022). Under these circumstances, one can consider the use of cluster wild bootstrapping (as implemented in the `wildmeta` package) as an alternative method for making inferences.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Hartung, J. (1999). An alternative method for meta-analysis. *Biometrical Journal*, **41**(8), 901–916. [https://doi.org/10.1002/\(SICI\)1521-4036\(199912\)41:8<901::AID-BIMJ901>3.0.CO;2-W](https://doi.org/10.1002/(SICI)1521-4036(199912)41:8<901::AID-BIMJ901>3.0.CO;2-W)
- Harville, D. A. (1977). Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, **72**(358), 320–338. <https://doi.org/10.2307/2286796>
- Joshi, M., Pustejovsky, J. E., & Beretvas, S. N. (2022). Cluster wild bootstrapping to handle dependent effect sizes in meta-analysis with a small number of studies. *Research Synthesis Methods*, **13**(4), 457–477. <https://doi.org/10.1002/jrsm.1554>
- Knapp, G., & Hartung, J. (2003). Improved tests for a random effects meta-regression with a single covariate. *Statistics in Medicine*, **22**(17), 2693–2710. <https://doi.org/10.1002/sim.1482>
- Langan, D., Higgins, J. P. T., Jackson, D., Bowden, J., Veroniki, A. A., Kontopantelis, E., Viechtbauer, W. & Simmonds, M. (2019). A comparison of heterogeneity variance estimators in simulated random-effects meta-analyses. *Research Synthesis Methods*, **10**(1), 83–98. <https://doi.org/10.1002/jrsm.1316>
- Pustejovsky, J. E. & Tipton, E. (2022). Meta-analysis with robust variance estimation: Expanding the range of working models. *Prevention Science*, **23**, 425–438. <https://doi.org/10.1007/s11121-021-01246-3>
- Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., & Timmer, J. (2009). Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, **25**(15), 1923–1929. <https://doi.org/10.1093/bioinformatics/btp355>
- Sánchez-Meca, J. & Marín-Martínez, F. (2008). Confidence intervals for the overall effect size in random-effects meta-analysis. *Psychological Methods*, **13**(1), 31–48. <https://doi.org/10.1037/1082-989x.13.1.31>

Sidik, K. & Jonkman, J. N. (2002). A simple confidence interval for meta-analysis. *Statistics in Medicine*, **21**(21), 3153–3159. <https://doi.org/10.1002/sim.1262>

Veroniki, A. A., Jackson, D., Viechtbauer, W., Bender, R., Bowden, J., Knapp, G., Kuss, O., Higgins, J. P., Langan, D., & Salanti, G. (2016). Methods to estimate the between-study variance and its uncertainty in meta-analysis. *Research Synthesis Methods*, **7**(1), 55–79. <https://doi.org/10.1002/jrsm.1164>

Viechtbauer, W. (2005). Bias and efficiency of meta-analytic variance estimators in the random-effects model. *Journal of Educational and Behavioral Statistics*, **30**(3), 261–293. <https://doi.org/10.3102/10769986030>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W., López-López, J. A., Sánchez-Meca, J., & Marín-Martínez, F. (2015). A comparison of procedures to test for moderators in mixed-effects meta-regression models. *Psychological Methods*, **20**(3), 360–374. <https://doi.org/10.1037/met0000023>

Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*. **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>

model.matrix.rma

Extract the Model Matrix from 'rma' Objects

Description

Function to extract the model matrix from objects of class "rma".

Usage

```
## S3 method for class 'rma'
model.matrix(object, asdf, ...)
```

Arguments

object	an object of class "rma".
asdf	logical to specify whether the model matrix should be turned into a data frame (the default is FALSE).
...	other arguments.

Value

The model matrix.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which a model matrix can be extracted.

[fitted](#) for a function to extract the fitted values.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### extract the model matrix
model.matrix(res)
```

pairmat	<i>Construct a Pairwise Contrast Matrix for 'rma' Objects</i>
---------	---

Description

Functions to construct a matrix of pairwise contrasts for objects of class "rma".

Usage

```
pairmat(x, btt, btt2, ...)
```

Arguments

x	an object of class "rma".
btt	vector of indices to specify for which coefficients pairwise contrasts should be constructed. Can also be a string to grep for. See 'Details'.
btt2	optional argument to specify a second set of coefficients that should also be included in the contrast matrix.
...	other arguments.

Value

When a meta-regression model includes a categorical moderator variable (i.e., a factor), there is often interest in testing whether the coefficients representing the various levels of the factor differ significantly from each other. The present function constructs the pairwise contrast matrix between all factor levels for a particular factor, which can be used together with the [anova](#) function to carry out such tests and the [predict](#) function to obtain corresponding confidence intervals.

The x argument is used to specify a meta-regression model and the btt argument the indices of the coefficients for which pairwise contrasts should be constructed. For example, with btt=2:4, contrasts are formed based on the second, third, and fourth coefficient of the model. Instead of

specifying the coefficient numbers, one can specify a string for `btt`. In that case, `grep` will be used to search for all coefficient names that match the string.

At times, it may be useful to include a second set of coefficients in the contrast matrix (not as pairwise contrasts, but as ‘main effects’). This can be done via the `btt2` argument.

When using the present function in a call to the `anova` or `predict` functions, argument `x` does not need to be specified, as the function will then automatically construct the contrast matrix based on the model object passed to the `anova` or `predict` function. See below for examples.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`rma.uni`, `rma.glmm`, and `rma.mv` for functions to fit meta-regression models for which pairwise contrasts may be useful.

`anova` for a function to carry out tests of the pairwise contrasts and `predict` to obtain corresponding confidence/prediction intervals.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### mixed-effects meta-regression model with the allocation method as a moderator;
### by removing the intercept term, we obtain the estimated average effect for each
### factor level from the model
res <- rma(yi, vi, mods = ~ 0 + alloc, data=dat)
res

### construct the contrast matrix for the 'alloc' factor
pairmat(res, btt=1:3)
pairmat(res, btt="alloc")

### test all pairwise contrasts
anova(res, X=pairmat(btt=1:3))
anova(res, X=pairmat(btt="alloc"))

### obtain the corresponding confidence intervals
predict(res, newmods=pairmat(btt="alloc"))

### test all pairwise contrasts adjusting for multiple testing
anova(res, X=pairmat(btt="alloc"), adjust="bonf")

### fit the same model, but including the intercept term; then 'alternate' is the
```

```

### reference level and the coefficients for 'random' and 'systematic' already
### represent pairwise contrasts with this reference level
res <- rma(yi, vi, mods = ~ alloc, data=dat)
res

### in this case, we want to include these coefficients directly in the contrast
### matrix (btt2=2:3) but also include the pairwise contrast between them (btt=2:3)
pairmat(res, btt=2:3, btt2=2:3)
pairmat(res, btt="alloc", btt2="alloc")

### test all pairwise contrasts
anova(res, X=pairmat(btt=2:3, btt2=2:3))
anova(res, X=pairmat(btt="alloc", btt2="alloc"))

### obtain the corresponding confidence intervals
predict(res, newmods=pairmat(btt="alloc", btt2="alloc"))

### meta-regression model with 'ablat' and 'alloc' as moderators
res <- rma(yi, vi, mods = ~ ablat + alloc, data=dat)
res

### test all pairwise contrasts between the 'alloc' levels (while controlling for 'ablat')
anova(res, X=pairmat(btt="alloc", btt2="alloc"))
anova(res, X=pairmat(btt="alloc", btt2="alloc"))

### obtain the corresponding confidence intervals
predict(res, newmods=pairmat(btt="alloc", btt2="alloc"))

### an example of a meta-regression model with more factors levels
dat <- dat.bangertdrowns2004
res <- rma(yi, vi, mods = ~ 0 + factor(grade), data=dat)
res

### test all pairwise contrasts between the 'grade' levels
anova(res, X=pairmat(btt="grade"))

### obtain the corresponding confidence intervals
predict(res, newmods=pairmat(btt="grade"))

### test all pairwise contrasts adjusting for multiple testing
anova(res, X=pairmat(btt="grade"), adjust="bonf")

```

Description

Function to carry out permutation tests for objects of class "rma.uni" and "rma.ls".

Usage

```
permutest(x, ...)

## S3 method for class 'rma.uni'
permutest(x, exact=FALSE, iter=1000, btt=x$btt,
          permci=FALSE, progbar=TRUE, digits, control, ...)

## S3 method for class 'rma.ls'
permutest(x, exact=FALSE, iter=1000, btt=x$btt, att=x$att,
          progbar=TRUE, digits, control, ...)
```

Arguments

<code>x</code>	an object of class "rma.uni" or "rma.ls".
<code>exact</code>	logical to specify whether an exact permutation test should be carried out (the default is FALSE). See 'Details'.
<code>iter</code>	integer to specify the number of iterations for the permutation test when not doing an exact test (the default is 1000).
<code>btt</code>	optional vector of indices (or list thereof) to specify which coefficients should be included in the Wald-type test. Can also be a string to grep for.
<code>att</code>	optional vector of indices (or list thereof) to specify which scale coefficients should be included in the Wald-type test. Can also be a string to grep for.
<code>permci</code>	logical to specify whether permutation-based confidence intervals (CIs) should also be constructed (the default is FALSE). Can also be a vector of indices to specify for which coefficients a permutation-based CI should be obtained.
<code>progbar</code>	logical to specify whether a progress bar should be shown (the default is TRUE).
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
<code>control</code>	list of control values for numerical comparisons (<code>comptol</code>) and for uniroot (i.e., <code>tol</code> and <code>maxiter</code>). The latter is only relevant when <code>permci=TRUE</code> . See 'Note'.
<code>...</code>	other arguments.

Details

For models without moderators, the permutation test is carried out by permuting the signs of the estimates. The (two-sided) p-value of the permutation test is then equal to the proportion of times that the absolute value of the test statistic under the permuted data is as extreme or more extreme than under the actually observed data. See Follmann and Proschan (1999) for more details.

For models with moderators, the permutation test is carried out by permuting the rows of the model matrix (i.e., X). The (two-sided) p-value for a particular model coefficient is then equal to the proportion of times that the absolute value of the test statistic for the coefficient under the permuted data is as extreme or more extreme than under the actually observed data. Similarly, for the omnibus test, the p-value is the proportion of times that the test statistic for the omnibus test is as extreme

or more extreme than the actually observed one (argument `btt` can be used to specify which coefficients should be included in this test). See Higgins and Thompson (2004) and Viechtbauer et al. (2015) for more details.

Exact versus Approximate Permutation Tests:

If `exact=TRUE`, the function will try to carry out an exact permutation test. An exact permutation test requires fitting the model to each possible permutation. However, the number of possible permutations increases rapidly with the number of estimates (i.e., k). For models without moderators, there are 2^k possible permutations of the signs. Therefore, for $k = 5$, there are 32 possible permutations, for $k = 10$, there are already 1024, and for $k = 20$, there are over one million such permutations.

For models with moderators, the increase in the number of possible permutations is even more severe. The total number of possible permutations of the model matrix is $k!$. Therefore, for $k = 5$, there are 120 possible permutations, for $k = 10$, there are 3,628,800, and for $k = 20$, there are over 10^{18} permutations of the model matrix.

Therefore, going through all possible permutations may become infeasible. Instead of using an exact permutation test, one can set `exact=FALSE` (which is also the default). In that case, the function approximates the exact permutation-based p-value(s) by going through a smaller number (as specified by the `iter` argument) of *random* permutations. Therefore, running the function twice on the same data can yield (slightly) different p-values. Setting `iter` sufficiently large ensures that the results become stable. For full reproducibility, one can also set the seed of the random number generator before running the function (see ‘Examples’). Note that if `exact=FALSE` and `iter` is actually larger than the number of iterations required for an exact permutation test, then an exact test will automatically be carried out.

For models with moderators, the exact permutation test actually only requires fitting the model to each *unique* permutation of the model matrix. The number of unique permutations will be smaller than $k!$ when the model matrix contains recurring rows. This may be the case when only including categorical moderators (i.e., factors) in the model or when any quantitative moderators included in the model can only take on a small number of unique values. When `exact=TRUE`, the function therefore uses an algorithm to restrict the test to only the unique permutations of the model matrix, which may make the use of the exact test feasible even when k is large.

One can also set `exact="i"` in which case the function simply returns the number of iterations required for an exact permutation test.

When using random permutations, the function ensures that the very first permutation will always correspond to the original data. This avoids p-values equal to 0.

Permutation-Based Confidence Intervals:

When `permci=TRUE`, the function also tries to obtain permutation-based confidence intervals (CIs) of the model coefficient(s). This is done by shifting the effect size estimates by some amount and finding the most extreme values for this amount for which the permutation-based test would just lead to non-rejection. The calculation of such CIs is computationally expensive and may take a long time to complete. For models with moderators, one can also set `permci` to a vector of indices to specify for which coefficient(s) a permutation-based CI should be obtained. When the algorithm fails to determine a particular CI bound, it will be shown as NA in the output.

Permutation Tests for Location-Scale Models:

The function also works with location-scale models (see rma.uni for details on such models). Permutation tests will then be carried out for both the location and scale parts of the model. However, note that permutation-based CIs are not available for location-scale models.

Value

An object of class "permutest.rma.uni". The object is a list containing the following components:

pval	p-value(s) based on the permutation test.
QMp	p-value for the omnibus test of moderators based on the permutation test.
zval.perm	values of the test statistics of the coefficients under the various permutations.
b.perm	the model coefficients under the various permutations.
QM.perm	the test statistic of the omnibus test of moderators under the various permutations.
ci.lb	lower bound of the confidence intervals for the coefficients (permutation-based when permci=TRUE).
ci.ub	upper bound of the confidence intervals for the coefficients (permutation-based when permci=TRUE).
...	some additional elements/values are passed on.

The results are formatted and printed with the `print` function. One can also use `coef` to obtain the table with the model coefficients, corresponding standard errors, test statistics, p-values, and confidence interval bounds. The permutation distribution(s) can be plotted with the `plot` function.

Note

The p-values obtained with permutation tests cannot reach conventional levels of statistical significance (i.e., $p \leq 0.05$) when k is very small. In particular, for models without moderators, the smallest possible (two-sided) p-value is 0.0625 when $k = 5$ and 0.03125 when $k = 6$. Therefore, the permutation test is only able to reject the null hypothesis at $\alpha = 0.05$ when k is at least equal to 6. For models with moderators, the smallest possible (two-sided) p-value for a particular model coefficient is 0.0833 when $k = 4$ and 0.0167 when $k = 5$ (assuming that each row in the model matrix is unique). Therefore, the permutation test is only able to reject the null hypothesis at $\alpha = 0.05$ when k is at least equal to 5. Consequently, permutation-based CIs can also only be obtained when k is sufficiently large.

When the number of permutations required for the exact test is so large as to be essentially indistinguishable from infinity (e.g., `factorial(200)`), the function will terminate with an error.

Determining whether a test statistic under the permuted data is as extreme or more extreme than under the actually observed data requires making $\geq$ or $\leq$ comparisons. To avoid problems due to the finite precision with which computers generally represent numbers (see [this](#) FAQ for details), the function uses a numerical tolerance (control argument `comptol`, which is set equal to `.Machine$double.eps^0.5` by default) when making such comparisons (e.g., instead of `sqrt(3)^2 >= 3`, which may evaluate to FALSE, we use `sqrt(3)^2 >= 3 - .Machine$double.eps^0.5`, which should evaluate to TRUE).

When obtaining permutation-based CIs, the function makes use of `uniroot`. By default, the desired accuracy is set equal to `.Machine$double.eps^0.25` and the maximum number of iterations to 100. The desired accuracy and the maximum number of iterations can be adjusted with the control argument (i.e., `control=list(tol=value, maxiter=value)`). Also, the interval searched for the CI bounds may be too narrow, leading to NA for a bound. In this case, one can try setting `control=list(distfac=value)` with a value larger than 1 to extend the interval (the value indicating a multiplicative factor by which to extend the width of the interval searched) or

control=list(extendInt="yes") to allow [unirroot](#) to extend the interval dynamically (in which case it can happen that a bound may try to drift to $\pm\infty$).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Follmann, D. A., & Proschan, M. A. (1999). Valid inference in random effects meta-analysis. *Biometrics*, **55**(3), 732–737. <https://doi.org/10.1111/j.0006-341x.1999.00732.x>
- Good, P. I. (2009). *Permutation, parametric, and bootstrap tests of hypotheses* (3rd ed.). New York: Springer.
- Higgins, J. P. T., & Thompson, S. G. (2004). Controlling the risk of spurious findings from meta-regression. *Statistics in Medicine*, **23**(11), 1663–1682. <https://doi.org/10.1002/sim.1752>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W., López-López, J. A., Sánchez-Meca, J., & Marín-Martínez, F. (2015). A comparison of procedures to test for moderators in mixed-effects meta-regression models. *Psychological Methods*, **20**(3), 360–374. <https://doi.org/10.1037/met0000023>
- Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*. **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>

See Also

[rma.uni](#) for the function to fit models for which permutation tests can be conducted.
[print](#) and [plot](#) for the print and plot methods and [coef](#) for a method to extract the model results table.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### random-effects model
res <- rma(yi, vi, data=dat)
res

## Not run:
### permutation test (approximate and exact)
set.seed(1234) # for reproducibility
permutest(res)
permutest(res, exact=TRUE)

## End(Not run)

### mixed-effects model with two moderators (absolute latitude and publication year)
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)
res
```

```

### number of iterations required for an exact permutation test
permutest(res, exact="i")

## Not run:
### permutation test (approximate only; exact not feasible)
set.seed(1234) # for reproducibility
permres <- permutest(res, iter=10000)
permres

### plot of the permutation distribution for absolute latitude
### dashed horizontal line: the observed value of the test statistic (in both tails)
### black curve: standard normal density (theoretical reference/null distribution)
### blue curve: kernel density estimate of the permutation distribution
### note: the tail area under the permutation distribution is larger
### than under a standard normal density (hence, the larger p-value)
plot(permres, beta=2, lwd=c(2,3,3,4), xlim=c(-5,5))

## End(Not run)

### mixed-effects model with a categorical and a quantitative moderator
res <- rma(yi, vi, mods = ~ ablat + alloc, data=dat)
res

## Not run:
### permutation test testing the allocation factor coefficients
set.seed(1234) # for reproducibility
permutest(res, btt="alloc")

## End(Not run)

```

plot.cumul.rma

Plot Method for 'cumul.rma' Objects

Description

Function to plot objects of class "cumul.rma".

Usage

```

## S3 method for class 'cumul.rma'
plot(x, yaxis, xlim, ylim, xlab, ylab,
     at, transf, atransf, targs, digits, cols,
     grid=TRUE, pch=19, cex=1, lwd=2, ...)

```

Arguments

x	an object of class "cumul.rma" obtained with cumul .
yaxis	either "tau2", "tau", "I2", or "H2" to specify what values should be placed on the y-axis. See 'Details'.

xlim	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
ylim	y-axis limits. If unspecified, the function sets the y-axis limits to some sensible values.
xlab	title for the x-axis. If unspecified, the function sets an appropriate axis title.
ylab	title for the y-axis. If unspecified, the function sets an appropriate axis title.
at	position of the x-axis tick marks and corresponding labels. If unspecified, the function sets the tick mark positions/labels to some sensible values.
transf	optional argument to specify a function to transform the pooled estimates (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
atransf	optional argument to specify a function to transform the x-axis labels (e.g., <code>atransf=exp</code> ; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified via <code>transf</code> or <code>atransf</code> .
digits	optional integer to specify the number of decimal places to which the tick mark labels of the x- and y-axis should be rounded. Can also be a vector of two integers, the first to specify the number of decimal places for the x-axis, the second for the y-axis labels (e.g., <code>digits=c(2,3)</code>). If unspecified, the function tries to set the argument to some sensible values.
cols	vector with two or more colors for visualizing the order of the cumulative results.
grid	logical to specify whether a grid should be added to the plot. Can also be a color name.
pch	plotting symbol to use. By default, a filled circle is used. See points for other options.
cex	symbol expansion factor.
lwd	line width.
...	other arguments.

Details

The function can be used to visualize the results from a cumulative meta-analysis as obtained with the [cumul](#) function.

The plot shows the pooled estimate on the x-axis and some measure of heterogeneity on the y-axis in the cumulative order of the results in the "cumul.rma" object. By default, τ^2 is shown on the y-axis for a random-effects model and I^2 otherwise, but one can also use argument `yaxis` to specify the measure of heterogeneity to place on the y-axis.

The color gradient of the points/lines indicates the order of the cumulative results (by default, light gray at the beginning, dark gray at the end). A different set of colors can be chosen via the `cols` argument. See 'Examples'.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[cumul](#) for the function to conduct a cumulative meta-analysis.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### random-effects model
res <- rma(yi, vi, data=dat)

### cumulative meta-analysis (in the order of publication year)
sav <- cumul(res, order=year)

### plot of the pooled estimates and tau^2 over time
plot(sav)

### illustrate some other plot options
plot(sav, yaxis="I2", ylim=c(0,100), transf=exp, xlim=c(0.25,0.55),
      lwd=5, cex=1.5, cols=c("green","blue","red"))
```

plot.gosh.rma

Plot Method for 'gosh.rma' Objects

Description

Function to plot objects of class "gosh.rma".

Usage

```
## S3 method for class 'gosh.rma'
plot(x, het="I2", pch=16, cex, out, col, alpha, border,
      xlim, ylim, xhist=TRUE, yhist=TRUE, hh=0.3, breaks,
      adjust, lwd, labels, ...)
```

Arguments

x	an object of class "gosh.rma" obtained with gosh .
het	character string to specify the heterogeneity measure to plot. Either "I2", "H2", "QE", "tau2", or "tau" (the last two only for random/mixed-effects models).
pch	plotting symbol to use. By default, a borderless filled circle is used. See points for other options.

cex	symbol expansion factor.
out	optional integer to specify the number of a study that may be a potential outlier. If specified, subsets containing the specified study are drawn in a different color than those not containing the study.
col	optional character string to specify the color of the points (if unspecified, points are drawn in black). When out is used, two colors should be specified (if unspecified, red is used for subsets containing the specified study and blue otherwise).
alpha	optional alpha transparency value for the points (0 means fully transparent and 1 means opaque). If unspecified, the function sets this to a sensible value.
border	optional character string to specify the color of the borders of the histogram bars. Set to FALSE to omit the borders.
xlim	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
ylim	y-axis limits. If unspecified, the function sets the y-axis limits to some sensible values.
xhist	logical to specify whether a histogram should be drawn for the x-axis (the default is TRUE).
yhist	logical to specify whether a histogram should be drawn for the y-axis (the default is TRUE).
hh	numeric value (or vector of two values) to adjust the height of the histogram(s). Must be between 0 and 1, but should not be too close to 0 or 1, as otherwise the plot cannot be drawn.
breaks	optional argument passed on to hist for choosing the (number of) breakpoints of the histogram(s).
adjust	optional argument passed on to density for adjusting the bandwidth of the kernel density estimate(s) (values larger than 1 result in more smoothing).
lwd	optional numeric value to specify the line width of the estimated densities. Set to 0 to omit the line(s).
labels	optional argument to specify the x-axis and y-axis labels (or passed on to pairs to specify the names of the variables in the scatter plot matrix).
...	other arguments.

Details

For models without moderators, the function draws a scatter plot of the pooled estimates on the x-axis against the chosen measure of heterogeneity on the y-axis for the various subsets. Histograms of the respective distributions (with kernel density estimates superimposed) are shown in the margins (when xhist=TRUE and yhist=TRUE).

For models with moderators, the function draws a scatter plot matrix (with the [pairs](#) function) of the chosen measure of heterogeneity and each of the model coefficients. Histograms of the variables plotted are shown along the diagonal, with kernel density estimates of the distributions superimposed. Arguments xlim, ylim, xhist, and yhist are then ignored, while argument hh can be used to compress/stretch the height of the distributions shown along the diagonal.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Olkin, I., Dahabreh, I. J., & Trikalinos, T. A. (2012). GOSH - a graphical display of study heterogeneity. *Research Synthesis Methods*, **3**(3), 214–223. <https://doi.org/10.1002/jrsm.1053>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/9781315>

See Also

[gosh](#) for the function to create the input to a GOSH plot.

Examples

```
### calculate log odds ratios and corresponding sampling variances
dat <- escalc(measure="OR", ai=ai, nli=nli, ci=ci, n2i=n2i, data=dat.egger2001)

### meta-analysis of all trials including ISIS-4 using an equal-effects model
res <- rma(yi, vi, data=dat, method="EE")

### fit FE model to all possible subsets (65535 models)
## Not run:
sav <- gosh(res, progbar=FALSE)

### create GOSH plot
### red points for subsets that include and blue points
### for subsets that exclude study 16 (the ISIS-4 trial)
plot(sav, out=16, breaks=100)

## End(Not run)
```

plot.infl.rma.uni

Plot Method for 'infl.rma.uni' Objects

Description

Function to plot objects of class "infl.rma.uni".

Usage

```
## S3 method for class 'infl.rma.uni'
plot(x, plotinf=TRUE, plotdfbs=FALSE, dfbsnew=FALSE, logcov=TRUE,
      slab.style=1, las=0, pch=21, bg, bg.infl, col.na, ...)
```

Arguments

<code>x</code>	an object of class "infl.rma.uni" obtained with influence .
<code>plotinf</code>	logical to specify whether the various case diagnostics should be plotted (the default is TRUE). Can also be a vector of up to 8 integers to specify which plots to draw. See 'Details' for the numbers corresponding to the various plots.
<code>plotdfbs</code>	logical to specify whether the DFBETAS values should be plotted (the default is FALSE). Can also be a vector of integers to specify for which coefficient(s) to plot the DFBETAS values.
<code>dfbsnew</code>	logical to specify whether a new device should be opened for plotting the DFBETAS values (the default is FALSE).
<code>logcov</code>	logical to specify whether the covariance ratios should be plotted on a log scale (the default is TRUE).
<code>slab.style</code>	integer to specify the style of the x-axis labels: 1 = study number, 2 = study label, 3 = abbreviated study label. Note that study labels, even when abbreviated, may be too long to fit in the margins (see argument <code>mar</code> for par to adjust the margin sizes).
<code>las</code>	integer between 0 and 3 to specify the alignment of the axis labels (see par). The most useful alternative to 0 is 3, so that the x-axis labels are drawn vertical to the axis.
<code>pch</code>	plotting symbol to use. By default, an open circle is used. See points for other options.
<code>bg</code>	optional character string to specify the background color of open plotting symbols. If unspecified, gray is used by default.
<code>bg.infl</code>	optional character string to specify the background color when the point is considered influential. If unspecified, red is used by default.
<code>col.na</code>	optional character string to specify the color for lines connecting two points with NA values in between. If unspecified, a light shade of gray is used by default.
<code>...</code>	other arguments.

Details

When `plotinf=TRUE`, the function plots the (1) externally standardized residuals, (2) DFFITS values, (3) Cook's distances, (4) covariance ratios, (5) leave-one-out τ^2 estimates, (6) leave-one-out (residual) heterogeneity test statistics, (7) hat values, and (8) weights. If `plotdfbs=TRUE`, the DFBETAS values are also plotted either after confirming the page change (if `dfbsnew=FALSE`) or on a separate device (if `dfbsnew=TRUE`).

A case (which is typically synonymous with study) may be considered to be 'influential' if at least one of the following is true:

- The absolute DFFITS value is larger than $3 \times \sqrt{p/(k-p)}$, where p is the number of model coefficients and k the number of cases.
- The lower tail area of a chi-square distribution with p degrees of freedom cut off by the Cook's distance is larger than 50%.
- The hat value is larger than $3 \times (p/k)$.

- Any DFBETAS value is larger than 1.

Cases which are considered influential with respect to any of these measures are indicated by the color specified for the `bg.infl` argument (the default is "red").

The cut-offs described above are indicated in the plot with horizontal reference lines. In addition, on the plot of the externally standardized residuals, horizontal reference lines are drawn at -1.96, 0, and 1.96. On the plot of the covariance ratios, a horizontal reference line is drawn at 1. On the plot of leave-one-out τ^2 estimates, a horizontal reference line is drawn at the τ^2 estimate based on all cases. On the plot of leave-one-out (residual) heterogeneity test statistics, horizontal reference lines are drawn at the test statistic based on all cases and at $k - p$, the degrees of freedom of the test statistic. On the plot of the hat values, a horizontal reference line is drawn at p/k . Since the sum of the hat values is equal to p , the value p/k indicates equal hat values for all k cases. Finally, on the plot of weights, a horizontal reference line is drawn at $100/k$, corresponding to the value for equal weights (in %) for all k cases. Note that all weights will automatically be equal to each other when using unweighted model fitting. Also, the hat values will be equal to the weights (except for their scaling) in models without moderators.

The chosen cut-offs are (somewhat) arbitrary. Substantively informed judgment should always be used when examining the influence of each case on the results.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W., & Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research Synthesis Methods*, **1**(2), 112–125. <https://doi.org/10.1002/jrsm.11>

See Also

[influence](#) for the function to compute the various model diagnostics.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
             data=dat.bcg, slab=paste(author, year, sep=", "))

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### compute the diagnostics
inf <- influence(res)

### plot the values
plot(inf)

### show the abbreviated study labels on the x-axis
```

```

op <- par(mar=c(8,4,4,2))
plot(inf, slab.style=3, las=3)
par(op)

### select which plots to show
plot(inf, plotinf=1:4)

### plot the DFBETAS values
plot(inf, plotinf=FALSE, plotdfbs=TRUE)

```

plot.permutest.rma.uni

Plot Method for 'permutest.rma.uni' Objects

Description

Function to plot objects of class "permutest.rma.uni".

Usage

```

## S3 method for class 'permutest.rma.uni'
plot(x, beta, alpha, QM=FALSE, QS=FALSE,
     breaks="Scott", freq=FALSE, col, border, col.out, col.ref, col.density,
     trim=0, adjust=1, lwd=c(2,0,0,4), legend=FALSE, ...)

```

Arguments

x	an object of class "permutest.rma.uni" obtained with permutest .
beta	optional vector of indices to specify which (location) coefficients should be plotted.
alpha	optional vector of indices to specify which scale coefficients should be plotted. Only relevant for location-scale models (see rma.uni).
QM	logical to specify whether the permutation distribution of the omnibus test of the (location) coefficients should be plotted (the default is FALSE).
QS	logical to specify whether the permutation distribution of the omnibus test of the scale coefficients should be plotted (the default is FALSE). Only relevant for location-scale models (see rma.uni).
breaks	argument to be passed on to the corresponding argument of hist to set (the method for determining) the (number of) breakpoints.
freq	logical to specify whether frequencies or probability densities should be plotted (the default is FALSE to plot densities).
col	optional character string to specify the color of the histogram bars.
border	optional character string to specify the color of the borders around the bars.
col.out	optional character string to specify the color of the bars that are more extreme than the observed test statistic (the default is a semi-transparent shade of red).

<code>col.ref</code>	optional character string to specify the color of the theoretical reference/null distribution that is superimposed on top of the histogram (the default is a dark shade of gray).
<code>col.density</code>	optional character string to specify the color of the kernel density estimate of the permutation distribution that is superimposed on top of the histogram (the default is blue).
<code>trim</code>	the fraction (up to 0.5) of observations to be trimmed from the tails of each permutation distribution before its histogram is plotted.
<code>adjust</code>	numeric value to be passed on to the corresponding argument of <code>density</code> (for adjusting the bandwidth of the kernel density estimate).
<code>lwd</code>	numeric vector to specify the width of the vertical lines corresponding to the value of the observed test statistic, of the theoretical reference/null distribution, of the density estimate, and of the vertical line at 0 (note: by default, the theoretical reference/null distribution and the density estimate both have a line width of 0 and are therefore not plotted).
<code>legend</code>	logical to specify whether a legend should be added to the plot (the default is FALSE).
<code>...</code>	other arguments.

Details

The function plots the permutation distribution of each model coefficient as a histogram.

For models with moderators, one can choose via argument `beta` which coefficients to plot (by default, all permutation distributions except that of the intercept are plotted). One can also choose to plot the permutation distribution of the omnibus test of the model coefficients (by setting `QM=TRUE`).

Arguments `breaks`, `freq`, `col`, and `border` are passed on to the `hist` function for the plotting.

Argument `trim` can be used to trim away a certain fraction of observations from the tails of each permutation distribution before its histogram is plotted. By setting this to a value above 0, one can quickly remove some of the extreme values that might lead to the bulk of the distribution getting squished together at the center (typically, a small value such as `trim=0.01` is sufficient for this purpose).

The observed test statistic is indicated as a vertical dashed line (in both tails for a two-sided test).

Argument `col.out` is used to specify the color for the bars in the histogram that are more extreme than the observed test statistic. The p-value of a permutation test corresponds to the area of these bars.

One can superimpose the theoretical reference/null distribution on top of the histogram (i.e., the distribution as assumed by the model). The p-value for the standard (i.e., non-permutation) test is the area that is more extreme than the observed test statistic under this reference/null distribution.

A kernel density estimate of the permutation distribution can also be superimposed on top of the histogram (as a smoothed representation of the permutation distribution).

Note that the theoretical reference/null distribution and the kernel density estimate of the permutation distribution are only shown when setting the line width for these elements greater than 0 via the `lwd` argument (e.g., `lwd=c(2, 2, 2, 4)`).

By setting the legend argument to TRUE, a legend is added to the plot. One can also use a keyword for this argument to specify the position of the legend (e.g., legend="topright"; see [legend](#) for options). Finally, this argument can also be a list, with elements x, y, inset, and cex, which are passed on to the corresponding arguments of the [legend](#) function for even more control (elements not specified are set to defaults).

For location-scale models (see [rma.uni](#) for details), one can also use arguments alpha and QS to specify which scale coefficients to plot and whether to also plot the permutation distribution of the omnibus test of the scale coefficients (by setting QS=TRUE).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[permutest](#) for the function to create permutest.rma.uni objects.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### random-effects model
res <- rma(yi, vi, data=dat)
res

## Not run:
### permutation test (exact)
permres <- permutest(res, exact=TRUE)
permres

### plot of the permutation distribution
### dashed horizontal line: the observed value of the test statistic (in both tails)
### black curve: standard normal density (theoretical reference/null distribution)
### blue curve: kernel density estimate of the permutation distribution
plot(permres, lwd=c(2,3,3,4))

### mixed-effects model with two moderators (absolute latitude and publication year)
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)
res

### permutation test (approximate)
set.seed(1234) # for reproducibility
permres <- permutest(res, iter=10000)
permres
```

```

### plot of the permutation distribution for absolute latitude
### note: the tail area under the permutation distribution is larger
### than under a standard normal density (hence, the larger p-value)
plot(permrres, beta=2, lwd=c(2,3,3,4), xlim=c(-5,5))

## End(Not run)

```

plot.rma

Plot Method for 'rma' Objects

Description

Functions to plot objects of class "rma.uni", "rma.mh", "rma.peto", and "rma.glmm".

Usage

```

## S3 method for class 'rma.uni'
plot(x, qqplot=FALSE, ...)

## S3 method for class 'rma.mh'
plot(x, qqplot=FALSE, ...)

## S3 method for class 'rma.peto'
plot(x, qqplot=FALSE, ...)

## S3 method for class 'rma.glmm'
plot(x, qqplot=FALSE, ...) # not currently implemented

## S3 method for class 'rma.mv'
plot(x, qqplot=FALSE, ...) # not currently implemented

```

Arguments

x	an object of class "rma.uni", "rma.mh", or "rma.peto". The method is not (yet) implemented for objects of class "rma.glmm" or "rma.mv".
qqplot	logical to specify whether a normal QQ plot should be drawn (the default is FALSE).
...	other arguments.

Details

Four plots are produced. If the model does not contain any moderators, then a forest plot, funnel plot, radial plot, and a plot of the standardized residuals is provided. If qqplot=TRUE, the last plot is replaced by a normal QQ plot of the standardized residuals.

If the model contains moderators, then a forest plot, funnel plot, plot of the standardized residuals against the fitted values, and a plot of the standardized residuals is provided. If qqplot=TRUE, the last plot is replaced by a normal QQ plot of the standardized residuals.

Note

If the number of studies is large, the forest plot may become difficult to read due to the small font size. Stretching the plotting device vertically should provide more space.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[forest](#) for forest plots, [funnel](#) for funnel plots, [radial](#) for radial plots, and [qqnorm](#) for normal QQ plots.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model
res <- rma(yi, vi, data=dat)

### plot results
plot(res, qqplot=TRUE)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### plot results
plot(res, qqplot=TRUE)
```

plot.rma.uni.selmodel *Plot Method for 'plot.rma.uni.selmodel' Objects*

Description

Function to plot objects of class "plot.rma.uni.selmodel".

Usage

```
## S3 method for class 'rma.uni.selmodel'
plot(x, xlim, ylim, n=1000, prec="max", scale=FALSE,
     ci=FALSE, reps=1000, shade=TRUE, rug=TRUE, add=FALSE,
     lty=c("solid","dotted"), lwd=c(2,1), ...)
```

Arguments

x	an object of class "rma.uni.selmodel" obtained with <code>selmodel</code> .
xlim	x-axis limits. Essentially the range of p-values for which the selection function should be drawn. If unspecified, the function sets the limits automatically.
ylim	y-axis limits. If unspecified, the function sets the limits automatically.
n	numeric value to specify for how many p-values within the x-axis limits the function value should be computed (the default is 1000).
prec	either a character string (with options "max", "min", "mean", or "median") or a numeric value. See 'Details'.
scale	logical to specify whether the function values should be rescaled to a 0 to 1 range (the default is FALSE).
ci	logical to specify whether a confidence interval should be drawn around the selection function (the default is FALSE). Can also be a string (with options "boot" or "wald"). See 'Details'.
reps	numeric value to specify the number of bootstrap samples to draw for generating the confidence interval bounds (the default is 1000).
shade	logical to specify whether the confidence interval region should be shaded (the default is TRUE). Can also be a character vector to specify the color for the shading.
rug	logical to specify whether the observed p-values should be added as tick marks on the x-axis (the default is TRUE).
add	logical to specify whether the function should be added to an existing plot (the default is FALSE).
lty	the line types for the selection function and the confidence interval bounds.
lwd	the line widths for the selection function and the confidence interval bounds.
...	other arguments.

Details

The function can be used to draw the estimated selection function based on objects of class "plot.rma.uni.selmodel".

When the selection function incorporates a measure of precision (which, strictly speaking, is really a measure of imprecision), one can specify for which level of precision the selection function should be drawn. When `prec="max"`, then the function is drawn for the *least* precise study (maximum imprecision), when `prec="min"`, then the function is drawn for the *most* precise study (minimum imprecision), while `prec="mean"` and `prec="median"` will show the function for the mean and median level of imprecision, respectively. Alternatively, one can specify a numeric value for argument `prec` to specify the precision value (where `prec="max"` corresponds to `prec=1` and higher levels of precision to `prec` values below 1).

When `ci=TRUE` (or equivalently, `ci="boot"`), a confidence interval is drawn around the selection function. The bounds of this interval are generated using parametric bootstrapping, with argument `reps` controlling the number of bootstrap samples to draw for generating the confidence interval bounds. When both `n` and `reps` are large, constructing the confidence interval can take some time.

For models where the selection function involves a single δ parameter, one can also set `ci="wald"`, in which case the confidence interval will be constructed based on the Wald-type CI of the δ parameter (doing so is much quicker than using parametric bootstrapping). This option is also available for step function models (even if they involve multiple δ parameters).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[selmodel](#) for the function to fit models for which the estimated selection function can be drawn.

Examples

```
### copy data into 'dat' and examine data
dat <- dat.hackshaw1998

### fit random-effects model using the log odds ratios
res <- rma(yi, vi, data=dat, method="ML")
res

### fit step selection model
sel1 <- selmodel(res, type="stepfun", steps=c(0.05, 0.10, 0.50, 1.00))

### plot selection function
plot(sel1, scale=TRUE)

### fit negative exponential selection model
sel2 <- selmodel(res, type="negexp")

### add selection function to the existing plot
plot(sel2, add=TRUE, col="blue")

### plot selection function with CI
plot(sel1, ci="wald")

### plot selection function with CI
plot(sel2, ci="wald")
```

plot.vif.rma

*Plot Method for 'vif.rma' Objects***Description**

Plot method for objects of class "vif.rma".

Usage

```
## S3 method for class 'vif.rma'
plot(x, breaks="Scott", freq=FALSE, col, border, col.out, col.density,
      trim=0, adjust=1, lwd=c(2,0), ...)
```

Arguments

x	an object of class "vif.rma" obtained with vif .
breaks	argument to be passed on to the corresponding argument of hist to set (the method for determining) the (number of) breakpoints.
freq	logical to specify whether frequencies (if TRUE) or probability densities should be plotted (the default is FALSE).
col	optional character string to specify the color of the histogram bars.
border	optional character string to specify the color of the borders around the bars.
col.out	optional character string to specify the color of the bars that are more extreme than the observed (G)VIF value (the default is a semi-transparent shade of red).
col.density	optional character string to specify the color of the kernel density estimate of the distribution that is superimposed on top of the histogram (the default is blue).
trim	the fraction (up to 0.5) of observations to be trimmed from the upper tail of each distribution before its histogram is plotted.
adjust	numeric value to be passed on to the corresponding argument of density (for adjusting the bandwidth of the kernel density estimate).
lwd	numeric vector to specify the width of the vertical lines corresponding to the value of the observed (G)VIFs and of the density estimate (note: by default, the density estimate has a line width of 0 and is therefore not plotted).
...	other arguments.

Details

The function plots the distribution of each (G)VIF as simulated under independence as a histogram.

Arguments `breaks`, `freq`, `col`, and `border` are passed on to the [hist](#) function for the plotting.

Argument `trim` can be used to trim away a certain fraction of observations from the upper tail of each distribution before its histogram is plotted. By setting this to a value above 0, one can quickly remove some of the extreme values that might lead to the bulk of the distribution getting squished together at the left (typically, a small value such as `trim=0.01` is sufficient for this purpose).

The observed (G)VIF value is indicated as a vertical dashed line. If the observed exceeds the upper plot limit, then this is indicated by an arrow pointing to the line.

Argument `col.out` is used to specify the color for the bars in the histogram that are more extreme than the observed (G)VIF value.

A kernel density estimate of the distribution can be superimposed on top of the histogram (as a smoothed representation of the distribution). Note that the kernel density estimate of the distribution is only shown when setting the line width for this element greater than 0 via the `lwd` argument (e.g., `lwd=c(2,2)`).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[vif](#) for the function to create `vif.rma` objects.

Examples

```
### copy data from Bangert-Drowns et al. (2004) into 'dat'
dat <- dat.bangertdrowns2004

### fit mixed-effects meta-regression model
res <- rma(yi, vi, mods = ~ length + wic + feedback + info + pers + imag + meta, data=dat)

### use the simulation approach to analyze the size of the VIFs
## Not run:
vifs <- vif(res, sim=TRUE, seed=1234)
vifs

### plot the simulated distributions of the VIFs
plot(vifs)

### add densities, trim away some extremes, and set break points
plot(vifs, lwd=c(2,2), trim=0.01, breaks=seq(1,2.2,by=0.05), adjust=1.5)

## End(Not run)
```

predict.matreg	<i>Predicted Values for 'matreg' Objects</i>
----------------	--

Description

The function computes predicted values, corresponding standard errors, and confidence intervals for objects of class "matreg".

Usage

```
## S3 method for class 'matreg'
predict(object, newmods, intercept, addx=FALSE,
        level, adjust=FALSE, digits, transf, targ, vcov=FALSE, ...)
```

Arguments

object	an object of class "matreg".
newmods	vector or matrix to specify the values of the moderator/predictor values for which the predicted values should be calculated. See 'Details'.
intercept	logical to specify whether the intercept should be included when calculating the predicted values for newmods. If unspecified, the intercept is automatically added when the model also included an intercept.
addx	logical to specify whether the values of the moderator/predictor variables should be added to the returned object. See 'Examples'.
level	numeric value between 0 and 100 to specify the confidence interval level (see here for details). If unspecified, the default is to take the value from the object.
adjust	logical to specify whether the width of confidence intervals should be adjusted using a Bonferroni correction (the default is FALSE).
digits	optional integer to specify the number of decimal places to which the printed results should be rounded.
transf	optional argument to specify a function to transform the predicted values and interval bounds (e.g., transf=exp; see also transf). If unspecified, no transformation is used.
targ	optional arguments needed by the function specified under transf.
vcov	logical to specify whether the variance-covariance matrix of the predicted values should also be returned (the default is FALSE).
...	other arguments.

Details

For models including p' moderator/predictor variables, new moderator/predictor values (for k_{new} hypothetical new studies/cases) must be specified by setting newmods equal to a $k_{new} \times p'$ matrix with the corresponding new moderator/predictor values (if newmods is a vector, then only a single predicted value is computed unless the model only includes a single moderator/predictor, in which

case predicted values corresponding to all the vector values are computed). If the model object includes an intercept (so that the model has $p' + 1$ coefficients), then it will be automatically added to newmods unless one sets `intercept=FALSE`; alternatively, if newmods is a $k_{new} \times (p' + 1)$ matrix, then the `intercept` argument is ignored and the first column of the matrix determines whether the intercept is included when computing the predicted values or not. If the matrix specified via newmods has row names, then these are used to label the predicted values in the output.

When computing multiple predicted values, one can set `adjust=TRUE` to obtain confidence intervals whose width is adjusted based on a Bonferroni correction (e.g., instead of 95% CIs, the function provides $(100-5/k_{new})\%$ CIs, where k_{new} denotes the number of predicted values computed).

Value

An object of class `c("predict.matreg", "list.rma")`. The object is a list containing the following components:

<code>pred</code>	predicted value(s).
<code>se</code>	corresponding standard error(s).
<code>ci.lb</code>	lower bound of the confidence interval(s).
<code>ci.ub</code>	upper bound of the confidence interval(s).
<code>X</code>	the moderator/predictor value(s) used to calculate the predicted values (only when <code>addx=TRUE</code>).
<code>...</code>	some additional elements/values.

If `vcov=TRUE`, then the returned object is a list with the first element equal to the one as described above and the second element equal to the variance-covariance matrix of the predicted values.

The object is formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function.

Note

Under the ‘Regular R Matrix’ case (see `matreg`), confidence intervals are constructed based on the critical values from a t-distribution with $k - p$ degrees of freedom, where p denotes the total number of coefficients (i.e., including the intercept term if the model includes one). Otherwise, critical values from a standard normal distribution (i.e., ± 1.96 for `level=95`) are used.

When using the `transf` option, the transformation is applied to the predicted values and the corresponding interval bounds. The standard errors are omitted from the printed output. Also, `vcov=TRUE` is ignored when using the `transf` option.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Examples

```
### fit a regression model with lm() to the 'mtcars' dataset
res <- lm(mpg ~ hp + wt + am, data=mtcars)

### obtain a predicted value
predict(res, newdata=data.frame(hp=120, wt=4.2, am=1), interval="confidence")

### covariance matrix of the dataset
S <- cov(mtcars)

### fit the same regression model using matreg()
res <- matreg(mpg ~ hp + wt + am, R=S, cov=TRUE,
              means=colMeans(mtcars), n=nrow(mtcars))

### obtain the same predicted value
predict(res, newmods=c(hp=120, wt=4.2, am=1))
```

predict.rma

Predicted Values for 'rma' Objects

Description

The function computes predicted values, corresponding standard errors, confidence intervals, and prediction intervals for objects of class "rma".

Usage

```
## S3 method for class 'rma'
predict(object, newmods, intercept, tau2.levels, gamma2.levels, hetvar, prob,
        addx=FALSE, level, adjust=FALSE, digits, transf, targs, vcov=FALSE, ...)

## S3 method for class 'rma.ls'
predict(object, newmods, intercept, addx=FALSE, newscale, addz=FALSE,
        prob, level, adjust=FALSE, digits, transf, targs, vcov=FALSE, ...)
```

Arguments

object	an object of class "rma" or "rma.ls".
newmods	optional vector or matrix to specify the values of the moderator values for which the predicted values should be calculated. See 'Details'.
intercept	logical to specify whether the intercept should be included when calculating the predicted values for newmods. If unspecified, the intercept is automatically added when the model also included an intercept.
tau2.levels	optional vector to specify the levels of the inner factor when computing prediction intervals. Only relevant for models of class "rma.mv" (see rma.mv) and when the model includes more than a single τ^2 value. See 'Details'.

gamma2.levels	optional vector to specify the levels of the inner factor when computing prediction intervals. Only relevant for models of class "rma.mv" (see rma.mv) and when the model includes more than a single γ^2 value. See 'Details'.
hetvar	optional numeric vector with the amount of heterogeneity for computing prediction intervals. See 'Details'.
prob	optional character string to obtain probabilistic estimates from predictive distributions. See 'Details'.
addx	logical to specify whether the values of the moderator variables should be added to the returned object. See 'Examples'.
newscale	optional vector or matrix to specify the values of the scale variables for which the predicted values should be calculated. Only relevant for location-scale models (see rma.uni). See 'Details'.
addz	logical to specify whether the values of the scale variables should be added to the returned object.
level	numeric value between 0 and 100 to specify the confidence and prediction interval level (see here for details). If unspecified, the default is to take the value from the object.
adjust	logical to specify whether the width of confidence/prediction intervals should be adjusted using a Bonferroni correction (the default is FALSE).
digits	optional integer to specify the number of decimal places to which the printed results should be rounded.
transf	optional argument to specify a function to transform the predicted values and interval bounds (e.g., transf=exp; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified under transf.
vcov	logical to specify whether the variance-covariance matrix of the predicted values should also be returned (the default is FALSE). Ignored when using the transf argument.
...	other arguments.

Details

For an equal-effects model, `predict(object)` returns the estimated (average) effect for the set of studies included in the meta-analysis. This is the same as the estimated intercept in the equal-effects model (i.e., $\hat{\theta}$).

For a random-effects model, `predict(object)` returns the estimated (average) effect in the hypothetical population of studies from which the set of studies included in the meta-analysis are assumed to be a random selection. This is the same as the estimated intercept in the random-effects model (i.e., $\hat{\mu}$).

For models including one or more moderators, `predict(object)` returns $\hat{y} = Xb$, where X denotes the model matrix (see [model.matrix](#)) and b the estimated model coefficient (see [coef](#)), or in words, the estimated (average) effects for values of the moderator(s) equal to those of the k studies included in the meta-analysis (i.e., the 'fitted values' for the k studies).

For models including p' moderator variables, new moderator values (for k_{new} hypothetical new studies) can be specified by setting `newmods` equal to a $k_{new} \times p'$ matrix with the corresponding

new moderator values (if `newmods` is a vector, then only a single predicted value is computed unless the model only includes a single moderator variable, in which case predicted values corresponding to all the vector values are computed). If the model object includes an intercept (so that the model matrix has $p' + 1$ columns), then it will be automatically added to `newmods` unless one sets `intercept=FALSE`; alternatively, if `newmods` is a $k_{new} \times (p' + 1)$ matrix, then the `intercept` argument is ignored and the first column of the matrix determines whether the intercept is included when computing the predicted values or not. Note that any factors in the original model get turned into the appropriate contrast variables within the `rma.uni` function, so that `newmods` should actually include the values for the contrast variables. If the matrix specified via `newmods` has row names, then these are used to label the predicted values in the output. Examples are shown below.

For random/mixed-effects models, a prediction interval is also computed (Riley et al., 2011, but see ‘Note’). The interval estimates where `level%` of the true effects fall in the population of studies (and hence where the true effect of a new study from the population of studies should fall in `level%` of the cases).

Analogously, argument `prob` can be used to specify a string along the lines of `prob="<0"` or `prob=">0"` to estimate based on the predictive distribution the proportion of true effects below/above 0 (or some other cutoff, such as a minimally clinically useful value). Equivalently, this is the probability that the true effect in a new study (from the same population of studies) will fall below/above the specified cutoff value (e.g., $\Pr(\theta_{i,new} < 0)$ or $\Pr(\theta_{i,new} > 0)$).

For random-effects models that were fitted with the `rma.mv` function, the model may actually include multiple τ^2 values (i.e., when the random argument includes an ‘`~ inner | outer`’ term and `struct="HCS"`, `struct="DIAG"`, `struct="HAR"`, or `struct="UN"`). In that case, the function will provide prediction intervals for each level of the inner factor (since the prediction intervals differ depending on the τ^2 value). Alternatively, one can use the `tau2.levels` argument to specify for which level(s) the prediction interval should be provided. If the model includes a second ‘`~ inner | outer`’ term with multiple γ^2 values, prediction intervals for each combination of levels of the inner factors will be provided. Alternatively, one can use the `tau2.levels` and `gamma2.levels` arguments to specify for which level combination(s) the prediction interval should be provided.

When using the `newmods` argument for mixed-effects models that were fitted with the `rma.mv` function, if the model includes multiple τ^2 (and multiple γ^2) values, then one must use the `tau2.levels` (and `gamma2.levels`) argument to specify the levels of the inner factor(s) (i.e., a vector of length k_{new}) to obtain the appropriate prediction interval(s). Alternatively, one can use the `hetvar` argument to specify a vector with the amount of heterogeneity for the predicted values.

For location-scale models fitted with the `rma.uni` function, one can use `newmods` to specify the values of the p' moderator variables included in the model and `newscale` to specify the values of the q' scale variables included in the model. Whenever `newmods` is specified, the function computes predicted effects for the specified moderators values. To obtain the corresponding prediction intervals, one must also specify the corresponding `newscale` values. If only `newscale` is specified (and not `newmods`), the function computes the predicted log-transformed τ^2 values (when using a log link) for the specified scale values. By setting `transf=exp`, one can then obtain the predicted τ^2 values.

When computing multiple predicted values, one can set `adjust=TRUE` to obtain confidence/prediction intervals whose width is adjusted based on a Bonferroni correction (e.g., instead of 95% CIs, the function provides $(100-5/k_{new})\%$ CIs, where k_{new} denotes the number of predicted values computed).

Value

An object of class `c("predict.rma", "list.rma")`. The object is a list containing the following components:

<code>pred</code>	predicted value(s).
<code>se</code>	corresponding standard error(s).
<code>ci.lb</code>	lower bound of the confidence interval(s).
<code>ci.ub</code>	upper bound of the confidence interval(s).
<code>pi.lb</code>	lower bound of the prediction interval(s) (only for random/mixed-effects models).
<code>pi.ub</code>	upper bound of the prediction interval(s) (only for random/mixed-effects models).
<code>prob</code>	probabilistic estimate(s) from the predictive distribution (only when <code>prob</code> was specified).
<code>tau2.level</code>	the level(s) of the inner factor (only for models of class <code>"rma.mv"</code> with multiple τ^2 values).
<code>gamma2.level</code>	the level(s) of the inner factor (only for models of class <code>"rma.mv"</code> with multiple γ^2 values).
<code>X</code>	the moderator value(s) used to calculate the predicted values (only when <code>addx=TRUE</code>).
<code>Z</code>	the scale value(s) used to calculate the predicted values (only when <code>addz=TRUE</code> and only for location-scale models).
<code>...</code>	some additional elements/values.

If `vcov=TRUE`, then the returned object is a list with the first element equal to the one as described above and the second element equal to the variance-covariance matrix of the predicted values.

The object is formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function.

Note

Confidence and prediction intervals are constructed based on the critical values from a standard normal distribution (i.e., ± 1.96 for `level=95`). When the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="adhoc"`, then a t-distribution with $k - p$ degrees of freedom is used, where p denotes the total number of columns of the model matrix (i.e., counting the intercept term if the model includes one).

For a random-effects model (where $p = 1$) fitted with the `rma.uni` function, note that this differs slightly from Riley et al. (2011), who suggest to use a t-distribution with $k - 2$ degrees of freedom for constructing the prediction interval. Neither a normal, nor a t-distribution with $k - 1$ or $k - 2$ degrees of freedom is correct; all of these are approximations. The computations are done in the way described above, so that the prediction interval is identical to the confidence interval when $\hat{\tau}^2 = 0$, which could be argued is the logical thing that should happen. If the prediction interval for a random-effects model should be computed as described by Riley et al. (2011), then one can use argument `predtype="Riley"` (and for mixed-effects meta-regression models, the function then uses $k - p - 1$ degrees of freedom).

The predicted values are based only on the fixed effects of the model. Best linear unbiased predictions (BLUPs) that combine the fitted values based on the fixed effects and the estimated contributions of the random effects can be obtained with `blup` (currently only for objects of class `"rma.uni"`).

When using the `transf` argument, the transformation is applied to the predicted values and the corresponding interval bounds. The standard errors are omitted from the printed output. Also, `vcov=TRUE` is ignored when using the `transf` argument.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. San Diego, CA: Academic Press.
- Riley, R. D., Higgins, J. P. T., & Deeks, J. J. (2011). Interpretation of random effects meta-analyses. *British Medical Journal*, **342**, d549. <https://doi.org/10.1136/bmj.d549>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*, **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>

See Also

`fitted` for a function to (only) extract the fitted values, `blup` for a function to compute BLUPs that combine the fitted values and predicted random effects, and `addpoly` to add polygons based on predicted values to a forest plot.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model
res <- rma(yi, vi, data=dat)

### estimated average log risk ratio with 95% CI/PI
predict(res, digits=2)

### estimated average risk ratio with 95% CI/PI
predict(res, transf=exp, digits=2)

### note: strictly speaking, the value obtained is the estimated median risk ratio
### because exponentiation is a non-linear transformation; but we can estimate the
### average risk ratio by using the integral transformation
predict(res, transf=transf.exp.int, targ=res$tau2, digits=2)

### predict() can automatically extract the tau^2 value needed for the integral
### transformation from the model object, so we don't need to specify it
```

```

predict(res, transf=transf.exp.int, digits=2)

### note: the prediction interval is not printed here, because the regular back-
### transformation gives the correct bounds for the PI (the integral transformation
### is only needed to obtain the estimated average risk ratio and its CI)

### estimate the probability that the true log risk ratio is below 0 in a new study
predict(res, digits=2, prob="<0")

### fit mixed-effects model with absolute latitude as a moderator
res <- rma(yi, vi, mods = ~ ablat, data=dat)

### predicted average risk ratios for given absolute latitude values
predict(res, transf=exp, addx=TRUE)

### predicted average risk ratios for 10-60 degrees absolute latitude
predict(res, newmods=c(10, 20, 30, 40, 50, 60), transf=exp, addx=TRUE)

### can also include the intercept term in the 'newmods' matrix
predict(res, newmods=cbind(1, c(10, 20, 30, 40, 50, 60)), transf=exp, addx=TRUE)

### apply a Bonferroni correction for obtaining the interval bounds
predict(res, newmods=cbind(1, c(10, 20, 30, 40, 50, 60)), transf=exp, addx=TRUE, adjust=TRUE)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### predicted average risk ratios for 10 and 60 degrees latitude in 1950 and 1980
predict(res, newmods=cbind(c(10,60,10,60),c(1950,1950,1980,1980)), transf=exp, addx=TRUE)

### predicted average risk ratios for 10 and 60 degrees latitude in 1970 (row names as labels)
predict(res, newmods=rbind(at10=c(10,1970), at60=c(60,1970)), transf=exp)

### fit mixed-effects model with two moderators (one of which is a factor)
res <- rma(yi, vi, mods = ~ ablat + factor(alloc), data=dat)

### examine how the factor was actually coded for the studies in the dataset
predict(res, addx=TRUE)

### predicted average risk ratios at 30 degrees for the three factor levels
### note: the contrast (dummy) variables need to be specified explicitly here
predict(res, newmods=c(30, 0, 0), addx=TRUE) # for alternate allocation
predict(res, newmods=c(30, 1, 0), addx=TRUE) # for random allocation
predict(res, newmods=c(30, 0, 1), addx=TRUE) # for systematic allocation

### can also use a named vector with arbitrary order and abbreviated variable names
predict(res, newmods=c(sys=0, ran=0, abl=30))
predict(res, newmods=c(sys=0, ran=1, abl=30))
predict(res, newmods=c(sys=1, ran=0, abl=30))

```

Description

Functions to print objects of class "anova.rma" and "list.anova.rma".

Usage

```
## S3 method for class 'anova.rma'
print(x, digits=x$digits, ...)
## S3 method for class 'list.anova.rma'
print(x, digits=x[[1]]$digits, ...)
```

Arguments

x	an object of class "anova.rma" or "list.anova.rma" obtained with anova .
digits	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
...	other arguments.

Details

For a Wald-type test of one or multiple model coefficients, the output includes the test statistic (either a chi-square or F-value) and the corresponding p-value.

When testing one or multiple contrasts, the output includes the estimated value of the contrast, its standard error, test statistic (either a z- or a t-value), and the corresponding p-value.

When comparing two model objects, the output includes:

- the number of parameters in the full and the reduced model.
- the AIC, BIC, AICc, and log-likelihood of the full and the reduced model.
- the value of the likelihood ratio test statistic.
- the corresponding p-value.
- the test statistic of the test for (residual) heterogeneity for the full and the reduced model.
- the estimate of τ^2 from the full and the reduced model. Suppressed for equal-effects models.
- amount (in percent) of heterogeneity in the reduced model that is accounted for in the full model (NA for "rma.mv" objects). This can be regarded as a pseudo R^2 statistic (Raudenbush, 2009). Note that the value may not be very accurate unless k is large (López-López et al., 2014).

The last two items are not provided when comparing "rma.mv" models.

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- López-López, J. A., Marín-Martínez, F., Sánchez-Meca, J., Van den Noortgate, W., & Viechtbauer, W. (2014). Estimation of the predictive power of the model in mixed-effects meta-regression: A simulation study. *British Journal of Mathematical and Statistical Psychology*, **67**(1), 30–48. <https://doi.org/10.1111/bmsp.12002>
- Raudenbush, S. W. (2009). Analyzing effect sizes: Random effects models. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 295–315). New York: Russell Sage Foundation.
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[anova](#) for the function to create `anova.rma` objects.

`print.confint.rma` *Print Methods for 'confint.rma' and 'list.confint.rma' Objects*

Description

Functions to print objects of class "`confint.rma`" and "`list.confint.rma`".

Usage

```
## S3 method for class 'confint.rma'
print(x, digits=x$digits, ...)
## S3 method for class 'list.confint.rma'
print(x, digits=x$digits, ...)
```

Arguments

<code>x</code>	an object of class " <code>confint.rma</code> " or " <code>list.confint.rma</code> " obtained with confint .
<code>digits</code>	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
<code>...</code>	other arguments.

Details

The output includes:

- estimate of the model coefficient or variance/correlation parameter
- lower bound of the confidence interval
- upper bound of the confidence interval

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[confint](#) for the functions to create `confint.rma` and `list.confint.rma` objects.

print.deltamethod	<i>Print Method for 'deltamethod' Objects</i>
-------------------	---

Description

Functions to print objects of class "deltamethod".

Usage

```
## S3 method for class 'deltamethod'
print(x, digits, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)
```

Arguments

<code>x</code>	an object of class "deltamethod".
<code>digits</code>	integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
<code>signif.stars</code>	logical to specify whether p-values should be encoded visually with 'significance stars'. Defaults to the <code>show.signif.stars</code> slot of options .
<code>signif.legend</code>	logical to specify whether the legend for the 'significance stars' should be printed. Defaults to the value for <code>signif.stars</code> .
<code>...</code>	other arguments.

Details

The output is a table with the estimated coefficients, corresponding standard errors, test statistics, p-values, and confidence interval bounds.

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

See Also

[deltamethod](#) for the function to create deltamethod objects.

print.escalc	<i>Print and Summary Methods for 'escalc' Objects</i>
--------------	---

Description

Function to print objects of class "escalc" (and to obtain inferences for the individual studies/rows in such an object).

Usage

```
## S3 method for class 'escalc'
print(x, digits=attr(x,"digits"), ...)

## S3 method for class 'escalc'
summary(object, out.names=c("sei","zi","pval","ci.lb","ci.ub"), var.names,
        H0=0, append=TRUE, replace=TRUE, level=95, olim, digits, transf, ...)
```

Arguments

x	an object of class "escalc" obtained with escalc .
object	an object of class "escalc" obtained with escalc .
digits	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
out.names	character string with four elements to specify the variable names for the standard errors, test statistics, and lower/upper confidence interval bounds.
var.names	character string with two elements to specify the variable names for the estimates and the corresponding sampling variances (the default is to take the value from the object if possible).
H0	numeric value to specify the value under the null hypothesis (the default is 0).
append	logical to specify whether the data frame specified via the object argument should be returned together with the additional variables that are calculated by the summary function (the default is TRUE).

replace	logical to specify whether existing values for sei, zi, ci.lb, and ci.ub in the data frame should be replaced. Only relevant when the data frame already contains these variables. If replace=TRUE (the default), all of the existing values will be overwritten. If replace=FALSE, only NA values will be replaced.
level	numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see here for details).
olim	argument to specify observation limits. If unspecified, no limits are used.
transf	argument to specify a function to transform the estimates and interval bounds (e.g., transf=exp; see also transf). If unspecified, no transformation is used. Any additional arguments needed for the function specified here can be passed via ...
...	other arguments.

Value

The `print.escalc` function formats and prints the data frame, so that the estimates and sampling variances are rounded (to the number of digits specified).

The `summary.escalc` function creates an object that is a data frame containing the original data (if `append=TRUE`) and the following components:

yi	estimates (transformed if transf is specified).
vi	corresponding sampling variances.
sei	corresponding standard errors.
zi	test statistics for testing $H_0: \theta_i = H_0$ (i.e., $(y_i - H_0)/sei$).
pval	corresponding p-values.
ci.lb	lower confidence interval bounds (transformed if transf is specified).
ci.ub	upper confidence interval bounds (transformed if transf is specified).

When the `transf` argument is specified, elements `vi`, `sei`, `zi`, and `pval` are not included (since these only apply to the untransformed estimates).

Note that the actual variable names above depend on the `out.names` (and `var.names`) arguments. If the data frame already contains variables with names as specified by the `out.names` argument, the values for these variables will be overwritten when `replace=TRUE` (which is the default). By setting `replace=FALSE`, only values that are NA will be replaced.

The `print.escalc` function again formats and prints the data frame, rounding the added variables to the number of digits specified.

Note

If some transformation function has been specified for the `transf` argument, then `yi`, `ci.lb`, and `ci.ub` will be transformed accordingly. However, `vi` and `sei` then still reflect the sampling variances and standard errors of the untransformed values.

The `summary.escalc` function computes `level%` Wald-type confidence intervals, which may or may not be the most accurate method for computing confidence intervals for the chosen effect size or outcome measure.

If the estimates are bounded (e.g., correlations are bounded between -1 and +1, proportions are bounded between 0 and 1), one can use the `olim` argument to enforce those limits (i.e., the estimates and confidence interval bounds cannot exceed those limits then).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[escalc](#) for the function to create `escalc` objects.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
dat

### apply summary function
summary(dat)
summary(dat, transf=exp)
```

print.fsn	<i>Print Method for 'fsn' Objects</i>
-----------	---------------------------------------

Description

Function to print objects of class "fsn".

Usage

```
## S3 method for class 'fsn'
print(x, digits=x$digits, ...)
```

Arguments

<code>x</code>	an object of class "fsn" obtained with fsn .
<code>digits</code>	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
<code>...</code>	other arguments.

Details

The output shows the results from the fail-safe N calculation.

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[fsm](#) for the function to create fsm objects.

print.gosh.rma	<i>Print Method for 'gosh.rma' Objects</i>
----------------	--

Description

Function to print objects of class "gosh.rma".

Usage

```
## S3 method for class 'gosh.rma'
print(x, digits=x$digits, ...)
```

Arguments

x	an object of class "gosh.rma" obtained with gosh .
digits	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
...	other arguments.

Details

The output shows how many model fits were attempted, how many succeeded, and summary statistics (i.e., the mean, minimum, first quartile, median, third quartile, and maximum) for the various measures of (residual) heterogeneity and the model coefficient(s) computed across all of the subsets.

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[gosh](#) for the function to create gosh.rma objects.

print.hc.rma.uni	<i>Print Method for 'hc.rma.uni' Objects</i>
------------------	--

Description

Function to print objects of class "hc.rma.uni".

Usage

```
## S3 method for class 'hc.rma.uni'
print(x, digits=x$digits, ...)
```

Arguments

x	an object of class "hc.rma.uni" obtained with hc .
digits	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
...	other arguments.

Details

The output is a data frame with two rows, the first (labeled rma) corresponding to the results based on the usual estimation method, the second (labeled hc) corresponding to the results based on the method by Henmi and Copas (2010). The data frame includes the following variables:

- the method used to estimate τ^2 (always DL for hc)
- the estimated amount of heterogeneity
- the pooled estimate
- the corresponding standard error (NA when transf argument has been used)
- the lower and upper confidence interval bounds

Value

The function returns the data frame invisibly.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[hc](#) for the function to create `hc.rma.uni` objects.

print.list.rma	<i>Print Method for 'list.rma' Objects</i>
----------------	--

Description

Function to print objects of class "list.rma".

Usage

```
## S3 method for class 'list.rma'
print(x, digits=x$digits, ...)
```

Arguments

<code>x</code>	an object of class "list.rma".
<code>digits</code>	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
<code>...</code>	other arguments.

Value

See the documentation of the function that creates the "list.rma" object for details on what is printed. Regardless of what is printed, a data frame with the results is also returned invisibly.

See [methods.list.rma](#) for some additional method functions for "list.rma" objects.

Author(s)

Wolfgang Viechtbauer (wvb@metafor-project.org, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

print.matreg

Print and Summary Methods for 'matreg' Objects

Description

Functions to print objects of class "matreg" and "summary.matreg".

Usage

```
## S3 method for class 'matreg'
print(x, digits, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)

## S3 method for class 'matreg'
summary(object, digits, ...)

## S3 method for class 'summary.matreg'
print(x, digits, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)
```

Arguments

x	an object of class "matreg" or "summary.matreg" (for print).
object	an object of class "matreg" (for summary).
digits	integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
signif.stars	logical to specify whether p-values should be encoded visually with 'significance stars'. Defaults to the show.signif.stars slot of options .
signif.legend	logical to specify whether the legend for the 'significance stars' should be printed. Defaults to the value for signif.stars.
...	other arguments.

Details

The output is a table with the estimated coefficients, corresponding standard errors, test statistics, p-values, and confidence interval bounds. When using summary, the output includes additional statistics, including R^2 and the omnibus test of the model coefficients (either an F- or a chi-square test).

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

See Also

[matreg](#) for the function to create matreg objects.

```
print.permutest.rma.uni
```

Print Method for 'permutest.rma.uni' Objects

Description

Function to print objects of class "permutest.rma.uni".

Usage

```
## S3 method for class 'permutest.rma.uni'
print(x, digits=x$digits, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)
```

Arguments

<code>x</code>	an object of class "permutest.rma.uni" obtained with permutest .
<code>digits</code>	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
<code>signif.stars</code>	logical to specify whether p-values should be encoded visually with 'significance stars'. Defaults to the <code>show.signif.stars</code> slot of options .
<code>signif.legend</code>	logical to specify whether the legend for the 'significance stars' should be printed. Defaults to the value for <code>signif.stars</code> .
<code>...</code>	other arguments.

Details

The output includes:

- the results of the omnibus test of moderators. Suppressed if the model includes only one coefficient (e.g., only an intercept, like in the equal- and random-effects models). The p-value is based on the permutation test.
- a table with the estimated coefficients, corresponding standard errors, test statistics, p-values, and confidence interval bounds. The p-values are based on permutation tests. If `permc1` was set to `TRUE`, then the permutation-based CI bounds are shown.

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[permutest](#) for the function to create `permutest.rma.uni` objects.

print.ranktest	<i>Print Method for 'ranktest' Objects</i>
----------------	--

Description

Function to print objects of class "ranktest".

Usage

```
## S3 method for class 'ranktest'
print(x, digits=x$digits, ...)
```

Arguments

x	an object of class "ranktest" obtained with ranktest .
digits	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
...	other arguments.

Details

The output includes:

- the estimated value of Kendall's tau rank correlation coefficient
- the corresponding p-value for the test that the true tau is equal to zero

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (wvb@metafor-project.org, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[ranktest](#) for the function to create ranktest objects.

print.regtest	<i>Print Method for 'regtest' Objects</i>
---------------	---

Description

Function to print objects of class "regtest".

Usage

```
## S3 method for class 'regtest'
print(x, digits=x$digits, ret.fit=x$ret.fit, ...)
```

Arguments

x	an object of class "regtest" obtained with regtest .
digits	integer to specify the number of decimal places to which the printed results should be rounded (the default is to take the value from the object).
ret.fit	logical to specify whether the full results from the fitted model should also be returned. If unspecified, the default is to take the value from the object.
...	other arguments.

Details

The output includes:

- the model used for the regression test
- the predictor used for the regression test
- the results from the fitted model (only when `ret.fit=TRUE`)
- the test statistic for the regression test
- the degrees of freedom of the test statistic (only if the test statistic follows a t-distribution)
- the corresponding p-value
- the 'limit estimate' and its corresponding CI (only for predictors "sei" "vi", "ninv", or "sqrtninv" and when the model does not contain any additional moderators)

Value

The function does not return an object.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[regtest](#) for the function to create regtest objects.

print.rma

Print and Summary Methods for 'rma' Objects

Description

Functions to print objects of class "rma.uni", "rma.mh", "rma.peto", "rma.glmm", "rma.glmm", and "rma.mv".

Usage

```
## S3 method for class 'rma.uni'
print(x, digits, showfit=FALSE, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)

## S3 method for class 'rma.mh'
print(x, digits, showfit=FALSE, ...)

## S3 method for class 'rma.peto'
print(x, digits, showfit=FALSE, ...)

## S3 method for class 'rma.glmm'
print(x, digits, showfit=FALSE, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)

## S3 method for class 'rma.mv'
print(x, digits, showfit=FALSE, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)

## S3 method for class 'rma'
summary(object, digits, ...)

## S3 method for class 'summary.rma'
print(x, digits, showfit=TRUE, signif.stars=getOption("show.signif.stars"),
      signif.legend=signif.stars, ...)
```

Arguments

<code>x</code>	an object of class <code>"rma.uni"</code> , <code>"rma.mh"</code> , <code>"rma.peto"</code> , <code>"rma.glmm"</code> , <code>"rma.mv"</code> , or <code>"summary.rma"</code> (for print).
<code>object</code>	an object of class <code>"rma"</code> (for summary).
<code>digits</code>	integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object. See also here for further details on how to control the number of digits in the output.
<code>showfit</code>	logical to specify whether the fit statistics and information criteria should be printed (the default is FALSE for print and TRUE for summary).
<code>signif.stars</code>	logical to specify whether p-values should be encoded visually with 'significance stars'. Defaults to the <code>show.signif.stars</code> slot of options .
<code>signif.legend</code>	logical to specify whether the legend for the 'significance stars' should be printed. Defaults to the value for <code>signif.stars</code> .
<code>...</code>	other arguments.

Details

The output includes:

- the log-likelihood, deviance, AIC, BIC, and AICc value (when setting `showfit=TRUE` or by default for summary).
- for objects of class `"rma.uni"` and `"rma.glmm"`, the amount of (residual) heterogeneity in the random/mixed-effects model (i.e., the estimate of τ^2 and its square root). Suppressed for equal-effects models. The (asymptotic) standard error of the estimate of τ^2 is also provided (where possible).
- for objects of `"rma.mv"`, a table providing information about the variance components and correlations in the model. For σ^2 components, the estimate and its square root are provided, in addition to the number of values/levels, whether the component was fixed or estimated, and the name of the grouping variable/factor. If the R argument was used to specify a known correlation matrix for a particular random effect, then this is also indicated. For models with an `'~ inner | outer'` formula term, the name of the inner and outer grouping variable/factor are given and the number of values/levels of these variables/factors. In addition, for each τ^2 component, the estimate and its square root are provided, the number of effect size estimates observed at each level of the inner grouping variable/factor (only for `struct="HCS"`, `struct="DIAG"`, `struct="HAR"`, and `struct="UN"`), and whether the component was fixed or estimated. Finally, either the estimate of ρ (for `struct="CS"`, `struct="AR"`, `struct="CAR"`, `struct="HAR"`, or `struct="HCS"`) or the entire estimated correlation matrix (for `struct="UN"`) between the levels of the inner grouping variable/factor is provided, again with information whether a particular correlation was fixed or estimated, and how often each combination of levels of the inner grouping variable/factor was observed across the levels of the outer grouping variable/factor. If there is a second `'~ inner | outer'` formula term, the same information as described above will be provided, but now for the γ^2 and ϕ components.
- for objects of class `"rma.uni"`:

- the I^2 statistic, which estimates (in percent) how much of the total variability in the effect size estimates (which is composed of heterogeneity plus sampling variability) can be attributed to heterogeneity among the true effects. For a meta-regression model, I^2 estimates how much of the unaccounted variability (which is composed of residual heterogeneity plus sampling variability) can be attributed to residual heterogeneity. See ‘Note’ for how I^2 is computed.
- the H^2 statistic, which estimates the ratio of the total amount of variability in the effect size estimates to the amount of sampling variability. For a meta-regression model, H^2 estimates the ratio of the unaccounted variability in the effect size estimates to the amount of sampling variability. See ‘Note’ for how H^2 is computed.
- the R^2 statistic, which estimates the amount of heterogeneity accounted for by the moderators included in the model and can be regarded as a pseudo R^2 statistic (Raudenbush, 2009). Only provided when fitting a model including moderators. See ‘Note’ for how R^2 is computed.
- for objects of class "rma.glmm", the amount of study level variability (only when using a model that models study level differences as a random effect).
- the results of the test for (residual) heterogeneity. This is the usual Q -test for heterogeneity when not including moderators in the model and the Q_E -test for residual heterogeneity when moderators are included. For objects of class "rma.glmm", the results from a Wald-type test and a likelihood ratio test are provided (see [rma.glmm](#) for more details).
- the results of the omnibus (Wald-type) test of the coefficients in the model (the indices of the coefficients tested are also indicated). Suppressed if the model includes only one coefficient (e.g., only an intercept, like in the equal- and random-effects models).
- a table with the estimated coefficients, corresponding standard errors, test statistics, p-values, and confidence interval bounds.
- the Cochran-Mantel-Haenszel test and Tarone’s test for heterogeneity (only when analyzing odds ratios using the Mantel-Haenszel method, i.e., for objects of class "rma.mh").

See [here](#) for details on the option to create styled/colored output with the help of the [crayon](#) package.

Value

The print functions do not return an object. The summary function returns the object passed to it (with additional class "summary.rma").

Note

For random-effects models, the I^2 statistic is computed with

$$I^2 = 100\% \times \frac{\hat{\tau}^2}{\hat{\tau}^2 + \tilde{v}},$$

where $\hat{\tau}^2$ is the estimated value of τ^2 and

$$\tilde{v} = \frac{(k-1) \sum w_i}{(\sum w_i)^2 - \sum w_i^2},$$

where $w_i = 1/v_i$ is the inverse of the sampling variance of the i th study ($\tilde{v}$ is equation 9 in Higgins & Thompson, 2002, and can be regarded as the ‘typical’ within-study variance of the effect size estimates). The H^2 statistic is computed with

$$H^2 = \frac{\hat{\tau}^2 + \tilde{v}}{\tilde{v}}.$$

Analogous equations are used for mixed-effects models.

Therefore, depending on the estimator of τ^2 used, the values of I^2 and H^2 will change. For random-effects models, I^2 and H^2 are often computed with $I^2 = (Q - (k - 1))/Q$ and $H^2 = Q/(k - 1)$, where Q denotes the statistic of the test for heterogeneity and k the number of studies included in the meta-analysis. The equations used in the **metafor** package to compute these statistics are more general and have the advantage that the values of I^2 and H^2 will be consistent with the estimated value of τ^2 (i.e., if $\hat{\tau}^2 = 0$, then $I^2 = 0$ and $H^2 = 1$ and if $\hat{\tau}^2 > 0$, then $I^2 > 0$ and $H^2 > 1$).

The two definitions of I^2 and H^2 actually coincide when using the DerSimonian-Laird estimator of τ^2 (i.e., the commonly used equations are actually special cases of the more general definitions given above). Therefore, if you prefer the more conventional definitions of these statistics, use `method="DL"` when fitting the random/mixed-effects model with the `rma.uni` function. The conventional definitions are also automatically used when fitting an equal-effects models.

For mixed-effects models, the pseudo R^2 statistic (Raudenbush, 2009) is computed with

$$R^2 = \frac{\hat{\tau}_{RE}^2 - \hat{\tau}_{ME}^2}{\hat{\tau}_{RE}^2},$$

where $\hat{\tau}_{RE}^2$ denotes the estimated value of τ^2 based on the random-effects model (i.e., the total amount of heterogeneity) and $\hat{\tau}_{ME}^2$ denotes the estimated value of τ^2 based on the mixed-effects model (i.e., the residual amount of heterogeneity). It can happen that $\hat{\tau}_{RE}^2 < \hat{\tau}_{ME}^2$, in which case R^2 is set to zero (and also if $\hat{\tau}_{RE}^2 = 0$). Again, the value of R^2 will change depending on the estimator of τ^2 used. This statistic is only computed when the random-effects model is nested within the mixed-effects model. You can also use the `anova` function to compute R^2 for any two models that are known to be nested. Note that the pseudo R^2 statistic may not be very accurate unless k is large (López-López et al., 2014).

For fixed-effects with moderators models, the R^2 statistic is simply the standard R^2 statistic (also known as the ‘coefficient of determination’) computed based on weighted least squares estimation. To be precise, the so-called ‘adjusted’ R^2 statistic is provided, since k is often relatively small in meta-analyses, in which case the adjustment is relevant.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Higgins, J. P. T., & Thompson, S. G. (2002). Quantifying heterogeneity in a meta-analysis. *Statistics in Medicine*, **21**(11), 1539–1558. <https://doi.org/10.1002/sim.1186>
- López-López, J. A., Marín-Martínez, F., Sánchez-Meca, J., Van den Noortgate, W., & Viechtbauer, W. (2014). Estimation of the predictive power of the model in mixed-effects meta-regression: A simulation study. *British Journal of Mathematical and Statistical Psychology*, **67**(1), 30–48. <https://doi.org/10.1111/bmsp.12002>

Raudenbush, S. W. (2009). Analyzing effect sizes: Random effects models. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 295–315). New York: Russell Sage Foundation.

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for the corresponding model fitting functions.

profile.rma

Profile Likelihood Plots for 'rma' Objects

Description

Functions to profile the (restricted) log-likelihood for objects of class "rma.uni", "rma.mv", "rma.uni.selmodel", and "rma.ls".

Usage

```
## S3 method for class 'rma.uni'
profile(fitted, xlim, ylim, steps=20, lltol=1e-03,
        progbar=TRUE, parallel="no", ncpus=1, cl, plot=TRUE, ...)

## S3 method for class 'rma.mv'
profile(fitted, sigma2, tau2, rho, gamma2, phi, xlim, ylim, steps=20, lltol=1e-03,
        progbar=TRUE, parallel="no", ncpus=1, cl, plot=TRUE, ...)

## S3 method for class 'rma.uni.selmodel'
profile(fitted, tau2, delta, xlim, ylim, steps=20, lltol=1e-03,
        progbar=TRUE, parallel="no", ncpus=1, cl, plot=TRUE, ...)

## S3 method for class 'rma.ls'
profile(fitted, alpha, xlim, ylim, steps=20, lltol=1e-03,
        progbar=TRUE, parallel="no", ncpus=1, cl, plot=TRUE, ...)

## S3 method for class 'profile.rma'
print(x, ...)
## S3 method for class 'profile.rma'
plot(x, xlim, ylim, pch=19, xlab, ylab, main, refline=TRUE, cline=FALSE, ...)
```

Arguments

fitted	an object of class "rma.uni", "rma.mv", "rma.uni.selmodel", or "rma.ls".
x	an object of class "profile.rma" (for plot and print).
sigma2	optional integer to specify for which σ^2 parameter the likelihood should be profiled.

tau2	optional integer to specify for which τ^2 parameter the likelihood should be profiled.
rho	optional integer to specify for which ρ parameter the likelihood should be profiled.
gamma2	optional integer to specify for which γ^2 parameter the likelihood should be profiled.
phi	optional integer to specify for which ϕ parameter the likelihood should be profiled.
delta	optional integer to specify for which δ parameter the likelihood should be profiled.
alpha	optional integer to specify for which α parameter the likelihood should be profiled.
xlim	optional vector to specify the lower and upper limit of the parameter over which the profiling should be done. If unspecified, the function sets these limits automatically.
ylim	optional vector to specify the y-axis limits when plotting the profiled likelihood. If unspecified, the function sets these limits automatically.
steps	number of points between xlim[1] and xlim[2] (inclusive) for which the likelihood should be evaluated (the default is 20). Can also be a numeric vector of length 2 or longer to specify for which parameter values the likelihood should be evaluated (in this case, xlim is automatically set to range(steps) if unspecified).
lltol	numerical tolerance used when comparing values of the profiled log-likelihood with the log-likelihood of the fitted model (the default is 1e-03).
progbar	logical to specify whether a progress bar should be shown (the default is TRUE).
parallel	character string to specify whether parallel processing should be used (the default is "no"). For parallel processing, set to either "snow" or "multicore". See 'Details'.
ncpus	integer to specify the number of processes to use in the parallel processing.
cl	optional cluster to use if parallel="snow". If unspecified, a cluster on the local machine is created for the duration of the call.
plot	logical to specify whether the profile plot should be drawn after profiling is finished (the default is TRUE).
pch	plotting symbol to use. By default, a filled circle is used. See points for other options.
refline	logical to specify whether the value of the parameter estimate should be indicated by a dotted vertical line and its log-likelihood value by a dotted horizontal line (the default is TRUE).
cline	logical to specify whether a horizontal reference line should be added to the plot that indicates the log-likelihood value corresponding to the 95% profile confidence interval (the default is FALSE). Can also be a numeric value between 0 and 100 to specify the confidence interval level.
xlab	title for the x-axis. If unspecified, the function sets an appropriate axis title.

ylab	title for the y-axis. If unspecified, the function sets an appropriate axis title.
main	title for the plot. If unspecified, the function sets an appropriate title.
...	other arguments.

Details

The function fixes a particular parameter of the model and then computes the maximized (restricted) log-likelihood over the remaining parameters of the model. By fixing the parameter of interest to a range of values, a profile of the (restricted) log-likelihood is constructed.

Selecting the Parameter(s) to Profile:

The parameters that can be profiled depend on the model object:

- For objects of class "rma.uni" obtained with the [rma.uni](#) function, the function profiles over τ^2 (not for equal-effects models).
- For objects of class "rma.mv" obtained with the [rma.mv](#) function, profiling is done by default over all variance and correlation components of the model. Alternatively, one can use the sigma2, tau2, rho, gamma2, or phi arguments to specify over which parameter the profiling should be done. Only one of these arguments can be used at a time. A single integer is used to specify the number of the parameter.
- For selection model objects of class "rma.uni.selmodel" obtained with the [selmodel](#) function, profiling is done by default over τ^2 (for models where this is an estimated parameter) and all selection model parameters. Alternatively, one can choose to profile only τ^2 by setting tau2=TRUE or one can select one of the selection model parameters to profile by specifying its number via the delta argument.
- For location-scale model objects of class "rma.ls" obtained with the [rma.uni](#) function, profiling is done by default over all α parameters that are part of the scale model. Alternatively, one can select one of the parameters to profile by specifying its number via the alpha argument.

Interpreting Profile Likelihood Plots:

A profile likelihood plot should show a single peak at the corresponding ML/REML estimate. If `refline=TRUE` (the default), the value of the parameter estimate is indicated by a dotted vertical line and its log-likelihood value by a dotted horizontal line. Hence, the intersection of these two lines should correspond to the peak (assuming that the model was fitted with ML/REML estimation).

When profiling a variance component (or some other parameter that cannot be negative), the peak may be at zero (if this corresponds to the ML/REML estimate of the parameter). In this case, the profiled log-likelihood should be a monotonically decreasing function of the parameter. Similarly, when profiling a correlation component, the peak may be at -1 or +1.

If the profiled log-likelihood has multiple peaks, this indicates that the likelihood surface is not unimodal. In such cases, the ML/REML estimate may correspond to a local optimum (when the intersection of the two dotted lines is not at the highest peak).

If the profile is flat (over the entire parameter space or large portions of it), then this suggests that at least some of the parameters of the model are not identifiable (and the parameter estimates obtained are to some extent arbitrary). See Raue et al. (2009) for some further discussion of parameter identifiability and the use of profile likelihoods to check for this.

The function checks whether any profiled log-likelihood value is actually larger than the log-likelihood of the fitted model (using a numerical tolerance of `lltol`). If so, a warning is issued as this might indicate that the optimizer did not identify the actual ML/REML estimate of the parameter profiled.

Parallel Processing:

Profiling requires repeatedly refitting the model, which can be slow when k is large and/or the model is complex (the latter especially applies to `"rma.mv"` objects and also to certain `"rma.uni.selmodel"` or `"rma.ls"` objects). On machines with multiple cores, one can try to speed things up by delegating the model fitting to separate worker processes, that is, by setting `parallel="snow"` or `parallel="multicore"` and `ncpus` to some value larger than 1. Parallel processing makes use of the `parallel` package, using the `makePSOCKcluster` and `parLapply` functions when `parallel="snow"` or using `mclapply` when `parallel="multicore"` (the latter only works on Unix/Linux-alikes).

Value

An object of class `"profile.rma"`. The object is a list (or list of such lists) containing the following components:

One of the following (depending on the parameter that was actually profiled):

<code>sigma2</code>	values of σ^2 over which the likelihood was profiled.
<code>tau2</code>	values of τ^2 over which the likelihood was profiled.
<code>rho</code>	values of ρ over which the likelihood was profiled.
<code>gamma2</code>	values of γ^2 over which the likelihood was profiled.
<code>phi</code>	values of ϕ over which the likelihood was profiled.
<code>delta</code>	values of δ over which the likelihood was profiled.
<code>alpha</code>	values of α over which the likelihood was profiled.

In addition, the following components are included:

<code>ll</code>	(restricted) log-likelihood values at the corresponding parameter values.
<code>beta</code>	a matrix with the estimated model coefficients at the corresponding parameter values.
<code>ci.lb</code>	a matrix with the lower confidence interval bounds of the model coefficients at the corresponding parameter values.
<code>ci.ub</code>	a matrix with the upper confidence interval bounds of the model coefficients at the corresponding parameter values.
<code>...</code>	some additional elements/values.

Note that the list is returned invisibly.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmuller, U., & Timmer, J. (2009). Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, **25**(15), 1923–1929. <https://doi.org/10.1093/bioinformatics/btp35>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*. **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>

See Also

[rma.uni](#), [rma.mv](#), and [selmodel](#) for functions to fit models for which profile likelihood plots can be drawn.

[confint](#) for functions to compute corresponding profile likelihood confidence intervals.

Examples

```
### calculate log odds ratios and corresponding sampling variances
dat <- escalc(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model using rma.uni()
res <- rma(yi, vi, data=dat)

### profile over tau^2
profile(res, progbar=FALSE)

### adjust xlim
profile(res, progbar=FALSE, xlim=c(0,1))

### specify tau^2 values at which to profile the likelihood
profile(res, progbar=FALSE, steps=c(seq(0,0.2,length=20),seq(0.3,1,by=0.1)))

### change data into long format
dat.long <- to.long(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, append=FALSE)

### set levels/labels for group ("con" = control/non-vaccinated, "exp" = experimental/vaccinated)
dat.long$group <- factor(dat.long$group, levels=c(2,1), labels=c("con","exp"))

### calculate log odds and corresponding sampling variances
dat.long <- escalc(measure="PLO", xi=out1, mi=out2, data=dat.long)
dat.long

### fit bivariate random-effects model using rma.mv()
res <- rma.mv(yi, vi, mods = ~ group, random = ~ group | study, struct="UN", data=dat.long)
res

### profile over tau^2_1, tau^2_2, and rho
### note: for rho, adjust region over which profiling is done ('zoom in' on area around estimate)
## Not run:
par(mfrow=c(2,2))
```

```

profile(res, tau2=1)
profile(res, tau2=2)
profile(res, rho=1, xlim=c(0.90, 0.98))
par(mfrow=c(1,1))

## End(Not run)

### an example where the peak of the likelihood profile is at 0
dat <- escalc(measure="RD", n1i=n1i, n2i=n2i, ai=ai, ci=ci, data=dat.hine1989)
res <- rma(yi, vi, data=dat)
profile(res, progbar=FALSE)

```

qqnorm.rma

Normal QQ Plots for 'rma' Objects

Description

Function to create normal QQ plots for objects of class "rma.uni", "rma.mh", and "rma.peto".

Usage

```

## S3 method for class 'rma.uni'
qqnorm(y, type="rstandard", pch=21, col, bg, grid=FALSE,
        envelope=TRUE, level=y$level, bonferroni=FALSE, reps=1000, smooth=TRUE, bass=0,
        label=FALSE, offset=0.3, pos=13, lty, ...)

## S3 method for class 'rma.mh'
qqnorm(y, type="rstandard", pch=21, col, bg, grid=FALSE,
        label=FALSE, offset=0.3, pos=13, ...)

## S3 method for class 'rma.peto'
qqnorm(y, type="rstandard", pch=21, col, bg, grid=FALSE,
        label=FALSE, offset=0.3, pos=13, ...)

## S3 method for class 'rma.glmm'
qqnorm(y, ...) # not currently implemented

## S3 method for class 'rma.mv'
qqnorm(y, ...) # not currently implemented

```

Arguments

y	an object of class "rma.uni", "rma.mh", or "rma.peto". The method is not (yet) implemented for objects of class "rma.glmm" or "rma.mv".
type	character string (either "rstandard" (default) or "rstudent") to specify whether standardized residuals or studentized deleted residuals should be used in creating the plot. See 'Details'.

<code>pch</code>	plotting symbol to use for the points. By default, an open circle is used. See points for other options.
<code>col</code>	optional character string to specify the (border) color of the points.
<code>bg</code>	optional character string to specify the background color of open plot symbols.
<code>grid</code>	logical to specify whether a grid should be added to the plot (the default is FALSE). Can also be a color name.
<code>envelope</code>	logical to specify whether a pseudo confidence envelope should be simulated and added to the plot (the default is TRUE). Can also be a color name. Only for objects of class <code>"rma.uni"</code> . See 'Details'.
<code>level</code>	numeric value between 0 and 100 to specify the level of the pseudo confidence envelope (see here for details). The default is to take the value from the object.
<code>bonferroni</code>	logical to specify whether the bounds of the envelope should be Bonferroni corrected.
<code>reps</code>	numeric value to specify the number of iterations for simulating the pseudo confidence envelope (the default is 1000).
<code>smooth</code>	logical to specify whether the results from the simulation should be smoothed (the default is TRUE).
<code>bass</code>	numeric value that controls the degree of smoothing (the default is 0).
<code>label</code>	argument to control the labeling of the points (the default is FALSE). See 'Details'.
<code>offset</code>	argument to control the distance between the points and the corresponding labels.
<code>pos</code>	argument to control the position of the labels.
<code>lty</code>	optional argument to specify the line type for the diagonal line and the pseudo confidence envelope. If unspecified, the function sets this to <code>c("solid", "dotted")</code> by default.
<code>...</code>	other arguments.

Details

The plot shows the theoretical quantiles of a normal distribution on the horizontal axis against the observed quantiles for either the standardized residuals (`type="rstandard"`, the default) or the externally standardized residuals (`type="rstudent"`) on the vertical axis (see [residuals](#) for details on the definition of these residual types).

For reference, a line is added to the plot with a slope of 1, going through the (0,0) point.

For objects of class `"rma.uni"`, it is also possible to add a pseudo confidence envelope to the plot. The envelope is created based on the quantiles of sets of pseudo residuals simulated from the given model (for details, see Cook & Weisberg, 1982). The number of sets simulated can be controlled with the `reps` argument. When `smooth=TRUE`, the simulated bounds are smoothed with Friedman's SuperSmoother (see [supsmu](#)). The `bass` argument can be set to a number between 0 and 10, with higher numbers indicating increasing smoothness. If `bonferroni=TRUE`, the envelope bounds are Bonferroni corrected, so that the envelope can be regarded as a confidence region for all k residuals simultaneously. The default however is `bonferroni=FALSE`, which makes the plot more sensitive to deviations from normality.

With the `label` argument, one can control whether points in the plot will be labeled (e.g., to identify outliers). If `label="all"` (or `label=TRUE`), all points in the plot will be labeled. If `label="out"`, points falling outside of the confidence envelope will be labeled (only available for objects of class `"rma.uni"`). Finally, one can also set this argument to a numeric value (between 1 and k), to specify how many of the most extreme points should be labeled (for example, with `label=1` only the most extreme point is labeled, while with `label=3`, the most extreme, and the second and third most extreme points are labeled). With the `offset` argument, one can adjust the distance between the labels and the corresponding points. The `pos` argument is the position specifier for the labels (1, 2, 3, and 4, respectively indicate positions below, to the left of, above, and to the right of the points; 13 places the labels below the points for points that fall below the reference line and above otherwise; 24 places the labels to the left of the points for points that fall above the reference line and to the right otherwise).

Value

A list with components:

<code>x</code>	the x-axis coordinates of the points that were plotted.
<code>y</code>	the y-axis coordinates of the points that were plotted.

Note that the list is returned invisibly.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression*. London: Chapman and Hall.

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/9781315>

Wang, M. C., & Bushman, B. J. (1998). Using the normal quantile plot to explore meta-analytic data sets. *Psychological Methods*, **3**(1), 46–54. <https://doi.org/10.1037/1082-989X.3.1.46>

See Also

[rma.uni](#), [rma.mh](#), and [rma.peto](#) for functions to fit models for which normal QQ plots can be drawn.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model
res <- rma(yi, vi, data=dat)
```

```

### draw QQ plot
qqnorm(res, grid=TRUE)

### fit mixed-effects model with absolute latitude as moderator
res <- rma(yi, vi, mods = ~ ablat, data=dat)

### draw QQ plot
qqnorm(res, grid=TRUE)

```

radial

*Radial Plots for 'rma' Objects***Description**

Function to create radial plots for objects of class "rma".

Usage

```

radial(x, ...)

## S3 method for class 'rma'
radial(x, center=FALSE, xlim, zlim, xlab, zlab,
       atz, aty, steps=7, level=x$level, digits=2,
       transf, targs, pch=21, col, bg, back, arc.res=100,
       cex, cex.lab, cex.axis, ...)

```

Arguments

x	an object of class "rma".
center	logical to specify whether the plot should be centered horizontally at the pooled estimate (the default is FALSE).
xlim	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
zlim	z-axis limits. If unspecified, the function sets the z-axis limits to some sensible values (note that the z-axis limits are the actual vertical limit of the plotting region).
xlab	title for the x-axis. If unspecified, the function sets an appropriate axis title.
zlab	title for the z-axis. If unspecified, the function sets an appropriate axis title.
atz	position for the z-axis tick marks and labels. If unspecified, these values are set by the function.
aty	position for the y-axis tick marks and labels. If unspecified, these values are set by the function.
steps	the number of tick marks for the y-axis (the default is 7). Ignored when argument aty is used.

<code>level</code>	numeric value between 0 and 100 to specify the level of the z-axis error region. The default is to take the value from the object.
<code>digits</code>	integer to specify the number of decimal places to which the tick mark labels of the y-axis should be rounded (the default is 2).
<code>transf</code>	argument to specify a function to transform the y-axis labels (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
<code>targs</code>	optional arguments needed by the function specified via <code>transf</code> .
<code>pch</code>	plotting symbol. By default, an open circle is used. See points for other options.
<code>col</code>	character string to specify the (border) color of the points.
<code>bg</code>	character string to specify the background color of open plot symbols.
<code>back</code>	character string to specify the background color of the z-axis error region. If unspecified, a shade of gray is used. Set to NA to suppress shading of the region.
<code>arc.res</code>	integer to specify the number of line segments (i.e., the resolution) when drawing the y-axis and confidence interval arcs (the default is 100).
<code>cex</code>	symbol expansion factor.
<code>cex.lab</code>	character expansion factor for axis labels.
<code>cex.axis</code>	character expansion factor for axis annotations.
<code>...</code>	other arguments.

Details

Radial plots (also sometimes called Galbraith plots) were introduced by Galbraith (1988a, 1988b, 1994) as a way to plot estimates with different precisions. In meta-analyses, they are an interesting alternative to forest plots, especially when the number of effect size estimates is large (in which case the forest plot would be very large).

For an equal-effects model, the plot shows the inverse of the standard errors on the horizontal axis (i.e., $1/\sqrt{v_i}$, where v_i is the sampling variance of the estimates) against the effect size estimates standardized by their corresponding standard errors on the vertical axis (i.e., $y_i/\sqrt{v_i}$). Since the vertical axis corresponds to standardized values, it is referred to as the z-axis within this function. On the right hand side of the plot, an arc is drawn (referred to as the y-axis within this function) corresponding to the estimates. A line projected from (0,0) through a particular point within the plot onto this arc indicates the value of the estimate for that point.

For a random-effects model, the function uses $1/\sqrt{v_i + \tau^2}$ for the horizontal axis, where τ^2 is the amount of heterogeneity as estimated based on the model. For the z-axis, $y_i/\sqrt{v_i + \tau^2}$ is used to compute standardized values of the effect size estimates.

The second (inner/smaller) arc that is drawn on the right hand side indicates the pooled estimate (in the middle of the arc) and the corresponding confidence interval (at the ends of the arc).

The shaded region in the plot is the z-axis error region. For `level=95` (or if this was the `level` value when the model was fitted), this corresponds to z-axis values equal to ± 1.96 . Under the assumptions of the equal/random-effects models, approximately 95% of the points should fall within this region.

When `center=TRUE`, the values on the y-axis are centered around the pooled estimate. As a result, the plot is centered horizontally at the pooled estimate.

If the z-axis label on the left is too close to the actual z-axis and/or the arc on the right is clipped, then this can be solved by increasing the margins on the right and/or left (see [par](#) and in particular the `mar` argument).

Note that radial plots cannot be drawn for models that contain moderators.

Value

A data frame with components:

<code>x</code>	the x-axis coordinates of the points that were plotted.
<code>y</code>	the y-axis coordinates of the points that were plotted.
<code>ids</code>	the study id numbers.
<code>slab</code>	the study labels.

Note that the data frame is returned invisibly.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Galbraith, R. F. (1988a). Graphical display of estimates having differing standard errors. *Technometrics*, **30**(3), 271–281. <https://doi.org/10.1080/00401706.1988.10488400>
- Galbraith, R. F. (1988b). A note on graphical presentation of estimated odds ratios from several clinical trials. *Statistics in Medicine*, **7**(8), 889–894. <https://doi.org/10.1002/sim.4780070807>
- Galbraith, R. F. (1994). Some applications of radial plots. *Journal of the American Statistical Association*, **89**(428), 1232–1242. <https://doi.org/10.1080/01621459.1994.10476864>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which radial plots can be drawn.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
dat

### fit equal-effects model
res <- rma(yi, vi, data=dat, method="EE")

### draw radial plot
radial(res)

### the line from (0,0) with a slope equal to the log risk ratio from the 4th study
```

```

### points to the corresponding effect size estimate on the arc (i.e., -1.44)
abline(a=0, b=dat$yi[4], lty="dotted")
dat$yi[4]

### meta-analysis of the log risk ratios using a random-effects model
res <- rma(yi, vi, data=dat)

### draw radial plot
radial(res)

### center the values around the pooled estimate
radial(res, center=TRUE)

### show risk ratio values on the y-axis arc
radial(res, transf=exp)

```

ranef

Best Linear Unbiased Predictions for 'rma.uni' and 'rma.mv' Objects

Description

Functions to compute best linear unbiased predictions (BLUPs) of the random effects for objects of class "rma.uni" and "rma.mv". Corresponding standard errors and prediction interval bounds are also provided.

Usage

```

## S3 method for class 'rma.uni'
ranef(object, level, digits, transf, targs, ...)
## S3 method for class 'rma.mv'
ranef(object, level, digits, transf, targs, verbose=FALSE, ...)

```

Arguments

object	an object of class "rma.uni" or "rma.mv".
level	numeric value between 0 and 100 to specify the prediction interval level (see here for details). If unspecified, the default is to take the value from the object.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
transf	optional argument to specify a function to transform the predicted values and interval bounds (e.g., transf=exp; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified under transf.
verbose	logical to specify whether output should be generated on the progress of the computations (the default is FALSE).
...	other arguments.

Value

For objects of class "rma.uni", an object of class "list.rma". The object is a list containing the following components:

pred	predicted values.
se	corresponding standard errors.
pi.lb	lower bound of the prediction intervals.
pi.ub	upper bound of the prediction intervals.
...	some additional elements/values.

The object is formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function.

For objects of class "rma.mv", a list of data frames with the same components as described above.

Note

For best linear unbiased predictions that combine the fitted values based on the fixed effects and the estimated contributions of the random effects, see `blup`.

For predicted/fitted values that are based only on the fixed effects of the model, see `fitted` and `predict`.

Equal-effects models do not contain random study effects. The BLUPs for these models will therefore be 0.

When using the `transf` argument, the transformation is applied to the predicted values and the corresponding interval bounds. The standard errors are then set equal to NA and are omitted from the printed output.

By default, a standard normal distribution is used to construct the prediction intervals. When the model was fitted with `test="t"`, `test="knha"`, `test="hksj"`, or `test="adhoc"`, then a t-distribution with $k - p$ degrees of freedom is used.

To be precise, it should be noted that the function actually computes empirical BLUPs (eBLUPs), since the predicted values are a function of the estimated variance component(s).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Kacker, R. N., & Harville, D. A. (1981). Unbiasedness of two-stage estimation and prediction procedures for mixed linear models. *Communications in Statistics, Theory and Methods*, **10**(13), 1249–1261. <https://doi.org/10.1080/03610928108828108>
- Raudenbush, S. W., & Bryk, A. S. (1985). Empirical Bayes meta-analysis. *Journal of Educational Statistics*, **10**(2), 75–98. <https://doi.org/10.3102/10769986010002075>
- Robinson, G. K. (1991). That BLUP is a good thing: The estimation of random effects. *Statistical Science*, **6**(1), 15–32. <https://doi.org/10.1214/ss/1177011926>
- Searle, S. R., Casella, G., & McCulloch, C. E. (1992). *Variance components*. Hoboken, NJ: Wiley.

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`rma.uni` and `rma.mv` for functions to fit models for which BLUPs of the random effects can be computed.

`predict` and `fitted` for functions to compute the predicted/fitted values based only on the fixed effects and `blup` for a function to compute BLUPs that combine the fitted values and predicted random effects.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### meta-analysis of the log risk ratios using a random-effects model
res <- rma(yi, vi, data=dat)

### BLUPs of the random effects
ranef(res)
```

ranktest	<i>Rank Correlation Test for Funnel Plot Asymmetry</i>
----------	--

Description

Function to carry out the rank correlation test for funnel plot asymmetry.

Usage

```
ranktest(x, vi, sei, subset, data, digits, ...)
```

Arguments

x	a vector with the effect size estimates or an object of class "rma".
vi	vector with the corresponding sampling variances. Ignored if x is an object of class "rma".
sei	vector with the corresponding standard errors (note: only one of the two, vi or sei, needs to be specified).
subset	optional (logical or numeric) vector to specify the subset of studies that should be included in the test. Ignored if x is an object of class "rma".
data	optional data frame containing the variables given to the arguments above.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded.
...	other arguments.

Details

The function carries out the rank correlation test as described by Begg and Mazumdar (1994). The test can be used to examine whether the effect size estimates and the corresponding sampling variances are correlated. A high correlation would indicate that the funnel plot is asymmetric, which may be a result of publication bias.

Via argument `x`, one can either pass a vector with the effect size estimates to the function (and then the corresponding sampling variances via `vi` or the standard errors via `sei`) or an object of class `"rma"`. When passing a model object to argument `x`, the model must be a model without moderators (i.e., either an equal- or a random-effects model).

Value

An object of class `"ranktest"`. The object is a list containing the following components:

<code>tau</code>	the estimated value of Kendall's tau rank correlation coefficient.
<code>pval</code>	the corresponding p-value for the test that the true tau value is equal to zero.

The results are formatted and printed with the `print` function.

Note

The method does not depend on the model fitted. Therefore, regardless of the model passed to the function, the results of the rank test will always be the same. See `regtest` for tests of funnel plot asymmetry that are based on regression models and model dependent.

The function makes use of the `cor.test` function with `method="kendall"`. If possible, an exact p-value is provided; otherwise, a large-sample approximation is used.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Begg, C. B., & Mazumdar, M. (1994). Operating characteristics of a rank correlation test for publication bias. *Biometrics*, **50**(4), 1088–1101. <https://doi.org/10.2307/2533446>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`regtest` for the regression test, `trimfill` for the trim and fill method, `tes` for the test of excess significance, `fsn` to compute the fail-safe N (file drawer analysis), and `selmodel` for selection models.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model
res <- rma(yi, vi, data=dat)

### carry out the rank correlation test
ranktest(res)

### can also pass the effect size estimates and corresponding sampling variances to the function
ranktest(yi, vi, data=dat)
```

rcalc	<i>Calculate the Variance-Covariance of Dependent Correlation Coefficients</i>
-------	--

Description

Function to calculate the variance-covariance matrix of correlation coefficients computed based on the same sample of subjects.

Usage

```
rcalc(x, ni, data, rtoz=FALSE, nfun="min", sparse=FALSE, ...)
```

Arguments

x	a formula of the form $r_i \sim \text{var1} + \text{var2} \mid \text{study}$. Can also be a correlation matrix or list thereof. See ‘Details’.
ni	vector to specify the sample sizes based on which the correlations were computed.
data	data frame containing the variables specified via the formula (and the sample sizes).
rtoz	logical to specify whether to transform the correlations via Fisher’s r-to-z transformation (the default is FALSE).
nfun	a character string to specify how the ‘common’ sample size within each study should be computed. Possible options are "min" (for the minimum), "harmonic" (for the harmonic mean), or "mean" (for the arithmetic mean). Can also be a function. See ‘Details’.
sparse	logical to specify whether the variance-covariance matrix should be returned as a sparse matrix (the default is FALSE).
...	other arguments.

Details

A meta-analysis of correlation coefficients may involve multiple correlation coefficients extracted from the same study. When these correlations are computed based on the same sample of subjects, then they are typically not independent. The `rcalc` function can be used to create a dataset with the correlation coefficients (possibly transformed with Fisher's r-to-z transformation) and the corresponding variance-covariance matrix. The dataset and variance-covariance matrix can then be further meta-analyzed using the `rma.mv` function.

When computing the covariance between two correlation coefficients, we can distinguish two cases:

1. In the first case, one of the variables involved in the two correlation coefficients is the same. For example, in r_{12} and r_{13} , variable 1 is common to both correlation coefficients. This is sometimes called the (partially) 'overlapping' case. The covariance between the two correlation coefficients, $\text{Cov}[r_{12}, r_{13}]$, then depends on the degree of correlation between variables 2 and 3 (i.e., r_{23}).
2. In the second case, none of the variables are common to both correlation coefficients. For example, this would be the case if we have correlations r_{12} and r_{34} based on 4 variables. This is sometimes called the 'non-overlapping' case. The covariance between the two correlation coefficients, $\text{Cov}[r_{12}, r_{34}]$, then depends on r_{13} , r_{14} , r_{23} , and r_{24} .

Equations to compute these covariances can be found, for example, in Steiger (1980) and Olkin and Finn (1990).

To use the `rcalc` function, one needs to construct a data frame that contains a study identifier (say `study`), two variable identifiers (say `var1` and `var2`), the corresponding correlation coefficients (say `ri`), and the sample sizes based on which the correlation coefficients were computed (say `ni`). Then the first argument should be a formula of the form `ri ~ var1 + var2 | study`, argument `ni` is set equal to the variable name containing the sample sizes, and the data frame containing these variables is specified via the `data` argument. When using the function for a single study, one can leave out the study identifier from the formula.

When argument `rtoz` is set to `TRUE`, then the correlations are transformed with Fisher's r-to-z transformation (Fisher, 1921) and the variance-covariance matrix is computed for the transformed values.

In some cases, the sample size may not be identical within a study (e.g., r_{12} may have been computed based on 120 subjects while r_{13} was computed based on 118 subjects due to 2 missing values in variable 3). For constructing the variance-covariance matrix, we need to assume a 'common' sample size for all correlation coefficients within the study. Argument `nfun` provides some options for how the common sample size should be computed. Possible options are `"min"` (for using the minimum sample size within a study as the common sample size), `"harmonic"` (for using the harmonic mean), or `"mean"` (for using the arithmetic mean). The default is `"min"`, which is a conservative choice (i.e., it will overestimate the sampling variances of coefficients that were computed based on a sample size that was actually larger than the minimum sample size). One can also specify a function via the `nfun` argument (which should take a numeric vector as input and return a single value).

Instead of specifying a formula, one can also pass a correlation matrix to the function via argument `x`. Argument `ni` then specifies the (common) sample size based on which the elements in the correlation matrix were computed. One can also pass a list of correlation matrices via argument `x`, in which case argument `ni` should be a vector of sample sizes of the same length as `x`.

Value

A list containing the following components:

<code>dat</code>	a data frame with the study identifier, the two variable identifiers, a variable pair identifier, the correlation coefficients (possibly transformed with Fisher's r-to-z transformation), and the (common) sample sizes.
<code>V</code>	corresponding variance-covariance matrix (given as a sparse matrix when <code>sparse=TRUE</code> ; otherwise a matrix with class <code>"vcovmat"</code>).

Note that a particular covariance can only be computed when all of the correlation coefficients involved in the covariance equation are included in the dataset. If one or more coefficients needed for the computation are missing, then the resulting covariance will also be missing (i.e., NA).

Note

For raw correlation coefficients, the variance-covariance matrix is computed with $n - 1$ in the denominator (instead of n as suggested in Steiger, 1980, and Olkin & Finn, 1990). This is more consistent with the usual equation for computing the sampling variance of a correlation coefficient (which also typically uses $n - 1$ in the denominator).

For raw and r-to-z transformed coefficients, the variance-covariance matrix will only be computed when the (common) sample size for a study is at least 5.

Author(s)

Wolfgang Viechtbauer (<wvbw@metafor-project.org>, <https://www.metafor-project.org>).

References

- Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron*, **1**, 1–32. <https://hdl.handle.net/2440/15169>
- Olkin, I., & Finn, J. D. (1990). Testing correlated correlations. *Psychological Bulletin*, **108**(2), 330–333. <https://doi.org/10.1037/0033-2909.108.2.330>
- Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, **87**(2), 245–251. <https://doi.org/10.1037/0033-2909.87.2.245>

See Also

[rma.mv](#) for a model fitting function that can be used to meta-analyze dependent correlation coefficients.

[dat.craft2003](#) for an illustrative example.

Examples

```
#####

### copy data into 'dat' and examine the first 12 rows
dat <- dat.craft2003
head(dat, 12)
```

```

### construct dataset and var-cov matrix of the correlations
tmp <- rcalc(ri ~ var1 + var2 | study, ni=ni, data=dat)
V <- tmp$V
dat <- tmp$dat

### examine data and var-cov matrix for study 1
dat[dat$study == 1,]
blsplit(V, dat$study, round, 4)$`1`

### examine data and var-cov matrix for study 6
dat[dat$study == 6,]
blsplit(V, dat$study, round, 4)$`6`

### examine data and var-cov matrix for study 17
dat[dat$study == 17,]
blsplit(V, dat$study, round, 4)$`17`

#####

### copy data into 'dat' and examine the first 12 rows
dat <- dat.craft2003
head(dat, 12)

### restructure data from study 1 into a correlation matrix
R1 <- diag(4)
R1[lower.tri(R1)] <- dat$ri[dat$study == 1]
R1[upper.tri(R1)] <- t(R1)[upper.tri(R1)]
rownames(R1) <- colnames(R1) <- c("perf", "acog", "asom", "conf")
R1

### restructure data from study 3 into a correlation matrix
R3 <- diag(4)
R3[lower.tri(R3)] <- dat$ri[dat$study == 3]
R3[upper.tri(R3)] <- t(R3)[upper.tri(R3)]
rownames(R3) <- colnames(R3) <- c("perf", "acog", "asom", "conf")
R3

### an example where a correlation matrix is passed to rcalc()
rcalc(R1, ni=142)

### an example where a list of correlation matrices is passed to rcalc()
tmp <- rcalc(list("1"=R1,"3"=R3), ni=c(142,37))
V <- tmp$V
dat <- tmp$dat

### examine data and var-cov matrix for study 1
dat[dat$id == 1,]
blsplit(V, dat$id, round, 4)$`1`

### examine data and var-cov matrix for study 3
dat[dat$id == 3,]
blsplit(V, dat$id, round, 4)$`3`

```

#####

regplot

Scatter Plots / Bubble Plots

Description

Function to create scatter plots / bubble plots based on meta-regression models.

Usage

```
regplot(x, ...)
```

```
## S3 method for class 'rma'
regplot(x, mod, pred=TRUE, ci=TRUE, pi=FALSE, shade=TRUE,
        xlim, ylim, predlim, olim, xlab, ylab, at, digits=2L,
        transf, atransf, targs, level=x$level,
        pch, psize, plim=c(0.5,3), col, bg, slab,
        grid=FALSE, refline, label=FALSE, offset=c(1,1), labsize=1,
        lcol, lwd, lty, legend=FALSE, xvals, ...)
```

```
## S3 method for class 'regplot'
points(x, ...)
```

Arguments

x	an object of class "rma.uni", "rma.mv", or "rma.glmm" including one or multiple moderators (or an object of class "regplot" for points).
mod	either a scalar to specify the position of the moderator variable in the model or a character string to specify the name of the moderator variable.
pred	logical to specify whether the (marginal) regression line based on the moderator should be added to the plot (the default is TRUE). Can also be an object from predict . See 'Details'.
ci	logical to specify whether the corresponding confidence interval bounds should be added to the plot (the default is TRUE).
pi	logical to specify whether the corresponding prediction interval bounds should be added to the plot (the default is FALSE).
shade	logical to specify whether the confidence/prediction interval regions should be shaded (the default is TRUE). Can also be a two-element character vector to specify the colors for shading the confidence and prediction interval regions (if shading only the former, a single color can also be specified).
xlim	x-axis limits. If unspecified, the function sets the x-axis limits to some sensible values.
ylim	y-axis limits. If unspecified, the function sets the y-axis limits to some sensible values.

predlim	argument to specify the limits of the (marginal) regression line. If unspecified, the limits are based on the range of the moderator variable.
olim	argument to specify observation limits. If unspecified, no limits are used.
xlab	title for the x-axis. If unspecified, the function sets an appropriate axis title.
ylab	title for the y-axis. If unspecified, the function sets an appropriate axis title.
at	position of the y-axis tick marks and corresponding labels. If unspecified, the function sets the tick mark positions/labels to some sensible values.
digits	integer to specify the number of decimal places to which the tick mark labels of the y-axis should be rounded. When specifying an integer (e.g., 2L), trailing zeros after the decimal mark are dropped for the y-axis labels. When specifying a numeric value (e.g., 2), trailing zeros are retained.
transf	argument to specify a function to transform the estimates and confidence/prediction interval bounds (e.g., <code>transf=exp</code> ; see also transf). If unspecified, no transformation is used.
atransf	argument to specify a function to transform the y-axis labels (e.g., <code>atransf=exp</code> ; see also transf). If unspecified, no transformation is used.
targs	optional arguments needed by the function specified via <code>transf</code> or <code>atransf</code> .
level	numeric value between 0 and 100 to specify the confidence/prediction interval level (see here for details). The default is to take the value from the object.
pch	plotting symbol to use for the estimates. By default, an open circle is used. Can also be a vector of values. See points for other options.
psize	numeric value to specify the point sizes for the estimates. If unspecified, the point sizes are a function of the model weights. Can also be a vector of values. Can also be a character string (either "seinv" or "vinv") to make the point sizes proportional to the inverse standard errors or inverse sampling variances.
plim	numeric vector of length 2 to scale the point sizes (ignored when a numeric value or vector is specified for <code>psize</code>). See 'Details'.
col	character string to specify the (border) color of the points. Can also be a vector.
bg	character string to specify the background color of open plot symbols. Can also be a vector.
slab	vector with labels for the k studies. If unspecified, the function tries to extract study labels from <code>x</code> .
grid	logical to specify whether a grid should be added to the plot. Can also be a color name for the grid.
refline	optional numeric value to specify the location of a horizontal reference line that should be added to the plot.
label	argument to control the labeling of the points (the default is FALSE). See 'Details'.
offset	argument to control the distance between the points and the corresponding labels. See 'Details'.
labsize	numeric value to control the size of the labels.

<code>lcol</code>	optional vector of (up to) four elements to specify the color of the regression line, of the confidence interval bounds, of the prediction interval bounds, and of the horizontal reference line.
<code>lty</code>	optional vector of (up to) four elements to specify the line type of the regression line, of the confidence interval bounds, of the prediction interval bounds, and of the horizontal reference line.
<code>lwd</code>	optional vector of (up to) four elements to specify the line width of the regression line, of the confidence interval bounds, of the prediction interval bounds, and of the horizontal reference line.
<code>legend</code>	logical to specify whether a legend should be added to the plot (the default is FALSE). See ‘Details’.
<code>xvals</code>	optional numeric vector to specify the values of the moderator for which predicted values should be computed. Needs to be specified when passing an object from <code>predict</code> to the <code>pred</code> argument. See ‘Details’.
<code>...</code>	other arguments.

Details

The function draws a scatter plot of the values of a moderator variable in a meta-regression model (on the x-axis) against the effect size estimates (on the y-axis). The regression line from the model (with corresponding confidence interval bounds) is added to the plot by default. These types of plots are also often referred to as ‘bubble plots’ as the points are typically drawn in different sizes to reflect their precision or weight in the model.

If the model includes multiple moderators, one must specify via argument `mod` either the position (as a number) or the name (as a string) of the moderator variable to place on the x-axis. The regression line then reflects the ‘marginal’ relationship between the chosen moderator and the estimates (i.e., all other moderators except the one being plotted are held constant at their means).

By default (i.e., when `psize` is not specified), the size of the points is a function of the square root of the model weights. This way, their area is proportional to the weights. However, the point sizes are rescaled so that the smallest point size is `plim[1]` and the largest point size is `plim[2]`. As a result, their relative sizes (i.e., areas) no longer exactly correspond to their relative weights. If exactly relative point sizes are desired, one can set `plim[2]` to NA, in which case the points are rescaled so that the smallest point size corresponds to `plim[1]` and all other points are scaled accordingly. As a result, the largest point may be very large. Alternatively, one can set `plim[1]` to NA, in which case the points are rescaled so that the largest point size corresponds to `plim[2]` and all other points are scaled accordingly. As a result, the smallest point may be very small. To avoid the latter, one can also set `plim[3]`, which enforces a minimal point size.

One can also set `psize` to a scalar (e.g., `psize=1`) to avoid that the points are drawn in different sizes. One can also specify the point sizes manually by passing a vector of the appropriate length to `psize`. Finally, one can also set `psize` to either `"seinv"` or `"vinv"` to make the point sizes proportional to the inverse standard errors or inverse sampling variances.

With the `label` argument, one can control whether points in the plot will be labeled. If `label="all"` (or `label=TRUE`), all points in the plot will be labeled. If `label="ciout"` or `label="piout"`, points falling outside of the confidence/prediction interval will be labeled. Alternatively, one can set this argument to a logical or numeric vector to specify which points should be labeled. The labels are placed above the points when they fall above the regression line and otherwise below. With the

offset argument, one can adjust the distance between the labels and the corresponding points. This can either be a single numeric value, which is used as a multiplicative factor for the point sizes (so that the distance between labels and points is larger for larger points) or a numeric vector with two values, where the first is used as an additive factor independent of the point sizes and the second again as a multiplicative factor for the point sizes. The values are given as percentages of the y-axis range. It may take some trial and error to find two values for the offset argument so that the labels are placed right next to the boundary of the points. With labsize, one can control the size of the labels.

One can also pass an object from [predict](#) to the pred argument. This can be useful when the meta-regression model reflects a more complex relationship between the moderator variable and the estimates (e.g., when using polynomials or splines) or when the model involves interactions. In this case, one also needs to specify the xvals argument. See ‘Examples’.

By setting the legend argument to TRUE, a legend is added to the plot. One can also use a keyword for this argument to specify the position of the legend (e.g., legend="topright"; see [legend](#) for options). Finally, this argument can also be a list, with elements x, y, inset, and cex, which are passed on to the corresponding arguments of the [legend](#) function for even more control (elements not specified are set to defaults).

Value

An object of class "regplot" with components:

slab	the study labels
ids	the study ids
xi	the x-axis coordinates of the points that were plotted.
yi	the y-axis coordinates of the points that were plotted.
pch	the plotting symbols of the points that were plotted.
psize	the point sizes of the points that were plotted.
col	the colors of the points that were plotted.
bg	the background colors of the points that were plotted.
label	logical vector indicating whether a point was labeled.

Note that the object is returned invisibly. Using `points.regplot`, one can redraw the points (and labels) in case one wants to superimpose the points on top of any elements that were added manually to the plot (see ‘Examples’).

Note

For certain types of models, it may not be possible to draw the prediction interval bounds (if this is the case, a warning will be issued).

For argument `slab` and when specifying vectors for arguments `pch`, `psize`, `col`, `bg`, and/or `label` (for a logical vector), the variables specified are assumed to be of the same length as the data passed to the model fitting function (and if the data argument was used in the original model fit, then the variables will be searched for within this data frame first). Any subsetting and removal of studies with missing values is automatically applied to the variables specified via these arguments.

If the estimates are bounded (e.g., correlations are bounded between -1 and +1, proportions are bounded between 0 and 1), one can use the `olim` argument to enforce those limits (i.e., the estimates and confidence/prediction interval bounds cannot exceed those limits then).

Author(s)

Wolfgang Viechtbauer (<wvbm@metafor-project.org>, <https://www.metafor-project.org>).

References

- Thompson, S. G., & Higgins, J. P. T. (2002). How should meta-regression analyses be undertaken and interpreted? *Statistics in Medicine*, **21**(11), 1559–1573. <https://doi.org/10.1002/sim.1187>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which scatter plots / bubble plots can be drawn.

Examples

```
### copy BCG vaccine data into 'dat'
dat <- dat.bcg

### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat)

#####

### fit mixed-effects model with absolute latitude as a moderator
res <- rma(yi, vi, mods = ~ ablat, data=dat)
res

### draw plot
regplot(res, mod="ablat", xlab="Absolute Latitude")

### adjust x-axis limits and back-transform to risk ratios
regplot(res, mod="ablat", xlab="Absolute Latitude", xlim=c(0,60), transf=exp)

### also extend the prediction limits for the regression line
regplot(res, mod="ablat", xlab="Absolute Latitude", xlim=c(0,60), predlim=c(0,60), transf=exp)

### add the prediction interval to the plot, add a reference line at 1, and add a legend
regplot(res, mod="ablat", pi=TRUE, xlab="Absolute Latitude",
        xlim=c(0,60), predlim=c(0,60), transf=exp, refline=1, legend=TRUE)

### label points outside of the prediction interval
regplot(res, mod="ablat", pi=TRUE, xlab="Absolute Latitude",
        xlim=c(0,60), predlim=c(0,60), transf=exp, refline=1, legend=TRUE,
        label="piout", labsz=0.8)

#####

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)
```

```

res

### plot the marginal relationships
regplot(res, mod="ablat", xlab="Absolute Latitude")
regplot(res, mod="year", xlab="Publication Year")

#####

### fit a quadratic polynomial meta-regression model
res <- rma(yi, vi, mods = ~ ablat + I(ablat^2), data=dat)
res

### compute predicted values using predict()
xs <- seq(0,60,length=601)
tmp <- predict(res, newmods=cbind(xs, xs^2))

### can now pass these results to the 'pred' argument (and have to specify xvals accordingly)
regplot(res, mod="ablat", pred=tmp, xlab="Absolute Latitude", xlim=c(0,60), xvals=xs)

### back-transform to risk ratios and add reference line
regplot(res, mod="ablat", pred=tmp, xlab="Absolute Latitude", xlim=c(0,60), xvals=xs,
        transf=exp, refline=1)

#####

### fit a model with an interaction between a quantitative and a categorical predictor
### (note: only for illustration purposes; this model is too complex for this dataset)
res <- rma(yi, vi, mods = ~ ablat * alloc, data=dat)
res

### draw bubble plot but do not add regression line or CI
tmp <- regplot(res, mod="ablat", xlab="Absolute Latitude", xlim=c(0,60), pred=FALSE, ci=FALSE)

### add regression lines for the three alloc levels
xs <- seq(0, 60, length=100)
preds <- predict(res, newmods=cbind(xs, 0, 0, 0, 0))
lines(xs, preds$pred, lwd=3)
preds <- predict(res, newmods=cbind(xs, 1, 0, xs, 0))
lines(xs, preds$pred, lwd=3)
preds <- predict(res, newmods=cbind(xs, 0, 1, 0, xs))
lines(xs, preds$pred, lwd=3)

### add points back to the plot (so they are on top of the lines)
points(tmp)

#####

### an example where we place a dichotomous variable on the x-axis

### dichotomize the 'random' variable
dat$random <- ifelse(dat$alloc == "random", 1, 0)

### fit mixed-effects model with this dummy variable as moderator

```

```

res <- rma(yi, vi, mods = ~ random, data=dat)
res

### draw bubble plot
regplot(res, mod="random")

### draw bubble plot and add a nicer x-axis
regplot(res, mod="random", xlab="Method of Treatment Allocation", xaxt="n")
axis(side=1, at=c(0,1), labels=c("Non-Random", "Random"))

#####

### an example where we place a categorical variable with more than two levels
### on the x-axis; this is done with a small trick, representing the moderator
### as a polynomial regression model

### fit mixed-effects model with a three-level factor
res <- rma(yi, vi, mods = ~ alloc, data=dat)
res

### compute the predicted pooled effect for each level of the factor
predict(res, newmods=rbind(alternate=c(0,0), random=c(1,0), systematic=c(0,1)))

### represent the three-level factor as a quadratic polynomial model
dat$anum <- as.numeric(factor(dat$alloc))
res <- rma(yi, vi, mods = ~ poly(anum, degree=2, raw=TRUE), data=dat)
res

### compute the predicted pooled effect for each level of the factor
### (note that these values are exactly the same as above)
pred <- predict(res, newmods=unname(poly(1:3, degree=2, raw=TRUE)))
pred

### draw bubble plot, placing the linear (1:3) term on the x-axis and add a
### nicer x-axis for the three levels
regplot(res, mod=2, pred=pred, xvals=c(1:3), xlim=c(1,3), xlab="Allocation Method", xaxt="n")
axis(side=1, at=1:3, labels=levels(factor(dat$alloc)))

#####

```

regtest

Regression Test for Funnel Plot Asymmetry

Description

Function to carry out (various versions of) Egger's regression test for funnel plot asymmetry.

Usage

```

regtest(x, vi, sei, ni, subset, data,
        model="rma", predictor="sei", ret.fit=FALSE, digits, ...)

```

Arguments

<code>x</code>	a vector with the effect size estimates or an object of class "rma".
<code>vi</code>	vector with the corresponding sampling variances. Ignored if <code>x</code> is an object of class "rma".
<code>sei</code>	vector with the corresponding standard errors (note: only one of the two, <code>vi</code> or <code>sei</code> , needs to be specified).
<code>ni</code>	optional vector with the corresponding sample sizes (only relevant when using the sample sizes (or a transformation thereof) as predictor).
<code>subset</code>	optional (logical or numeric) vector to specify the subset of studies that should be included in the test. Ignored if <code>x</code> is an object of class "rma".
<code>data</code>	optional data frame containing the variables given to the arguments above.
<code>model</code>	either "rma" or "lm" to specify the type of model to use for the regression test. See 'Details'.
<code>predictor</code>	either "sei" "vi", "ni", "ninv", "sqrtni", or "sqrtninv" to specify the predictor to use for the regression test. See 'Details'.
<code>ret.fit</code>	logical to specify whether the full results from the fitted model should also be returned.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded.
<code>...</code>	other arguments.

Details

Various tests for funnel plot asymmetry have been suggested in the literature, including the rank correlation test by Begg and Mazumdar (1994) and the regression test by Egger et al. (1997). Extensions, modifications, and further developments of the regression test are described (among others) by Macaskill et al. (2001), Sterne and Egger (2005), Harbord et al. (2006), Peters et al. (2006), Rücker et al. (2008), and Moreno et al. (2009). The various versions of the regression test differ in terms of the model (either a weighted regression model with a multiplicative dispersion term or a fixed/mixed-effects meta-regression model is used), in terms of the predictor variable that the estimates are hypothesized to be related to when publication bias is present (suggested predictors include the standard error, the sampling variance, and the sample size or transformations thereof), and in terms of the effect size measure used (e.g., for 2×2 table data, one has the choice between various measures). The idea behind the various tests is the same though: If there is a relationship between the estimates and the chosen predictor, then this usually implies asymmetry in the funnel plot, which in turn may be an indication of publication bias. The `regtest` function can be used to carry out various versions of the regression test.

Via argument `x`, one can either pass a vector with the effect size estimates to the function (and then the corresponding sampling variances via `vi` or the standard errors via `sei`) or an object of class "rma".

The model type for the regression test is chosen via the `model` argument, with `model="lm"` for a weighted regression model with a multiplicative dispersion term or `model="rma"` for a (mixed-effects) meta-regression model (the default).

The predictor for the test is chosen via the `predictor` argument:

- `predictor="sei"` for the standard errors (the default),
- `predictor="vi"` for the sampling variances,
- `predictor="ni"` for the sample sizes,
- `predictor="ninv"` for the inverse of the sample sizes,
- `predictor="sqrtni"` for the square root of the sample sizes, or
- `predictor="sqrtninv"` for the inverse square root of the sample sizes.

The effect size measure used for the regression test is simply determined by the values passed to the function or the measure that was used in fitting the original model (when passing an object of class `"rma"` to the function).

When using the sample sizes (or a transformation thereof) as the predictor, one can use the `ni` argument to specify the sample sizes. When `x` is a vector with the estimates and it was computed with [escalc](#), then the sample sizes should automatically be stored as an attribute of `x` and `ni` does not need to be specified. This should also be the case when passing an object of class `"rma"` to the function and the input to the model fitting function came from [escalc](#).

When passing an object of class `"rma"` to the function, arguments such as `method`, `weighted`, and `test` as used during the initial model fitting are also used for the regression test. If the model already included one or more moderators, then `regtest` will add the chosen predictor to the moderator(s) already included in the model. This way, one can test for funnel plot asymmetry after accounting first for the influence of the moderator(s) already included in the model.

The model used for conducting the regression test can also be used to obtain a 'limit estimate' of the pooled effect. In particular, when the standard errors, sampling variances, or inverse (square root) sample sizes are used as the predictor, the model intercept in essence reflects the estimate under infinite precision. This is sometimes (cautiously) interpreted as the pooled estimate adjusted for publication bias.

Value

An object of class `"regtest"`. The object is a list containing the following components:

<code>model</code>	the model used for the regression test.
<code>predictor</code>	the predictor used for the regression test.
<code>zval</code>	the value of the test statistic.
<code>pval</code>	the corresponding p-value
<code>dfs</code>	the degrees of freedom of the test statistic (if the test is based on a t-distribution).
<code>fit</code>	the full results from the fitted model.
<code>est</code>	the limit estimate (only for predictors <code>"sei"</code> , <code>"vi"</code> , <code>"ninv"</code> , or <code>"sqrtninv"</code> and when the model does not contain any additional moderators; <code>NULL</code> otherwise).
<code>ci.lb</code>	lower bound of the confidence interval for the limit estimate.
<code>ci.ub</code>	upper bound of the confidence intervals for the limit estimate.

The results are formatted and printed with the [print](#) function.

Note

The classical ‘Egger test’ is obtained by setting `model="lm"` and `predictor="sei"`. For the random/mixed-effects version of the test, set `model="rma"` (this is the default). See Sterne and Egger (2005) for details on these two types of models/tests.

When conducting a classical ‘Egger test’, the test of the limit estimate is the same as the ‘precision-effect test’ (PET) of Stanley and Doucouliagos (2014). The limit estimate when using the sampling variance as predictor is sometimes called the ‘precision-effect estimate with SE’ (PEESE) (Stanley & Doucouliagos, 2014). A conditional procedure where we use the limit estimate when PET is not significant (i.e., when using the standard error as predictor) and the PEESE (i.e., when using the sampling variance as predictor) when PET is significant is sometimes called the PET-PEESE procedure (Stanley & Doucouliagos, 2014).

All of the tests do not directly test for publication bias, but for a relationship between the estimates and the chosen predictor. If such a relationship is present, then this usually implies asymmetry in the funnel plot, which in turn may be an indication of publication bias. However, it is important to keep in mind that there can be other reasons besides publication bias that could lead to asymmetry in the funnel plot. For example, the sampling variances can be inherently corrected with the estimates for various effect size measures, which can induce artificial asymmetry in the funnel plot unrelated to publication bias (Macaskill et al., 2001; Moreno et al., 2009; Peters et al., 2006; Pustejovsky & Rodgers, 2019; Zwetsloot et al., 2017). Using the sample sizes (or some transformation thereof) as the predictor can mitigate this issue.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Begg, C. B., & Mazumdar, M. (1994). Operating characteristics of a rank correlation test for publication bias. *Biometrics*, **50**(4), 1088–1101. <https://doi.org/10.2307/2533446>
- Egger, M., Davey Smith, G., Schneider, M., & Minder, C. (1997). Bias in meta-analysis detected by a simple, graphical test. *British Medical Journal*, **315**(7109), 629–634. <https://doi.org/10.1136/bmj.315.7109.629>
- Harbord, R. M., Egger, M., & Sterne, J. A. C. (2006). A modified test for small-study effects in meta-analyses of controlled trials with binary endpoints. *Statistics in Medicine*, **25**(20), 3443–3457. <https://doi.org/10.1002/sim.2380>
- Macaskill, P., Walter, S. D., & Irwig, L. (2001). A comparison of methods to detect publication bias in meta-analysis. *Statistics in Medicine*, **20**(4), 641–654. <https://doi.org/10.1002/sim.698>
- Moreno, S. G., Sutton, A. J., Ades, A. E., Stanley, T. D., Abrams, K. R., Peters, J. L., & Cooper, N. J. (2009). Assessment of regression-based methods to adjust for publication bias through a comprehensive simulation study. *BMC Medical Research Methodology*, **9**, 2. <https://doi.org/10.1186/1471-2288-9-2>
- Peters, J. L., Sutton, A. J., Jones, D. R., Abrams, K. R., & Rushton, L. (2006). Comparison of two methods to detect publication bias in meta-analysis. *Journal of the American Medical Association*, **295**(6), 676–680. <https://doi.org/10.1001/jama.295.6.676>
- Pustejovsky, J. E., & Rodgers, M. A. (2019). Testing for funnel plot asymmetry of standardized mean differences. *Research Synthesis Methods*, **10**(1), 57–71. <https://doi.org/10.1002/jrsm.1332>
- Rücker, G., Schwarzer, G., & Carpenter, J. (2008). Arcsine test for publication bias in meta-analyses with binary outcomes. *Statistics in Medicine*, **27**(5), 746–763. <https://doi.org/10.1002/sim.2971>

- Stanley, T. D., & Doucouliagos, H. (2014). Meta-regression approximations to reduce publication selection bias. *Research Synthesis Methods*, 5(1), 60–78. <https://doi.org/10.1002/jrsm.1095>
- Sterne, J. A. C., & Egger, M. (2005). Regression methods to detect publication and other bias in meta-analysis. In H. R. Rothstein, A. J. Sutton, & M. Borenstein (Eds.) *Publication bias in meta-analysis: Prevention, assessment, and adjustments* (pp. 99–110). Chichester, England: Wiley.
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, 36(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Zwetsloot, P.-P., Van Der Naald, M., Sena, E. S., Howells, D. W., IntHout, J., De Groot, J. A. H., Chamuleau, S. A. J., MacLeod, M. R., & Wever, K. E. (2017). Standardized mean differences cause funnel plot distortion in publication bias assessments. *eLife*, 6, e24260. <https://doi.org/10.7554/elife.24260>

See Also

[ranktest](#) for the rank correlation test, [trimfill](#) for the trim and fill method, [tes](#) for the test of excess significance, [fsn](#) to compute the fail-safe N (file drawer analysis), and [selmodel](#) for selection models.

Examples

```
### copy data into 'dat' and examine data
dat <- dat.egger2001

### calculate log odds ratios and corresponding sampling variances (but remove ISIS-4 trial)
dat <- escalc(measure="OR", ai=ai, nli=nli, ci=ci, n2i=n2i, data=dat, subset=-16)

### fit random-effects model
res <- rma(yi, vi, data=dat)
res

### classical Egger test
regtest(res, model="lm")

### mixed-effects meta-regression version of the Egger test
regtest(res)

### same tests, but passing the estimates to the function directly
regtest(yi, vi, data=dat, model="lm")
regtest(yi, vi, data=dat)

### if dat$yi is computed with escalc(), sample size information is stored in attributes
dat$yi

### then this will also work
regtest(yi, vi, data=dat, predictor="ni")

### similarly when passing a model object to the function
regtest(res, model="lm", predictor="ni")
regtest(res, model="lm", predictor="ninv")
regtest(res, predictor="ni")
regtest(res, predictor="ninv")
```

```

### otherwise have to supply sample sizes manually
dat$yi <- c(dat$yi) # this removes the 'ni' attribute from 'yi'
dat$nitotal <- with(dat, n1i + n2i)
regtest(yi, vi, ni=nitotal, data=dat, predictor="ni")
res <- rma(yi, vi, data=dat)
regtest(res, predictor="ni", ni=nitotal, data=dat)

### standard funnel plot (with standard errors on the y-axis)
funnel(res, refline=0)

### regression test (by default the standard errors are used as predictor)
reg <- regtest(res)
reg

### add regression line to funnel plot
se <- seq(0,1.8,length=100)
lines(coef(reg$fit)[1] + coef(reg$fit)[2]*se, se, lwd=3)

### regression test (using the sampling variances as predictor)
reg <- regtest(res, predictor="vi")

### add regression line to funnel plot (using the sampling variances as predictor)
lines(coef(reg$fit)[1] + coef(reg$fit)[2]*se^2, se, lwd=3, lty="dotted")

### add legend
legend("bottomright", inset=0.02, lty=c("solid","dotted"), lwd=3, cex=0.9, bg="white",
      legend=c("Standard Errors as Predictor",
               "Sampling Variances as Predictor"))

### testing for asymmetry after accounting for the influence of a moderator
res <- rma(yi, vi, mods = ~ year, data=dat)
regtest(res, model="lm")
regtest(res)

```

replmiss

Replace Missing Values in a Vector

Description

Function to replace missing (NA) values in a vector.

Usage

```
replmiss(x, y, data)
```

Arguments

x	vector that may include one or more missing values.
y	either a scalar or a vector of the same length as x with the value(s) to replace missing values with.
data	optional data frame containing the variables given to the arguments above.

Value

Vector `x` with the missing values replaced based on the scalar or vector `y`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

Examples

```
x <- c(4,2,7,NA,1,NA,5)
x <- replmiss(x,0)
x

x <- c(4,2,7,NA,1,NA,5)
y <- c(2,3,6,5,8,1,2)
x <- replmiss(x,y)
x
```

reporter

Dynamically Generated Analysis Reports for 'rma.uni' Objects

Description

Function to dynamically generate an analysis report for objects of class `"rma.uni"`.

Usage

```
reporter(x, ...)

## S3 method for class 'rma.uni'
reporter(x, dir, filename, format="html_document", open=TRUE,
         digits, forest, funnel, footnotes=FALSE, verbose=TRUE, ...)
```

Arguments

<code>x</code>	an object of class <code>"rma.uni"</code> .
<code>dir</code>	optional character string to specify the directory for creating the report. If unspecified, <code>tempdir</code> will be used.
<code>filename</code>	optional character string to specify the filename (without file extension) for the report. If unspecified, the function sets a filename automatically.
<code>format</code>	output format for the report (either <code>html_document</code> , <code>pdf_document</code> , or <code>word_document</code>). Can be abbreviated. See 'Note'.
<code>open</code>	logical to specify whether the report should be opened after it has been generated (the default is <code>TRUE</code>). See 'Note'.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.

forest	either a logical which will suppress the drawing of the forest plot when set to FALSE or a character string with arguments to be added to the call to <code>forest</code> for generating the forest plot.
funnel	either a logical which will suppress the drawing of the funnel plot when set to FALSE or a character string with arguments to be added to the call to <code>funnel</code> for generating the funnel plot.
footnotes	logical to specify whether additional explanatory footnotes should be added to the report (the default is FALSE).
verbose	logical to specify whether information on the progress of the report generation should be provided (the default is TRUE).
...	other arguments.

Details

The function dynamically generates an analysis report based on the model object. The report includes information about the model that was fitted, the distribution of the effect size estimates, the pooled estimate based on the fitted model, tests and statistics that are informative about potential heterogeneity in the true effects, checks for outliers and/or influential studies, and tests for funnel plot asymmetry. By default, a forest plot and a funnel plot are also provided (these can be suppressed by setting `forest=FALSE` and/or `funnel=FALSE`).

Value

The function generates either a html, pdf, or docx file and returns (invisibly) the path to the generated document.

Note

Since the report is created based on an R Markdown document that is generated by the function, the `rmarkdown` package and `pandoc` must be installed.

To render the report into a pdf document (i.e., using `format="pdf_document"`) requires a LaTeX installation. If LaTeX is not already installed, you could try using the `tinytex` package to install a lightweight LaTeX distribution based on TeX Live.

Once the report is generated, the function opens the output file (either a .html, .pdf, or .docx file) with an appropriate application (if `open=TRUE`). This will only work when an appropriate application for the file type is installed and associated with the extension.

If `filename` is unspecified, the default is to use `report`, followed by an underscore (i.e., `_`) and the name of the object passed to the function. Both the R Markdown file (with extension `.rmd`) and the actual report (with extension `.html`, `.pdf`, or `.docx`) are named accordingly. To generate the report, the model object is also saved to a file (with the same filename as above, but with extension `.rdata`). Also, files `references.bib` and `apa.csl` are copied to the same directory (these files are needed to generate the references in APA format).

Since the report is put together based on predefined text blocks, the writing is not very elegant. Also, using personal pronouns ('I' or 'we') does not make sense for such a report, so a lot of passive voice is used.

The generated report provides an illustration of how the results of the model can be reported, but is not a substitute for a careful examination of the results.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#) for the function to fit models for which an analysis report can be generated.

Examples

```
### copy BCG vaccine data into 'dat'
dat <- dat.bcg

### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat,
             slab=paste(author, ", ", year, sep=""))

### fit random-effects model
res <- rma(yi, vi, data=dat)

## Not run:
### generate report
reporter(res)

## End(Not run)
```

residuals.rma

Residual Values based on 'rma' Objects

Description

Functions to compute residuals and standardized versions thereof for models fitted with the [rma.uni](#), [rma.mh](#), [rma.peto](#), and [rma.mv](#) functions.

Usage

```
## S3 method for class 'rma'
residuals(object, type="response", ...)

## S3 method for class 'rma.uni'
rstandard(model, digits, type="marginal", ...)
## S3 method for class 'rma.mh'
rstandard(model, digits, ...)
## S3 method for class 'rma.peto'
```

```

rstandard(model, digits, ...)
## S3 method for class 'rma.mv'
rstandard(model, digits, cluster, ...)

## S3 method for class 'rma.uni'
rstudent(model, digits, progbar=FALSE, ...)
## S3 method for class 'rma.mh'
rstudent(model, digits, progbar=FALSE, ...)
## S3 method for class 'rma.peto'
rstudent(model, digits, progbar=FALSE, ...)
## S3 method for class 'rma.mv'
rstudent(model, digits, progbar=FALSE, cluster,
          reestimate=TRUE, parallel="no", ncpus=1, cl, ...)

```

Arguments

<code>object</code>	an object of class "rma" (for residuals).
<code>type</code>	the type of residuals which should be returned. For residuals, the alternatives are: "response" (default), "rstandard", "rstudent", and "pearson". For <code>rstandard.rma.uni</code> , the alternatives are: "marginal" (default) and "conditional". See 'Details'.
<code>model</code>	an object of class "rma" (for residuals) or an object of class "rma.uni", "rma.mh", "rma.peto", or "rma.mv" (for <code>rstandard</code> and <code>rstudent</code>).
<code>cluster</code>	optional vector to specify a clustering variable to use for computing cluster-level multivariate standardized residuals (only for "rma.mv" objects).
<code>reestimate</code>	logical to specify whether variance/correlation components should be re-estimated after deletion of the <i>i</i> th case when computing externally standardized residuals for "rma.mv" objects (the default is TRUE).
<code>parallel</code>	character string to specify whether parallel processing should be used (the default is "no"). For parallel processing, set to either "snow" or "multicore". See 'Note'.
<code>ncpus</code>	integer to specify the number of processes to use in the parallel processing.
<code>cl</code>	optional cluster to use if <code>parallel="snow"</code> . If unspecified, a cluster on the local machine is created for the duration of the call.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
<code>progbar</code>	logical to specify whether a progress bar should be shown (only for <code>rstudent</code>) (the default is FALSE).
<code>...</code>	other arguments.

Details

The observed residuals (obtained with `residuals`) are simply equal to the 'observed - fitted' values. These can be obtained with `residuals(object)` (using the default `type="response"`).

Dividing the observed residuals by the model-implied standard errors of the estimates yields Pearson (or semi-standardized) residuals. These can be obtained with `residuals(object, type="pearson")`.

Dividing the observed residuals by their corresponding standard errors yields (internally) standardized residuals. These can be obtained with `rstandard(model)` or `residuals(object, type="rstandard")`.

With `rstudent(model)` (or `residuals(object, type="rstudent")`), one can obtain the externally standardized residuals (also called standardized deleted residuals or (externally) studentized residuals). The externally standardized residual for the i th case is obtained by deleting the i th case from the dataset, fitting the model based on the remaining cases, calculating the predicted value for the i th case based on the fitted model, taking the difference between the observed and the predicted value for the i th case (which yields the deleted residual), and then standardizing the deleted residual based on its standard error.

If a particular case fits the model, its standardized residual follows (asymptotically) a standard normal distribution. A large standardized residual for a case therefore may suggest that the case does not fit the assumed model (i.e., it may be an outlier).

For "rma.uni" objects, `rstandard(model, type="conditional")` computes conditional residuals, which are the deviations of the effect size estimates from the best linear unbiased predictions (BLUPs) of the study-specific true effects (see [blup](#)).

For "rma.mv" objects, one can specify a clustering variable (via the `cluster` argument). If specified, `rstandard(model)` and `rstudent(model)` also compute cluster-level multivariate (internally or externally) standardized residuals. If all estimates within a cluster fit the model, then the multivariate standardized residual for the cluster follows (asymptotically) a chi-square distribution with k_i degrees of freedom (where k_i denotes the number of estimates within the cluster).

See also [influence.rma.uni](#) and [influence.rma.mv](#) for other leave-one-out diagnostics that are useful for detecting influential cases in models fitted with the [rma.uni](#) and [rma.mv](#) functions.

Value

Either a vector with the residuals of the requested type (for residuals) or an object of class "list.rma", which is a list containing the following components:

<code>resid</code>	observed residuals (for <code>rstandard</code>) or deleted residuals (for <code>rstudent</code>).
<code>se</code>	corresponding standard errors.
<code>z</code>	standardized residuals (internally standardized for <code>rstandard</code> or externally standardized for <code>rstudent</code>).

When a clustering variable is specified for "rma.mv" objects, the returned object is a list with the first element (named `obs`) as described above and a second element (named `cluster`) of class "list.rma" with:

<code>X2</code>	cluster-level multivariate standardized residuals.
<code>k</code>	number of estimates within the clusters.

The object is formatted and printed with [print](#). To format the results as a data frame, one can use the [as.data.frame](#) function.

Note

The externally standardized residuals (obtained with `rstudent`) are calculated by refitting the model k times (where k denotes the number of cases). Depending on how large k is, it may take a few moments to finish the calculations. For complex models fitted with `rma.mv`, this can become computationally expensive.

On machines with multiple cores, one can try to speed things up by delegating the model fitting to separate worker processes, that is, by setting `parallel="snow"` or `parallel="multicore"` and `ncpus` to some value larger than 1 (only for objects of class `"rma.mv"`). Parallel processing makes use of the `parallel` package, using the `makePSOCKcluster` and `parLapply` functions when `parallel="snow"` or using `mclapply` when `parallel="multicore"` (the latter only works on Unix/Linux-alikes).

Alternatively (or in addition to using parallel processing), one can also set `reestimate=FALSE`, in which case any variance/correlation components in the model are not re-estimated after deleting the i th case from the dataset. Doing so only yields an approximation to the externally standardized residuals (and the cluster-level multivariate standardized residuals) that ignores the influence of the i th case on the variance/correlation components, but is considerably faster (and often yields similar results).

It may not be possible to fit the model after deletion of the i th case from the dataset. This will result in NA values for that case when calling `rstudent`.

Also, for `"rma.mv"` objects with a clustering variable specified, it may not be possible to compute the cluster-level multivariate standardized residual for a particular cluster (if the var-cov matrix of the residuals within a cluster is not of full rank). This will result in NA for that cluster.

The variable specified via `cluster` is assumed to be of the same length as the data originally passed to the `rma.mv` function (and if the `data` argument was used in the original model fit, then the variable will be searched for within this data frame first). Any subsetting and removal of studies with missing values that was applied during the model fitting is also automatically applied to the variable specified via the `cluster` argument.

For objects of class `"rma.mh"` and `"rma.peto"`, `rstandard` actually computes Pearson (or semi-standardized) residuals.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. San Diego, CA: Academic Press.

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/9781315>

Viechtbauer, W., & Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research Synthesis Methods*, **1**(2), 112–125. <https://doi.org/10.1002/jrsm.11>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which the various types of residuals can be computed.

[influence.rma.uni](#) and [influence.rma.mv](#) for other model diagnostics.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model
res <- rma(yi, vi, data=dat)

### compute the studentized residuals
rstudent(res)

### fit mixed-effects model with absolute latitude as moderator
res <- rma(yi, vi, mods = ~ ablat, data=dat)

### compute the studentized residuals
rstudent(res)
```

rma.glmm

Meta-Analysis via Generalized Linear (Mixed-Effects) Models

Description

Function to fit meta-analytic equal-, fixed-, and random-effects models and (mixed-effects) meta-regression models using a generalized linear (mixed-effects) model framework. See below and the introduction to the [metafor-package](#) for more details on these models.

Usage

```
rma.glmm(ai, bi, ci, di, n1i, n2i, x1i, x2i, t1i, t2i, xi, mi, ti, ni,
         mods, measure, data, slab, subset,
         add=1/2, to="only0", drop00=TRUE, intercept=TRUE,
         model="UM.FS", method="ML", coding=1/2, cor=FALSE, test="z",
         level=95, btt, nAGQ=7, verbose=FALSE, digits, control, ...)
```

Arguments

These arguments pertain to data input:

ai	see below and the documentation of the escalc function for more details.
bi	see below and the documentation of the escalc function for more details.
ci	see below and the documentation of the escalc function for more details.
di	see below and the documentation of the escalc function for more details.

n1i	see below and the documentation of the escalc function for more details.
n2i	see below and the documentation of the escalc function for more details.
x1i	see below and the documentation of the escalc function for more details.
x2i	see below and the documentation of the escalc function for more details.
t1i	see below and the documentation of the escalc function for more details.
t2i	see below and the documentation of the escalc function for more details.
xi	see below and the documentation of the escalc function for more details.
mi	see below and the documentation of the escalc function for more details.
ti	see below and the documentation of the escalc function for more details.
ni	see below and the documentation of the escalc function for more details.
mods	optional argument to include one or more moderators in the model. A single moderator can be given as a vector of length k specifying the values of the moderator. Multiple moderators are specified by giving a matrix with k rows and as many columns as there are moderator variables. Alternatively, a model formula can be used to specify the model. See ‘Details’.
measure	character string to specify the effect size measure to use for the meta-analysis. Possible options are "OR" for the (log transformed) odds ratio, "IRR" for the (log transformed) incidence rate ratio, "PLO" for the (logit transformed) proportion, or "IRLN" for the (log transformed) incidence rate.
data	optional data frame containing the data supplied to the function.
slab	optional vector with labels for the k studies.
subset	optional (logical or numeric) vector to specify the subset of studies that should be used for the analysis.

These arguments pertain to handling of zero cells/counts/frequencies:

add	non-negative number to specify the amount to add to zero cells, counts, or frequencies when calculating the effect size estimates of the individual studies. See below and the documentation of the escalc function for more details.
to	character string to specify when the values under add should be added (either "only0", "all", "if0all", or "none"). See below and the documentation of the escalc function for more details.
drop00	logical to specify whether studies with no cases/events (or only cases) in both groups should be dropped. See the documentation of the escalc function for more details.

These arguments pertain to the model / computations and output:

intercept	logical to specify whether an intercept should be added to the model (the default is TRUE).
model	character string to specify the general model type for the analysis. Either "UM.FS" (the default), "UM.RS", "CM.EL", or "CM.AL". See ‘Details’.
method	character string to specify whether an equal- or a random-effects model should be fitted. An equal-effects model is fitted when using method="EE". A random-effects model is fitted by setting method="ML" (the default). See ‘Details’.

coding	numeric scalar to specify how the group variable should be coded in the random effects structure for random/mixed-effects models (the default is 1/2). See ‘Note’.
cor	logical to specify whether the random study effects should be allowed to be correlated with the random group effects for random/mixed-effects models when <code>model="UM.RS"</code> (the default is FALSE). See ‘Note’.
test	character string to specify how test statistics and confidence intervals for the fixed effects should be computed. By default (<code>test="z"</code>), Wald-type tests and CIs are obtained, which are based on a standard normal distribution. When <code>test="t"</code> , a t-distribution is used instead. See ‘Details’ and also here for some recommended practices.
level	numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see here for details).
btt	optional vector of indices to specify which coefficients to include in the omnibus test of moderators. Can also be a string to grep for. See ‘Details’.
nAGQ	positive integer to specify the number of points per axis for evaluating the adaptive Gauss-Hermite approximation to the log-likelihood. The default is 7. Setting this to 1 corresponds to the Laplacian approximation. See ‘Note’.
verbose	logical to specify whether output should be generated on the progress of the model fitting (the default is FALSE). Can also be an integer. Values > 1 generate more verbose output. See ‘Note’.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is 4. See also here for further details on how to control the number of digits in the output.
control	optional list of control values for the estimation algorithms. If unspecified, default values are defined inside the function. See ‘Note’.
...	additional arguments.

Details

Specifying the Data:

The function can be used in combination with the following effect size measures:

- `measure="OR"` for (log transformed) odds ratios,
- `measure="IRR"` for (log transformed) incidence rate ratios,
- `measure="PLO"` for (logit transformed) proportions (i.e., log odds),
- `measure="IRLN"` for (log transformed) incidence rates.

The [escalc](#) function describes the data/arguments that should be specified/used for these measures.

Specifying the Model:

A variety of model types are available when analyzing 2×2 table data (i.e., when `measure="OR"`) or two-group event count data (i.e., when `measure="IRR"`):

- `model="UM.FS"` for an unconditional generalized linear mixed-effects model with fixed study effects,

- `model="UM.RS"` for an unconditional generalized linear mixed-effects model with random study effects,
- `model="CM.AL"` for a conditional generalized linear mixed-effects model (approximate likelihood),
- `model="CM.EL"` for a conditional generalized linear mixed-effects model (exact likelihood).

For `measure="OR"`, models `"UM.FS"` and `"UM.RS"` are essentially (mixed-effects) logistic regression models, while for `measure="IRR"`, these models are (mixed-effects) Poisson regression models. The difference between `"UM.FS"` and `"UM.RS"` is how study level variability is modeled. One can choose between using fixed study effects (which means that k dummy variables are added to the model) or random study effects (which means that random effects corresponding to the levels of the study factor are added to the model).

The conditional model (`model="CM.EL"`) avoids having to model study level variability by conditioning on the total numbers of cases/events in each study. For `measure="OR"`, this leads to a non-central hypergeometric distribution for the data within each study and the corresponding model is then a (mixed-effects) conditional logistic model. Fitting this model can be difficult and computationally expensive. When the number of cases in each study is small relative to the group sizes, one can approximate the exact likelihood by a binomial distribution, which leads to a regular (mixed-effects) logistic regression model (`model="CM.AL"`). For `measure="IRR"`, the conditional model leads directly to a binomial distribution for the data within each study and the resulting model is again a (mixed-effects) logistic regression model (no approximate likelihood model is needed here).

When analyzing proportions (i.e., `measure="PLO"`) or incidence rates (i.e., `measure="IRLN"`) of individual groups, the model type is always a (mixed-effects) logistic or Poisson regression model, respectively (i.e., the `model` argument is not relevant here).

Aside from choosing the general model type, one has to decide whether to fit an equal- or a random-effects model to the data. An *equal-effects model* is fitted by setting `method="EE"`. A *random-effects model* is fitted by setting `method="ML"` (the default). Note that random-effects models with dichotomous data are often referred to as ‘binomial-normal’ models in the meta-analytic literature. Analogously, for event count data, such models could be referred to as ‘Poisson-normal’ models.

One or more moderators can be included in a model via the `mods` argument. A single moderator can be given as a (row or column) vector of length k specifying the values of the moderator. Multiple moderators are specified by giving an appropriate model matrix (i.e., X) with k rows and as many columns as there are moderator variables (e.g., `mods = cbind(mod1, mod2, mod3)`, where `mod1`, `mod2`, and `mod3` correspond to the names of the variables for three moderator variables). The intercept is added to the model matrix by default unless `intercept=FALSE`.

Alternatively, one can use standard [formula](#) syntax to specify the model. In this case, the `mods` argument should be set equal to a one-sided formula of the form `mods = ~ model` (e.g., `mods = ~ mod1 + mod2 + mod3`). Interactions, polynomial/spline terms, and factors can be easily added to the model in this manner. When specifying a model formula via the `mods` argument, the `intercept` argument is ignored. Instead, the inclusion/exclusion of the intercept is controlled by the specified formula (e.g., `mods = ~ 0 + mod1 + mod2 + mod3` or `mods = ~ mod1 + mod2 + mod3 - 1` would lead to the removal of the intercept).

Equal-, Saturated-, and Random/Mixed-Effects Models:

When fitting a particular model, actually up to three different models are fitted within the function:

- the equal-effects model (i.e., where τ^2 is set to 0),

- the saturated model (i.e., the model with a deviance of 0), and
- the random/mixed-effects model (i.e., where τ^2 is estimated) (only if `method="ML"`).

The saturated model is obtained by adding as many dummy variables to the model as needed so that the model deviance is equal to zero. Even when `method="ML"`, the equal- and saturated models are also fitted, as they are used to compute the test statistics for the Wald-type and likelihood ratio tests for (residual) heterogeneity (see below).

Omnibus Test of Moderators:

For models including moderators, an omnibus test of all model coefficients is conducted that excludes the intercept (the first coefficient) if it is included in the model. If no intercept is included in the model, then the omnibus test includes all coefficients in the model including the first. Alternatively, one can manually specify the indices of the coefficients to test via the `btt` ('betas to test') argument (i.e., to test $H_0: \beta_{j \in \text{btt}} = 0$, where $\beta_{j \in \text{btt}}$ is the set of coefficients to be tested). For example, with `btt=c(3, 4)`, only the third and fourth coefficients from the model are included in the test (if an intercept is included in the model, then it corresponds to the first coefficient in the model). Instead of specifying the coefficient numbers, one can specify a string for `btt`. In that case, `grep` will be used to search for all coefficient names that match the string. The omnibus test is called the Q_M -test and follows asymptotically a chi-square distribution with m degrees of freedom (with m denoting the number of coefficients tested) under the null hypothesis (that the true value of all coefficients tested is equal to 0).

Categorical Moderators:

Categorical moderator variables can be included in the model via the `mods` argument in the same way that appropriately (dummy) coded categorical variables can be included in linear models. One can either do the dummy coding manually or use a model formula together with the `factor` function to automate the coding (note that string/character variables in a model formula are automatically converted to factors).

Tests and Confidence Intervals:

By default, tests of individual coefficients in the model (and the corresponding confidence intervals) are based on a standard normal distribution, while the omnibus test is based on a chi-square distribution (see above). As an alternative, one can set `test="t"`, in which case tests of individual coefficients and confidence intervals are based on a t-distribution with $k - p$ degrees of freedom, while the omnibus test then uses an F-distribution with m and $k - p$ degrees of freedom (with k denoting the total number of estimates included in the analysis and p the total number of model coefficients including the intercept if it is present). Note that `test="t"` is not the same as `test="knha"` in `rma.uni`, as no adjustment to the standard errors of the estimated coefficients is made.

Tests for (Residual) Heterogeneity:

Two different tests for (residual) heterogeneity are automatically carried out by the function. The first is a Wald-type test, which tests the coefficients corresponding to the dummy variables added in the saturated model for significance. The second is a likelihood ratio test, which tests the same set of coefficients, but does so by computing -2 times the difference in the log-likelihoods of the equal-effects and the saturated models. These two tests are not identical for the types of models fitted by the `rma.glmm` function and may even lead to conflicting conclusions.

Calculating Effect Size Estimates of the Individual Studies:

The various models do not require the calculation of the effect size estimates of the individual studies (e.g., the observed log odds ratios of the k studies) and directly make use of the cell/event counts. Zero cells/events are not a problem (except in extreme cases, such as when one of the two outcomes never occurs or when there are no events in any of the studies). Therefore, it is unnecessary to add some constant to the cell/event counts when there are zero cells/events.

However, for plotting and various other functions, it is necessary to calculate the effect size estimates for the k studies. Here, zero cells/events can be problematic, so adding a constant value to the cell/event counts ensures that all k values can be calculated. The `add` and `to` arguments are used to specify what value should be added to the cell/event counts and under what circumstances when calculating the estimates. The documentation of the `escalc` function explains how the `add` and `to` arguments work. Note that `drop00` is set to `TRUE` by default, since studies where $a_i=c_i=0$ or $b_i=d_i=0$ or studies where $x_{1i}=x_{2i}=0$ are uninformative about the size of the effect.

Value

An object of class `c("rma.glmm", "rma")`. The object is a list containing the following components:

<code>beta</code>	estimated coefficients of the model.
<code>se</code>	standard errors of the coefficients.
<code>zval</code>	test statistics of the coefficients.
<code>pval</code>	corresponding p-values.
<code>ci.lb</code>	lower bound of the confidence intervals for the coefficients.
<code>ci.ub</code>	upper bound of the confidence intervals for the coefficients.
<code>vb</code>	variance-covariance matrix of the estimated coefficients.
<code>tau2</code>	estimated amount of (residual) heterogeneity. Always 0 when <code>method="EE"</code> .
<code>sigma2</code>	estimated amount of study level variability (only for <code>model="UM.RS"</code>).
<code>k</code>	number of studies included in the analysis.
<code>p</code>	number of coefficients in the model (including the intercept).
<code>m</code>	number of coefficients included in the omnibus test of moderators.
<code>QE.Wld</code>	Wald-type test statistic of the test for (residual) heterogeneity.
<code>QEp.Wld</code>	corresponding p-value.
<code>QE.LRT</code>	likelihood ratio test statistic of the test for (residual) heterogeneity.
<code>QEp.LRT</code>	corresponding p-value.
<code>QM</code>	test statistic of the omnibus test of moderators.
<code>QMp</code>	corresponding p-value.
<code>I2</code>	value of I^2 .
<code>H2</code>	value of H^2 .
<code>int.only</code>	logical that indicates whether the model is an intercept-only model.
<code>yi, vi, X</code>	the vector of estimates, the corresponding sampling variances, and the model matrix.
<code>fit.stats</code>	a list with the log-likelihood, deviance, AIC, BIC, and AICc values.
<code>...</code>	some additional elements/values.

Methods

The results of the fitted model are formatted and printed with the `print` function. If fit statistics should also be given, use `summary` (or use the `fitstats` function to extract them).

Note

When `measure="OR"` or `measure="IRR"`, `model="UM.FS"` or `model="UM.RS"`, and `method="ML"`, one has to choose a coding scheme for the group variable in the random effects structure. When `coding=1/2` (the default), the two groups are coded with $+1/2$ and $-1/2$ (i.e., contrast coding), which is invariant under group label switching.

When `coding=1`, the first group is coded with 1 and the second group with 0. Finally, when `coding=0`, the first group is coded with 0 and the second group with 1. Note that these coding schemes are not invariant under group label switching.

When `model="UM.RS"` and `method="ML"`, one has to decide whether the random study effects are allowed to be correlated with the random group effects. By default (i.e., when `cor=FALSE`), no such correlation is allowed (which is typically an appropriate assumption when `coding=1/2`). When using a different coding scheme for the group variable (i.e., `coding=1` or `coding=0`), allowing the random study and group effects to be correlated (i.e., using `cor=TRUE`) is usually recommended.

Fitting the various types of models requires several different iterative algorithms:

- For `model="UM.FS"` and `model="CM.AL"`, iteratively reweighted least squares (IWLS) as implemented in the `glm` function is used for fitting the equal-effects and the saturated models. For `method="ML"`, adaptive Gauss-Hermite quadrature as implemented in the `glmer` function is used. The same applies when `model="CM.EL"` is used in combination with `measure="IRR"` or when `measure="PLO"` or `measure="IRLN"` (regardless of the model type).
- For `model="UM.RS"`, adaptive Gauss-Hermite quadrature as implemented in the `glmer` function is used to fit all of the models.
- For `model="CM.EL"` and `measure="OR"`, the quasi-Newton method optimizer as implemented in the `nlminb` function is used by default for fitting the equal-effects and the saturated models. For `method="ML"`, the same algorithm is used, together with adaptive quadrature as implemented in the `integrate` function (for the integration over the density of the non-central hypergeometric distribution). Standard errors of the parameter estimates are obtained by inverting the Hessian, which is numerically approximated using the `hessian` function from the `numDeriv` package. One can also set `control=list(hesspack="pracma")` or `control=list(hesspack="calculus")`, in which case the `pracma::hessian` or `calculus::hessian` functions from the respective packages are used instead for approximating the Hessian. When τ^2 is estimated to be smaller than 10^{-4} , then τ^2 is effectively treated as zero for computing the standard errors (which helps to avoid numerical problems in approximating the Hessian). This cutoff can be adjusted via the `tau2tol` control argument (e.g., `control=list(tau2tol=0)` to switch off this behavior). One can also choose a different optimizer from `optim` via the `control` argument (e.g., `control=list(optimizer="BFGS")` or `control=list(optimizer="Nelder-Mead")`). Besides `nlminb` and one of the methods from `optim`, one can also choose one of the optimizers from the `minqa` package (i.e., `uobyqa`, `newuoa`, or `bobyqa`), one of the (derivative-free) algorithms from the `nloptr` package, the Newton-type algorithm implemented in `nlm`, the various algorithms implemented in the `dfoptim` package (`hjk` for the Hooke-Jeeves, `nmk` for the Nelder-Mead, and `mads` for the Mesh Adaptive Direct Searches algorithm), the quasi-Newton type optimizers `ucminf` and `lbfgsb3c` and the subspace-searching simplex algorithm `subplex` from the packages of the same name, the

Barzilai-Borwein gradient decent method implemented in `BBoptim`, the `Rcgmin` and `Rvmin` optimizers, or the parallelized version of the L-BFGS-B algorithm implemented in `optimParallel` from the package of the same name.

The optimizer name must be given as a character string (i.e., in quotes). Additional control parameters can be specified via the `optCtrl` elements of the control argument (e.g., `control=list(optCtrl=list(iter.max=1000, rel.tol=1e-8))`). For `nloptr`, the default is to use the BOBYQA implementation from that package with a relative convergence criterion of $1e-8$ on the function value (i.e., log-likelihood), but this can be changed via the `algorithm` and `fstop_rel` arguments (e.g., `control=list(optimizer="nloptr", optCtrl=list(algorithm="NLOPT_LN_SBPLX", fstop_rel=1e-6))`). For `optimParallel`, the control argument `ncpus` can be used to specify the number of cores to use for the parallelization (e.g., `control=list(optimizer="optimParallel", ncpus=2)`).

When `model="CM.EL"` and `measure="OR"`, actually `model="CM.AL"` is used first to obtain starting values for `optim`, so either 4 (if `method="EE"`) or 6 (if `method="ML"`) models need to be fitted in total.

Various additional control parameters can be adjusted via the control argument:

- `glmCtrl` is a list of named arguments to be passed on to the control argument of the `glm` function,
- `glmerCtrl` is a list of named arguments to be passed on to the control argument of the `glmer` function,
- `intCtrl` is a list of named arguments (i.e., `rel.tol` and subdivisions) to be passed on to the `integrate` function, and
- `hessianCtrl` is a list of named arguments to be passed on to the `method.args` argument of the `hessian` function. Most important is the `r` argument, which is set to 16 by default (i.e., `control=list(hessianCtrl=list(r=16))`). If the Hessian cannot be inverted, it may be necessary to adjust the `r` argument to a different number (e.g., try `r=4`, `r=6`, or `r=8`).

Also, for `glmer`, the `nAGQ` argument is used to specify the number of quadrature points. The default value is 7, which should provide sufficient accuracy in the evaluation of the log-likelihood in most cases, but at the expense of speed. Setting this to 1 corresponds to the Laplacian approximation (which is faster, but less accurate). Note that `glmer` does not allow values of `nAGQ > 1` when `model="UM.RS"` and `method="ML"`, so this value is automatically set to 1 for this model.

Instead of `glmer`, one can also choose to use `mixed_model` from the `GLMMadaptive` package or `glmmTMB` from the `glmmTMB` package for the model fitting. This is done by setting `control=list(package="GLMMadaptive")` or `control=list(package="glmmTMB")`, respectively.

Information on the progress of the various algorithms can be obtained by setting `verbose=TRUE`. Since fitting the various models can be computationally expensive, this option is useful to determine how the model fitting is progressing. One can also set `verbose` to an integer (`verbose=2` yields even more information and `verbose=3` also sets `option(warn=1)` temporarily).

For `model="CM.EL"` and `measure="OR"`, optimization involves repeated calculation of the density of the non-central hypergeometric distribution. When `method="ML"`, this also requires integration over the same density. This is currently implemented in a rather brute-force manner and may not be numerically stable, especially when models with moderators are fitted. Stability can be improved by scaling the moderators in a similar manner (i.e., don't use a moderator that is coded 0 and 1, while another uses values in the 1000s). For models with an intercept and moderators, the function

actually rescales (non-dummy) variables to z-scores during the model fitting (results are given after back-scaling, so this should be transparent to the user). For models without an intercept, this is not done, so sensitivity analyses are highly recommended here (to ensure that the results do not depend on the scaling of the moderators). Also, if a warning is issued that the standard errors of the fixed effects are unusually small, one should try sensitivity analyses with different optimizers and/or adjusted settings for the `hessianCtrl` and `tau2tol` control arguments.

Finally, there is also (experimental!) support for the following measures:

- `measure="RR"` for log transformed risk ratios,
- `measure="RD"` for raw risk differences,
- `measure="PLN"` for log transformed proportions,
- `measure="PR"` for raw proportions,

(the first two only for models "UM.FS" and "UM.RS") by using log and identity links for the binomial models. However, model fitting with these measures will often lead to numerical problems.

Via the (undocumented) `link` argument, one can also directly adjust the link function that is used (by default, measures "OR" and "PLO" use a "logit" link, measures "RR" and "PLN" use a "log" link, measures "RD" and "PR" use an "identity" link, and measures "IRR" and "IRLN" use a "log" link). See [family](#) for alternative options. Changing these defaults is only recommended for users familiar with the consequences and the interpretation of the resulting estimates (when misused, the results could be meaningless).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

Code for computing the density of the non-central hypergeometric distribution comes from the [MCMCpack](#) package, which in turn is based on Liao and Rosen (2001).

References

- Agresti, A. (2002). *Categorical data analysis* (2nd. ed). Hoboken, NJ: Wiley.
- Bagos, P. G., & Nikolopoulos, G. K. (2009). Mixed-effects Poisson regression models for meta-analysis of follow-up studies with constant or varying durations. *The International Journal of Biostatistics*, **5**(1). <https://doi.org/10.2202/1557-4679.1168>
- Jackson, D., Law, M., Stijnen, T., Viechtbauer, W., & White, I. R. (2018). A comparison of seven random-effects models for meta-analyses that estimate the summary odds ratio. *Statistics in Medicine*, **37**(7), 1059–1085. <https://doi.org/10.1002/sim.7588>
- Liao, J. G., & Rosen, O. (2001). Fast and stable algorithms for computing and sampling from the noncentral hypergeometric distribution. *American Statistician*, **55**(4), 366–369. <https://doi.org/10.1198/000313001753>
- Simmonds, M. C., & Higgins, J. P. T. (2016). A general framework for the use of logistic regression models in meta-analysis. *Statistical Methods in Medical Research*, **25**(6), 2858–2877. <https://doi.org/10.1177/0962280214534409>
- Stijnen, T., Hamza, T. H., & Ozdemir, P. (2010). Random effects meta-analysis of event outcome in the framework of the generalized linear mixed model with applications in sparse data. *Statistics in Medicine*, **29**(29), 3046–3067. <https://doi.org/10.1002/sim.4040>

Turner, R. M., Omar, R. Z., Yang, M., Goldstein, H., & Thompson, S. G. (2000). A multilevel model framework for meta-analysis of clinical trials with binary outcomes. *Statistics in Medicine*, **19**(24), 3417–3432. [https://doi.org/10.1002/1097-0258\(20001230\)19:24<3417::aid-sim614>3.0.co;2-1](https://doi.org/10.1002/1097-0258(20001230)19:24<3417::aid-sim614>3.0.co;2-1)

van Houwelingen, H. C., Zwinderman, K. H., & Stijnen, T. (1993). A bivariate approach to meta-analysis. *Statistics in Medicine*, **12**(24), 2273–2284. <https://doi.org/10.1002/sim.4780122405>

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), and [rma.mv](#) for other model fitting functions.

[dat.nielweise2007](#), [dat.nielweise2008](#), [dat.collins1985a](#), and [dat.pritz1997](#) for further examples of the use of the `rma.glmm` function.

For rare event data, see also the [rema](#) package for a version of the conditional logistic model that uses a permutation approach for making inferences.

Examples

```
#####

### random-effects model using rma.uni() (standard RE model analysis)
rma(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, method="ML")

### random-effects models using rma.glmm() (requires 'lme4' package)

## Not run:
### unconditional model with fixed study effects (the default)
rma.glmm(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, model="UM.FS")

### unconditional model with random study effects
rma.glmm(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, model="UM.RS")

### conditional model with approximate likelihood
rma.glmm(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, model="CM.AL")

### conditional model with exact likelihood
### note: fitting this model may take a bit of time, so be patient
rma.glmm(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, model="CM.EL")

## End(Not run)

#####

### try some alternative measures

## Not run:
rma.glmm(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
rma.glmm(measure="RD", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

## End(Not run)
```

```
#####

### meta-analysis of proportions

## Not run:
dat <- dat.debruin2009

### binomial-normal model (with logit link) = mixed-effects logistic model
res <- rma.glmm(measure="PLO", xi=xi, ni=ni, data=dat)
predict(res, transf=transf.ilogit)

### binomial-normal model with measure="PLN" (uses a log link)
res <- rma.glmm(measure="PLN", xi=xi, ni=ni, data=dat)
predict(res, transf=exp)

### binomial-normal model with measure="PR" (uses an identity link)
res <- rma.glmm(measure="PR", xi=xi, ni=ni, data=dat)
predict(res)

### binomial-normal model (with probit link) = mixed-effects probit model
res <- rma.glmm(measure="PLO", xi=xi, ni=ni, data=dat, link="probit")
predict(res, transf=pnorm)

### further link functions that one could consider here
res <- rma.glmm(measure="PLO", xi=xi, ni=ni, data=dat, link="cauchit")
predict(res, transf=pcauchy)
res <- rma.glmm(measure="PLO", xi=xi, ni=ni, data=dat, link="cloglog")
predict(res, transf=\(x) 1-exp(-exp(x)))

## End(Not run)

#####
```

rma.mh

Meta-Analysis via the Mantel-Haenszel Method

Description

Function to fit equal-effects models to 2×2 table and person-time data via the Mantel-Haenszel method. See below and the introduction to the [metafor-package](#) for more details on these models.

Usage

```
rma.mh(ai, bi, ci, di, n1i, n2i, x1i, x2i, t1i, t2i,
       measure="OR", data, slab, subset,
       add=1/2, to="only0", drop00=TRUE,
       correct=TRUE, level=95, verbose=FALSE, digits, ...)
```

Arguments

These arguments pertain to data input:

ai	vector with the 2×2 table frequencies (upper left cell). See below and the documentation of the escalc function for more details.
bi	vector with the 2×2 table frequencies (upper right cell). See below and the documentation of the escalc function for more details.
ci	vector with the 2×2 table frequencies (lower left cell). See below and the documentation of the escalc function for more details.
di	vector with the 2×2 table frequencies (lower right cell). See below and the documentation of the escalc function for more details.
n1i	vector with the group sizes or row totals (first group). See below and the documentation of the escalc function for more details.
n2i	vector with the group sizes or row totals (second group). See below and the documentation of the escalc function for more details.
x1i	vector with the number of events (first group). See below and the documentation of the escalc function for more details.
x2i	vector with the number of events (second group). See below and the documentation of the escalc function for more details.
t1i	vector with the total person-times (first group). See below and the documentation of the escalc function for more details.
t2i	vector with the total person-times (second group). See below and the documentation of the escalc function for more details.
measure	character string to specify the effect size measure to use for the meta-analysis. Possible options are "RR" for the (log transformed) risk ratio, "OR" for the (log transformed) odds ratio, "RD" for the risk difference, "IRR" for the (log transformed) incidence rate ratio, or "IRD" for the incidence rate difference.
data	optional data frame containing the data supplied to the function.
slab	optional vector with labels for the k studies.
subset	optional (logical or numeric) vector to specify the subset of studies that should be used for the analysis.

These arguments pertain to handling of zero cells/counts/frequencies:

add	non-negative number to specify the amount to add to zero cells or even counts when calculating the effect size estimates of the individual studies. Can also be a vector of two numbers, where the first number is used in the calculation of the estimates and the second number is used when applying the Mantel-Haenszel method. See below and the documentation of the escalc function for more details.
to	character string to specify when the values under add should be added (either "only0", "all", "if0all", or "none"). Can also be a character vector, where the first string again applies when calculating the estimates and the second string when applying the Mantel-Haenszel method. See below and the documentation of the escalc function for more details.

`drop00` logical to specify whether studies with no cases/events (or only cases) in both groups should be dropped when calculating the effect size estimates (the values for such studies are set to NA). Can also be a vector of two logicals, where the first applies to the calculation of the estimates and the second when applying the Mantel-Haenszel method. See below and the documentation of the [escalc](#) function for more details.

These arguments pertain to the model / computations and output:

`correct` logical to specify whether to apply a continuity correction when computing the Cochran-Mantel-Haenszel test statistic.

`level` numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see [here](#) for details).

`verbose` logical to specify whether output should be generated on the progress of the model fitting (the default is FALSE).

`digits` optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is 4. See also [here](#) for further details on how to control the number of digits in the output.

`...` additional arguments.

Details

Specifying the Data:

When the effect size measure is either the risk ratio (`measure="RR"`), odds ratio (`measure="OR"`), or risk difference (`measure="RD"`), the studies are assumed to provide data in terms of 2×2 tables of the form:

	outcome 1	outcome 2	total
group 1	ai	bi	n1i
group 2	ci	di	n2i

where ai, bi, ci, and di denote the cell frequencies and n1i and n2i the row totals. For example, in a set of randomized clinical trials (RCTs) or cohort studies, group 1 and group 2 may refer to the treatment/exposed and placebo/control/non-exposed group, respectively, with outcome 1 denoting some event of interest (e.g., death) and outcome 2 its complement. In a set of case-control studies, group 1 and group 2 may refer to the group of cases and the group of controls, with outcome 1 denoting, for example, exposure to some risk factor and outcome 2 non-exposure. For these effect size measures, one needs to specify the cell frequencies via the ai, bi, ci, and di arguments (or alternatively, one can use the ai, ci, n1i, and n2i arguments).

Alternatively, when the effect size measure is the incidence rate ratio (`measure="IRR"`) or the incidence rate difference (`measure="IRD"`), the studies are assumed to provide data in terms of tables of the form:

	events	person-time
group 1	x1i	t1i
group 2	x2i	t2i

where $x1i$ and $x2i$ denote the number of events in the first and the second group, respectively, and $t1i$ and $t2i$ the corresponding total person-times at risk.

Mantel-Haenszel Method:

An approach for aggregating data of these types was suggested by Mantel and Haenszel (1959) and later extended by various authors (see references). The Mantel-Haenszel method provides a weighted estimate under an equal-effects model. The method is particularly advantageous when aggregating a large number of studies with small sample sizes (the so-called sparse data or increasing strata case).

When analyzing odds ratios, the Cochran-Mantel-Haenszel (CMH) test (Cochran, 1954; Mantel & Haenszel, 1959) and Tarone's test for heterogeneity (Tarone, 1985) are also provided (by default, the CMH test statistic is computed with the continuity correction; this can be switched off with `correct=FALSE`). When analyzing incidence rate ratios, the Mantel-Haenszel (MH) test (Rothman et al., 2008) for person-time data is also provided (again, the `correct` argument controls whether the continuity correction is applied). When analyzing risk ratios, odds ratios, or incidence rate ratios, the printed results are given both in terms of the log and the raw units (for easier interpretation).

Calculating Effect Size Estimates of the Individual Studies:

The Mantel-Haenszel method itself does not require the calculation of the effect size estimates of the individual studies (e.g., the observed log odds ratios of the k studies) and directly makes use of the cell/event counts. Zero cells/events are not a problem (except in extreme cases, such as when one of the two outcomes never occurs in any of the 2×2 tables or when there are no events for one of the two groups in any of the tables). Therefore, it is unnecessary to add some constant to the cell/event counts when there are zero cells/events.

However, for plotting and various other functions, it is necessary to calculate the effect size estimates for the k studies. Here, zero cells/events can be problematic, so adding a constant value to the cell/event counts ensures that all k values can be calculated. The `add` and `to` arguments are used to specify what value should be added to the cell/event counts and under what circumstances when calculating the estimates and when applying the Mantel-Haenszel method. Similarly, the `drop00` argument is used to specify how studies with no cases/events (or only cases) in both groups should be handled. The documentation of the [escalc](#) function explains how the `add`, `to`, and `drop00` arguments work. If only a single value for these arguments is specified (as per default), then these values are used when calculating the estimates and no adjustment to the cell/event counts is made when applying the Mantel-Haenszel method. Alternatively, when specifying two values for these arguments, the first value applies when calculating the estimates and the second value when applying the Mantel-Haenszel method.

Note that `drop00` is set to `TRUE` by default. Therefore, the effect size estimates for studies where `ai=ci=0` or `bi=di=0` or studies where `x1i=x2i=0` are set to `NA`. When applying the Mantel-Haenszel method, such studies are not explicitly dropped (unless the second value of `drop00` argument is also set to `TRUE`), but this is practically not necessary, as they do not actually influence the results (assuming no adjustment to the cell/event counts are made when applying the Mantel-Haenszel method).

Value

An object of class `c("rma.mh", "rma")`. The object is a list containing the following components:

beta	aggregated log risk ratio, log odds ratio, risk difference, log rate ratio, or rate difference.
se	standard error of the aggregated value.
zval	test statistics of the aggregated value.
pval	corresponding p-value.
ci.lb	lower bound of the confidence interval.
ci.ub	upper bound of the confidence interval.
QE	test statistic of the test for heterogeneity.
QEp	corresponding p-value.
MH	Cochran-Mantel-Haenszel test statistic (measure="OR") or Mantel-Haenszel test statistic (measure="IRR").
MHp	corresponding p-value.
TA	test statistic of Tarone's test for heterogeneity (only when measure="OR").
TAp	corresponding p-value (only when measure="OR").
k	number of studies included in the analysis.
yi, vi	the vector of estimates and corresponding sampling variances.
fit.stats	a list with the log-likelihood, deviance, AIC, BIC, and AICc values under the unrestricted and restricted likelihood.
...	some additional elements/values.

Methods

The results of the fitted model are formatted and printed with the `print` function. If fit statistics should also be given, use `summary` (or use the `fitstats` function to extract them).

The `residuals`, `rstandard`, and `rstudent` functions extract raw and standardized residuals. Leave-one-out diagnostics can be obtained with `leave1out`.

Forest, funnel, radial, L'Abbé, and Baujat plots can be obtained with `forest`, `funnel`, `radial`, `labbe`, and `baujat`. The `qqnorm` function provides normal QQ plots of the standardized residuals. One can also call `plot` on the fitted model object to obtain various plots at once.

A cumulative meta-analysis (i.e., adding one observation at a time) can be obtained with `cumul`.

Other extractor functions include `coef`, `vcov`, `se`, `logLik`, `deviance`, `AIC`, and `BIC`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Cochran, W. G. (1954). Some methods for strengthening the common χ^2 tests. *Biometrics*, **10**(4), 417–451. <https://doi.org/10.2307/3001616>
- Greenland, S., & Robins, J. M. (1985). Estimation of a common effect parameter from sparse follow-up data. *Biometrics*, **41**(1), 55–68. <https://doi.org/10.2307/2530643>

- Mantel, N., & Haenszel, W. (1959). Statistical aspects of the analysis of data from retrospective studies of disease. *Journal of the National Cancer Institute*, **22**(4), 719–748. <https://doi.org/10.1093/jnci/22.4.719>
- Nurminen, M. (1981). Asymptotic efficiency of general noniterative estimators of common relative risk. *Biometrika*, **68**(2), 525–530. <https://doi.org/10.1093/biomet/68.2.525>
- Robins, J., Breslow, N., & Greenland, S. (1986). Estimators of the Mantel-Haenszel variance consistent in both sparse data and large-strata limiting models. *Biometrics*, **42**(2), 311–323. <https://doi.org/10.2307/253105>
- Rothman, K. J., Greenland, S., & Lash, T. L. (2008). *Modern epidemiology* (3rd ed.). Philadelphia: Lippincott Williams & Wilkins.
- Sato, T., Greenland, S., & Robins, J. M. (1989). On the variance estimator for the Mantel-Haenszel risk difference. *Biometrics*, **45**(4), 1323–1324. <https://www.jstor.org/stable/2531784>
- Tarone, R. E. (1981). On summary estimators of relative risk. *Journal of Chronic Diseases*, **34**(9-10), 463–468. [https://doi.org/10.1016/0021-9681\(81\)90006-0](https://doi.org/10.1016/0021-9681(81)90006-0)
- Tarone, R. E. (1985). On heterogeneity tests based on efficient scores. *Biometrika*, **72**(1), 91–95. <https://doi.org/10.1093/biomet/72.1.91>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.glmm](#), [rma.peto](#), and [rma.mv](#) for other model fitting functions.

Examples

```
### meta-analysis of the (log) odds ratios using the Mantel-Haenszel method
rma.mh(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### meta-analysis of the (log) risk ratios using the Mantel-Haenszel method
rma.mh(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
```

rma.mv	<i>Meta-Analysis via Multivariate/Multilevel Linear (Mixed-Effects) Models</i>
--------	--

Description

Function to fit meta-analytic multivariate/multilevel fixed- and random/mixed-effects models with or without moderators via linear (mixed-effects) models. See below and the introduction to the [metafor-package](#) for more details on these models.

Usage

```
rma.mv(yi, V, W, mods, data, slab, subset,
       random, struct="CS", intercept=TRUE, method="REML",
       test="z", dfs="residual", level=95, btt,
       R, Rscale="cor", sigma2, tau2, rho, gamma2, phi,
       cvvc=FALSE, sparse=FALSE, verbose=FALSE, digits, control, ...)
```

Arguments

These arguments pertain to data input:

<code>yi</code>	vector of length k with the effect size estimates. See ‘Details’.
<code>V</code>	vector of length k with the corresponding sampling variances or a $k \times k$ variance-covariance matrix of the sampling errors. See ‘Details’.
<code>W</code>	optional argument to specify a vector of length k with user-defined weights or a $k \times k$ user-defined weight matrix. See ‘Details’.
<code>mods</code>	optional argument to include one or more moderators in the model. A single moderator can be given as a vector of length k specifying the values of the moderator. Multiple moderators are specified by giving a matrix with k rows and as many columns as there are moderator variables. Alternatively, a model formula can be used to specify the model. See ‘Details’.
<code>data</code>	optional data frame containing the data supplied to the function.
<code>slab</code>	optional vector with labels for the k estimates.
<code>subset</code>	optional (logical or numeric) vector to specify the subset of studies (or more precisely, rows of the dataset) that should be used for the analysis.

These arguments pertain to the model / computations and output:

<code>random</code>	either a single one-sided formula or list of one-sided formulas to specify the random-effects structure of the model. See ‘Details’.
<code>struct</code>	character string to specify the variance structure of an $\sim$ inner outer formula in the random argument. Either "CS" for compound symmetry, "HCS" for heteroscedastic compound symmetry, "UN" or "GEN" for an unstructured variance-covariance matrix, "ID" for a scaled identity matrix, "DIAG" for a diagonal matrix, "AR" for an AR(1) autoregressive structure, "HAR" for a heteroscedastic AR(1) autoregressive structure, "CAR" for a continuous-time autoregressive structure, or one of "SPEXP", "SPGAU", "SPLIN", "SPRAT", or "SPSPH" for one of the spatial correlation structures. See ‘Details’.
<code>intercept</code>	logical to specify whether an intercept should be added to the model (the default is TRUE). Ignored when <code>mods</code> is a formula.
<code>method</code>	character string to specify whether the model should be fitted via maximum likelihood ("ML") or via restricted maximum likelihood ("REML") estimation (the default is "REML").
<code>test</code>	character string to specify how test statistics and confidence intervals for the fixed effects should be computed. By default (<code>test="z"</code>), Wald-type tests and CIs are obtained, which are based on a standard normal distribution. When <code>test="t"</code> , a t-distribution is used instead. See ‘Details’ and also here for some recommended practices.
<code>dfs</code>	character string to specify how the (denominator) degrees of freedom should be calculated when <code>test="t"</code> . Either <code>dfs="residual"</code> or <code>dfs="contain"</code> . Can also be a numeric vector with the degrees of freedom for each model coefficient. See ‘Details’.

level	numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see here for details).
btt	optional vector of indices to specify which coefficients to include in the omnibus test of moderators. Can also be a string to grep for. See ‘Details’.
R	an optional named list of known correlation matrices corresponding to (some of) the components specified via the random argument. See ‘Details’.
Rscale	character string, integer, or logical to specify how matrices specified via the R argument should be scaled. See ‘Details’.
sigma2	optional numeric vector (of the same length as the number of random intercept components specified via the random argument) to fix the corresponding σ^2 value(s). A specific σ^2 value can be fixed by setting the corresponding element of this argument to the desired value. A specific σ^2 value will be estimated if the corresponding element is set equal to NA. See ‘Details’.
tau2	optional numeric value (for struct="CS", "AR", "CAR", or a spatial correlation structure) or vector (for struct="HCS", "UN", or "HAR") to fix the amount of (residual) heterogeneity for the levels of the inner factor corresponding to an $\sim$ inner outer formula specified in the random argument. A numeric value fixes a particular τ^2 value, while NA means that the value should be estimated. See ‘Details’.
rho	optional numeric value (for struct="CS", "HCS", "AR", "HAR", "CAR", or a spatial correlation structure) or vector (for struct="UN") to fix the correlation between the levels of the inner factor corresponding to an $\sim$ inner outer formula specified in the random argument. A numeric value fixes a particular ρ value, while NA means that the value should be estimated. See ‘Details’.
gamma2	as tau2 argument, but for a second $\sim$ inner outer formula specified in the random argument. See ‘Details’.
phi	as rho argument, but for a second $\sim$ inner outer formula specified in the random argument. See ‘Details’.
cvvc	logical to specify whether to calculate the variance-covariance matrix of the variance/correlation components (can also be set to "varcov" or "varcor"). See ‘Details’.
sparse	logical to specify whether the function should use sparse matrix objects to the extent possible (can speed up model fitting substantially for certain models). See ‘Note’.
verbose	logical to specify whether output should be generated on the progress of the model fitting (the default is FALSE). Can also be an integer. Values > 1 generate more verbose output. See ‘Note’.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is 4. See also here for further details on how to control the number of digits in the output.
control	optional list of control values for the estimation algorithms. If unspecified, default values are defined inside the function. See ‘Note’.
...	additional arguments.

Details

Specifying the Data:

The function can be used in combination with any of the usual effect size measures used in meta-analyses (e.g., log risk ratios, log odds ratios, risk differences, mean differences, standardized mean differences, log transformed ratios of means, raw correlation coefficients, correlation coefficients transformed with Fisher's r -to- z transformation), or, more generally, any set of estimates (with corresponding sampling variances) one would like to meta-analyze. Simply specify the effect size estimates via the `yi` argument and the corresponding sampling variances via the `V` argument. In case the sampling errors are correlated, then one can specify the entire variance-covariance matrix of the sampling errors via the `V` argument.

The `escalc` function can be used to compute a wide variety of effect size measures (and the corresponding sampling variances) based on various summary statistics. Equations for computing the covariance between the sampling errors for a variety of different effect size measures can be found, for example, in Gleser and Olkin (2009), Lajeunesse (2011), and Wei and Higgins (2013). For raw and Fisher r -to- z transformed correlations, one can find suitable equations, for example, in Steiger (1980). The latter are implemented in the `rcalc` function. See also `vcalc` for a function that can be used to construct or approximate the variance-covariance matrix of dependent estimates for a wide variety of circumstances. See also [here](#) for some recommendations on a general workflow for meta-analyses involving complex dependency structures.

Specifying Fixed Effects:

With `rma.mv(yi, V)`, a fixed-effects model is fitted to the data (note: arguments `struct`, `sigma2`, `tau2`, `rho`, `gamma2`, `phi`, `R`, and `Rscale` are not relevant then and are ignored). The model is then simply given by $y \sim N(\theta, V)$, where y is a (column) vector with the effect size estimates, θ is the (average) true effect, and V is the variance-covariance matrix of the sampling errors (if a vector of sampling variances is provided via the `V` argument, then V is assumed to be diagonal). Note that the argument is `V`, not `v` (`R` is case sensitive!).

One or more moderators can be included in the model via the `mods` argument. A single moderator can be given as a (row or column) vector of length k specifying the values of the moderator. Multiple moderators are specified by giving an appropriate model matrix (i.e., X) with k rows and as many columns as there are moderator variables (e.g., `mods = cbind(mod1, mod2, mod3)`, where `mod1`, `mod2`, and `mod3` correspond to the names of the variables for the three moderator variables). The intercept is added to the model matrix by default unless `intercept=FALSE`.

Alternatively, one can use standard `formula` syntax to specify the model. In this case, the `mods` argument should be set equal to a one-sided formula of the form `mods = ~ model` (e.g., `mods = ~ mod1 + mod2 + mod3`). Interactions, polynomial/spline terms, and factors can be easily added to the model in this manner. When specifying a model formula via the `mods` argument, the `intercept` argument is ignored. Instead, the inclusion/exclusion of the intercept is controlled by the specified formula (e.g., `mods = ~ 0 + mod1 + mod2 + mod3` or `mods = ~ mod1 + mod2 + mod3 - 1` would lead to the removal of the intercept). One can also directly specify moderators via the `yi` argument (e.g., `rma.mv(yi ~ mod1 + mod2 + mod3, V)`). In that case, the `mods` argument is ignored and the inclusion/exclusion of the intercept is again controlled by the specified formula.

With moderators included, the model is then given by $y \sim N(X\beta, V)$, where X denotes the model matrix containing the moderator values (and with the first column equal to 1s for the intercept term if it is included) and β is a column vector containing the corresponding model coefficients. The model coefficients (i.e., β) are then estimated with $b = (X'WX')^{-1}X'Wy$, where $W = V^{-1}$ is the weight matrix (without moderators, X is just a column vector of 1's). With the `W` argument, one can also specify user-defined weights (or a weight matrix).

Specifying Random Effects:

One can fit random/mixed-effects models to the data by specifying the desired random effects structure via the `random` argument. The `random` argument is either a single one-sided formula or a list of one-sided formulas. One formula type that can be specified via this argument is of the form `random = ~ 1 | id`. Such a formula adds a random effect corresponding to the grouping variable `id` to the model. Estimates with the same value of the `id` variable receive the same value of the random effect, while estimates with different values of the `id` variable are assumed to be independent. The variance component corresponding to such a formula is denoted by σ^2 . An arbitrary number of such formulas can be specified as a list of formulas (e.g., `random = list(~ 1 | id1, ~ 1 | id2)`), with variance components σ_1^2, σ_2^2 , and so on. Nested random effects of this form can also be added using `random = ~ 1 | id1/id2`, which adds a random effect corresponding to the grouping variable `id1` and a random effect corresponding to `id2` within `id1` to the model. This can be extended to models with even more levels of nesting (e.g., `random = ~ 1 | id1/id2/id3`). Random effects of this form are useful to model clustering (and hence non-independence) induced by a multilevel structure in the data (e.g., estimates derived from the same paper, lab, research group, or species may be more similar to each other than estimates derived from different papers, labs, research groups, or species). See, for example, Konstantopoulos (2011) and Nakagawa and Santos (2012) for more details.

See [dat.konstantopoulos2011](#), [dat.bornmann2007](#), [dat.obrien2003](#), and [dat.crede2010](#) for examples of multilevel meta-analyses.

In addition or alternatively to specifying one or multiple `~ 1 | id` terms, the `random` argument can also contain a formula of the form `~ inner | outer`. Estimates with the same value of the outer grouping variable share correlated random effects corresponding to the levels of the inner grouping variable, while estimates with different values of the outer grouping variable are assumed to be independent (note that the inner variable is automatically treated as a factor). The `struct` argument is used to specify the variance structure corresponding to the inner variable. With `struct="CS"`, a compound symmetric structure is assumed (i.e., a single variance component τ^2 corresponding to the $j = 1, \dots, J$ levels of the inner variable and a single correlation coefficient ρ for the correlation between the different levels). With `struct="HCS"`, a heteroscedastic compound symmetric structure is assumed (with τ_j^2 denoting the variance component corresponding to the j th level of the inner variable and a single correlation coefficient ρ for the correlation between the different levels). With `struct="UN"`, an unstructured (but positive definite) variance-covariance matrix is assumed (with τ_j^2 as described above and correlation coefficient $\rho_{jj'}$ for the combination of the j th and j' th level of the inner variable). For example, for an inner variable with four levels, these structures correspond to variance-covariance matrices of the form:

$$\begin{array}{ccc} \text{struct}=\text{"CS"} & \text{struct}=\text{"HCS"} & \text{struct}=\text{"UN"} \\ \begin{bmatrix} \tau^2 & & & \\ \rho\tau^2 & \tau^2 & & \\ \rho\tau^2 & \rho\tau^2 & \tau^2 & \\ \rho\tau^2 & \rho\tau^2 & \rho\tau^2 & \tau^2 \end{bmatrix} & \begin{bmatrix} \tau_1^2 & & & \\ \rho\tau_2\tau_1 & \tau_2^2 & & \\ \rho\tau_3\tau_1 & \rho\tau_3\tau_2 & \tau_3^2 & \\ \rho\tau_4\tau_1 & \rho\tau_4\tau_2 & \rho\tau_4\tau_3 & \tau_4^2 \end{bmatrix} & \begin{bmatrix} \tau_1^2 & & & \\ \rho_{21}\tau_2\tau_1 & \tau_2^2 & & \\ \rho_{31}\tau_3\tau_1 & \rho_{32}\tau_3\tau_2 & \tau_3^2 & \\ \rho_{41}\tau_4\tau_1 & \rho_{42}\tau_4\tau_2 & \rho_{43}\tau_4\tau_3 & \tau_4^2 \end{bmatrix} \end{array}$$

Structures `struct="ID"` and `struct="DIAG"` are just like `struct="CS"` and `struct="HCS"`, respectively, except that ρ is set to 0, so that we either get a scaled identity matrix or a diagonal matrix.

With the outer term corresponding to a study identification variable and the inner term to a variable indicating the treatment type or study arm, such a random effect could be used to estimate how strongly different treatment effects within the same study are correlated and/or whether the

amount of heterogeneity differs across different treatment types or study arms. Network meta-analyses (also known as mixed treatment comparisons) will also typically require such a random effect (e.g., Salanti et al., 2008). The meta-analytic bivariate model (e.g., van Houwelingen, Arends, & Stijnen, 2002) can also be fitted in this manner (see the examples below). The inner term could also correspond to a variable indicating different types of outcomes measured within the same study, which allows for fitting multivariate models with multiple correlated effects per study (e.g., Berkey et al., 1998; Kalaian & Raudenbush, 1996).

See [dat.berkey1998](#), [dat.assink2016](#), [dat.kalaian1996](#), [dat.dagostino1998](#), and [dat.craft2003](#) for examples of multivariate meta-analyses with multiple outcomes. See [dat.knapp2017](#), [dat.mccurdy2020](#), and [dat.tannersmith2016](#) for further examples of multilevel/multivariate models with complex data structures (see also [here](#) for a related discussion of a recommended workflow for such cases). See [dat.kearon1998](#) for an example using a bivariate model to analyze sensitivity and specificity. See [dat.hasselblad1998](#), [dat.pagliaro1992](#), [dat.lopez2019](#), and [dat.senn2013](#) for examples of network meta-analyses.

For meta-analyses of studies reporting estimates at multiple time points, it may also be reasonable to assume that the true effects are correlated over time according to an autoregressive structure (Ishak et al., 2007; Trikalinos & Olkin, 2012). For this purpose, one can choose `struct="AR"`, corresponding to a structure with a single variance component τ^2 and AR(1) autocorrelation among the values of the random effect. The values of the inner variable should then reflect the various time points, with increasing values reflecting later time points. This structure assumes equally spaced time points, so the actual values of the inner variable are not relevant, only their ordering. One can also use `struct="HAR"`, which allows for fitting a heteroscedastic AR(1) structure (with τ_j^2 denoting the variance component of the j th measurement occasion). Finally, when time points are not evenly spaced, one might consider using `struct="CAR"` for a continuous-time autoregressive structure, in which case the values of the inner variable should reflect the exact time points of the measurement occasions. For example, for an inner variable with four time points, these structures correspond to variance-covariance matrices of the form:

$$\begin{array}{ccc} \text{struct}="AR" & \text{struct}="HAR" & \text{struct}="CAR" \\ \begin{bmatrix} \tau^2 & & & \\ \rho\tau^2 & \tau^2 & & \\ \rho^2\tau^2 & \rho\tau^2 & \tau^2 & \\ \rho^3\tau^2 & \rho^2\tau^2 & \rho\tau^2 & \tau^2 \end{bmatrix} & \begin{bmatrix} \tau_1^2 & & & \\ \rho\tau_2\tau_1 & \tau_2^2 & & \\ \rho^2\tau_3\tau_1 & \rho\tau_3\tau_2 & \tau_3^2 & \\ \rho^3\tau_4\tau_1 & \rho^2\tau_4\tau_2 & \rho\tau_4\tau_3 & \tau_4^2 \end{bmatrix} & \begin{bmatrix} \tau^2 & & & \\ \rho^{|t_2-t_1|}\tau^2 & \tau^2 & & \\ \rho^{|t_3-t_1|}\tau^2 & \rho^{|t_3-t_2|}\tau^2 & \tau^2 & \\ \rho^{|t_4-t_1|}\tau^2 & \rho^{|t_4-t_2|}\tau^2 & \rho^{|t_4-t_3|}\tau^2 & \tau^2 \end{bmatrix} \end{array}$$

See [dat.fine1993](#) and [dat.ishak2007](#) for examples involving such structures.

For estimates that have a known spatial configuration, various spatial correlation structures are also available. For these structures, the formula is of the form `random = ~ var1 + var2 + ... | outer`, where `var1`, `var2`, and so on are variables to specify the spatial coordinates (e.g., longitude and latitude) based on which distances (by default Euclidean) will be computed. Let d denote the distance between two points that share the same value of the outer variable (if all true effects are allowed to be spatially correlated, simply set `outer` to a variable that is a constant). Then the correlation between the true effects corresponding to these two points is a function of d and the parameter ρ . The following table shows the types of spatial correlation structures that can be specified and the equations for the correlation. The covariance between the true effects is then the correlation times τ^2 .

structure	struct	correlation
exponential	"SPEXP"	$\exp(-d/\rho)$

Gaussian	"SPGAU"	$\exp(-d^2/\rho^2)$
linear	"SPLIN"	$(1 - d/\rho)I(d < \rho)$
rational quadratic	"SPRAT"	$1 - (d/\rho)^2/(1 + (d/\rho)^2)$
spherical	"SPSPH"	$(1 - 1.5(d/\rho) + 0.5(d/\rho)^3)I(d < \rho)$

Note that $I(d < \rho)$ is equal to 1 if $d < \rho$ and 0 otherwise. The parameterization of the various structures is based on Pinheiro and Bates (2000). Instead of Euclidean distances, one can also use other distance measures by setting (the undocumented) argument `dist` to either "maximum" for the maximum distance between two points (supremum norm), to "manhattan" for the absolute distance between the coordinate vectors (L1 norm), or to "gcd" for the great-circle distance (WGS84 ellipsoid method). In the latter case, only two variables, namely the longitude and latitude (in decimal degrees, with minus signs for West and South), must be specified.

If a distance matrix has already been computed, one can also pass this matrix as a list element to the `dist` argument. In this case, one should use a formula of the form `random = ~ id | outer`, where `id` are location identifiers, with corresponding row/column names in the distance matrix specified via the `dist` argument.

See [dat.maire2019](#) for an example of a meta-analysis with a spatial correlation structure.

An `~ inner | outer` formula can also be used to add random effects to the model corresponding to a set of predictor variables when `struct="GEN"`. Here, the inner term is used to specify one or multiple variables (e.g., `random = ~ var1 + var2 | outer`) and corresponding 'random slopes' are added to the model (and a 'random intercept' unless the intercept is removed from the inner term). The variance-covariance matrix of the random effects added in this manner is assumed to be a general unstructured (but positive definite) matrix. Such a random effects structure may be useful in a meta-analysis examining the dose-response relationship between a moderator variable and the size of the true effect (sometimes called a 'dose-response meta-analysis').

See [dat.obrien2003](#) for an example of a meta-analysis examining a dose-response relationship.

The random argument can also contain a second formula of the form `~ inner | outer` (but no more!). A second formula of this form works exactly described as above, but its variance components are denoted by γ^2 and its correlation components by ϕ . The `struct` argument should then be of length 2 to specify the variance-covariance structure for the first and second component, respectively.

When the random argument contains a formula of the form `~ 1 | id`, one can use the (optional) argument `R` to specify a corresponding known correlation matrix for the random effect (i.e., `R = list(id = Cor)`, where `Cor` is the correlation matrix). In that case, estimates with the same value of the `id` variable receive the same value for the random effect, while estimates with different values of the `id` variable receive values that are correlated as specified in the corresponding correlation matrix given via the `R` argument. The column/row names of the correlation matrix given via the `R` argument must therefore correspond to the unique values of the `id` variable. When the random argument contains multiple formulas of the form `~ 1 | id`, one can specify known correlation matrices for none, some, or all of those terms (e.g., with `random = list(~ 1 | id1, ~ 1 | id2)`, one could specify `R = list(id1 = Cor1)` or `R = list(id1 = Cor1, id2 = Cor2)`, where `Cor1` and `Cor2` are the correlation matrices corresponding to the grouping variables `id1` and `id2`, respectively).

Such a random effect with a known (or at least approximately known) correlation structure is useful in a variety of contexts. For example, such a component can be used to account for the correlations induced by the shared phylogenetic history among organisms (e.g., plants, fungi, animals). In that case, `~ 1 | species` is used to specify the species and argument `R` is used to specify

the phylogenetic correlation matrix of the species studied in the meta-analysis. The corresponding variance component then indicates how much variance/heterogeneity is attributable to the specified phylogeny. See Nakagawa and Santos (2012) for more details. As another example, in a genetic meta-analysis studying disease association for several single nucleotide polymorphisms (SNPs), linkage disequilibrium (LD) among the SNPs can induce an approximately known degree of correlation among the effects. In that case, `~ 1 | snp` could be used to specify the SNPs and `R` the corresponding LD correlation matrix for the SNPs included in the meta-analysis.

The `Rscale` argument controls how matrices specified via the `R` argument are scaled. With `Rscale="none"` (or `Rscale=0` or `Rscale=FALSE`), no scaling is used. With `Rscale="cor"` (or `Rscale=1` or `Rscale=TRUE`), the `cov2cor` function is used to ensure that the matrices are correlation matrices (assuming they were covariance matrices to begin with). With `Rscale="cor0"` (or `Rscale=2`), first `cov2cor` is used and then the elements of each correlation matrix are scaled with $(R - \min(R)) / (1 - \min(R))$ (this ensures that a correlation of zero in a phylogenetic correlation matrix corresponds to the split at the root node of the tree comprising the species that are actually analyzed). Finally, `Rscale="cov0"` (or `Rscale=3`) only rescales with $R - \min(R)$ (which ensures that a phylogenetic covariance matrix is rooted at the lowest split).

See [dat.moura2021](#) and [dat.lim2014](#) for examples of meta-analyses with phylogenetic correlation structures.

Together with the variance-covariance matrix of the sampling errors (i.e., V), the specified random effects structure of the model implies a particular ‘marginal’ variance-covariance matrix of the effect size estimates. Once estimates of the variance components (i.e., of the σ^2 , τ^2 , ρ , γ^2 , and/or ϕ values) have been obtained (either using maximum likelihood or restricted maximum likelihood estimation), the estimated marginal variance-covariance matrix can be constructed (denoted by M). The model coefficients (i.e., β) are then estimated with $b = (X'WX')^{-1}X'Wy$, where $W = M^{-1}$ is the weight matrix. With the `W` argument, one can again specify user-defined weights (or a weight matrix).

Fixing Variance/Correlation Components:

Arguments `sigma2`, `tau2`, `rho`, `gamma2`, and `phi` can be used to fix particular variance/correlation components at a given value. This is useful for sensitivity analyses (e.g., for plotting the regular/restricted log-likelihood as a function of a particular variance/correlation component), likelihood ratio tests, or for imposing a desired variance-covariance structure on the data.

For example, if `random = list(~ 1 | id1, ~ 1 | id2)` or `random = ~ 1 | id1/id2`, then `sigma2` must be of length 2 (corresponding to σ_1^2 and σ_2^2) and a fixed value can be assigned to either or both variance components. Setting a particular component to NA means that the component will be estimated by the function (e.g., `sigma2=c(0,NA)` would fix σ_1^2 to 0 and estimate σ_2^2).

Argument `tau2` is only relevant when the `random` argument contains an `~ inner | outer` formula. In that case, if the `tau2` argument is used, it must be either of length 1 (for "CS", "ID", "AR", "CAR", or one of the spatial correlation structures) or of the same length as the number of unique values of the inner variable (for "HCS", "DIAG", "UN", or "HAR"). A numeric value in the `tau2` argument then fixes the corresponding variance component to that value, while NA means that the component will be estimated. Similarly, if argument `rho` is used, it must be either of length 1 (for "CS", "HCS", "AR", "HAR", or one of the spatial correlation structures) or of length $J(J-1)/2$ (for "UN"), where J denotes the number of unique values of the inner variable. Again, a numeric value fixes the corresponding correlation, while NA means that the correlation will be estimated. For example, with `struct="CS"` and `rho=0`, the variance-covariance matrix of the inner variable will be diagonal with τ^2 along the diagonal. For `struct="UN"`, the values specified under `rho`

should be given in column-wise order (e.g., for an inner variable with four levels, the order would be $\rho_{21}, \rho_{31}, \rho_{41}, \rho_{32}, \rho_{42}, \rho_{43}$).

Similarly, arguments `gamma2` and `phi` are only relevant when the random argument contains a second `~ inner | outer` formula. The arguments then work exactly as described above.

Omnibus Test of Moderators:

For models including moderators, an omnibus test of all model coefficients is conducted that excludes the intercept (the first coefficient) if it is included in the model. If no intercept is included in the model, then the omnibus test includes all coefficients in the model including the first. Alternatively, one can manually specify the indices of the coefficients to test via the `btt` ('betas to test') argument (i.e., to test $H_0: \beta_{j \in \text{btt}} = 0$, where $\beta_{j \in \text{btt}}$ is the set of coefficients to be tested). For example, with `btt=c(3, 4)`, only the third and fourth coefficients from the model are included in the test (if an intercept is included in the model, then it corresponds to the first coefficient in the model). Instead of specifying the coefficient numbers, one can specify a string for `btt`. In that case, `grep` will be used to search for all coefficient names that match the string. The omnibus test is called the Q_M -test and follows asymptotically a chi-square distribution with m degrees of freedom (with m denoting the number of coefficients tested) under the null hypothesis (that the true value of all coefficients tested is equal to 0).

Categorical Moderators:

Categorical moderator variables can be included in the model via the `mods` argument in the same way that appropriately (dummy) coded categorical variables can be included in linear models. One can either do the dummy coding manually or use a model formula together with the `factor` function to automate the coding (note that string/character variables in a model formula are automatically converted to factors).

Tests and Confidence Intervals:

By default, tests of individual coefficients in the model (and the corresponding confidence intervals) are based on a standard normal distribution, while the omnibus test is based on a chi-square distribution (see above). As an alternative, one can set `test="t"`, in which case tests of individual coefficients and confidence intervals are based on a t-distribution with $k - p$ degrees of freedom, while the omnibus test then uses an F-distribution with m and $k - p$ degrees of freedom (with k denoting the total number of estimates included in the analysis and p the total number of model coefficients including the intercept if it is present). Note that `test="t"` is not the same as `test="knha"` in `rma.uni`, as no adjustment to the standard errors of the estimated coefficients is made.

The method for calculating the (denominator) degrees of freedom described above (which corresponds to `dfs="residual"`) is quite simplistic and may lead to tests with inflated Type I error rates and confidence intervals that are too narrow on average. As an alternative, one can set `dfs="contain"` (which automatically also sets `test="t"`), in which case the degrees of freedom for the test of a particular model coefficient, b_j , are determined by checking whether x_j , the corresponding column of the model matrix X , varies at the level corresponding to a particular random effect in the model. If such a random effect can be found, then the degrees of freedom are set to $l - p$, where l denotes the number of unique values of this random effect (i.e., for an `~ 1 | id` term, the number of unique values of the `id` variable and for an `~ inner | outer` term, the number of unique values of the outer variable). If no such random effect can be found, then $k - p$ is used as the degrees of freedom. For the omnibus F-test, the minimum of the degrees of freedom of all coefficients involved in the test is used as the denominator degrees of freedom. This approach for

calculating the degrees of freedom should often lead to tests with better control of the Type I error rate and confidence intervals with closer to nominal coverage rates (see also [here](#)).

One can also set `dfs` to a numeric vector with the desired values for the degrees of freedom for testing the model coefficients (e.g., if some other method for determining the degrees of freedom was used).

Tests and Confidence Intervals for Variance/Correlation Components:

Depending on the random effects structure specified, the model may include one or multiple variance/correlation components. Profile likelihood confidence intervals for such components can be obtained using the `confint` function. Corresponding likelihood ratio tests can be obtained using the `anova` function (by comparing two models where the size of the component to be tested is constrained to some null value in the reduced model). It is also always a good idea to examine plots of the (restricted) log-likelihood as a function of the variance/correlation components in the model using the `profile` function to check for parameter identifiability (see ‘Note’).

Test for (Residual) Heterogeneity:

A test for (residual) heterogeneity is automatically carried out by the function. Without moderators in the model, this test is the generalized/weighted least squares extension of Cochran’s Q -test, which tests whether the variability in the estimates is larger than one would expect based on sampling variability (and the given covariances among the sampling errors) alone. A significant test suggests that the true effects are heterogeneous. When moderators are included in the model, this is the Q_E -test for residual heterogeneity, which tests whether the variability in the estimates that is not accounted for by the moderators included in the model is larger than one would expect based on sampling variability (and the given covariances among the sampling errors) alone.

Variance-Covariance Matrix of the Variance/Correlation Components:

In some cases, one might want to obtain the variance-covariance matrix of the variance/correlation components of the model (i.e., of the estimated σ^2 , τ^2 , ρ , γ^2 , and ϕ values). The function will try to calculate this matrix when `cvvc=TRUE` (or equivalently, when `cvvc="varcor"`). When `struct="UN"`, one can also set `cvvc="varcov"` in which case the variance-covariance matrix is given for the variance and covariance components (instead of the correlation components). The element of the model object that contains the resulting variance-covariance matrix is called ‘vvc’. See `matreg` for an example making use of such a matrix.

Value

An object of class `c("rma.mv", "rma")`. The object is a list containing the following components:

<code>beta</code>	estimated coefficients of the model.
<code>se</code>	standard errors of the coefficients.
<code>zval</code>	test statistics of the coefficients.
<code>pval</code>	corresponding p-values.
<code>ci.lb</code>	lower bound of the confidence intervals for the coefficients.
<code>ci.ub</code>	upper bound of the confidence intervals for the coefficients.
<code>vb</code>	variance-covariance matrix of the estimated coefficients.
<code>sigma2</code>	estimated σ^2 value(s).
<code>tau2</code>	estimated τ^2 value(s).

rho	estimated ρ value(s).
gamma2	estimated γ^2 value(s).
phi	estimated ϕ value(s).
k	number of estimates included in the analysis.
p	number of coefficients in the model (including the intercept).
m	number of coefficients included in the omnibus test of moderators.
QE	test statistic of the test for (residual) heterogeneity.
QEp	corresponding p-value.
QM	test statistic of the omnibus test of moderators.
QMp	corresponding p-value.
int.only	logical that indicates whether the model is an intercept-only model.
yi, V, X	the estimates, the corresponding variance-covariance matrix of the sampling errors, and the model matrix.
M	the estimated marginal variance-covariance matrix of the estimates.
fit.stats	a list with the log-likelihood, deviance, AIC, BIC, and AICc values.
vvc	variance-covariance matrix of the variance/correlation components (NA when cvvc=FALSE).
...	some additional elements/values.

Methods

The results of the fitted model are formatted and printed with the `print` function. If fit statistics should also be given, use `summary` (or use the `fitstats` function to extract them). Full versus reduced model comparisons in terms of fit statistics and likelihood ratio tests can be obtained with `anova`. Wald-type tests for sets of model coefficients or linear combinations thereof can be obtained with the same function. Tests and confidence intervals based on (cluster) robust methods can be obtained with `robust`.

Predicted/fitted values can be obtained with `predict` and `fitted`. For best linear unbiased predictions, see `ranef`.

The `residuals`, `rstandard`, and `rstudent` functions extract raw and standardized residuals. See `influence` for additional model diagnostics (e.g., to determine influential studies). For models with moderators, variance inflation factors can be obtained with `vif`.

Confidence intervals for any variance/correlation components in the model can be obtained with `confint`.

For random/mixed-effects models, the `profile` function can be used to obtain a plot of the (restricted) log-likelihood as a function of a specific variance/correlation component of the model. For models with moderators, `regplot` draws scatter plots / bubble plots, showing the (marginal) relationship between the estimates and a selected moderator from the model.

Other extractor functions include `coef`, `vcov`, `se`, `logLik`, `deviance`, `AIC`, `BIC`, `hatvalues`, and `weights`.

Note

Argument `V` also accepts a list of variance-covariance matrices for the estimates. From the list elements, the full (block diagonal) variance-covariance matrix is then automatically constructed. For this to work correctly, the list elements (and the values in the matrices therein) must be in the same order as the effect size estimates passed via the `yi` argument.

Model fitting is done via numerical optimization over the model parameters. By default, `nlminb` is used for the optimization. One can also chose a different optimizer from `optim` via the control argument (e.g., `control=list(optimizer="BFGS")` or `control=list(optimizer="Nelder-Mead")`). Besides `nlminb` and one of the methods from `optim`, one can also choose one of the optimizers from the `minqa` package (i.e., `uobyqa`, `newuoa`, or `bobyqa`), one of the (derivative-free) algorithms from the `nloptr` package, the Newton-type algorithm implemented in `nlm`, the various algorithms implemented in the `dfoptim` package (`hjk` for the Hooke-Jeeves, `nmk` for the Nelder-Mead, and `mads` for the Mesh Adaptive Direct Searches algorithm), the quasi-Newton type optimizers `ucminf` and `lbfgsb3c` and the subspace-searching simplex algorithm `subplex` from the packages of the same name, the Barzilai-Borwein gradient decent method implemented in `BBoptim`, the `Rcgmin` and `Rvmin` optimizers, or the parallelized version of the L-BFGS-B algorithm implemented in `optimParallel` from the package of the same name.

The optimizer name must be given as a character string (i.e., in quotes). Additional control parameters can be specified via the control argument (e.g., `control=list(iter.max=1000, rel.tol=1e-8)`). For `nloptr`, the default is to use the BOBYQA implementation from that package with a relative convergence criterion of $1e-8$ on the function value (i.e., log-likelihood), but this can be changed via the `algorithm` and `fstop_rel` arguments (e.g., `control=list(optimizer="nloptr", algorithm="NLOPT_LN_SBPLX", fstop_rel=1e-6)`). For `optimParallel`, the control argument `ncpus` can be used to specify the number of cores to use for the parallelization (e.g., `control=list(optimizer="optimParallel", ncpus=2)`). Control argument `mfmexit` (which is `Inf` by default and is independent of the control arguments of the various optimizers) hard exits when the specified number of iterations has been exceeded. Control argument `nearpd` can be set to `TRUE` to force the marginal variance-covariance matrix of the estimates to be positive definite (using the `nearPD` function from the `Matrix` package).

When using the `cvvc` argument, the variance-covariance matrix of the variance/correlation components are obtained by inverting the Hessian, which is numerically approximated using the `hessian` function from the `numDeriv` package. Note that these computations may not be numerically stable, especially when the estimates are close to their parameter bounds. One can set control argument `hessianCtrl` to a list of named arguments to be passed on to the `method.args` argument of the `hessian` function (the default is `control=list(hessianCtrl=list(r=8))`). One can also set `control=list(hesspack="pracma")` or `control=list(hesspack="calculus")` in which case the `pracma::hessian` or `calculus::hessian` functions from the respective packages are used instead for approximating the Hessian.

At the moment, the starting values are not chosen in a terribly clever way and could be far off. As a result, the optimizer may be slow to converge or may even get stuck at a local maximum. One can set the starting values manually for the various variance/correlation components in the model via the control argument by specifying the vectors `sigma2.init`, `tau2.init`, `rho.init`, `gamma2.init`, and/or `phi.init` as needed. Especially for complex models, it is a good idea to try out different starting values to make sure that the same estimates are obtained.

Information on the progress of the optimization algorithm can be obtained by setting `verbose=TRUE` (this won't work when using parallelization). Since fitting complex models with many random effects can be computationally expensive, this option is useful to determine how the model fitting is

progressing. One can also set `verbose` to an integer (`verbose=2` yields even more information and `verbose=3` also sets `option(warn=1)` temporarily).

Whether a particular variance/correlation component is actually identifiable needs to be carefully examined when fitting complex models. The function does some limited checking internally to fix variances and/or correlations to zero when it appears that insufficient information is available to estimate a particular parameter. For example, if a particular factor only has a single level, the corresponding variance component is set to 0 (this check can be switched off with `control=list(check.k.gtr.1=FALSE)`). However, it is strongly advised in general to do post model fitting checks to make sure that the likelihood surface around the ML/REML estimates is not flat for some of the parameter estimates (which would imply that the estimates are essentially arbitrary). For example, one can plot the (restricted) log-likelihood as a function of each variance/correlation component in the model to make sure that each profile plot shows a clear peak at the corresponding ML/REML estimate. The `profile` function can be used for this purpose.

Finally, note that the model fitting is not done in a very efficient manner at the moment, which is partly a result of allowing for crossed random effects and correlations across the entire dataset (e.g., when using the `R` argument). As a result, the function works directly with the entire $k \times k$ (marginal) variance-covariance matrix of the estimates (instead of working with smaller blocks in a block diagonal structure). As a result, model fitting can be slow for large k . However, when the variance-covariance structure is actually sparse, a lot of speed can be gained by setting `sparse=TRUE`, in which case sparse matrix objects are used (via the `Matrix` package). Also, when model fitting appears to be slow, setting `verbose=TRUE` is useful to obtain information on how the model fitting is progressing.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Berkey, C. S., Hoaglin, D. C., Antczak-Bouckoms, A., Mosteller, F., & Colditz, G. A. (1998). Meta-analysis of multiple outcomes by regression with random effects. *Statistics in Medicine*, **17**(22), 2537–2550. [https://doi.org/10.1002/\(sici\)1097-0258\(19981130\)17:22<2537::aid-sim953>3.0.co;2-c](https://doi.org/10.1002/(sici)1097-0258(19981130)17:22<2537::aid-sim953>3.0.co;2-c)
- Gleser, L. J., & Olkin, I. (2009). Stochastically dependent effect sizes. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 357–376). New York: Russell Sage Foundation.
- Ishak, K. J., Platt, R. W., Joseph, L., Hanley, J. A., & Caro, J. J. (2007). Meta-analysis of longitudinal studies. *Clinical Trials*, **4**(5), 525–539. <https://doi.org/10.1177/1740774507083567>
- Kalaian, H. A., & Raudenbush, S. W. (1996). A multivariate mixed linear model for meta-analysis. *Psychological Methods*, **1**(3), 227–235. <https://doi.org/10.1037/1082-989X.1.3.227>
- Konstantopoulos, S. (2011). Fixed effects and variance components estimation in three-level meta-analysis. *Research Synthesis Methods*, **2**(1), 61–76. <https://doi.org/10.1002/jrsm.35>
- Lajeunesse, M. J. (2011). On the meta-analysis of response ratios for studies with correlated and multi-group designs. *Ecology*, **92**(11), 2049–2055. <https://doi.org/10.1890/11-0423.1>
- Nakagawa, S., & Santos, E. S. A. (2012). Methodological issues and advances in biological meta-analysis. *Evolutionary Ecology*, **26**(5), 1253–1274. <https://doi.org/10.1007/s10682-012-9555-5>
- Pinheiro, J. C., & Bates, D. (2000). *Mixed-effects models in S and S-PLUS*. New York: Springer.

- Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, **87**(2), 245–251. <https://doi.org/10.1037/0033-2909.87.2.245>
- Salanti, G., Higgins, J. P. T., Ades, A. E., & Ioannidis, J. P. A. (2008). Evaluation of networks of randomized trials. *Statistical Methods in Medical Research*, **17**(3), 279–301. <https://doi.org/10.1177/0962280207080643>
- Trikalinos, T. A., & Olkin, I. (2012). Meta-analysis of effect sizes reported at multiple time points: A multivariate approach. *Clinical Trials*, **9**(5), 610–620. <https://doi.org/10.1177/1740774512453218>
- van Houwelingen, H. C., Arends, L. R., & Stijnen, T. (2002). Advanced methods in meta-analysis: Multivariate approach and meta-regression. *Statistics in Medicine*, **21**(4), 589–624. <https://doi.org/10.1002/sim.1040>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Wei, Y., & Higgins, J. P. (2013). Estimating within-study covariances in multivariate meta-analysis with multiple outcomes. *Statistics in Medicine*, **32**(7), 1191–1205. <https://doi.org/10.1002/sim.5679>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), and [rma.glmm](#) for other model fitting functions.

Examples

```
### calculate log odds ratios and corresponding sampling variances
dat <- escalc(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
dat

### fit random-effects model using rma.uni()
rma(yi, vi, data=dat)

### fit random-effects model using rma.mv()
### note: sigma^2 in this model is the same as tau^2 from the previous model
rma.mv(yi, vi, random = ~ 1 | trial, data=dat)

### change data into long format
dat.long <- to.long(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, append=FALSE)
dat.long

### set levels/labels for group ("con" = control/non-vaccinated, "exp" = experimental/vaccinated)
dat.long$group <- factor(dat.long$group, levels=c(2,1), labels=c("con","exp"))
dat.long

### calculate log odds and corresponding sampling variances
dat.long <- escalc(measure="PLO", xi=out1, mi=out2, data=dat.long)
dat.long

### fit bivariate random-effects model using rma.mv()
res <- rma.mv(yi, vi, mods = ~ group, random = ~ group | study, struct="UN", data=dat.long)
res
```

rma.peto

*Meta-Analysis via Peto's Method***Description**

Function to fit equal-effects models to 2×2 table data via Peto's method. See below and the introduction to the [metafor-package](#) for more details on these models.

Usage

```
rma.peto(ai, bi, ci, di, n1i, n2i,
         data, slab, subset,
         add=1/2, to="only0", drop00=TRUE,
         level=95, verbose=FALSE, digits, ...)
```

Arguments

These arguments pertain to data input:

ai	vector with the 2×2 table frequencies (upper left cell). See below and the documentation of the escalc function for more details.
bi	vector with the 2×2 table frequencies (upper right cell). See below and the documentation of the escalc function for more details.
ci	vector with the 2×2 table frequencies (lower left cell). See below and the documentation of the escalc function for more details.
di	vector with the 2×2 table frequencies (lower right cell). See below and the documentation of the escalc function for more details.
n1i	vector with the group sizes or row totals (first group). See below and the documentation of the escalc function for more details.
n2i	vector with the group sizes or row totals (second group). See below and the documentation of the escalc function for more details.
data	optional data frame containing the data supplied to the function.
slab	optional vector with labels for the k studies.
subset	optional (logical or numeric) vector to specify the subset of studies that should be used for the analysis.

These arguments pertain to handling of zero cells/counts/frequencies:

add	non-negative number to specify the amount to add to zero cells when calculating the effect size estimates of the individual studies. Can also be a vector of two numbers, where the first number is used in the calculation of the estimates and the second number is used when applying Peto's method. See below and the documentation of the escalc function for more details.
-----	--

to	character string to specify when the values under add should be added (either "only0", "all", "if0all", or "none"). Can also be a character vector, where the first string again applies when calculating the effect size estimates and the second string when applying Peto's method. See below and the documentation of the escalc function for more details.
drop00	logical to specify whether studies with no cases (or only cases) in both groups should be dropped when calculating the effect size estimates (the values for such studies are set to NA). Can also be a vector of two logicals, where the first applies to the calculation of the estimates and the second when applying Peto's method. See below and the documentation of the escalc function for more details.

These arguments pertain to the model / computations and output:

level	numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see here for details).
verbose	logical to specify whether output should be generated on the progress of the model fitting (the default is FALSE).
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is 4. See also here for further details on how to control the number of digits in the output.
...	additional arguments.

Details

Specifying the Data:

The studies are assumed to provide data in terms of 2×2 tables of the form:

	outcome 1	outcome 2	total
group 1	ai	bi	n1i
group 2	ci	di	n2i

where ai, bi, ci, and di denote the cell frequencies and n1i and n2i the row totals. For example, in a set of randomized clinical trials (RCTs) or cohort studies, group 1 and group 2 may refer to the treatment/exposed and placebo/control/non-exposed group, respectively, with outcome 1 denoting some event of interest (e.g., death) and outcome 2 its complement. In a set of case-control studies, group 1 and group 2 may refer to the group of cases and the group of controls, with outcome 1 denoting, for example, exposure to some risk factor and outcome 2 non-exposure.

Peto's Method:

An approach for aggregating data of this type was suggested by Peto (see Yusuf et al., 1985). The method provides a weighted estimate of the (log) odds ratio under an equal-effects model. The method is particularly advantageous when the event of interest is rare, but it should only be used when the group sizes within the individual studies are not too dissimilar and the effect sizes are generally small (Greenland & Salvani, 1990; Sweeting et al., 2004; Bradburn et al., 2007). Note that the printed results are given both in terms of the log and the raw units (for easier interpretation).

Calculating Effect Size Estimates of the Individual Studies:

Peto's method itself does not require the calculation of the observed log odds ratios of the individual studies and directly makes use of the cell frequencies in the 2×2 tables. Zero cells are not a problem (except in extreme cases, such as when one of the two outcomes never occurs in any of the tables). Therefore, it is unnecessary to add some constant to the cell counts when there are zero cells.

However, for plotting and various other functions, it is necessary to calculate the observed log odds ratios for the k studies. Here, zero cells can be problematic, so adding a constant value to the cell counts ensures that all k values can be calculated. The `add` and `to` arguments are used to specify what value should be added to the cell frequencies and under what circumstances when calculating the observed log odds ratios and when applying Peto's method. Similarly, the `drop00` argument is used to specify how studies with no cases (or only cases) in both groups should be handled. The documentation of the `escalc` function explains how the `add`, `to`, and `drop00` arguments work. If only a single value for these arguments is specified (as per default), then these values are used when calculating the observed log odds ratios and no adjustment to the cell counts is made when applying Peto's method. Alternatively, when specifying two values for these arguments, the first value applies when calculating the observed log odds ratios and the second value when applying Peto's method.

Note that `drop00` is set to `TRUE` by default. Therefore, the observed log odds ratios for studies where `ai=ci=0` or `bi=di=0` are set to `NA`. When applying Peto's method, such studies are not explicitly dropped (unless the second value of `drop00` argument is also set to `TRUE`), but this is practically not necessary, as they do not actually influence the results (assuming no adjustment to the cell counts are made when applying Peto's method).

Value

An object of class `c("rma.peto", "rma")`. The object is a list containing the following components:

<code>beta</code>	aggregated log odds ratio.
<code>se</code>	standard error of the aggregated value.
<code>zval</code>	test statistics of the aggregated value.
<code>pval</code>	corresponding p-value.
<code>ci.lb</code>	lower bound of the confidence interval.
<code>ci.ub</code>	upper bound of the confidence interval.
<code>QE</code>	test statistic of the test for heterogeneity.
<code>QEp</code>	corresponding p-value.
<code>k</code>	number of studies included in the analysis.
<code>yi, vi</code>	the vector of individual log odds ratios and corresponding sampling variances.
<code>fit.stats</code>	a list with the log-likelihood, deviance, AIC, BIC, and AICc values under the unrestricted and restricted likelihood.
<code>...</code>	some additional elements/values.

Methods

The results of the fitted model are formatted and printed with the `print` function. If fit statistics should also be given, use `summary` (or use the `fitstats` function to extract them).

The `residuals`, `rstandard`, and `rstudent` functions extract raw and standardized residuals. Leave-one-out diagnostics can be obtained with `leave1out`.

Forest, funnel, radial, L'Abbé, and Baujat plots can be obtained with `forest`, `funnel`, `radial`, `labbe`, and `baujat`. The `qqnorm` function provides normal QQ plots of the standardized residuals. One can also call `plot` on the fitted model object to obtain various plots at once.

A cumulative meta-analysis (i.e., adding one observation at a time) can be obtained with `cumul`.

Other extractor functions include `coef`, `vcov`, `se`, `logLik`, `deviance`, `AIC`, and `BIC`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Bradburn, M. J., Deeks, J. J., Berlin, J. A., & Localio, A. R. (2007). Much ado about nothing: A comparison of the performance of meta-analytical methods with rare events. *Statistics in Medicine*, **26**(1), 53–77. <https://doi.org/10.1002/sim.2528>
- Greenland, S., & Salvan, A. (1990). Bias in the one-step method for pooling study results. *Statistics in Medicine*, **9**(3), 247–252. <https://doi.org/10.1002/sim.4780090307>
- Sweeting, M. J., Sutton, A. J., & Lambert, P. C. (2004). What to add to nothing? Use and avoidance of continuity corrections in meta-analysis of sparse data. *Statistics in Medicine*, **23**(9), 1351–1375. <https://doi.org/10.1002/sim.1761>
- Yusuf, S., Peto, R., Lewis, J., Collins, R., & Sleight, P. (1985). Beta blockade during and after myocardial infarction: An overview of the randomized trials. *Progress in Cardiovascular Disease*, **27**(5), 335–371. [https://doi.org/10.1016/s0033-0620\(85\)80003-7](https://doi.org/10.1016/s0033-0620(85)80003-7)
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`rma.uni`, `rma.glmm`, `rma.mh`, and `rma.mv` for other model fitting functions.

`dat.collins1985a`, `dat.collins1985b`, and `dat.yusuf1985` for further examples of the use of the `rma.peto` function.

Examples

```
### meta-analysis of the (log) odds ratios using Peto's method
rma.peto(ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
```

Description

Function to fit meta-analytic equal-, fixed-, and random-effects models and (mixed-effects) meta-regression models using a linear (mixed-effects) model framework. See below and the introduction to the [metafor-package](#) for more details on these models.

Usage

```
rma.uni(yi, vi, sei, weights, ai, bi, ci, di, n1i, n2i,
        x1i, x2i, t1i, t2i, m1i, m2i, sd1i, sd2i,
        xi, mi, ri, ti, fi, pi, sdi, r2i, mini, maxi, ni, mods, scale,
        measure="GEN", data, slab, subset,
        add=1/2, to="only0", drop00=FALSE, intercept=TRUE,
        method="REML", weighted=TRUE, test="z",
        level=95, btt, att, tau2, verbose=FALSE, digits, control, ...)
```

```
rma(yi, vi, sei, weights, ai, bi, ci, di, n1i, n2i,
    x1i, x2i, t1i, t2i, m1i, m2i, sd1i, sd2i,
    xi, mi, ri, ti, fi, pi, sdi, r2i, mini, maxi, ni, mods, scale,
    measure="GEN", data, slab, subset,
    add=1/2, to="only0", drop00=FALSE, intercept=TRUE,
    method="REML", weighted=TRUE, test="z",
    level=95, btt, att, tau2, verbose=FALSE, digits, control, ...)
```

Arguments

These arguments pertain to data input:

<code>yi</code>	vector of length k with the effect size estimates. See ‘Details’.
<code>vi</code>	vector of length k with the corresponding sampling variances. See ‘Details’.
<code>sei</code>	vector of length k with the corresponding standard errors (only relevant when not using <code>vi</code>). See ‘Details’.
<code>weights</code>	optional argument to specify a vector of length k with user-defined weights. See ‘Details’.
<code>ai</code>	see below and the documentation of the escalc function for more details.
<code>bi</code>	see below and the documentation of the escalc function for more details.
<code>ci</code>	see below and the documentation of the escalc function for more details.
<code>di</code>	see below and the documentation of the escalc function for more details.
<code>n1i</code>	see below and the documentation of the escalc function for more details.
<code>n2i</code>	see below and the documentation of the escalc function for more details.
<code>x1i</code>	see below and the documentation of the escalc function for more details.

x2i	see below and the documentation of the escalc function for more details.
t1i	see below and the documentation of the escalc function for more details.
t2i	see below and the documentation of the escalc function for more details.
m1i	see below and the documentation of the escalc function for more details.
m2i	see below and the documentation of the escalc function for more details.
sd1i	see below and the documentation of the escalc function for more details.
sd2i	see below and the documentation of the escalc function for more details.
xi	see below and the documentation of the escalc function for more details.
mi	see below and the documentation of the escalc function for more details.
ri	see below and the documentation of the escalc function for more details.
ti	see below and the documentation of the escalc function for more details.
fi	see below and the documentation of the escalc function for more details.
pi	see below and the documentation of the escalc function for more details.
sdi	see below and the documentation of the escalc function for more details.
r2i	see below and the documentation of the escalc function for more details.
mini	see below and the documentation of the escalc function for more details.
maxi	see below and the documentation of the escalc function for more details.
ni	see below and the documentation of the escalc function for more details.
mods	optional argument to include one or more moderators in the model. A single moderator can be given as a vector of length k specifying the values of the moderator. Multiple moderators are specified by giving a matrix with k rows and as many columns as there are moderator variables. Alternatively, a model formula can be used to specify the model. See ‘Details’.
scale	optional argument to include one or more predictors for the scale part in a location-scale model. See ‘Details’.
measure	character string to specify the type of data supplied to the function. See ‘Details’.
data	optional data frame containing the data supplied to the function.
slab	optional vector with labels for the k studies.
subset	optional (logical or numeric) vector to specify the subset of studies that should be used for the analysis.

These arguments pertain to handling of zero cells/counts/frequencies:

add	see the documentation of the escalc function.
to	see the documentation of the escalc function.
drop00	see the documentation of the escalc function.

These arguments pertain to the model / computations and output:

intercept	logical to specify whether an intercept should be added to the model (the default is TRUE). Ignored when mods is a formula.
-----------	---

method	character string to specify whether an equal- or a random-effects model should be fitted. An equal-effects model is fitted when using <code>method="EE"</code> . A random-effects model is fitted by setting <code>method</code> equal to one of the following: <code>"DL"</code> , <code>"HE"</code> (or <code>"CO"</code>), <code>"HS"</code> , <code>"Hsk"</code> , <code>"SJ"</code> , <code>"ML"</code> , <code>"REML"</code> , <code>"EB"</code> , <code>"PM"</code> , <code>"GENQ"</code> , <code>"PMM"</code> , or <code>"GENQM"</code> . The default is <code>"REML"</code> . See ‘Details’.
weighted	logical to specify whether weighted (default) or unweighted estimation should be used to fit the model (the default is <code>TRUE</code>).
test	character string to specify how test statistics and confidence intervals for the fixed effects should be computed. By default (<code>test="z"</code>), Wald-type tests and CIs are obtained, which are based on a standard normal distribution. When <code>test="t"</code> , a t-distribution is used instead. When <code>test="knha"</code> , the method by Knapp and Hartung (2003) is used. See ‘Details’ and also here for some recommended practices.
level	numeric value between 0 and 100 to specify the confidence interval level (the default is 95; see here for details).
btt	optional vector of indices to specify which coefficients to include in the omnibus test of moderators. Can also be a string to grep for. See ‘Details’.
att	optional vector of indices to specify which scale coefficients to include in the omnibus test. Only relevant for location-scale models. See ‘Details’.
tau2	optional numeric value to specify the amount of (residual) heterogeneity in a random- or mixed-effects model (instead of estimating it). Useful for sensitivity analyses (e.g., for plotting results as a function of τ^2). If unspecified, the value of τ^2 is estimated from the data.
verbose	logical to specify whether output should be generated on the progress of the model fitting (the default is <code>FALSE</code>). Can also be an integer. Values > 1 generate more verbose output. See ‘Note’.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is 4. See also here for further details on how to control the number of digits in the output.
control	optional list of control values for the iterative estimation algorithms. If unspecified, default values are defined inside the function. See ‘Note’.
...	additional arguments.

Details

Specifying the Data:

The function can be used in combination with any of the usual effect size measures used in meta-analyses (e.g., log risk ratios, log odds ratios, risk differences, mean differences, standardized mean differences, log transformed ratios of means, raw correlation coefficients, correlation coefficients transformed with Fisher’s *r*-to-*z* transformation), or, more generally, any set of estimates (with corresponding sampling variances) one would like to analyze. Simply specify the estimates via the `yi` argument and the corresponding sampling variances via the `vi` argument. Instead of specifying `vi`, one can specify the standard errors (the square root of the sampling variances) via the `sei` argument. The [escalc](#) function can be used to compute a wide variety of effect size measures (and the corresponding sampling variances) based on various summary statistics. If the estimates were computed with some other software and they are one of the types of measures

implemented in [escalc](#), one can use the `measure` argument to declare what the estimates actually represent (this is for example useful so that plots based on a model object automatically use correct axis labels).

Note that the second argument of the function is used to specify the sampling variances of the estimates, not the standard errors. So, assuming that the variable with the standard errors is called `'stderrs'`, writing `rma(yi, stderrs)` would incorrectly pass the standard errors of the estimates to the second argument for the sampling variances, which would yield incorrect results. Instead, `rma(yi, sei=stderrs)` or `rma(yi, stderrs^2)` would yield correct results. See [here](#) for further details.

Alternatively, the function can automatically calculate the values of a chosen effect size measure (and the corresponding sampling variances) when supplied with the necessary data. The [escalc](#) function describes which measures are currently implemented and what data/arguments should then be specified/used. The `measure` argument should then be set to the desired measure.

Specifying the Model:

The function can be used to fit equal-, fixed-, and random-effects models, as well as (mixed-effects) meta-regression models including one or multiple moderators (the difference between the various models is described in detail on the introductory [metafor-package](#) help page).

Assuming the effect size estimates and corresponding sampling variances are supplied via the `yi` and `vi` arguments, an *equal-effects model* can be fitted with `rma(yi, vi, method="EE")`. Setting `method="FE"` fits a *fixed-effects model* (see [here](#) for a discussion of this model and how the interpretation of these models differ despite yielding identical results). Weighted estimation (with inverse-variance weights) is used by default. User-defined weights can be supplied via the `weights` argument. Unweighted estimation can be used by setting `weighted=FALSE` (which is the same as setting the weights equal to a constant).

A *random-effects model* can be fitted with the same code but setting the `method` argument to one of the various estimators for the amount of heterogeneity:

- `method="DL"` for the DerSimonian-Laird estimator (DerSimonian & Laird, 1986; Raudenbush, 2009),
- `method="HE"` for the Hedges estimator (Cochran, 1954; Hedges, 1983, 1992),
- `method="HS"` for the Hunter-Schmidt estimator (Hunter & Schmidt, 1990; Viechtbauer et al., 2015),
- `method="Hsk"` for the Hunter-Schmidt estimator with a small sample-size correction (Bran-nick et al., 2019),
- `method="SJ"` for the Sidik-Jonkman estimator (Sidik & Jonkman, 2005b, 2007),
- `method="ML"` for the maximum likelihood estimator (Hardy & Thompson, 1996; Raudenbush, 2009),
- `method="REML"` for the restricted maximum likelihood estimator (Viechtbauer, 2005; Raudenbush, 2009)
- `method="EB"` for the empirical Bayes estimator (Morris, 1983; Berkey et al. 1995),
- `method="PM"` for the Paule-Mandel estimator (Paule & Mandel, 1982; Viechtbauer et al., 2015),
- `method="GENQ"` for the generalized Q-statistic estimator (DerSimonian & Kacker, 2007; Jackson et al., 2014),
- `method="PMM"` for the median-unbiased Paule-Mandel estimator (Viechtbauer, 2021),
- `method="GENQM"` for the median-unbiased generalized Q-statistic estimator (Viechtbauer, 2021).

For a description of the various estimators, see Brannick et al. (2019), DerSimonian and Kacker (2007), Raudenbush (2009), Veroniki et al. (2016), Viechtbauer (2005), and Viechtbauer et al. (2015). Note that the Hedges estimator is also called the ‘variance component estimator’ or ‘Cochran estimator’ (hence, one can also use `method="VC"` or `method="CO"` to choose this estimator), the Sidik-Jonkman estimator is also called the ‘model error variance estimator’, the empirical Bayes estimator is actually identical to the Paule-Mandel estimator (Viechtbauer et al., 2015), and the generalized Q-statistic estimator is a general method-of-moments estimator (DerSimonian & Kacker, 2007) requiring the specification of weights (the HE and DL estimators are just special cases with equal and inverse sampling variance weights, respectively). Finally, the two median-unbiased estimators are versions of the Paule-Mandel and generalized Q-statistic estimators that equate the respective estimating equations not to their expected values, but to the medians of their theoretical distributions (Viechtbauer, 2021).

One or more moderators can be included in a model via the `mods` argument. A single moderator can be given as a (row or column) vector of length k specifying the values of the moderator. Multiple moderators are specified by giving an appropriate model matrix (i.e., X) with k rows and as many columns as there are moderator variables (e.g., `mods = cbind(mod1, mod2, mod3)`, where `mod1`, `mod2`, and `mod3` correspond to the names of the variables for three moderator variables). The intercept is added to the model matrix by default unless `intercept=FALSE`.

Alternatively, one can use standard [formula](#) syntax to specify the model. In this case, the `mods` argument should be set equal to a one-sided formula of the form `mods = ~ model` (e.g., `mods = ~ mod1 + mod2 + mod3`). Interactions, polynomial/spline terms, and factors can be easily added to the model in this manner. When specifying a model formula via the `mods` argument, the `intercept` argument is ignored. Instead, the inclusion/exclusion of the intercept is controlled by the specified formula (e.g., `mods = ~ 0 + mod1 + mod2 + mod3` or `mods = ~ mod1 + mod2 + mod3 - 1` would lead to the removal of the intercept).

When the effect size estimates and corresponding sampling variances are supplied via the `yi` and `vi` (or `sei`) arguments, one can also specify moderators via the `yi` argument (e.g., `rma(yi ~ mod1 + mod2 + mod3, vi)`). In that case, the `mods` argument is ignored and the inclusion/exclusion of the intercept is again controlled by the specified formula.

Omnibus Test of Moderators:

For models including moderators, an omnibus test of all model coefficients is conducted that excludes the intercept (the first coefficient) if it is included in the model. If no intercept is included in the model, then the omnibus test includes all coefficients in the model including the first. Alternatively, one can manually specify the indices of the coefficients to test via the `btt` (‘betas to test’) argument (i.e., to test $H_0: \beta_{j \in \text{btt}} = 0$, where $\beta_{j \in \text{btt}}$ is the set of coefficients to be tested). For example, with `btt=c(3,4)`, only the third and fourth coefficients from the model are included in the test (if an intercept is included in the model, then it corresponds to the first coefficient in the model). Instead of specifying the coefficient numbers, one can specify a string for `btt`. In that case, [grep](#) will be used to search for all coefficient names that match the string. The omnibus test is called the Q_M -test and follows asymptotically a chi-square distribution with m degrees of freedom (with m denoting the number of coefficients tested) under the null hypothesis (that the true value of all coefficients tested is equal to 0).

Categorical Moderators:

Categorical moderator variables can be included in the model via the `mods` argument in the same way that appropriately (dummy) coded categorical variables can be included in linear models. One can either do the dummy coding manually or use a model formula together with the [factor](#)

function to automate the coding (note that string/character variables in a model formula are automatically converted to factors). An example to illustrate these different approaches is provided below.

Tests and Confidence Intervals:

By default, tests of individual coefficients in the model (and the corresponding confidence intervals) are based on a standard normal distribution, while the omnibus test is based on a chi-square distribution (see above). As an alternative, one can set `test="t"`, in which case tests of individual coefficients and confidence intervals are based on a t-distribution with $k - p$ degrees of freedom, while the omnibus test then uses an F-distribution with m and $k - p$ degrees of freedom (with k denoting the total number of estimates included in the analysis and p the total number of model coefficients including the intercept if it is present). Furthermore, when `test="knha"` (or equivalently, `test="hksj"`), the method by Hartung (1999), Sidik and Jonkman (2002), and Knapp and Hartung (2003) (the Knapp-Hartung method; also referred to as the Hartung-Knapp-Sidik-Jonkman method) is used, which applies an adjustment to the standard errors of the estimated coefficients (to account for the uncertainty in the estimate of the amount of (residual) heterogeneity) and uses t- and F-distributions as described above (see also [here](#)). Finally, one can set `test="adhoc"`, in which case the Knapp-Hartung method is used, but with the restriction that the adjustment to the standard errors can never result in adjusted standard errors that are smaller than the unadjusted ones (see Jackson et al., 2017, section 4.3).

Test for (Residual) Heterogeneity:

A test for (residual) heterogeneity is automatically carried out by the function. Without moderators in the model, this is simply Cochran's Q -test (Cochran, 1954), which tests whether the variability in the estimates is larger than would be expected based on sampling variability alone. A significant test suggests that the true effects are heterogeneous. When moderators are included in the model, this is the Q_E -test for residual heterogeneity, which tests whether the variability in the estimates not accounted for by the moderators included in the model is larger than would be expected based on sampling variability alone.

Location-Scale Models:

The function can also be used to fit so-called 'location-scale models' (Viechtbauer & López-López, 2022). In such models, one can specify not only predictors for the size of the average true effect (i.e., for their 'location'), but also predictors for the amount of heterogeneity in the true effects (i.e., for their 'scale'). The model is given by

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_{p'} x_{ip'} + u_i + \varepsilon_i,$$

$$u_i \sim N(0, \tau_i^2), \varepsilon_i \sim N(0, v_i),$$

$$\log(\tau_i^2) = \alpha_0 + \alpha_1 z_{i1} + \alpha_2 z_{i2} + \dots + \alpha_{q'} z_{iq'},$$

where $x_{i1}, \dots, x_{ip'}$ are the values of the p' predictor variables that may be related to the size of the average true effect (letting $p = p' + 1$ denote the total number of location coefficients in the model including the model intercept β_0) and $z_{i1}, \dots, z_{iq'}$ are the values of the q' scale variables that may be related to the amount of heterogeneity in the true effects (letting $q = q' + 1$ denote the total number of scale coefficients in the model including the model intercept α_0). Location variables can be specified via the `mods` argument as described above (e.g., `mods = ~ mod1 + mod2 + mod3`). Scale variables can be specified via the `scale` argument (e.g., `scale = ~ var1 + var2 + var3`). A log link is used for specifying the relationship between the scale variables and the amount of

heterogeneity so that τ_i^2 is guaranteed to be non-negative (one can also set (the undocumented) argument `link="identity"` to use an identity link, but this is more likely to lead to estimation problems). Estimates of the location and scale coefficients can be obtained either with maximum likelihood (`method="ML"`) or restricted maximum likelihood (`method="REML"`) estimation. An omnibus test of the scale coefficients is conducted as described above (where the `att` argument can be used to specify which scale coefficients to include in the test).

Value

An object of class `c("rma.uni", "rma")`. The object is a list containing the following components:

<code>beta</code>	estimated coefficients of the model.
<code>se</code>	standard errors of the coefficients.
<code>zval</code>	test statistics of the coefficients.
<code>pval</code>	corresponding p-values.
<code>ci.lb</code>	lower bound of the confidence intervals for the coefficients.
<code>ci.ub</code>	upper bound of the confidence intervals for the coefficients.
<code>vb</code>	variance-covariance matrix of the estimated coefficients.
<code>tau2</code>	estimated amount of (residual) heterogeneity. Always 0 when <code>method="EE"</code> .
<code>se.tau2</code>	standard error of the estimated amount of (residual) heterogeneity.
<code>k</code>	number of studies included in the analysis.
<code>p</code>	number of coefficients in the model (including the intercept).
<code>m</code>	number of coefficients included in the omnibus test of moderators.
<code>QE</code>	test statistic of the test for (residual) heterogeneity.
<code>QEp</code>	corresponding p-value.
<code>QM</code>	test statistic of the omnibus test of moderators.
<code>QMp</code>	corresponding p-value.
<code>I2</code>	value of I^2 . See print for more details.
<code>H2</code>	value of H^2 . See print for more details.
<code>R2</code>	value of R^2 . See print for more details.
<code>int.only</code>	logical that indicates whether the model is an intercept-only model.
<code>yi, vi, X</code>	the vector of estimates, the corresponding sampling variances, and the model matrix.
<code>fit.stats</code>	a list with the log-likelihood, deviance, AIC, BIC, and AICc values under the unrestricted and restricted likelihood.
<code>...</code>	some additional elements/values.

For location-scale models, the object is of class `c("rma.ls", "rma.uni", "rma")` and includes the following components in addition to the ones listed above:

<code>alpha</code>	estimated scale coefficients of the model.
<code>se.alpha</code>	standard errors of the coefficients.

<code>zval.alpha</code>	test statistics of the coefficients.
<code>pval.alpha</code>	corresponding p-values.
<code>ci.lb.alpha</code>	lower bound of the confidence intervals for the coefficients.
<code>ci.ub.alpha</code>	upper bound of the confidence intervals for the coefficients.
<code>va</code>	variance-covariance matrix of the estimated coefficients.
<code>tau2</code>	as above, but now a vector of values.
<code>q</code>	number of scale coefficients in the model (including the intercept).
<code>QS</code>	test statistic of the omnibus test of the scale coefficients.
<code>QSp</code>	corresponding p-value.
<code>...</code>	some additional elements/values.

Methods

The results of the fitted model are formatted and printed with the `print` function. If fit statistics should also be given, use `summary` (or use the `fitstats` function to extract them). Full versus reduced model comparisons in terms of fit statistics and likelihood ratio tests can be obtained with `anova`. Wald-type tests for sets of model coefficients or linear combinations thereof can be obtained with the same function. Permutation tests for the model coefficient(s) can be obtained with `permutest`. Tests and confidence intervals based on (cluster) robust methods can be obtained with `robust`.

Predicted/fitted values can be obtained with `predict` and `fitted`. For best linear unbiased predictions, see `blup` and `ranef`.

The `residuals`, `rstandard`, and `rstudent` functions extract raw and standardized residuals. Additional model diagnostics (e.g., to determine influential studies) can be obtained with the `influence` function. For models without moderators, leave-one-out diagnostics can also be obtained with `leave1out`. For models with moderators, variance inflation factors can be obtained with `vif`.

A confidence interval for the amount of (residual) heterogeneity in the random/mixed-effects model can be obtained with `confint`. For location-scale models, `confint` can provide confidence intervals for the scale coefficients.

Forest, funnel, radial, L'Abbé, and Baujat plots can be obtained with `forest`, `funnel`, `radial`, `labbe`, and `baujat` (radial and L'Abbé plots only for models without moderators). The `qqnorm` function provides normal QQ plots of the standardized residuals. One can also call `plot` on the fitted model object to obtain various plots at once. For random/mixed-effects models, the `profile` function can be used to obtain a plot of the (restricted) log-likelihood as a function of τ^2 . For location-scale models, `profile` draws analogous plots based on the scale coefficients. For models with moderators, `regplot` draws scatter plots / bubble plots, showing the (marginal) relationship between the estimates and a selected moderator from the model.

Tests for funnel plot asymmetry (which may be indicative of publication bias) can be obtained with `ranktest` and `regtest`. For models without moderators, the `trimfill` method can be used to carry out a trim and fill analysis and `hc` provides a random-effects model analysis that is more robust to publication bias (based on the method by Henmi & Copas, 2010). The test of 'excess significance' can be carried out with the `tes` function. The fail-safe N (based on a file drawer analysis) can be computed using `fsn`. Selection models can be fitted with the `selmodel` function.

For models without moderators, a cumulative meta-analysis (i.e., adding one observation at a time) can be obtained with `cumul`.

Other extractor functions include `coef`, `vcov`, `se`, `logLik`, `deviance`, `AIC`, `BIC`, `hatvalues`, and `weights`.

Note

While the HS, HSk, HE, DL, SJ, and GENQ estimators of τ^2 are based on closed-form solutions, the ML, REML, and EB estimators must be obtained iteratively. For this, the function makes use of the Fisher scoring algorithm, which is robust to poor starting values and usually converges quickly (Harville, 1977; Jennrich & Sampson, 1976). By default, the starting value is set equal to the value of the Hedges (HE) estimator and the algorithm terminates when the change in the estimated value of τ^2 is smaller than 10^{-5} from one iteration to the next. The maximum number of iterations is 100 by default (which should be sufficient in most cases). Information on the progress of the algorithm can be obtained by setting `verbose=TRUE`. One can also set `verbose` to an integer (`verbose=2` yields even more information and `verbose=3` also sets `option(warn=1)` temporarily).

A different starting value, threshold, and maximum number of iterations can be specified via the `control` argument by setting `control=list(tau2.init=value, threshold=value, maxiter=value)`. The step length of the Fisher scoring algorithm can also be adjusted by a desired factor with `control=list(stepadj=value)` (values below 1 will reduce the step length). If using `verbose=TRUE` shows the estimate jumping around erratically (or cycling through a few values), decreasing the step length (and increasing the maximum number of iterations) can often help with convergence (e.g., `control=list(stepadj=0.5, maxiter=1000)`).

The PM, PMM, and GENQM estimators also involve iterative algorithms, which make use of the `uniroot` function. By default, the desired accuracy (`tol`) is set equal to `.Machine$double.eps^0.25` and the maximum number of iterations (`maxiter`) to 100 (as above). The upper bound of the interval searched (`tau2.max`) is set to the larger of 100 and $10 \times \text{mad}(y_i)^2$ (i.e., 10 times the squared median absolute deviation of the estimates computed with the `mad` function). These values can be adjusted with `control=list(tol=value, maxiter=value, tau2.max=value)`.

All of the heterogeneity estimators except SJ can in principle yield negative estimates for the amount of (residual) heterogeneity. However, negative estimates of τ^2 are outside of the parameter space. For the HS, HSk, HE, DL, and GENQ estimators, negative estimates are therefore truncated to zero. For the ML, REML, and EB estimators, the Fisher scoring algorithm makes use of step halving (Jennrich & Sampson, 1976) to guarantee a non-negative estimate. Finally, for the PM, PMM, and GENQM estimators, the lower bound of the interval searched is set to zero by default. For those brave enough to step into risky territory, there is the option to set the lower bound for all these estimators to some other value besides zero (even a negative one) with `control=list(tau2.min=value)`, but the lowest value permitted is $-\min(v_i)$ (to ensure that the marginal variances are always non-negative).

The Hunter-Schmidt estimator for the amount of heterogeneity is defined in Hunter and Schmidt (1990) only in the context of the random-effects model when analyzing correlation coefficients. A general version of this estimator for random- and mixed-effects models not specific to any particular effect size measure is described in Viechtbauer (2005) and Viechtbauer et al. (2015) and is implemented here.

The Sidik-Jonkman estimator starts with a crude estimate of τ^2 , which is then updated as described in Sidik and Jonkman (2005b, 2007). If, instead of the crude estimate, one wants to use a better a priori estimate, one can do so by passing this value via `control=list(tau2.init=value)`.

One can also specify a vector of estimators via the `method` argument (e.g., `rma(yi, vi, method=c("REML", "DL"))`). The various estimators are then applied in turn until one converges. This is mostly useful for sim-

ulation studies where an estimator (like the REML estimator) is not guaranteed to converge and one can then substitute one (like the DL estimator) that does not involve iterative methods and is guaranteed to provide an estimate.

Estimates with non-positive sampling variances are problematic. If a sampling variance is equal to zero, then its weight will be $1/0$ for equal-effects models when using weighted estimation. Switching to unweighted estimation is a possible solution then. For random/mixed-effects model, some estimators of τ^2 are undefined when there is at least one sampling variance equal to zero. Other estimators may work, but it may still be necessary to switch to unweighted model fitting, especially when the estimate of τ^2 converges to zero.

When including moderators in the model, it is possible that the model matrix is not of full rank (i.e., there is a linear relationship between the moderator variables included in the model). The function automatically tries to reduce the model matrix to full rank by removing redundant predictors, but if this fails the model cannot be fitted and an error will be issued. Deleting (redundant) moderator variables from the model as needed should solve this problem.

Some general words of caution about the assumptions underlying the models:

- The sampling variances (i.e., the v_i values) are treated as if they are known constants, even though in practice they are usually estimates themselves. This implies that the distributions of the test statistics and corresponding confidence intervals are only exact and have nominal coverage when the within-study sample sizes are large (i.e., when the error in the sampling variance estimates is small). Certain effect size measures (e.g., the arcsine square root transformed risk difference and Fisher's r -to- z transformed correlation coefficient) are based on variance stabilizing transformations that also help to make the assumption of known sampling variances much more reasonable.
- When fitting a mixed/random-effects model, τ^2 is estimated and then treated as a known constant thereafter. This ignores the uncertainty in the estimate of τ^2 . As a consequence, the standard errors of the parameter estimates tend to be too small, yielding test statistics that are too large and confidence intervals that are not wide enough. The Knapp and Hartung (2003) adjustment (i.e., using `test="knha"`) can be used to counter this problem, yielding test statistics and confidence intervals whose properties are closer to nominal.
- Most effect size measures do not have exactly normal sampling distributions as assumed under the various models. However, the normal approximation usually becomes more accurate for most measures as the within-study sample sizes increase. Therefore, sufficiently large within-study sample sizes are (usually) needed to be certain that the tests and confidence intervals have nominal levels/coverage. Again, certain measures (e.g., Fisher's r -to- z transformed correlation coefficient) may be preferable from this perspective as well.

For location-scale models, model fitting is done via numerical optimization over the model parameters. By default, `nlminb` is used for the optimization. One can also choose a different optimizer from `optim` via the control argument (e.g., `control=list(optimizer="BFGS")` or `control=list(optimizer="Nelder-Mead")`). Besides `nlminb` and one of the methods from `optim`, one can also choose one of the optimizers from the `minqa` package (i.e., `uobyqa`, `newuoa`, or `bobyqa`), one of the (derivative-free) algorithms from the `nloptr` package, the Newton-type algorithm implemented in `nlm`, the various algorithms implemented in the `dfoptim` package (`hjk` for the Hooke-Jeeves, `nmk` for the Nelder-Mead, and `mads` for the Mesh Adaptive Direct Searches algorithm), the quasi-Newton type optimizers `ucminf` and `lbfgsb3c` and the subspace-searching simplex algorithm `subplex` from the packages of the same name, the Barzilai-Borwein gradient decent method implemented in `BBoptim`,

the `Rcgmin` and `Rvmmmin` optimizers, or the parallelized version of the L-BFGS-B algorithm implemented in `optimParallel` from the package of the same name. When using an identity link with `link="identity"`, constrained optimization (to ensure non-negative τ_i^2 values) as implemented in `constrOptim` is used by default. Alternative optimizers in this case are the `solnp` solver from the `Rsolnp` package, `nloptr`, or the augmented Lagrangian algorithms `constrOptim.nl` and `auglag` from the `alabama` package.

The optimizer name must be given as a character string (i.e., in quotes). Additional control parameters can be specified via the `control` argument (e.g., `control=list(iter.max=1000, rel.tol=1e-8)`). For `nloptr`, the default is to use the BOBYQA implementation from that package with a relative convergence criterion of $1e-8$ on the function value (i.e., log-likelihood), but this can be changed via the `algorithm` and `ftol_rel` arguments (e.g., `control=list(optimizer="nloptr", algorithm="NLOPT_LN_SBPLX", ftol_rel=1e-6)`) (note: when using `optimizer="nloptr"` in combination with an identity link, the "NLOPT_LN_COBYLA" algorithm is automatically used, since it allows for inequality constraints). For `optimParallel`, the `control` argument `ncpus` can be used to specify the number of cores to use for the parallelization (e.g., `control=list(optimizer="optimParallel", ncpus=2)`). Control argument `mfmexit` (which is 10^5 by default and is independent of the control arguments of the various optimizers) hard exits when the specified number of iterations has been exceeded.

Under certain circumstances (e.g., when the amount of heterogeneity is very small for certain combinations of values for the scale variables and scale coefficients), the values of the scale coefficients may try to drift towards minus or plus infinity, which can lead to problems with the optimization. One can impose constraints on the scale coefficients via `control=list(alpha.min=minval, alpha.max=maxval)` where `minval` and `maxval` are either scalars or vectors of the appropriate length.

Finally, for location-scale models, the standard errors of the scale coefficients are obtained by inverting the Hessian, which is numerically approximated using the `hessian` function from the `numDeriv` package. This may fail (especially when using an identity link), leading to NA values for the standard errors and hence test statistics, p-values, and confidence interval bounds. One can set `control` argument `hessianCtrl` to a list of named arguments to be passed on to the `method.args` argument of the `hessian` function (the default is `control=list(hessianCtrl=list(r=8))`). One can also set `control=list(hesspack="pracma")` or `control=list(hesspack="calculus")` in which case the `pracma::hessian` or `calculus::hessian` functions from the respective packages are used instead for approximating the Hessian.

Even if the Hessian can be approximated and inverted, the standard errors may be unreasonably large when the likelihood surface is very flat around the estimated scale coefficients. This is more likely to happen when k is small and when the amount of heterogeneity is very small under some conditions as defined by the scale coefficients/variables. Setting constraints on the scale coefficients as described above can also help to mitigate this issue.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Berkey, C. S., Hoaglin, D. C., Mosteller, F., & Colditz, G. A. (1995). A random-effects regression model for meta-analysis. *Statistics in Medicine*, **14**(4), 395–411. <https://doi.org/10.1002/sim.4780140406>

- Brannick, M. T., Potter, S. M., Benitez, B., & Morris, S. B. (2019). Bias and precision of alternate estimators in meta-analysis: Benefits of blending Schmidt–Hunter and Hedges approaches. *Organizational Research Methods*, **22**(2), 490–514. <https://doi.org/10.1177/1094428117741966>
- Cochran, W. G. (1954). The combination of estimates from different experiments. *Biometrics*, **10**(1), 101–129. <https://doi.org/10.2307/3001666>
- DerSimonian, R., & Laird, N. (1986). Meta-analysis in clinical trials. *Controlled Clinical Trials*, **7**(3), 177–188. [https://doi.org/10.1016/0197-2456\(86\)90046-2](https://doi.org/10.1016/0197-2456(86)90046-2)
- DerSimonian, R., & Kacker, R. (2007). Random-effects model for meta-analysis of clinical trials: An update. *Contemporary Clinical Trials*, **28**(2), 105–114. <https://doi.org/10.1016/j.cct.2006.04.004>
- Hardy, R. J. & Thompson, S. G. (1996). A likelihood approach to meta-analysis with random effects. *Statistics in Medicine*, **15**(6), 619–629. [https://doi.org/10.1002/\(SICI\)1097-0258\(19960330\)15:6<619::AID-388-395>3.0.CO;2-W](https://doi.org/10.1002/(SICI)1097-0258(19960330)15:6<619::AID-388-395>3.0.CO;2-W)
- Hartung, J. (1999). An alternative method for meta-analysis. *Biometrical Journal*, **41**(8), 901–916. [https://doi.org/10.1002/\(SICI\)1521-4036\(199912\)41:8<901::AID-BIMJ901>3.0.CO;2-W](https://doi.org/10.1002/(SICI)1521-4036(199912)41:8<901::AID-BIMJ901>3.0.CO;2-W)
- Harville, D. A. (1977). Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, **72**(358), 320–338. <https://doi.org/10.2307/2286796>
- Hedges, L. V. (1983). A random effects model for effect sizes. *Psychological Bulletin*, **93**(2), 388–395. <https://doi.org/10.1037/0033-2909.93.2.388>
- Hedges, L. V. (1992). Meta-analysis. *Journal of Educational Statistics*, **17**(4), 279–296. <https://doi.org/10.3102/10769>
- Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. San Diego, CA: Academic Press.
- Henmi, M., & Copas, J. B. (2010). Confidence intervals for random effects meta-analysis and robustness to publication bias. *Statistics in Medicine*, **29**(29), 2969–2983. <https://doi.org/10.1002/sim.4029>
- Hunter, J. E., & Schmidt, F. L. (1990). *Methods of meta-analysis: Correcting error and bias in research findings*. Thousand Oaks, CA: Sage.
- Jackson, D., Turner, R., Rhodes, K. & Viechtbauer, W. (2014). Methods for calculating confidence and credible intervals for the residual between-study variance in random effects meta-regression models. *BMC Medical Research Methodology*, **14**, 103. <https://doi.org/10.1186/1471-2288-14-103>
- Jackson, D., Law, M., Rücker, G., & Schwarzer, G. (2017). The Hartung-Knapp modification for random-effects meta-analysis: A useful refinement but are there any residual concerns? *Statistics in Medicine*, **36**(25), 3923–3934. <https://doi.org/10.1002/sim.7411>
- Jennrich, R. I., & Sampson, P. F. (1976). Newton-Raphson and related algorithms for maximum likelihood variance component estimation. *Technometrics*, **18**(1), 11–17. <https://doi.org/10.2307/1267911>
- Knapp, G., & Hartung, J. (2003). Improved tests for a random effects meta-regression with a single covariate. *Statistics in Medicine*, **22**(17), 2693–2710. <https://doi.org/10.1002/sim.1482>
- Morris, C. N. (1983). Parametric empirical Bayes inference: Theory and applications. *Journal of the American Statistical Association*, **78**(381), 47–55. <https://doi.org/10.2307/2287098>
- Paule, R. C., & Mandel, J. (1982). Consensus values and weighting factors. *Journal of Research of the National Bureau of Standards*, **87**(5), 377–385. <https://doi.org/10.6028/jres.087.022>
- Raudenbush, S. W. (2009). Analyzing effect sizes: Random effects models. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 295–315). New York: Russell Sage Foundation.
- Sidik, K. & Jonkman, J. N. (2002). A simple confidence interval for meta-analysis. *Statistics in Medicine*, **21**(21), 3153–3159. <https://doi.org/10.1002/sim.1262>

- Sidik, K., & Jonkman, J. N. (2005a). A note on variance estimation in random effects meta-regression. *Journal of Biopharmaceutical Statistics*, **15**(5), 823–838. <https://doi.org/10.1081/BIP-200067915>
- Sidik, K., & Jonkman, J. N. (2005b). Simple heterogeneity variance estimation for meta-analysis. *Journal of the Royal Statistical Society, Series C*, **54**(2), 367–384. <https://doi.org/10.1111/j.1467-9876.2005.00489.>
- Sidik, K., & Jonkman, J. N. (2007). A comparison of heterogeneity variance estimators in combining results of studies. *Statistics in Medicine*, **26**(9), 1964–1981. <https://doi.org/10.1002/sim.2688>
- Veroniki, A. A., Jackson, D., Viechtbauer, W., Bender, R., Bowden, J., Knapp, G., Kuss, O., Higgins, J. P., Langan, D., & Salanti, G. (2016). Methods to estimate the between-study variance and its uncertainty in meta-analysis. *Research Synthesis Methods*, **7**(1), 55–79. <https://doi.org/10.1002/jrsm.1164>
- Viechtbauer, W. (2005). Bias and efficiency of meta-analytic variance estimators in the random-effects model. *Journal of Educational and Behavioral Statistics*, **30**(3), 261–293. <https://doi.org/10.3102/10769986030>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W. (2021). Median-unbiased estimators for the amount of heterogeneity in meta-analysis. *European Congress of Methodology*, Valencia, Spain. <https://www.wvbauer.com/lib/exe/fetch.php/talks:2>
- Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*. **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>
- Viechtbauer, W., López-López, J. A., Sánchez-Meca, J., & Marín-Martínez, F. (2015). A comparison of procedures to test for moderators in mixed-effects meta-regression models. *Psychological Methods*, **20**(3), 360–374. <https://doi.org/10.1037/met0000023>

See Also

[rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for other model fitting functions.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit a random-effects model using the log risk ratios and sampling variances as input
### note: method="REML" is the default, so one could leave this out
rma(yi, vi, data=dat, method="REML")

### fit a random-effects model using the log risk ratios and standard errors as input
### note: the second argument of rma() is for the *sampling variances*, so we use the
### named argument 'sei' to supply the standard errors to the function
dat$sei <- sqrt(dat$vi)
rma(yi, sei=sei, data=dat)

### fit a random-effects model supplying the 2x2 table cell frequencies to the function
### and requesting that log risk ratios (measure="RR") should be computed based on the data
rma(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat)

### fit a mixed-effects model with two moderators (absolute latitude and publication year)
rma(yi, vi, mods=cbind(ablat, year), data=dat)

### using a model formula to specify the same model
```

```

rma(yi, vi, mods = ~ ablat + year, data=dat)

### using a model formula as part of the yi argument
rma(yi ~ ablat + year, vi, data=dat)

### manual dummy coding of the allocation factor
alloc.random    <- ifelse(dat$alloc == "random",    1, 0)
alloc.alternate  <- ifelse(dat$alloc == "alternate", 1, 0)
alloc.systematic <- ifelse(dat$alloc == "systematic", 1, 0)

### test the allocation factor (in the presence of the other moderators)
### note: 'alternate' is the reference level of the allocation factor,
###       since this is the dummy/level we leave out of the model
### note: the intercept is the first coefficient, so with btt=2:3 we test
###       coefficients 2 and 3, corresponding to the coefficients for the
###       allocation factor
rma(yi, vi, mods = ~ alloc.random + alloc.systematic + year + ablat, data=dat, btt=2:3)

### using a model formula to specify the same model
rma(yi, vi, mods = ~ factor(alloc) + year + ablat, data=dat, btt=2:3)

### factor() is not needed as character variables are automatically converted to factors
res <- rma(yi, vi, mods = ~ alloc + year + ablat, data=dat, btt=2:3)
res

### test all pairwise differences between the 'alloc' levels
anova(res, X=pairmat(btt="alloc"))

### subgrouping versus using a single model with a factor (subgrouping provides
### an estimate of tau^2 within each subgroup, but the number of studies in each
### subgroup is quite small; the model with the allocation factor provides a
### single estimate of tau^2 based on a larger number of studies, but assumes
### that tau^2 is the same within each subgroup)
res.a <- rma(yi, vi, data=dat, subset=(alloc=="alternate"))
res.r <- rma(yi, vi, data=dat, subset=(alloc=="random"))
res.s <- rma(yi, vi, data=dat, subset=(alloc=="systematic"))
res.a
res.r
res.s
res <- rma(yi, vi, mods = ~ 0 + factor(alloc), data=dat)
res

#####

### demonstrating that Q_E + Q_M = Q_Total for fixed-effects models
### note: this does not work for random/mixed-effects models, since Q_E and
### Q_Total are calculated under the assumption that tau^2 = 0, while the
### calculation of Q_M incorporates the estimate of tau^2
res <- rma(yi, vi, data=dat, method="FE")
res # this gives Q_Total
res <- rma(yi, vi, mods = ~ ablat + year, data=dat, method="FE")
res # this gives Q_E and Q_M
res$QE + res$QM

```

```

### decomposition of Q_E into subgroup Q-values
res <- rma(yi, vi, mods = ~ factor(alloc), data=dat)
res

res.a <- rma(yi, vi, data=dat, subset=(alloc=="alternate"))
res.r <- rma(yi, vi, data=dat, subset=(alloc=="random"))
res.s <- rma(yi, vi, data=dat, subset=(alloc=="systematic"))

res.a$QE # Q-value within subgroup "alternate"
res.r$QE # Q-value within subgroup "random"
res.s$QE # Q-value within subgroup "systematic"

res$QE
res.a$QE + res.r$QE + res.s$QE

#####

### an example of a location-scale model
dat <- dat.bangertdrowns2004

### fit a standard random-effects model
res <- rma(yi, vi, data=dat)
res

### fit the same model as a location-scale model
res <- rma(yi, vi, scale = ~ 1, data=dat)
res

### check that we obtain the same estimate for tau^2
predict(res, newscale=1, transf=exp)

### add the total sample size (per 100) as a location and scale predictor
dat$ni100 <- dat$ni/100
res <- rma(yi, vi, mods = ~ ni100, scale = ~ ni100, data=dat)
res

### variables in the location and scale parts can differ
res <- rma(yi, vi, mods = ~ ni100 + meta, scale = ~ ni100 + imag, data=dat)
res

```

Description

Function to obtain cluster-robust tests and confidence intervals (also known as robust variance estimation) of the model coefficients for objects of class "rma".

Usage

```
robust(x, cluster, ...)

## S3 method for class 'rma.uni'
robust(x, cluster, adjust=TRUE, clubSandwich=FALSE, digits, ...)
## S3 method for class 'rma.mv'
robust(x, cluster, adjust=TRUE, clubSandwich=FALSE, digits, ...)
```

Arguments

<code>x</code>	an object of class "rma.uni" or "rma.mv".
<code>cluster</code>	vector to specify the clustering variable to use for constructing the sandwich estimator of the variance-covariance matrix.
<code>adjust</code>	logical to specify whether a small-sample correction should be applied to the variance-covariance matrix.
<code>clubSandwich</code>	logical to specify whether the clubSandwich package should be used to obtain the cluster-robust tests and confidence intervals.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
<code>...</code>	other arguments.

Details

The function constructs a cluster-robust estimate of the variance-covariance matrix of the model coefficients based on a sandwich-type estimator and then computes tests and confidence intervals of the model coefficients. This function will often be part of a general workflow for meta-analyses involving complex dependency structures as described [here](#).

By default, tests of individual coefficients and confidence intervals are based on a t-distribution with $n - p$ degrees of freedom, while the omnibus test uses an F-distribution with m and $n - p$ degrees of freedom, where n is the number of clusters, p denotes the total number of model coefficients (including the intercept if it is present), and m denotes the number of coefficients tested by the omnibus test. This is sometimes called the ‘residual’ method for approximating the (denominator) degrees of freedom.

When `adjust=TRUE` (the default), the cluster-robust estimate of the variance-covariance matrix is multiplied by the factor $n/(n - p)$, which serves as a small-sample adjustment that tends to improve the performance of the method when the number of clusters is small. This is sometimes called the ‘CR1’ adjustment/estimator (in contrast to ‘CR0’ when `adjust=FALSE`).

For an even better small-sample adjustment, one can set `clubSandwich=TRUE` in which case the **clubSandwich** package is used to obtain the cluster-robust tests and confidence intervals. The variance-covariance matrix of the model coefficients is then estimated using the ‘bias-reduced linearization’ adjustment proposed by Bell and McCaffrey (2002) and further developed in Tipton (2015) and Pustejovsky and Tipton (2018). This is sometimes called the ‘CR2’ adjustment/estimator. The degrees of freedom of the t-tests are then estimated using a Satterthwaite approximation. F-tests are then based on an approximate Hotelling’s T-squared reference distribution, with denominator degrees of freedom estimated using a method by Zhang (2012, 2013), as further described in Tipton and Pustejovsky (2015).

Value

An object of class "robust.rma". The object is a list containing the following components:

beta	estimated coefficients of the model.
se	robust standard errors of the coefficients.
zval	test statistics of the coefficients.
pval	corresponding p-values.
ci.lb	lower bound of the confidence intervals for the coefficients.
ci.ub	upper bound of the confidence intervals for the coefficients.
vb	robust variance-covariance matrix of the estimated coefficients.
QM	test statistic of the omnibus test of moderators.
QMp	corresponding p-value.
...	some additional elements/values.

The results are formatted and printed with the `print.rma.uni` and `print.rma.mv` functions (depending on the type of model).

Predicted/fitted values based on "robust.rma" objects can be obtained with the `predict` function. Tests for sets of model coefficients or linear combinations thereof can be obtained with the `anova` function.

Note

The variable specified via `cluster` is assumed to be of the same length as the data originally passed to the `rma.uni` or `rma.mv` functions (and if the `data` argument was used in the original model fit, then the variable will be searched for within this data frame first). Any subsetting and removal of studies with missing values that was applied during the model fitting is also automatically applied to the variable specified via the `cluster` argument.

The idea of the robust (sandwich-type) estimator for models with unspecified heteroscedasticity can be traced back to Eicker (1967), Huber (1967), and White (1980, 1984). Hence, the method in general is often referred to as the Eicker-Huber-White method. Some small-sample improvements to the method are described by MacKinnon and White (1985). The extension to the cluster-robust estimator can be found in Froot (1989) and Williams (2000), which is also related to the GEE approach by Liang and Zeger (1986). Cameron and Miller (2015) provide an extensive overview of cluster-robust methods. Sidik and Jonkman (2005, 2006) introduced robust methods in the meta-analytic context for standard random/mixed-effects models. The use of cluster-robust methods for multivariate/multilevel meta-analytic models was introduced by Hedges, Tipton, and Johnson (2010).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Bell, R. M., & McCaffry, D. F. (2002). Bias reduction in standard errors for linear regression with multi-stage samples. *Survey Methodology*, **28**(2), 169–181. <https://www150.statcan.gc.ca/n1/en/catalogue/12-001>

- Cameron, A. C., & Miller, D. L. (2015). A practitioner's guide to cluster-robust inference. *Journal of Human Resources*, **50**(2), 317–372. <https://doi.org/10.3368/jhr.50.2.317>
- Eicker, F. (1967). Limit theorems for regressions with unequal and dependent errors. In L. M. LeCam & J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 59–82). Berkeley: University of California Press.
- Froot, K. A. (1989). Consistent covariance matrix estimation with cross-sectional dependence and heteroskedasticity in financial data. *Journal of Financial and Quantitative Analysis*, **24**(3), 333–355. <https://doi.org/10.2307/2330815>
- Hedges, L. V., Tipton, E., & Johnson, M. C. (2010). Robust variance estimation in meta-regression with dependent effect size estimates. *Research Synthesis Methods*, **1**(1), 39–65. <https://doi.org/10.1002/jrsm.5>
- Huber, P. (1967). The behavior of maximum-likelihood estimates under nonstandard conditions. In L. M. LeCam & J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 221–233). University of California Press.
- Liang, K. Y., & Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, **73**(1), 13–22. <https://doi.org/10.1093/biomet/73.1.13>
- MacKinnon, J. G., & White, H. (1985). Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics*, **29**(3), 305–325. [https://doi.org/10.1016/0304-4076\(85\)90158-7](https://doi.org/10.1016/0304-4076(85)90158-7)
- Pustejovsky, J. E., & Tipton, E. (2018). Small-sample methods for cluster-robust variance estimation and hypothesis testing in fixed effects models. *Journal of Business & Economic Statistics*, **36**(4), 672–683. <https://doi.org/10.1080/07350015.2016.1247004>
- Tipton, E. (2015). Small sample adjustments for robust variance estimation with meta-regression. *Psychological Methods*, **20**(3), 375–393. <https://doi.org/10.1037/met0000011>
- Tipton, E., & Pustejovsky, J. E. (2015). Small-sample adjustments for tests of moderators and model fit using robust variance estimation in meta-regression. *Journal of Educational and Behavioral Statistics*, **40**(6), 604–634. <https://doi.org/10.3102/1076998615606099>
- Sidik, K., & Jonkman, J. N. (2005). A note on variance estimation in random effects meta-regression. *Journal of Biopharmaceutical Statistics*, **15**(5), 823–838. <https://doi.org/10.1081/BIP-200067915>
- Sidik, K., & Jonkman, J. N. (2006). Robust variance estimation for random effects meta-analysis. *Computational Statistics & Data Analysis*, **50**(12), 3681–3701. <https://doi.org/10.1016/j.csda.2005.07.019>
- White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, **48**(4), 817–838. <https://doi.org/10.2307/1912934>
- White, H. (1984). *Asymptotic theory for econometricians*. Orlando, FL: Academic Press.
- Williams, R. L. (2000). A note on robust variance estimation for cluster-correlated data. *Biometrics*, **56**(2), 645–646. <https://doi.org/10.1111/j.0006-341x.2000.00645.x>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Zhang, J.-T. (2012). An approximate Hotelling T²-test for heteroscedastic one-way MANOVA. *Open Journal of Statistics*, **2**(1), 1–11. <https://doi.org/10.4236/ojs.2012.21001>
- Zhang, J.-T. (2013). Tests of linear hypotheses in the ANOVA under heteroscedasticity. *International Journal of Advanced Statistics and Probability*, **1**, 9–24. <https://doi.org/10.14419/ijasp.v1i2.908>

See Also

[rma.uni](#) and [rma.mv](#) for functions to fit models for which cluster-robust tests and confidence intervals can be obtained.

Examples

```
#####

### copy data from Bangert-Drowns et al. (2004) into 'dat'
dat <- dat.bangertdrowns2004

### fit random-effects model
res <- rma(yi, vi, data=dat)
res

### use cluster-robust inference methods
robust(res, cluster=id)

### use methods from the clubSandwich package
robust(res, cluster=id, clubSandwich=TRUE)

### fit meta-regression model
res <- rma(yi, vi, mods = ~ length, data=dat)
res

### use cluster-robust inference methods
robust(res, cluster=id)

### use methods from the clubSandwich package
robust(res, cluster=id, clubSandwich=TRUE)

#####

### copy data from Konstantopoulos (2011) into 'dat'
dat <- dat.konstantopoulos2011

### fit multilevel random-effects model
res <- rma.mv(yi, vi, random = ~ 1 | district/school, data=dat)
res

### use cluster-robust inference methods
robust(res, cluster=district)

### use methods from the clubSandwich package
robust(res, cluster=district, clubSandwich=TRUE)

#####

### copy data from Berkey et al. (1998) into 'dat'
dat <- dat.berkey1998

### variables v1i and v2i correspond to the 2x2 var-cov matrices of the studies;
```

```

### so use these variables to construct the V matrix (note: since v1i and v2i are
### var-cov matrices and not correlation matrices, set vi=1 for all rows)
V <- vcalc(vi=1, cluster=author, rvars=c(v1i, v2i), data=dat)

### fit multivariate model
res <- rma.mv(yi, V, mods = ~ 0 + outcome, random = ~ outcome | trial, struct="UN", data=dat)
res

### use cluster-robust inference methods
robust(res, cluster=trial)

### use methods from the clubSandwich package
robust(res, cluster=trial, clubSandwich=TRUE)

#####

```

se

Extract the Standard Errors from 'rma' Objects

Description

Function to extract the standard errors from objects of class "rma".

Usage

```

se(object, ...)
## Default S3 method:
se(object, ...)
## S3 method for class 'rma'
se(object, ...)

```

Arguments

object an object of class "rma".
 ... other arguments.

Value

A vector with the standard errors.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`rma.uni`, `rma.mh`, `rma.peto`, `rma.glmm`, and `rma.mv` for functions to fit models for which standard errors can be extracted.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)
res

### extract the model coefficients
coef(res)

### extract the standard errors
se(res)
```

selmodel	<i>Selection Models</i>
----------	-------------------------

Description

Function to fit selection models.

Usage

```
selmodel(x, ...)

## S3 method for class 'rma.uni'
selmodel(x, type, alternative="greater", prec, subset, delta,
         steps, decreasing=FALSE, verbose=FALSE, digits, control, ...)
```

Arguments

x	an object of class "rma.uni".
type	character string to specify the type of selection model. Possible options are "beta", "halfnorm", "negexp", "logistic", "power", "negexppow", "stepfun", "trunc", and "truncest". Can be abbreviated.
alternative	character string to specify the sidedness of the hypothesis when testing the effect size estimates. Possible options are "greater" (the default), "less", or "two.sided". Can be abbreviated.
prec	optional character string to specify the measure of precision (only relevant for selection models that can incorporate this into the selection function). Possible options are "sei", "vi", "ninv", or "sqrtninv".

subset	optional (logical or numeric) vector to specify the subset of studies to which the selection function applies.
delta	optional numeric vector (of the same length as the number of selection model parameters) to fix the corresponding δ value(s). A δ value can be fixed by setting the corresponding element of this argument to the desired value. A δ value will be estimated if the corresponding element is set equal to NA.
steps	numeric vector of one or more values that can or must be specified for certain selection functions.
decreasing	logical to specify whether the δ values in a step function selection model must be a monotonically decreasing function of the p-values (the default is FALSE). Only relevant when type="stepfun".
verbose	logical to specify whether output should be generated on the progress of the model fitting (the default is FALSE). Can also be an integer. Values > 1 generate more verbose output. See 'Note'.
digits	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
control	optional list of control values for the estimation algorithm. See 'Note'.
...	other arguments.

Details

Selection models are a general class of models that attempt to model the process by which the studies included in a meta-analysis may have been influenced by some form of publication bias. If a particular selection model is an adequate approximation for the underlying selection process, then the model provides estimates of the parameters of interest (e.g., the average true effect and the amount of heterogeneity in the true effects) that are 'corrected' for this selection process (i.e., they are estimates of the parameters in the population of studies before any selection has taken place). The present function fits a variety of such selection models. To do so, one should pass an object fitted with the `rma.uni` function to the first argument. The model that will then be fitted is of the same form as the original model combined with the specific selection model chosen (see below for possible options). For example, if the original model was a random-effects model, then a random-effects selection model will be fitted. Similarly, if the original model included moderators, then they will also be accounted for in the selection model fitted. Model fitting is done via maximum likelihood (ML) estimation over the fixed- and random-effects parameters (e.g., μ and τ^2 in a random-effects model) and the selection model parameters.

Argument type determines the specific type of selection model that should be fitted. Many selection models are based on the idea that selection may have taken place based on the p-values of the studies. In particular, let y_i and v_i denote the effect size estimate and the corresponding sampling variance of the i th study. Then $z_i = y_i/\sqrt{v_i}$ is the (Wald-type) test statistic for testing the null hypothesis $H_0: \theta_i = 0$ and $p_i = 1 - \Phi(z_i)$ (if alternative="greater"), $p_i = \Phi(z_i)$ (if alternative="less"), or $p_i = 2(1 - \Phi(|z_i|))$ (if alternative="two.sided") the corresponding (one- or two-sided) p-value, where $\Phi()$ denotes the cumulative distribution function of a standard normal distribution. Finally, let $w(p_i)$ denote some function that specifies the relative likelihood of selection given the p-value of a study.

If $w(p_i) > w(p_{i'})$ when $p_i < p_{i'}$ (i.e., $w(p_i)$ is larger for smaller p-values), then alternative="greater" implies selection in favor of increasingly significant positive effects, alternative="less" implies

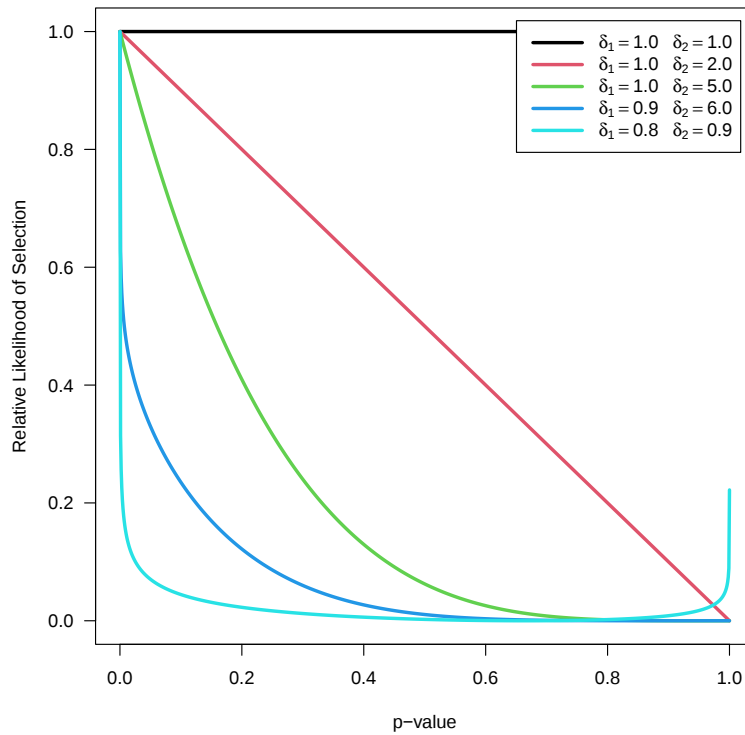
selection in favor of increasingly significant negative effects, and `alternative="two.sided"` implies selection in favor of increasingly significant effects regardless of their direction.

Beta Selection Model:

When `type="beta"`, the function can be used to fit the ‘beta selection model’ by Citkowicz and Vevea (2017). For this model, the selection function is given by

$$w(p_i) = p_i^{\delta_1-1} \times (1 - p_i)^{\delta_2-1}$$

where $\delta_1 > 0$ and $\delta_2 > 0$. The null hypothesis $H_0: \delta_1 = \delta_2 = 1$ represents the case where there is no selection according to the model. The figure below illustrates with some examples how the relative likelihood of selection can depend on the p-value for various combinations of δ_1 and δ_2 . Note that the model allows for a non-monotonic selection function.



As suggested by Pustejovsky (2024), the model can be modified by truncating p-values smaller or larger than certain thresholds. The modified selection function is then given by

$$w(p_i) = \tilde{p}_i^{\delta_1-1} \times (1 - \tilde{p}_i)^{\delta_2-1}$$

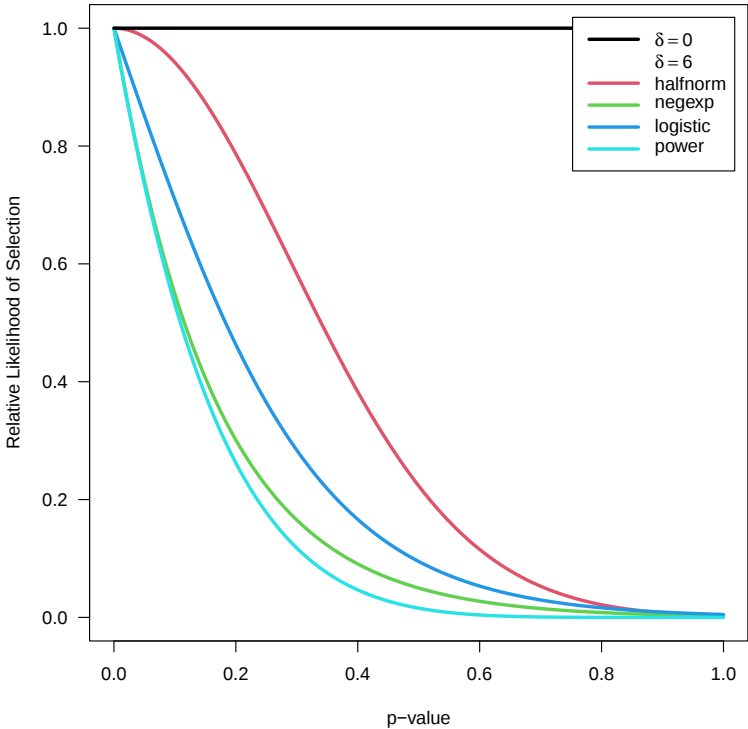
where $\tilde{p}_i = \min(\max(\alpha_1, p_i), \alpha_2)$. To fit such a selection model, one should specify the two α values (with $0 < \alpha < 1$) via the `steps` argument.

Half-Normal, Negative-Exponential, Logistic, and Power Selection Models:

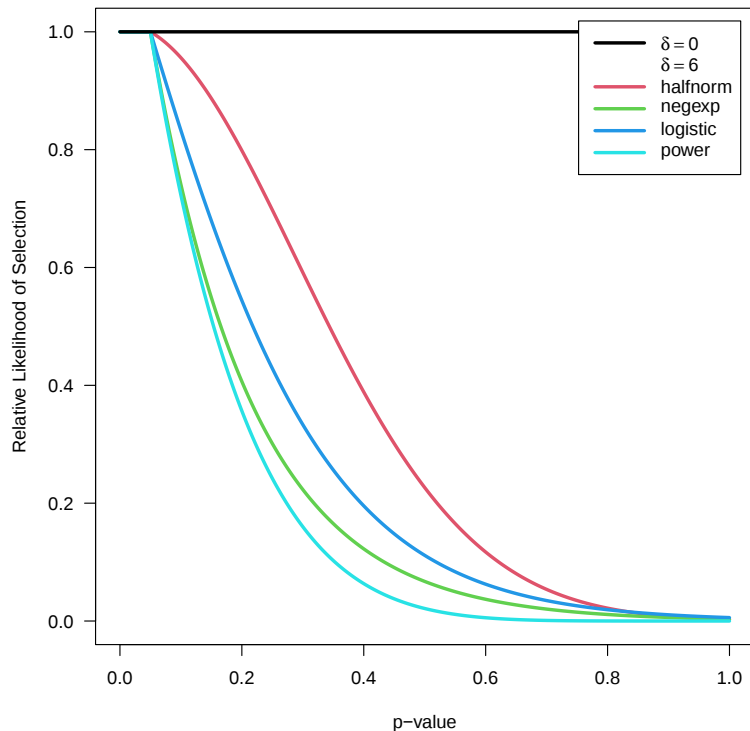
Preston et al. (2004) suggested the first three of the following selection functions:

name	type	selection function
half-normal	"halfnorm"	$w(p_i) = \exp(-\delta \times p_i^2)$
negative-exponential	"negexp"	$w(p_i) = \exp(-\delta \times p_i)$
logistic	"logistic"	$w(p_i) = 2 \times \exp(-\delta \times p_i)/(1 + \exp(-\delta \times p_i))$
power	"power"	$w(p_i) = (1 - p_i)^\delta$

The power selection model is added here as it has similar properties as the models suggested by Preston et al. (2004). For all models, assume $\delta \geq 0$, so that all functions imply a monotonically decreasing relationship between the p-value and the selection probability. For all functions, H_0 : $\delta = 0$ implies no selection. The figure below shows the relative likelihood of selection as a function of the p-value for $\delta = 0$ and for the various selection functions when $\delta = 6$.



Here, these functions are extended to allow for the possibility that $w(p_i) = 1$ for p-values below a certain significance threshold denoted by α (e.g., to model the case that the relative likelihood of selection is equally high for all significant studies but decreases monotonically for p-values above the significance threshold). To fit such a selection model, one should specify the α value (with $0 < \alpha < 1$) via the steps argument. There should be at least one observed p-value below and one observed p-value above the chosen threshold to fit these models. The selection functions are then given by $\min(1, w(p_i)/w(\alpha))$. The figure below shows some examples of the relative likelihood of selection when steps=0.05.



Preston et al. (2004) also suggested selection functions where the relative likelihood of selection not only depends on the p-value, but also the precision (e.g., standard error) of the estimate (if two studies have similar p-values, it may be plausible to assume that the larger / more precise study has a higher probability of selection). These selection functions (plus the corresponding power function) are given by:

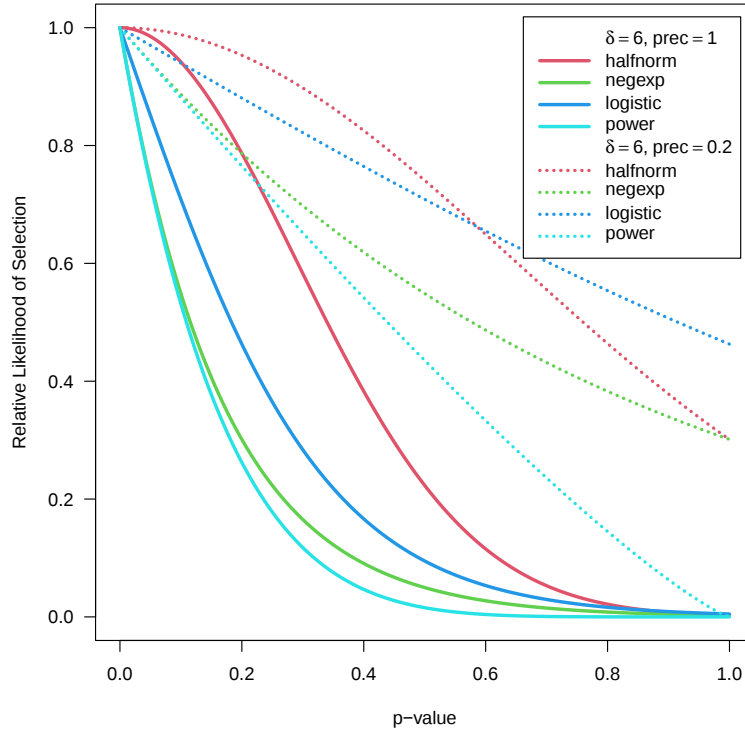
name	type	selection function
half-normal	"halfnorm"	$w(p_i) = \exp(-\delta \times \text{prec}_i \times p_i^2)$
negative-exponential	"negexp"	$w(p_i) = \exp(-\delta \times \text{prec}_i \times p_i)$
logistic	"logistic"	$w(p_i) = 2 \times \exp(-\delta \times \text{prec}_i \times p_i) / (1 + \exp(-\delta \times \text{prec}_i \times p_i))$
power	"power"	$w(p_i) = (1 - p_i)^{\delta \times \text{prec}_i}$

where $\text{prec}_i = \sqrt{v_i}$ (i.e., the standard error of the i th study) according to Preston et al. (2004). Here, this idea is generalized to allow the user to specify the specific measure of precision to use (via the `prec` argument). Possible options are:

- `prec="sei"` for the standard errors,
- `prec="vi"` for the sampling variances,
- `prec="ninv"` for the inverse of the sample sizes,
- `prec="sqrtninv"` for the inverse square root of the sample sizes.

Using some function of the sample sizes as a measure of precision is only possible when information about the sample sizes is actually stored within the object passed to the `selmodel` function. See 'Note'.

Note that prec_i is really a measure of imprecision (with higher values corresponding to lower precision). Also, regardless of the specific measure chosen, the values are actually rescaled with $\text{prec}_i = \text{prec}_i / \max(\text{prec}_i)$ inside of the function, such that $\text{prec}_i = 1$ for the least precise study and $\text{prec}_i < 1$ for the remaining studies (the rescaling does not actually change the fit of the model, it only helps to improve the stability of model fitting algorithm). The figure below shows some examples of the relative likelihood of selection using these selection functions for two different precision values (note that lower values of prec lead to a higher likelihood of selection).



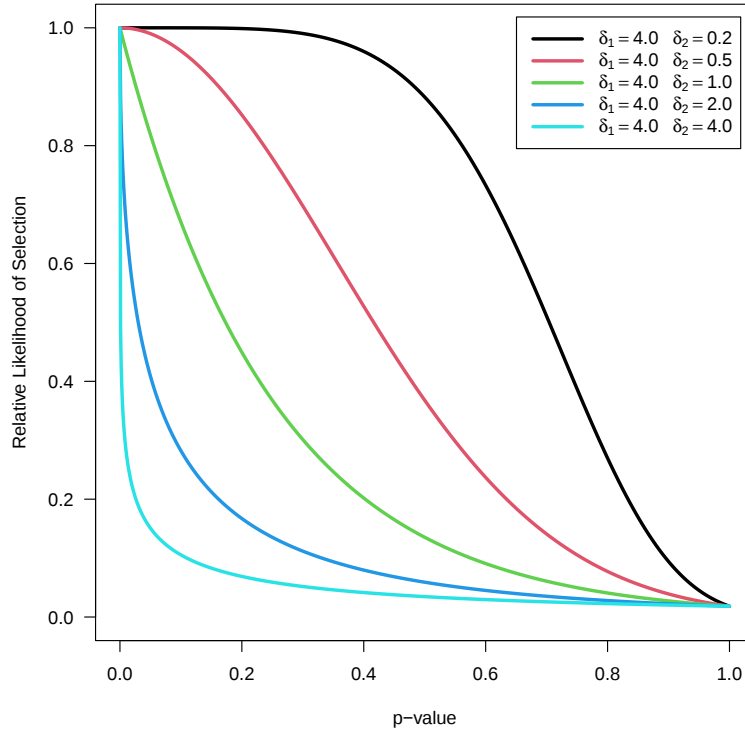
One can also use the `steps` argument as described above in combination with these selection functions (studies with p-values below the chosen threshold then have $w(p_i) = 1$ regardless of their exact p-value or precision).

Negative Exponential Power Selection Model:

As an extension of the half-normal and negative-exponential models, one can also choose `type="negexppow"` for a 'negative exponential power selection model'. The selection function for this model is given by

$$w(p_i) = \exp(-\delta_1 \times p_i^{1/\delta_2})$$

where $\delta_1 \geq 0$ and $\delta_2 \geq 0$ (see Begg & Mazumdar, 1994, although here a different parameterization is used, such that increasing δ_2 leads to more severe selection). The figure below shows some examples of this selection function when holding δ_1 constant while increasing δ_2 .



This model affords greater flexibility in the shape of the selection function, but requires the estimation of the additional power parameter (the half-normal and negative-exponential models are therefore special cases when fixing δ_2 to 0.5 or 1, respectively). $H_0: \delta_1 = 0$ again implies no selection, but so does $H_0: \delta_2 = 0$.

One can again use the steps argument to specify a single significance threshold, α , so that $w(p_i) = 1$ for p-values below this threshold and otherwise $w(p_i)$ follows the selection function as given above. One can also use the prec argument to specify a measure of precision in combination with this model, which leads to the selection function

$$w(p_i) = \exp(-\delta_1 \times \text{prec}_i \times p_i^{1/\delta_2})$$

and hence is the logical extension of the negative exponential power selection model that also incorporates some measure of precision into the selection process.

Step Function Selection Models:

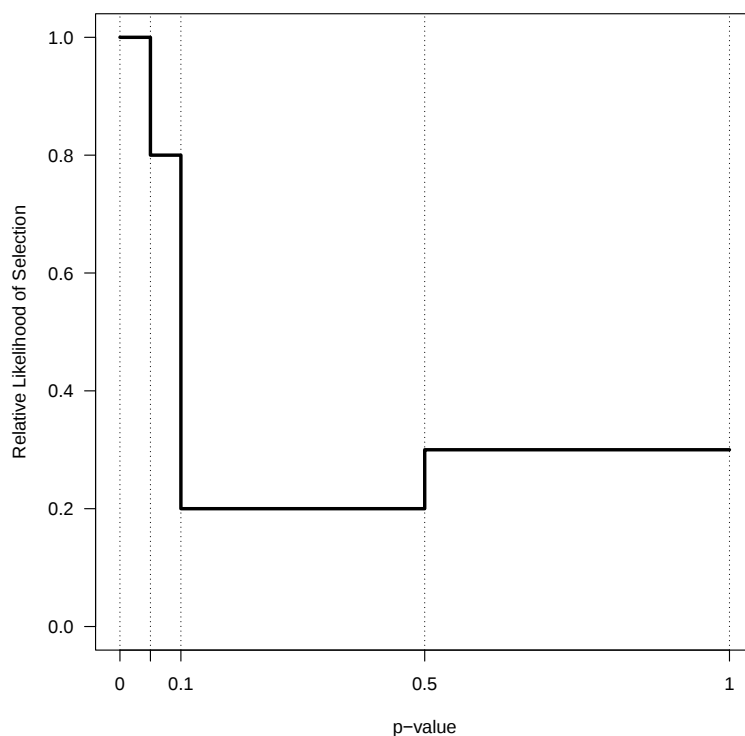
When type="stepfun", the function can be used to fit 'step function models' as described by Iyengar and Greenhouse (1988), Hedges (1992), Vevea and Hedges (1995), Vevea and Woods (2005), and others. For these models, one must specify one or multiple values via the steps argument, which define intervals in which the relative likelihood of selection is constant. Let

$$\alpha_1 < \alpha_2 < \dots < \alpha_c$$

denote these cutpoints sorted in increasing order, with the constraint that $\alpha_c = 1$ (if the highest value specified via steps is not 1, the function will automatically add this cutpoint), and define $\alpha_0 = 0$. The selection function is then given by $w(p_i) = \delta_j$ for $\alpha_{j-1} < p_i \leq \alpha_j$ where $\delta_j \geq 0$.

To make the model identifiable, we set $\delta_1 = 1$. The δ_j values therefore denote the likelihood of selection in the various intervals relative to the interval for p-values between 0 and α_1 . Hence, the null hypothesis $H_0: \delta_j = 1$ for $j = 1, \dots, c$ implies no selection.

For example, if `steps=c(0.05, 0.10, 0.50, 1)`, then δ_2 is the likelihood of selection for p-values between 0.05 and 0.10, δ_3 is the likelihood of selection for p-values between 0.10 and 0.50, and δ_4 is the likelihood of selection for p-values between 0.50 and 1 relative to the likelihood of selection for p-values between 0 and 0.05. The figure below shows the corresponding selection function for some arbitrarily chosen δ_j values.



There should be at least one observed p-value within each interval to fit this model. If there are no p-values between $\alpha_0 = 0$ and α_1 (i.e., within the first interval for which $\delta_1 = 1$), then estimates of $\delta_2, \dots, \delta_c$ will try to drift to infinity. If there are no p-values between α_{j-1} and α_j for $j = 2, \dots, c$, then δ_j will try to drift to zero. In either case, results should be treated with great caution. A common practice is then to collapse and/or adjust the intervals until all intervals contain at least one study. By setting `ptable=TRUE`, the function simply returns the p-value table and does not attempt any model fitting.

Note that when `alternative="greater"` or `alternative="less"` (i.e., when we assume that the relative likelihood of selection is not only related to the p-values of the studies, but also the directionality of the effects), then it would usually make sense to divide conventional levels of significance (e.g., 0.05) by 2 before passing these values to the `steps` argument. For example, if we think that studies were selected for positive effects that are significant at two-tailed $\alpha = 0.05$, then we should use `alternative="greater"` in combination with `steps=c(0.025, 1)`.

When specifying a single cutpoint in the context of a random-effects model (typically `steps=c(0.025, 1)` with either `alternative="greater"` or `alternative="less"`), this model is sometimes called the ‘three-parameter selection model’ (3PSM), corresponding to the parameters μ , τ^2 , and

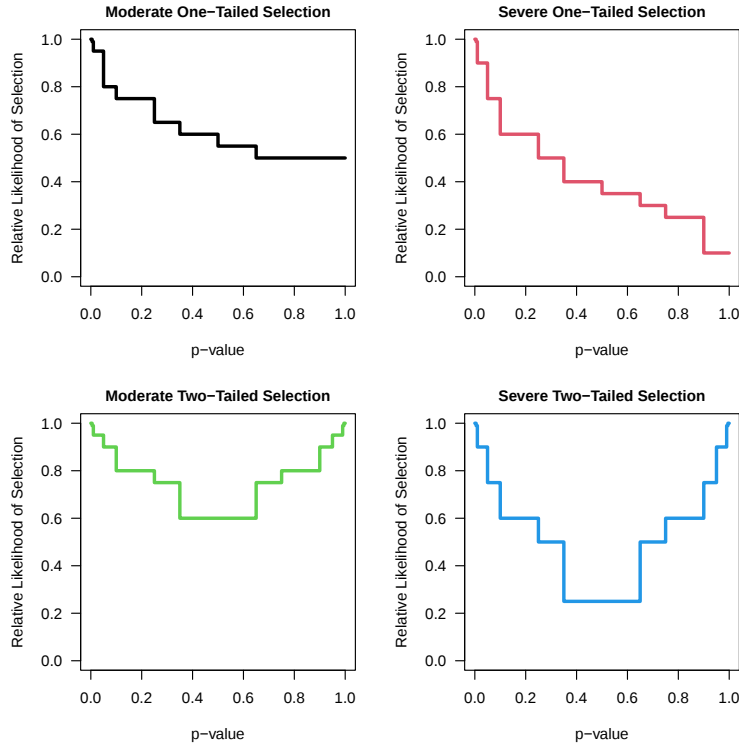
δ_2 (e.g., Carter et al., 2019; McShane et al., 2016; Pustejovsky & Rodgers, 2019). The same idea but in the context of an equal-effects model was also described by Iyengar and Greenhouse (1988).

Note that δ_j (for $j = 2, \dots, c$) can be larger than 1 (implying a greater likelihood of selection for p-values in the corresponding interval relative to the first interval). With `control=list(delta.max=1)`, one can enforce that the likelihood of selection for p-values above the first cutpoint can never be greater than the likelihood of selection for p-values below it. This constraint should be used with caution, as it may force δ_j estimates to fall on the boundary of the parameter space. Alternatively, one can set `decreasing=TRUE`, in which case the δ_j values must be a monotonically decreasing function of the p-values (which also forces $\delta_j \leq 1$). This feature should be considered experimental.

One of the challenges when fitting this model with many cutpoints is the large number of parameters that need to be estimated (which is especially problematic when the number of studies is small). An alternative approach suggested by Vevea and Woods (2005) is to fix the δ_j values to some a priori chosen values instead of estimating them. One can then conduct a sensitivity analysis by examining the results (e.g., the estimates of μ and τ^2 in a random-effects model) for a variety of different sets of δ_j values (reflecting more or less severe forms of selection). This can be done by specifying the δ_j values via the `delta` argument. Table 1 in Vevea and Woods (2005) provides some illustrative examples of moderate and severe selection functions for one- and two-tailed selection. The code below creates a data frame that contains these functions.

```
tab <- data.frame(
  steps = c(0.005, 0.01, 0.05, 0.10, 0.25, 0.35, 0.50, 0.65, 0.75, 0.90, 0.95, 0.99, 0.995, 1),
  delta.mod.1 = c(1, 0.99, 0.95, 0.80, 0.75, 0.65, 0.60, 0.55, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50),
  delta.sev.1 = c(1, 0.99, 0.90, 0.75, 0.60, 0.50, 0.40, 0.35, 0.30, 0.25, 0.10, 0.10, 0.10, 0.10),
  delta.mod.2 = c(1, 0.99, 0.95, 0.90, 0.80, 0.75, 0.60, 0.60, 0.75, 0.80, 0.90, 0.95, 0.99, 1.00),
  delta.sev.2 = c(1, 0.99, 0.90, 0.75, 0.60, 0.50, 0.25, 0.25, 0.50, 0.60, 0.75, 0.90, 0.99, 1.00))
```

The figure below shows the corresponding selection functions.



These four functions are “merely examples and should not be regarded as canonical” (Vevea & Woods, 2005).

Truncated Distribution Selection Model:

When `type="trunc"`, the model assumes that the relative likelihood of selection depends not on the p-value but on the value of the effect size estimate of a study. Let y_c denote a single cutpoint (which can be specified via argument `steps` and which is assumed to be 0 when unspecified). Let

$$w(y_i) = \begin{cases} 1 & \text{if } y_i > y_c \\ \delta_1 & \text{if } y_i \leq y_c \end{cases}$$

denote the selection function when `alternative="greater"` and

$$w(y_i) = \begin{cases} 1 & \text{if } y_i < y_c \\ \delta_1 & \text{if } y_i \geq y_c \end{cases}$$

when `alternative="less"` (note that `alternative="two.sided"` is not an option for this type of selection model). Therefore, when `alternative="greater"`, δ_1 denotes the likelihood of selection for effect size estimates that fall below the chosen cutpoint relative to those that fall above it (and vice-versa when `alternative="less"`). Hence, the null hypothesis $H_0: \delta_1 = 1$ implies no selection.

In principle, it is also possible to obtain a maximum likelihood estimate of the cutpoint. For this, one can set `type="truncest"`, in which case the selection function is given by

$$w(y_i) = \begin{cases} 1 & \text{if } y_i > \delta_2 \\ \delta_1 & \text{if } y_i \leq \delta_2 \end{cases}$$

when `alternative="greater"` and analogously when `alternative="less"`. Therefore, instead of specifying the cutpoint via the `steps` argument, it is estimated via δ_2 . Note that estimating both δ_1 and δ_2 simultaneously is typically very difficult (the likelihood surface is often quite rugged with multiple local optima) and will require a large number of studies. The implementation of this selection function should be considered experimental.

Models similar to those described above were proposed by Rust et al. (1990) and Formann (2008), but made various simplifying assumptions (e.g., Formann assumed $\delta_1 = 0$) and did not account for the heteroscedastic nature of the sampling variances of the effect size estimates, nor did they allow for heterogeneity in the true effects or the influence of moderators.

Subsets of Studies Affected and Unaffected by Publication Bias:

In some meta-analyses, some of the studies are known (or are assumed) to be free of publication bias (e.g., preregistered studies). In this case, the selection function should only apply to a subset of the studies (e.g., the non-registered studies). Using the `subset` argument, one can specify to which subset of studies the selection function should apply (note though that all studies are still included in the model fitting). The argument can either be a logical or a numeric vector, but note that what is specified is applied to the set of data originally passed to the `rma.uni` function, so a logical vector must be of the same length as the original dataset (and if the data argument was used in the original model fit, then any variables specified in creating a logical vector will be searched for within this data frame first). Any subsetting and removal of studies with missing values is automatically applied to the variables specified via these arguments.

Value

An object of class `c("rma.uni", "rma")`. The object is a list containing the same components as a regular `c("rma.uni", "rma")` object, but the parameter estimates are based on the selection model. Most importantly, the following elements are modified based on the selection model:

<code>beta</code>	estimated coefficients of the model.
<code>se</code>	standard errors of the coefficients.
<code>zval</code>	test statistics of the coefficients.
<code>pval</code>	corresponding p-values.
<code>ci.lb</code>	lower bound of the confidence intervals for the coefficients.
<code>ci.ub</code>	upper bound of the confidence intervals for the coefficients.
<code>vb</code>	variance-covariance matrix of the estimated coefficients.
<code>tau2</code>	estimated amount of (residual) heterogeneity. Always 0 when <code>method="EE"</code> .
<code>se.tau2</code>	standard error of the estimated amount of (residual) heterogeneity.

In addition, the object contains the following additional elements:

<code>delta</code>	estimated selection model parameter(s).
<code>se.delta</code>	corresponding standard error(s).
<code>zval.delta</code>	corresponding test statistic(s).
<code>pval.delta</code>	corresponding p-value(s).
<code>ci.lb.delta</code>	lower bound of the confidence intervals for the parameter(s).
<code>ci.ub.delta</code>	upper bound of the confidence intervals for the parameter(s).

LRT	test statistic of the likelihood ratio test for the selection model parameter(s).
LRTdf	degrees of freedom for the likelihood ratio test.
LRTp	p-value for the likelihood ratio test.
LRT.tau2	test statistic of the likelihood ratio test for testing $H_0: \tau^2 = 0$ (NA when fitting an equal-effects model).
LRTp.tau2	p-value for the likelihood ratio test.
ptable	frequency table for the observed p-values falling into the intervals defined by the steps argument (NA when steps is not specified).
...	some additional elements/values.

Methods

The results of the fitted model are formatted and printed with the `print` function. The estimated selection function can be drawn with `plot`.

The `profile` function can be used to obtain a plot of the log-likelihood as a function of τ^2 and/or the selection model parameter(s) of the model. Corresponding confidence intervals can be obtained with the `confint` function.

Note

Model fitting is done via numerical optimization over the model parameters. By default, `optim` with method "BFGS" is used for the optimization. One can also chose a different optimizer from `optim` via the control argument (e.g., `control=list(optimizer="Nelder-Mead")`). Besides one of the methods from `optim`, one can also choose the quasi-Newton algorithm in `nlminb`, one of the optimizers from the minqa package (i.e., `uobyqa`, `newuoa`, or `bobyqa`), one of the (derivative-free) algorithms from the `nloptr` package, the Newton-type algorithm implemented in `nlm`, the various algorithms implemented in the `dfoptim` package (`hjk` for the Hooke-Jeeves, `nmk` for the Nelder-Mead, and `mads` for the Mesh Adaptive Direct Searches algorithm), the quasi-Newton type optimizers `ucminf` and `lbfgsb3c` and the subspace-searching simplex algorithm `subplex` from the packages of the same name, the Barzilai-Borwein gradient decent method implemented in `BBoptim`, the `Rcgmin` and `Rvmmmin` optimizers, or the parallelized version of the L-BFGS-B algorithm implemented in `optimParallel` from the package of the same name.

The optimizer name must be given as a character string (i.e., in quotes). Additional control parameters can be specified via the control argument (e.g., `control=list(maxit=1000, reltol=1e-8)`). For `nloptr`, the default is to use the BOBYQA implementation from that package with a relative convergence criterion of $1e-8$ on the function value (i.e., log-likelihood), but this can be changed via the `algorithm` and `ftop_rel` arguments (e.g., `control=list(optimizer="nloptr", algorithm="NLOPT_LN_SBPLX", ftop_rel=1e-6)`). For `optimParallel`, the control argument `ncpus` can be used to specify the number of cores to use for the parallelization (e.g., `control=list(optimizer="optimParallel", ncpus=2)`).

All selection models (except for `type="stepfun"`, `type="trunc"`, and `type="truncst"`) require repeated evaluations of an integral, which is done via adaptive quadrature as implemented in the `integrate` function. One can adjust the arguments of the `integrate` function via control element `intCtrl`, which is a list of named arguments (e.g., `control = list(intCtrl = list(rel.tol=1e-4, subdivisions=100))`).

The starting values for the fixed effects, the τ^2 value (only relevant in random/mixed-effects selection models), and the δ parameter(s) are chosen automatically by the function, but one can also set the starting values manually via the `control` argument by specifying a vector of the appropriate length for `beta.init`, a single value for `tau2.init`, and a vector of the appropriate length for `delta.init`.

By default, the δ parameter(s) are constrained to a certain range, which improves the stability of the optimization algorithm. For all models, the maximum is set to 100 and the minimum to 0 (except for `type="beta"`, where the minimum for both parameters is $1e-5$, and when `type="stepfun"` with `decreasing=TRUE`, in which case the maximum is set to 1). These defaults can be changed via the `control` argument by specifying a scalar or a vector of the appropriate length for `delta.min` and/or `delta.max`. For example, `control=list(delta.max=Inf)` lifts the upper bound. Note that when a parameter estimate drifts close to its imposed bound, a warning will be issued.

A difficulty with fitting the beta selection model (i.e., `type="beta"`) is the behavior of $w(p_i)$ when $p_i = 0$ or $p_i = 1$. When $\delta_1 < 1$ or $\delta_2 < 1$, then this leads to selection weights equal to infinity, which causes problems when computing the likelihood function. Following Citkowitz and Vevea (2017), this problem can be avoided by censoring p-values too close to 0 or 1. The specific censoring point can be set via the `pval.min` element of the `control` argument. The default for this selection model is `control=list(pval.min=1e-5)`. A similar issue arises for the power selection model (i.e., `type="power"`) when $p_i = 1$. Again, `pval.min=1e-5` is used to circumvent this issue. For all other selection models, the default is `pval.min=0`.

The variance-covariance matrix corresponding to the estimates of the fixed effects, the τ^2 value (only relevant in random/mixed-effects selection models), and the δ parameter(s) is obtained by inverting the Hessian, which is numerically approximated using the `hessian` function from the `numDeriv` package. This may fail, leading to NA values for the standard errors and hence test statistics, p-values, and confidence interval bounds. One can set `control` argument `hessianCtrl` to a list of named arguments to be passed on to the `method.args` argument of the `hessian` function (the default is `control=list(hessianCtrl=list(r=6))`). One can also set `control=list(hesspack="pracma")` or `control=list(hesspack="calculus")` in which case the `pracma::hessian` or `calculus::hessian` functions from the respective packages are used instead for approximating the Hessian. When τ^2 is estimated to be smaller than either 10^{-4} or $\min(v_1, \dots, v_k)/10$ (where v_i denotes the sampling variances of the i th study), then τ^2 is effectively treated as zero for computing the standard errors (which helps to avoid numerical problems in approximating the Hessian). This cutoff can be adjusted via the `tau2tol` control argument (e.g., `control=list(tau2tol=0)` to switch off this behavior). Similarly, for `type="beta"` and `type="stepfun"`, δ estimates below 10^{-4} are treated as effectively zero for computing the standard errors. In this case, the corresponding standard errors are NA. This cutoff can be adjusted via the `deltatol` control argument (e.g., `control=list(deltatol=0)` to switch off this behavior).

Information on the progress of the optimization algorithm can be obtained by setting `verbose=TRUE` (this won't work when using parallelization). One can also set `verbose` to an integer (`verbose=2` yields even more information and `verbose=3` also show the progress visually by drawing the selection function as the optimization proceeds).

For selection functions where the `prec` argument is relevant, using a function of the sample sizes as the measure of precision (i.e., `prec="ninv"` or `prec="sqrtninv"`) is only possible when information about the sample sizes is actually stored within the object passed to the `selmodel` function. That should automatically be the case when the effect size estimates were computed with the `escalc` function or when the estimates were computed within the model fitting function. On the other hand, this will not be the case when `rma.uni` was used together with the `yi` and `vi` arguments and the `yi`

and v_i values were *not* computed with `escalc`. In that case, it is still possible to pass information about the sample sizes to the `rma.uni` function (e.g., use `rma.uni(yi, vi, ni=ni, data=dat)`, where data frame `dat` includes a variable called `ni` with the sample sizes).

Finally, the automatic rescaling of the chosen precision measure can be switched off by setting `scaleprec=FALSE`.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Begg, C. B., & Mazumdar, M. (1994). Operating characteristics of a rank correlation test for publication bias. *Biometrics*, **50**(4), 1088–1101. <https://doi.org/10.2307/2533446>
- Carter, E. C., Schönbrodt, F. D., Gervais, W. M., & Hilgard, J. (2019). Correcting for bias in psychology: A comparison of meta-analytic methods. *Advances in Methods and Practices in Psychological Science*, **2**(2), 115–144. <https://doi.org/10.1177/2515245919847196>
- Citkowicz, M., & Vevea, J. L. (2017). A parsimonious weight function for modeling publication bias. *Psychological Methods*, **22**(1), 28–41. <https://doi.org/10.1037/met0000119>
- Formann, A. K. (2008). Estimating the proportion of studies missing for meta-analysis due to publication bias. *Contemporary Clinical Trials*, **29**(5), 732–739. <https://doi.org/10.1016/j.cct.2008.05.004>
- Hedges, L. V. (1992). Modeling publication selection effects in meta-analysis. *Statistical Science*, **7**(2), 246–255. <https://doi.org/10.1214/ss/1177011364>
- Iyengar, S., & Greenhouse, J. B. (1988). Selection models and the file drawer problem. *Statistical Science*, **3**(1), 109–117. <https://doi.org/10.1214/ss/1177013012>
- McShane, B. B., Bockenholt, U., & Hansen, K. T. (2016). Adjusting for publication bias in meta-analysis: An evaluation of selection methods and some cautionary notes. *Perspectives on Psychological Science*, **11**(5), 730–749. <https://doi.org/10.1177/1745691616662243>
- Preston, C., Ashby, D., & Smyth, R. (2004). Adjusting for publication bias: Modelling the selection process. *Journal of Evaluation in Clinical Practice*, **10**(2), 313–322. <https://doi.org/10.1111/j.1365-2753.2003.0045>
- Pustejovsky, J. E., & Rodgers, M. A. (2019). Testing for funnel plot asymmetry of standardized mean differences. *Research Synthesis Methods*, **10**(1), 57–71. <https://doi.org/10.1002/jrsm.1332>
- Pustejovsky, J. E. (2024). Beta-density selection models for meta-analysis. <https://jepusto.com/posts/beta-density-s>
- Rust, R. T., Lehmann, D. R., & Farley, J. U. (1990). Estimating publication bias in meta-analysis. *Journal of Marketing Research*, **27**(2), 220–226. <https://doi.org/10.1177/002224379002700209>
- Vevea, J. L., & Hedges, L. V. (1995). A general linear model for estimating effect size in the presence of publication bias. *Psychometrika*, **60**(3), 419–435. <https://doi.org/10.1007/BF02294384>
- Vevea, J. L., & Woods, C. M. (2005). Publication bias in research synthesis: Sensitivity analysis using a priori weight functions. *Psychological Methods*, **10**(4), 428–443. <https://doi.org/10.1037/1082-989X.10.4.428>

See Also

`rma.uni` for the function to fit models which can be extended with selection models.

Examples

```
#####

### example from Citkowicz and Vevea (2017) for beta selection model

# copy data into 'dat' and examine data
dat <- dat.baskerville2012
dat

# fit random-effects model
res <- rma(smd, se^2, data=dat, method="ML", digits=3)
res

# funnel plot
funnel(res, ylim=c(0,0.6), xlab="Standardized Mean Difference")

# fit beta selection model
## Not run:
sel <- selmodel(res, type="beta")
sel

# plot the selection function
plot(sel, ylim=c(0,40))

# only apply the selection function to studies with a quality score below 10
sel <- selmodel(res, type="beta", subset=score<10)
sel

# use a truncated beta selection function (need to switch optimizers to get convergence)
sel <- selmodel(res, type="beta", steps=c(0.025,0.975), control=list(optimizer="nlminb"))
sel

## End(Not run)

# fit mixed-effects meta-regression model with 'blind' dummy variable as moderator
res <- rma(smd, se^2, data=dat, mods = ~ blind, method="ML", digits=3)
res

# predicted average effect for studies that do not and that do use blinding
predict(res, newmods=c(0,1))

# fit beta selection model
## Not run:
sel <- selmodel(res, type="beta")
sel
predict(sel, newmods=c(0,1))

## End(Not run)

#####

### example from Preston et al. (2004)
```

```

# copy data into 'dat' and examine data
dat <- dat.hahn2001
dat

### meta-analysis of (log) odds ratios using the Mantel-Haenszel method
res <- rma.mh(measure="OR", ai=ai, nli=nli, ci=ci, n2i=n2i, data=dat, digits=2, slab=study)
res

# calculate log odds ratios and corresponding sampling variances
dat <- escalc(measure="OR", ai=ai, nli=nli, ci=ci, n2i=n2i, data=dat, drop00=TRUE)
dat

# fit equal-effects model
res <- rma(yi, vi, data=dat, method="EE")

# predicted odds ratio (with 95% CI)
predict(res, transf=exp, digits=2)

# funnel plot
funnel(res, atransf=exp, at=log(c(0.01,0.1,1,10,100)), ylim=c(0,2))

# fit half-normal, negative-exponential, logistic, and power selection models
## Not run:
sel1 <- selmodel(res, type="halfnorm", alternative="less")
sel2 <- selmodel(res, type="negexp", alternative="less")
sel3 <- selmodel(res, type="logistic", alternative="less")
sel4 <- selmodel(res, type="power", alternative="less")

# plot the selection functions
plot(sel1)
plot(sel2, add=TRUE, col="blue")
plot(sel3, add=TRUE, col="red")
plot(sel4, add=TRUE, col="green")

# add legend
legend("topright", inset=0.02, lty="solid", lwd=2, col=c("black","blue","red","green"),
      legend=c("Half-normal", "Negative-exponential", "Logistic", "Power"))

# show estimates of delta (and corresponding SEs)
tab <- data.frame(delta = c(sel1$delta, sel2$delta, sel3$delta, sel4$delta),
                  se = c(sel1$se.delta, sel2$se.delta, sel3$se.delta, sel4$se.delta))
rownames(tab) <- c("Half-normal", "Negative-exponential", "Logistic", "Power")
round(tab, 2)

# predicted odds ratios (with 95% CI)
predict(res, transf=exp, digits=2)
predict(sel1, transf=exp, digits=2)
predict(sel2, transf=exp, digits=2)
predict(sel3, transf=exp, digits=2)
predict(sel4, transf=exp, digits=2)

## End(Not run)

```

```

# fit selection models including standard error as precision measure (note: using
# scaleprec=FALSE here since Preston et al. (2004) did not use the rescaling)
## Not run:
sel1 <- selmodel(res, type="halfnorm", prec="sei", alternative="less", scaleprec=FALSE)
sel2 <- selmodel(res, type="negexp", prec="sei", alternative="less", scaleprec=FALSE)
sel3 <- selmodel(res, type="logistic", prec="sei", alternative="less", scaleprec=FALSE)
sel4 <- selmodel(res, type="power", prec="sei", alternative="less", scaleprec=FALSE)

# show estimates of delta (and corresponding SEs)
tab <- data.frame(delta = c(sel1$delta, sel2$delta, sel3$delta, sel4$delta),
                  se = c(sel1$se.delta, sel2$se.delta, sel3$se.delta, sel4$se.delta))
rownames(tab) <- c("Half-normal", "Negative-exponential", "Logistic", "Power")
round(tab, 2)

# predicted odds ratio (with 95% CI)
predict(res, transf=exp, digits=2)
predict(sel1, transf=exp, digits=2)
predict(sel2, transf=exp, digits=2)
predict(sel3, transf=exp, digits=2)
predict(sel4, transf=exp, digits=2)

## End(Not run)

#####

### meta-analysis on the effect of environmental tobacco smoke on lung cancer risk

# copy data into 'dat' and examine data
dat <- dat.hackshaw1998
dat

# fit random-effects model
res <- rma(yi, vi, data=dat, method="ML")
res

# funnel plot
funnel(res, atransf=exp, at=log(c(0.25,0.5,1,2,4,8)), ylim=c(0,0.8))

# step function selection model
## Not run:
sel <- selmodel(res, type="stepfun", alternative="greater", steps=c(0.025,0.10,0.50,1))
sel

# plot the selection function
plot(sel)

# truncated distribution selection model (with steps=0 by default)
sel <- selmodel(res, type="trunc")
sel

## End(Not run)

```

```
#####

### meta-analysis on the effect of the color red on attractiveness ratings

# copy data into 'dat', select only results for male raters, and examine data
dat <- dat.lehmann2018
dat <- dat[dat$Gender == "Males",]
dat[c(1,6,48:49)]

# fit random-effects model
res <- rma(yi, vi, data=dat, method="ML")
res

# step function selection model (3PSM)
## Not run:
sel <- selmodel(res, type="stepfun", alternative="greater", steps=0.025)
sel

# step function selection model that only applies to the non-preregistered studies
sel <- selmodel(res, type="stepfun", alternative="greater", steps=0.025,
                 subset=Preregistered=="Not Pre-Registered")
sel

## End(Not run)

#####

### validity of student ratings example from Vevea & Woods (2005)

# copy data into 'dat' and examine data
dat <- dat.cohen1981
dat[c(1,4,5)]

# calculate r-to-z transformed correlations and corresponding sampling variances
dat <- escalc(measure="ZCOR", ri=ri, ni=ni, data=dat[c(1,4,5)])
dat

# fit random-effects model
res <- rma(yi, vi, data=dat, method="ML", digits=3)
res

# predicted average correlation (with 95% CI)
predict(res, transf=transf.ztor)

# funnel plot
funnel(res, ylim=c(0,0.4))

# selection functions from Vevea & Woods (2005)
tab <- data.frame(
  steps = c(0.005, 0.01, 0.05, 0.10, 0.25, 0.35, 0.50, 0.65, 0.75, 0.90, 0.95, 0.99, 0.995, 1),
  delta.mod.1 = c(1, 0.99, 0.95, 0.80, 0.75, 0.65, 0.60, 0.55, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50),
  delta.sev.1 = c(1, 0.99, 0.90, 0.75, 0.60, 0.50, 0.40, 0.35, 0.30, 0.25, 0.10, 0.10, 0.10, 0.10),
  delta.mod.2 = c(1, 0.99, 0.95, 0.90, 0.80, 0.75, 0.60, 0.60, 0.75, 0.80, 0.90, 0.95, 0.99, 1.00),
```

```

delta.sev.2 = c(1, 0.99, 0.90, 0.75, 0.60, 0.50, 0.25, 0.25, 0.50, 0.60, 0.75, 0.90, 0.99, 1.00))

# apply step function model with a priori chosen selection weights
## Not run:
sel <- lapply(tab[-1], function(delta) selmodel(res, type="stepfun", steps=tab$steps, delta=delta))

# estimates (transformed correlation) and tau^2 values
sav <- data.frame(estimate = round(c(res$beta, sapply(sel, function(x) x$beta)), 2),
                  varcomp = round(c(res$tau2, sapply(sel, function(x) x$tau2)), 3))
sav

## End(Not run)

#####

```

simulate.rma

Simulate Method for 'rma' Objects

Description

Function to simulate estimates based on "rma" model objects.

Usage

```

## S3 method for class 'rma'
simulate(object, nsim=1, seed=NULL, olim, ...)

```

Arguments

object	an object of class "rma".
nsim	number of response vectors to simulate (defaults to 1).
seed	an object to specify if and how the random number generator should be initialized ('seeded'). Either NULL or an integer that will be used in a call to <code>set.seed</code> before simulating the response vectors. If set, the value is saved as the "seed" attribute of the returned value. The default, NULL will not change the random generator state, and return <code>.Random.seed</code> as the "seed" attribute; see 'Value'.
olim	argument to specify observation limits for the simulated values. If unspecified, no limits are used.
...	other arguments.

Details

The model specified via `object` must be a model fitted with either the `rma.uni` or `rma.mv` functions.

Value

A data frame with `nsim` columns with the simulated estimates.

The data frame comes with an attribute "seed". If argument `seed` is `NULL`, the attribute is the value of `.Random.seed` before the simulation was started; otherwise it is the value of the `seed` argument with a "kind" attribute with value `as.list(RNGkind())`.

Note

If the estimates are bounded (e.g., correlations are bounded between -1 and +1, proportions are bounded between 0 and 1), one can use the `olim` argument to enforce those limits when simulating the values (i.e., the simulated values cannot exceed those limits then).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#) and [rma.mv](#) for functions to fit models for which simulated estimates can be generated.

Examples

```
### copy BCG vaccine data into 'dat'
dat <- dat.bcg

### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat)
dat

### fit random-effects model
res <- rma(yi, vi, data=dat)
res

### simulate 5 sets of new estimates based on the fitted model
newdat <- simulate(res, nsim=5, seed=1234)
newdat
```

tes	<i>Test of Excess Significance</i>
-----	------------------------------------

Description

Function to conduct the test of excess significance.

Usage

```
tes(x, vi, sei, subset, data, H0=0, alternative="two.sided", alpha=0.05, theta, tau2,
    test, tes.alternative="greater", progbar=TRUE, tes.alpha=0.10, digits, ...)

## S3 method for class 'tes'
print(x, digits=x$digits, ...)
```

Arguments

These arguments pertain to data input:

x	a vector with the effect size estimates or an object of class "rma".
vi	vector with the corresponding sampling variances. Ignored if x is an object of class "rma".
sei	vector with the corresponding standard errors (note: only one of the two, vi or sei, needs to be specified).
subset	optional (logical or numeric) vector to specify the subset of studies that should be included. Ignored if x is an object of class "rma".
data	optional data frame containing the variables given to the arguments above.

These arguments pertain to the tests of the effect size estimates:

H0	numeric value to specify the value under the null hypothesis (the default is 0).
alternative	character string to specify the sidedness of the hypothesis when testing the effect size estimates. Possible options are "two.sided" (the default), "greater", or "less". Can be abbreviated.
alpha	alpha level for testing the effect size estimates (the default is 0.05).

These arguments pertain to the power of the tests:

theta	optional numeric value to specify the value of the (average) true effect under the alternative hypothesis. If unspecified, it will be estimated based on the data or the value is taken from the "rma" object.
tau2	optional numeric value to specify the amount of heterogeneity in the true effects. If unspecified, the true effects are assumed to be homogeneous or the value is taken from the "rma" object.

These arguments pertain to the test of excess significance:

<code>test</code>	optional character string to specify the type of test to use for conducting the test of excess significance. Possible options are "chi2", "binom", or "exact". Can be abbreviated. If unspecified, the function chooses the type of test based on the data.
<code>tes.alternative</code>	character string to specify the sidedness of the hypothesis for the test of excess significance. Possible options are "greater" (the default), "two.sided", or "less". Can be abbreviated.
<code>progbar</code>	logical to specify whether a progress bar should be shown (the default is TRUE). Only relevant when conducting an exact test.
<code>tes.alpha</code>	alpha level for the test of excess significance (the default is 0.10). Only relevant for finding the 'limit estimate'.

Miscellaneous arguments:

<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded.
<code>...</code>	other arguments.

Details

The function carries out the test of excess significance described by Ioannidis and Trikalinos (2007). The test can be used to examine whether the observed number of significant findings is greater than the number of significant findings expected given the power of the tests. An overabundance of significant tests may suggest that the collection of studies is not representative of all studies conducted on a particular topic.

Via argument `x`, one can either pass a vector with the effect size estimates to the function (and then the corresponding sampling variances via `vi` or the standard errors via `sei`) or an object of class "rma".

The effect size estimates are tested for significance based on a standard Wald-type test, that is, by comparing

$$z_i = \frac{y_i - H_0}{\sqrt{v_i}}$$

against the appropriate critical value(s) of a standard normal distribution (e.g., ± 1.96 for `alternative="two.sided"` and `alpha=0.05`, which are the defaults). Let O denote the observed number of significant tests.

Given a particular value for the (average) true effect denoted by θ (which, if it is unspecified, is determined by computing the inverse-variance weighted average of the effect size estimates or the value is taken from the model object), let $1 - \beta_i$ denote the power of the i th test (where β_i denotes the Type II error probability). If $\tau^2 > 0$, let $1 - \beta_i$ denote the expected power (computed based on integrating the power over a normal distribution with mean θ and variance τ^2). Let $E = \sum_{i=1}^k (1 - \beta_i)$ denote the expected number of significant tests.

The test of excess significance then tests if O is significantly greater (if `tes.alternative="greater"`) than E . This can be done using Pearson's chi-square test (if `test="chi2"`), a binomial test (if `test="binomial"`), or an exact test (if `test="exact"`). The latter is described in Francis (2013). If argument `test` is unspecified, the default is to do an exact test if the number of elements in the sum that needs to be computed is less than or equal to 10^6 and to do a chi-square test otherwise.

One can also iteratively find the value of θ such that the p-value of the test of excess significance is equal to `tes.alpha` (which is 0.10 by default). The resulting value is called the ‘limit estimate’ and is denoted θ_{lim} by Ioannidis and Trikalinos (2007). Note that the limit estimate is not computable if the p-value is larger than `tes.alpha` even if $\theta = H_0$.

Value

An object of class "tes". The object is a list containing the following components:

<code>k</code>	the number of studies included in the analysis.
<code>O</code>	the observed number of significant tests.
<code>E</code>	the expected number of significant tests.
<code>OEratio</code>	the ratio of <code>O</code> over <code>E</code> .
<code>test</code>	the type of test conducted.
<code>pval</code>	the p-value of the test of excess significance.
<code>power</code>	the (estimated) power of the tests.
<code>sig</code>	logical vector indicating which tests were significant.
<code>theta</code>	the value of θ used for computing the power of the tests.
<code>theta.lim</code>	the ‘limit estimate’ (i.e., θ_{lim}).
<code>...</code>	some additional elements/values.

The results are formatted and printed with the `print` function.

Note

When `tes.alternative="greater"` (the default), then the function tests if O is significantly greater than E and hence this is indeed a test of excess significance. When `tes.alternative="two.sided"`, then the function tests if O differs significantly from E in either direction and hence it would be more apt to describe this as a test of (in)consistency (between O and E). Finally, one can also set `tes.alternative="less"`, in which case the function tests if O is significantly lower than E , which could be considered a test of excess non-significance.

When `tes.alternative="two.sided"`, one can actually compute two limit estimates. The function attempts to compute both.

The function computes the significance and power of the studies based on Wald-type tests regardless of the effect size measure used as input. This works as an adequate approximation as long as the within-study sample sizes are not too small.

Note that the test is not a test for publication bias but a test whether the set of studies includes an unusual number of significant findings given the power of the studies. The general usefulness of the test and its usefulness under particular circumstances (e.g., when there is substantial heterogeneity in the true effects) has been the subject of considerable debate. See Francis (2013) and the commentaries on this article in the same issue of the journal.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Francis, G. (2013). Replication, statistical consistency, and publication bias. *Journal of Mathematical Psychology*, **57**(5), 153–169. <https://doi.org/10.1016/j.jmp.2013.02.003>
- Ioannidis, J. P. A., & Trikalinos, T. A. (2007). An exploratory test for an excess of significant findings. *Clinical Trials*, **4**(3), 245–253. <https://doi.org/10.1177/1740774507079441>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[regtest](#) for the regression test, [ranktest](#) for the rank correlation test, [trimfill](#) for the trim and fill method, [fsn](#) to compute the fail-safe N (file drawer analysis), and [selmodel](#) for selection models.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=x.a, ni=n.a, ci=x.p, ni=n.p, data=dat.dorn2007)

### conduct test of excess significance (using test="chi2" to speed things up)
tes(yi, vi, data=dat, test="chi2")

### same as fitting an EE model and then passing the object to the function
res <- rma(yi, vi, data=dat, method="EE")
tes(res, test="chi2")

### illustrate limit estimate (value of theta where p-value of test is equal to tes.alpha)
thetas <- seq(0,1,length=101)
pvals <- sapply(thetas, function(theta) tes(yi, vi, data=dat, test="chi2", theta=theta)$pval)
plot(thetas, pvals, type="o", pch=19, ylim=c(0,1))
sav <- tes(yi, vi, data=dat, test="chi2")
abline(h=sav$tes.alpha, lty="dotted")
abline(v=sav$theta.lim, lty="dotted")

### examine significance of test as a function of alpha (to examine 'significance chasing')
alphas <- seq(0.01,0.99,length=101)
pvals <- sapply(alphas, function(alpha) tes(yi, vi, data=dat, test="chi2", alpha=alpha)$pval)
plot(alphas, pvals, type="o", pch=19, ylim=c(0,1))
abline(v=0.05, lty="dotted")
abline(h=0.10, lty="dotted")
```

to.long

Convert Data from Vector to Long Format

Description

Function to convert summary data in vector format to the corresponding long format.

Usage

```
to.long(measure, ai, bi, ci, di, n1i, n2i, x1i, x2i, t1i, t2i,
        m1i, m2i, sd1i, sd2i, xi, mi, ri, ti, sdi, ni, data, slab, subset,
        add=1/2, to="none", drop00=FALSE, vlong=FALSE, append=TRUE, var.names)
```

Arguments

measure	a character string to specify the effect size measure corresponding to the summary data supplied. See ‘Details’ and the documentation of the escalc function for possible options.
ai	vector with the 2×2 table frequencies (upper left cell).
bi	vector with the 2×2 table frequencies (upper right cell).
ci	vector with the 2×2 table frequencies (lower left cell).
di	vector with the 2×2 table frequencies (lower right cell).
n1i	vector with the group sizes or row totals (first group/row).
n2i	vector with the group sizes or row totals (second group/row).
x1i	vector with the number of events (first group).
x2i	vector with the number of events (second group).
t1i	vector with the total person-times (first group).
t2i	vector with the total person-times (second group).
m1i	vector with the means (first group or time point).
m2i	vector with the means (second group or time point).
sd1i	vector with the standard deviations (first group or time point).
sd2i	vector with the standard deviations (second group or time point).
xi	vector with the frequencies of the event of interest.
mi	vector with the frequencies of the complement of the event of interest or the group means.
ri	vector with the raw correlation coefficients.
ti	vector with the total person-times.
sdi	vector with the standard deviations.
ni	vector with the sample/group sizes.
data	optional data frame containing the variables given to the arguments above.
slab	optional vector with labels for the studies.
subset	optional (logical or numeric) vector to specify the subset of studies that should be included in the data frame returned by the function.
add	see the documentation of the escalc function.
to	see the documentation of the escalc function.
drop00	see the documentation of the escalc function.
vlong	optional logical whether a very long format should be used (only relevant for 2×2 or 1×2 table data).

append	logical to specify whether the data frame specified via the data argument (if one has been specified) should be returned together with the long format data (the default is TRUE). Can also be a character or numeric vector to specify which variables from data to append.
var.names	optional character vector with variable names (the length depends on the data type). If unspecified, the function sets appropriate variable names by default.

Details

The `escalc` function describes a wide variety of effect size measures that can be computed for a meta-analysis. The summary data used to compute those measures are typically contained in vectors, each element corresponding to a study. The `to.long` function takes this information and constructs a long format dataset from these data.

For example, in various fields (such as the health and medical sciences), the response variable measured is often dichotomous (binary), so that the data from a study comparing two different groups can be expressed in terms of a 2×2 table, such as:

	outcome 1	outcome 2	total
group 1	ai	bi	n1i
group 2	ci	di	n2i

where ai, bi, ci, and di denote the cell frequencies (i.e., the number of individuals falling into a particular category) and n1i and n2i the row totals (i.e., the group sizes).

The cell frequencies in k such 2×2 tables can be specified via the ai, bi, ci, and di arguments (or alternatively, via the ai, ci, n1i, and n2i arguments). The function then creates the corresponding long format dataset. The measure argument should then be set equal to one of the effect size measures that can be computed based on this type of data, such as "RR", "OR", "RD" (it is not relevant which specific measure is chosen, as long as it corresponds to the specified summary data). See the documentation of the `escalc` function for more details on the types of data formats available.

The long format for data of this type consists of two rows per study, a factor indicating the study (default name `study`), a dummy variable indicating the group (default name `group`, coded as 1 and 2), and two variables indicating the number of individuals experiencing outcome 1 or outcome 2 (default names `out1` and `out2`). Alternatively, if `vlong=TRUE`, then the long format consists of four rows per study, a factor indicating the study (default name `study`), a dummy variable indicating the group (default name `group`, coded as 1 and 2), a dummy variable indicating the outcome (default name `outcome`, coded as 1 and 2), and a variable indicating the frequency of the respective outcome (default name `freq`).

The default variable names can be changed via the `var.names` argument (must be of the appropriate length, depending on the data type).

The examples below illustrate the use of this function.

Value

A data frame with either k , $2 \times k$, or $4 \times k$ rows and an appropriate number of columns (depending on the data type) with the data in long format. If `append=TRUE` and a data frame was specified via the `data` argument, then the data in long format are appended to the original data frame (with rows repeated an appropriate number of times).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[escalc](#) for a function to compute effect size estimates (and corresponding sampling variances) based on similar inputs.

[to.table](#) for a function to turn similar inputs into tabular form.

Examples

```
### convert data to long format
dat.bcg
dat.long <- to.long(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
dat.long

### extra long format
dat <- to.long(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, vlong=TRUE)
dat

### select variables to append
dat.long <- to.long(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
                   data=dat.bcg, append=c("author", "year"))
dat.long
dat.long <- to.long(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
                   data=dat.bcg, append=2:3)
dat.long

### convert data to long format
dat.long <- to.long(measure="IRR", x1i=x1i, x2i=x2i, t1i=t1i, t2i=t2i,
                   data=dat.hart1999, var.names=c("id", "group", "events", "ptime"))
dat.long

### convert data to long format
dat.long <- to.long(measure="MD", m1i=m1i, sd1i=sd1i, n1i=n1i,
                   m2i=m2i, sd2i=sd2i, n2i=n2i, data=dat.normand1999,
                   var.names=c("id", "group", "mean", "sd", "n"))
dat.long
```

Description

Function to convert summary data in vector format to the corresponding table format.

Usage

```
to.table(measure, ai, bi, ci, di, n1i, n2i, x1i, x2i, t1i, t2i,
         m1i, m2i, sd1i, sd2i, xi, mi, ri, ti, sdi, ni, data, slab, subset,
         add=1/2, to="none", drop00=FALSE, rows, cols)
```

Arguments

measure	a character string to specify the effect size measure corresponding to the summary data supplied. See ‘Details’ and the documentation of the escalc function for possible options.
ai	vector with the 2×2 table frequencies (upper left cell).
bi	vector with the 2×2 table frequencies (upper right cell).
ci	vector with the 2×2 table frequencies (lower left cell).
di	vector with the 2×2 table frequencies (lower right cell).
n1i	vector with the group sizes or row totals (first group/row).
n2i	vector with the group sizes or row totals (second group/row).
x1i	vector with the number of events (first group).
x2i	vector with the number of events (second group).
t1i	vector with the total person-times (first group).
t2i	vector with the total person-times (second group).
m1i	vector with the means (first group or time point).
m2i	vector with the means (second group or time point).
sd1i	vector with the standard deviations (first group or time point).
sd2i	vector with the standard deviations (second group or time point).
xi	vector with the frequencies of the event of interest.
mi	vector with the frequencies of the complement of the event of interest or the group means.
ri	vector with the raw correlation coefficients.
ti	vector with the total person-times.
sdi	vector with the standard deviations.
ni	vector with the sample/group sizes.
data	optional data frame containing the variables given to the arguments above.
slab	optional vector with labels for the studies.
subset	optional (logical or numeric) vector to specify the subset of studies that should be included in the array returned by the function.
add	see the documentation of the escalc function.
to	see the documentation of the escalc function.
drop00	see the documentation of the escalc function.
rows	optional vector with row/group names.
cols	optional vector with column/outcome names.

Details

The `escalc` function describes a wide variety of effect size measures that can be computed for a meta-analysis. The summary data used to compute those measures are typically contained in vectors, each element corresponding to a study. The `to.table` function takes this information and constructs an array of k tables from these data.

For example, in various fields (such as the health and medical sciences), the response variable measured is often dichotomous (binary), so that the data from a study comparing two different groups can be expressed in terms of a 2×2 table, such as:

	outcome 1	outcome 2	total
group 1	ai	bi	n1i
group 2	ci	di	n2i

where ai, bi, ci, and di denote the cell frequencies (i.e., the number of individuals falling into a particular category) and n1i and n2i the row totals (i.e., the group sizes).

The cell frequencies in k such 2×2 tables can be specified via the ai, bi, ci, and di arguments (or alternatively, via the ai, ci, n1i, and n2i arguments). The function then creates the corresponding $2 \times 2 \times k$ array of tables. The measure argument should then be set equal to one of the effect size measures that can be computed based on this type of data, such as "RR", "OR", "RD" (it is not relevant which specific measure is chosen, as long as it corresponds to the specified summary data). See the documentation of the `escalc` function for more details on the types of data formats available.

The examples below illustrate the use of this function.

Value

An array with k elements each consisting of either 1 or 2 rows and an appropriate number of columns.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`escalc` for a function to compute effect size estimates (and corresponding sampling variances) based on similar inputs.

`to.long` for a function to turn similar inputs into a long format dataset.

Examples

```
### create tables
dat <- to.table(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg,
               data=dat.bcg, slab=paste(author, year, sep=" "),
               rows=c("Vaccinated", "Not Vaccinated"), cols=c("TB+", "TB-"))

dat

### create tables
dat <- to.table(measure="IRR", x1i=x1i, x2i=x2i, t1i=t1i, t2i=t2i,
               data=dat.hart1999, slab=paste(study, year, sep=" "),
               rows=c("Warfarin Group", "Placebo/Control Group"))

dat

### create tables
dat <- to.table(measure="MD", m1i=m1i, sd1i=sd1i, n1i=n1i,
               m2i=m2i, sd2i=sd2i, n2i=n2i, data=dat.normand1999,
               slab=source, rows=c("Specialized Care", "Routine Care"))

dat
```

to.wide	<i>Convert Data from a Long to a Wide Format</i>
---------	--

Description

Function to convert data given in long format to a wide format.

Usage

```
to.wide(data, study, grp, ref, grpvars, postfix=c(".1",".2"),
        addid=TRUE, addcomp=TRUE, adddesign=TRUE, minlen=2,
        var.names=c("id","comp","design"))
```

Arguments

data	a data frame in long format.
study	either the name (given as a character string) or the position (given as a single number) of the study variable in the data frame.
grp	either the name (given as a character string) or the position (given as a single number) of the group variable in the data frame.
ref	optional character string to specify the reference group (must be one of the groups in the group variable). If not given, the most frequently occurring group is used as the reference group.
grpvars	either the names (given as a character vector) or the positions (given as a numeric vector) of the group-level variables.
postfix	a character string of length 2 giving the affix that is placed after the names of the group-level variables for the first and second group.

<code>addid</code>	logical to specify whether a row id variable should be added to the data frame (the default is TRUE).
<code>addcomp</code>	logical to specify whether a comparison id variable should be added to the data frame (the default is TRUE).
<code>adddesign</code>	logical to specify whether a design id variable should be added to the data frame (the default is TRUE).
<code>minlen</code>	integer to specify the minimum length of the shortened group names for the comparison and design id variables (the default is 2).
<code>var.names</code>	character vector with three elements to specify the name of the id, comparison, and design variables (the defaults are "id", "comp", and "design", respectively).

Details

A meta-analytic dataset may be structured in a ‘long’ format, where each row in the dataset corresponds to a particular study group (e.g., treatment arm). Using this function, such a dataset can be restructured into a ‘wide’ format, where each group within a study is contrasted against a particular reference group.

The `study` and `grp` arguments are used to specify the study and group variables in the dataset (either as character strings or as numbers indicating the column positions of these variables in the dataset). Optional argument `ref` is used to specify the reference group (this must be one of the groups in the `grp` variable). Argument `grpvars` is used to specify (either as a character vector or by giving the column positions) of those variables in the dataset that correspond to group-level outcomes (the remaining variables are treated as study-level outcomes).

The dataset is restructured so that a two-group study will yield a single row in the restructured dataset, contrasting the first group against the second/reference group. For studies with more than two groups (often called ‘multiarm’ or ‘multitreatment’ studies in the medical literature), the reference group is repeated as many times as needed (so a three-group study would yield two rows in the restructured dataset, contrasting two groups against a common reference group).

If a study does not include the reference group, then another group from the study will be used as the reference group. This group is chosen based on the factor levels of the `grp` variable (i.e., the last level that occurs in the study becomes the reference group).

To distinguish the names of the group-level outcome variables for the two first and second group in the restructured dataset, the strings given for the `postfix` argument are placed after the respective variable names.

If requested, row id, comparison id, and design id variables are added to the restructured dataset. The row id is simply a unique number for each row in the dataset. The comparison id variable indicates which two groups have been compared against each other. The design id variable indicates which groups were included in a particular study. The group names are shortened for the comparison and design variables (to at least `minlen`; the actual length might be longer to ensure uniqueness of the group names).

Value

A data frame with rows contrasting groups against a reference group and an appropriate number of columns (depending on the number of group-level outcome variables).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[contrmat](#) for a function to construct a contrast matrix based on a dataset in wide format.

[dat.hasselblad1998](#), [dat.lopez2019](#), [dat.obrien2003](#), [dat.pagliaro1992](#), [dat.senn2013](#) for illustrative examples.

Examples

```
### data in long format
dat <- dat.senn2013
dat <- dat[c(1,4,3,2,5,6)]
dat

### restructure to wide format
dat <- to.wide(dat, study="study", grp="treatment", ref="placebo", grpvars=4:6)
dat

### data in long format
dat <- dat.hasselblad1998
dat

### restructure to wide format
dat <- to.wide(dat, study="study", grp="trt", ref="no_contact", grpvars=6:7)
dat
```

transf

Transformation Functions

Description

Functions to carry out various types of transformations that are useful for meta-analyses.

Usage

```
transf.rtoz(xi)
transf.ztor(xi)
transf.logit(xi)
transf.ilogit(xi)
transf.probit(xi)
transf.iprobit(xi)
```

```
transf.arcsin(xi)
transf.iarcsin(xi)
transf.pft(xi, ni)
transf.ipft(xi, ni)
transf.ipft.hm(xi, targs)
transf.isqrt(xi)
transf.irft(xi, ti)
transf.iirft(xi, ti)
transf.ahw(xi)
transf.iahw(xi)
transf.abt(xi)
transf.iabt(xi)
transf.r2toz(xi)
transf.ztor2(xi)
transf.ztor.int(xi, targs)
transf.exp.int(xi, targs)
transf.ilogit.int(xi, targs)
transf.iprobit.int(xi, targs)
transf.iarcsin.int(xi, targs)
transf.iahw.int(xi, targs)
transf.iabt.int(xi, targs)
transf.ztor.mode(xi, targs)
transf.exp.mode(xi, targs)
transf.ilogit.mode(xi, targs)
transf.iprobit.mode(xi, targs)
transf.iarcsin.mode(xi, targs)
transf.iahw.mode(xi, targs)
transf.iabt.mode(xi, targs)
transf.dtou1(xi)
transf.dtou2(xi)
transf.dtou3(xi)
transf.dtoovl(xi)
transf.dtocles(xi)
transf.clestod(xi)
transf.dtocles.int(xi, targs)
transf.dtocliffd(xi)
transf.dtobesd(xi)
transf.dtomd(xi, targs)
transf.dtorpb(xi, n1i, n2i)
transf.dtorbis(xi, n1i, n2i)
transf.rpbtorbis(xi, pi)
transf.rtorpb(xi, pi)
transf.rtod(xi, n1i, n2i)
transf.rpbtod(xi, n1i, n2i)
transf.lnortord(xi, pc)
transf.lnortorr(xi, pc)
transf.lnortod.norm(xi)
transf.lnortod.logis(xi)
```

```

transf.dtolnor.norm(xi)
transf.dtolnor.logis(xi)
transf.lnortortet.pearson(xi)
transf.lnortortet.digby(xi)

```

Arguments

<code>xi</code>	vector of values to be transformed.
<code>ni</code>	vector of sample sizes.
<code>n1i</code>	vector of sample sizes for the first group.
<code>n2i</code>	vector of sample sizes for the second group.
<code>ti</code>	vector of person-times at risk.
<code>pc</code>	control group risk (either a single value or a vector).
<code>pi</code>	proportion of individuals falling into the first of the two groups that is created by the dichotomization.
<code>targs</code>	list with additional arguments for the transformation function. See ‘Details’.

Details

The following transformation functions are currently implemented:

- `transf.rtoz`: Fisher’s r-to-z transformation for correlation coefficients (same as `atanh(x)`).
- `transf.ztor`: inverse of the former (i.e., the z-to-r transformation; same as `tanh(x)`).
- `transf.logit`: logit (log odds) transformation for proportions (same as `qlogis(x)`).
- `transf.ilogit`: inverse of the former (same as `plogis(x)`).
- `transf.probit`: probit transformation for proportions (same as `qnorm(x)`).
- `transf.iprobit`: inverse of the former (same as `pnorm(x)`).
- `transf.arcsin`: arcsine square root transformation for proportions.
- `transf.iarcsin`: inverse of the former.
- `transf.pft`: Freeman-Tukey (double arcsine) transformation for proportions. See Freeman and Tukey (1950). The `xi` argument is used to specify the proportions and the `ni` argument the corresponding sample sizes.
- `transf.ipft`: inverse of the former. See Miller (1978).
- `transf.ipft.hm`: inverse of the former, using the harmonic mean of the sample sizes for the back-transformation. See Miller (1978). The sample sizes are specified via the `targs` argument (the list element should be called `ni`).
- `transf.isqrt`: inverse of the square root transformation (i.e., function to square a number).
- `transf.irft`: Freeman-Tukey transformation for incidence rates. See Freeman and Tukey (1950). The `xi` argument is used to specify the incidence rates and the `ti` argument the corresponding person-times at risk.
- `transf.iirft`: inverse of the former.

- `transf.ahw`: transformation of coefficient alpha as suggested by Hakstian and Whalen (1976), except that $1 - (1 - \alpha)^{1/3}$ is used (so that the transformed values are a monotonically increasing function of the α values).
- `transf.iahw`: inverse of the former.
- `transf.abt`: transformation of coefficient alpha as suggested by Bonett (2002), except that $-\log(1 - \alpha)$ is used (so that the transformed values are a monotonically increasing function of the α values).
- `transf.iabt`: inverse of the former.
- `transf.r2toz`: variance stabilizing transformation for the coefficient of determination, given by $z_i = \frac{1}{2} \log \left(\frac{1 + \sqrt{R_i^2}}{1 - \sqrt{R_i^2}} \right)$ (see Olkin & Finn, 1995, but with the additional $\frac{1}{2}$ factor for consistency with the usual r-to-z transformation).
- `transf.ztor2`: inverse of the former.
- `transf.ztor.int`: integral transformation function for the z-to-r transformation. See 'Note'.
- `transf.exp.int`: integral transformation function for the exponential transformation. See 'Note'.
- `transf.ilogit.int`: integral transformation function for the inverse logit transformation. See 'Note'.
- `transf.iprobit.int`: integral transformation function for the inverse probit transformation. See 'Note'.
- `transf.iarcsin.int`: integral transformation function for the inverse arcsine square root transformation. See 'Note'.
- `transf.iahw.int`: integral transformation function for the `transf.iahw` transformation. See 'Note'.
- `transf.iahw.int`: integral transformation function for the `transf.iabt` transformation. See 'Note'.
- `transf.ztor.mode`: function to determine the mode of an atanh-normal variable.
- `transf.exp.mode`: function to determine the mode of a log-normal variable.
- `transf.ilogit.mode`: function to determine the mode of a logit-normal variable.
- `transf.iprobit.mode`: function to determine the mode of a probit-normal variable.
- `transf.iarcsin.mode`: function to determine the mode of an arcsine-square-root-normal variable.
- `transf.iahw.mode`: function to determine the mode of a `transf.ahw`-normal variable.
- `transf.iabt.mode`: function to determine the mode of a `transf.abt`-normal variable.
- `transf.dtou1`: transformation of standardized mean differences to Cohen's U_1 values (Cohen, 1988). Under the assumption that the data for those in the first (say, treated) and second (say, control) group are normally distributed with equal variances but potentially different means, Cohen's U_1 indicates the proportion of non-overlap between the two distributions (i.e., when $d = 0$, then U_1 is equal to 0, which goes to 1 as d increases).
- `transf.dtou2`: transformation of standardized mean differences to Cohen's U_2 values (Cohen, 1988). Under the same assumptions as above, Cohen's U_2 indicates the proportion in the first group that exceeds the same proportion in the second group (i.e., when $d = 0$, then U_2 is equal to 0.5, which goes to 1 as d increases).

- `transf.dtou3`: transformation of standardized mean differences to Cohen's U_3 values (Cohen, 1988). Under the same assumptions as above, Cohen's U_3 indicates the proportion of individuals in the first group that have a higher value than the mean of those in the second group (i.e., when $d = 0$, then U_3 is equal to 0.5, which goes to 1 as d increases).
- `transf.dtoovl`: transformation of standardized mean differences to overlapping coefficient values under the same assumptions as above (Inman & Bardley, 1989). Note that $1 - U_1$ is *not* the same as the overlapping coefficient (see Grice & Barrett, 2014).
- `transf.dtocles`: transformation of standardized mean differences to common language effect size (CLES) values (McGraw & Wong, 1992) (also called the probability of superiority). A CLES value indicates the probability that a randomly sampled individual from the first group has a higher value than a randomly sampled individual from the second group (i.e., when $d = 0$, then the CLES is equal to 0.5, which goes to 1 as d increases).
- `transf.clestod`: inverse of the former.
- `transf.dtocles.int`: integral transformation function for the `transf.dtocles` transformation. See 'Note'.
- `transf.dtocliffd`: transformation of standardized mean differences to Cliff's delta values.
- `transf.dtobesd`: transformation of standardized mean differences to binomial effect size display values (Rosenthal & Rubin, 1982). Note that the function only provides the proportion of individuals in the first group scoring above the median (of the scores of the two group combined). The proportion of individuals in the second group scoring above the median is simply one minus this value.
- `transf.dtomd`: transformation of standardized mean differences to mean differences given a known standard deviation (which needs to be specified via the `targs` argument). For example: `transf.dtomd(0.5, targs=15)`.
- `transf.dtorpb`: transformation of standardized mean differences to point-biserial correlations. Arguments `n1i` and `n2i` denote the number of individuals in the first and second group, respectively. If `n1i` and `n2i` are not specified, the function assumes $n1i = n2i$ and uses the approximate formula $r_{pb} = \frac{d}{\sqrt{d^2 + 4}}$. If `n1i` and `n2i` are specified, the function uses the exact transformation formula $r_{pb} = \frac{d}{\sqrt{d^2 + h}}$, where $h = \frac{m}{n_1} + \frac{m}{n_2}$ and $m = n_1 + n_2 - 2$ (Jacobs & Viechtbauer, 2017).
- `transf.dtorbis`: transformation of standardized mean differences to biserial correlations. Like `transf.dtorpb`, but the point-biserial correlations are then transformed to biserial correlations with $r_{bis} = \frac{\sqrt{p(1-p)}}{f(z_p)} r_{pb}$, where $p = \frac{n_1}{n_1 + n_2}$ and $f(z_p)$ denotes the density of the standard normal distribution at value z_p , which is the point for which $P(Z > z_p) = p$, with Z denoting a random variable following a standard normal distribution (Jacobs & Viechtbauer, 2017).
- `transf.rpbtorbis`: transformation of point-biserial correlations to biserial correlations. Argument `pi` denotes the proportion of individuals falling into the first of the two groups that is created by the dichotomization (hence, $1 - \pi$ falls into the second group). If `pi` is not specified, the function assumes $\pi = 0.5$, which corresponds to dichotomization at the median. The transformation is carried out as described for `transf.dtorbis`.
- `transf.rtorpb`: transformation of Pearson product-moment correlations to the corresponding point-biserial correlations, when one of the two variables is dichotomized. Argument `pi` can be used to denote the proportion of individuals falling into the first of the two groups that

is created by the dichotomization (hence, $1-p_i$ falls into the second group). If p_i is not specified, the function assumes $p_i=0.5$, which corresponds to dichotomization at the median. This function is simply the inverse of `transf.rpbtorbis`.

- `transf.rtod`: transformation of Pearson product-moment correlations to the corresponding standardized mean differences, when one of the two variables is dichotomized. Arguments `n1i` and `n2i` can be used to denote the number of individuals in the first and second group created by the dichotomization. If `n1i` and `n2i` are not specified, the function assumes `n1i = n2i`. This function is simply the inverse of `transf.dtorbis`.
- `transf.rpbtd`: transformation of point-biserial correlations to standardized mean differences. This is simply the inverse of `transf.dtorpb`.
- `transf.lnortod`: transformation of log odds ratios to risk differences, assuming a particular value for the control group risk (which needs to be specified via the `pc` argument).
- `transf.lnortorr`: transformation of log odds ratios to risk ratios, assuming a particular value for the control group risk (which needs to be specified via the `pc` argument). Note that this function transforms to risk ratios, *not* log risk ratios.
- `transf.lnortod.norm`: transformation of log odds ratios to standardized mean differences (assuming normal distributions) (Cox & Snell, 1989).
- `transf.lnortod.logis`: transformation of log odds ratios to standardized mean differences (assuming logistic distributions) (Chinn, 2000).
- `transf.dtolnor.norm`: transformation of standardized mean differences to log odds ratios (assuming normal distributions) (Cox & Snell, 1989).
- `transf.dtolnor.logis`: transformation of standardized mean differences to log odds ratios (assuming logistic distributions) (Chinn, 2000).
- `transf.lnortortet.pearson`: transformation of log odds ratios to tetrachoric correlations as suggested by Pearson (1900).
- `transf.lnortortet.digby`: transformation of log odds ratios to tetrachoric correlations as suggested by Digby (1983).

Value

A vector with the transformed values.

Note

The integral transformation method for a transformation function $h(z)$ is given by

$$\int_{\text{lower}}^{\text{upper}} h(z)f(z)dz$$

using the limits `targs$lower` and `targs$upper`, where $f(z)$ is the density of a normal distribution with mean equal to `xi` and variance equal to `targs$tau2`. By default, `targs$lower` and `targs$upper` are set to reasonable values and, if possible, `targs$tau2` is extracted from the model object in functions where such transformation functions are typically applied (e.g., `predict`). An example is provided below.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Bonett, D. G. (2002). Sample size requirements for testing and estimating coefficient alpha. *Journal of Educational and Behavioral Statistics*, **27**(4), 335–340. <https://doi.org/10.3102/10769986027004335>
- Chinn, S. (2000). A simple method for converting an odds ratio to effect size for use in meta-analysis. *Statistics in Medicine*, **19**(22), 3127–3131. [https://doi.org/10.1002/1097-0258\(20001130\)19:22<3127::ai](https://doi.org/10.1002/1097-0258(20001130)19:22<3127::ai)
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Cox, D. R., & Snell, E. J. (1989). *Analysis of binary data* (2nd ed.). London: Chapman & Hall.
- Digby, P. G. N. (1983). Approximating the tetrachoric correlation coefficient. *Biometrics*, **39**(3), 753–757. <https://doi.org/10.2307/2531104>
- Fisher, R. A. (1921). On the “probable error” of a coefficient of correlation deduced from a small sample. *Metron*, **1**, 1–32. <https://hdl.handle.net/2440/15169>
- Freeman, M. F., & Tukey, J. W. (1950). Transformations related to the angular and the square root. *Annals of Mathematical Statistics*, **21**(4), 607–611. <https://doi.org/10.1214/aoms/1177729756>
- Grice, J. W., & Barrett, P. T. (2014). A note on Cohen’s overlapping proportions of normal distributions. *Psychological Reports*, **115**(3), 741–747. <https://doi.org/10.2466/03.pr0.115c29z4>
- Hakstian, A. R., & Whalen, T. E. (1976). A k-sample significance test for independent alpha coefficients. *Psychometrika*, **41**(2), 219–231. <https://doi.org/10.1007/BF02291840>
- Inman, H. F., & Bradley Jr, E. L. (1989). The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities. *Communications in Statistics, Theory and Methods*, **18**(10), 3851–3874. <https://doi.org/10.1080/03610928908830127>
- Jacobs, P., & Viechtbauer, W. (2017). Estimation of the biserial correlation and its sampling variance for use in meta-analysis. *Research Synthesis Methods*, **8**(2), 161–180. <https://doi.org/10.1002/jrsm.1218>
- McGraw, K. O., & Wong, S. P. (1992). A common language effect size statistic. *Psychological Bulletin*, **111**(2), 361–365. <https://doi.org/10.1037/0033-2909.111.2.361>
- Miller, J. J. (1978). The inverse of the Freeman-Tukey double arcsine transformation. *American Statistician*, **32**(4), 138. <https://doi.org/10.1080/00031305.1978.10479283>
- Olkin, I., & Finn, J. D. (1995). Correlations redux. *Psychological Bulletin*, **118**(1), 155–164. <https://doi.org/10.1037/0033-2909.118.1.155>
- Pearson, K. (1900). Mathematical contributions to the theory of evolution. VII. On the correlation of characters not quantitatively measurable. *Philosophical Transactions of the Royal Society of London, Series A*, **195**, 1–47. <https://doi.org/10.1098/rsta.1900.0022>
- Rosenthal, R., & Rubin, D. B. (1982). A simple, general purpose display of magnitude of experimental effect. *Journal of Educational Psychology*, **74**(2), 166–169. <https://doi.org/10.1037/0022-0663.74.2.166>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model
```

```

res <- rma(yi, vi, data=dat)

### average risk ratio with 95% CI (but technically, this provides an
### estimate of the median risk ratio, not the mean risk ratio!)
predict(res, transf=exp)

### average risk ratio with 95% CI using the integral transformation
predict(res, transf=transf.exp.int, targs=list(tau2=res$tau2, lower=-4, upper=4))

### this also works
predict(res, transf=transf.exp.int, targs=list(tau2=res$tau2))

### this as well
predict(res, transf=transf.exp.int)

```

trimfill

Trim and Fill Analysis for 'rma.uni' Objects

Description

Function to carry out a trim and fill analysis for objects of class "rma.uni".

Usage

```

trimfill(x, ...)

## S3 method for class 'rma.uni'
trimfill(x, side, estimator="L0", maxiter=100, verbose=FALSE, ilim, ...)

```

Arguments

x	an object of class "rma.uni".
side	optional character string (either "left" or "right") to specify on which side of the funnel plot the missing studies should be imputed. If left unspecified, the side is chosen within the function depending on the results of the regression test (see regtest for details on this test).
estimator	character string (either "L0", "R0", or "Q0") to specify the estimator for the number of missing studies (the default is "L0").
maxiter	integer to specify the maximum number of iterations for the trim and fill method (the default is 100).
verbose	logical to specify whether output should be generated on the progress of the iterative algorithm used as part of the trim and fill method (the default is FALSE).
ilim	limits for the imputed values. If unspecified, no limits are used.
...	other arguments.

Details

The trim and fill method is a nonparametric (rank-based) data augmentation technique proposed by Duval and Tweedie (2000a, 2000b; see also Duval, 2005). The method can be used to estimate the number of studies missing from a meta-analysis due to suppression of the most extreme results on one side of the funnel plot. The method then augments the observed data so that the funnel plot is more symmetric and recomputes the pooled estimate based on the complete data. The trim and fill method can only be used in the context of an equal- or a random-effects model (i.e., in models without moderators). The method should not be regarded as a way of yielding a more ‘valid’ estimate of the pooled effect, but as a way of examining the sensitivity of the results to one particular selection mechanism (i.e., one particular form of publication bias).

Value

An object of class `c("rma.uni.trimfill", "rma.uni", "rma")`. The object is a list containing the same components as objects created by `rma.uni`, except that the data are augmented by the trim and fill method. The following components are also added:

<code>k0</code>	estimated number of missing studies.
<code>side</code>	either "left" or "right", indicating on which side of the funnel plot the missing studies (if any) were imputed.
<code>se.k0</code>	standard error of <code>k0</code> .
<code>p.k0</code>	p-value for the test of H_0 : no missing studies on the chosen side (only when <code>estimator="R0"</code> ; NA otherwise).
<code>yi</code>	the observed effect size estimates plus the imputed values (if there are any).
<code>vi</code>	the corresponding sampling variances
<code>fill</code>	a logical vector indicating which of the values in <code>yi</code> are the observed (FALSE) and the imputed (TRUE) data.

The results of the fitted model after the data augmentation are printed with the `print` function. Calling `funnel` on the object provides a funnel plot of the observed and imputed data.

Note

Three different estimators for the number of missing studies were proposed by Duval and Tweedie (2000a, 2000b). Based on these articles and Duval (2005), "R0" and "L0" are recommended. An advantage of estimator "R0" is that it provides a test of the null hypothesis that the number of missing studies (on the chosen side) is zero.

If the estimates are bounded (e.g., correlations are bounded between -1 and +1, proportions are bounded between 0 and 1), one can use the `ilim` argument to enforce those limits when imputing values (i.e., imputed values cannot exceed those limits then).

The model used during the trim and fill procedure is the same as used by the original model object. Hence, if an equal-effects model is passed to the function, then an equal-effects model is also used during the trim and fill procedure and the results provided are also based on an equal-effects model. This would be an ‘equal-equal’ approach. Similarly, if a random-effects model is passed to the function, then the same model is used as part of the trim and fill procedure and for the final analysis. This would be a ‘random-random’ approach. However, one can also easily fit a different model for the final analysis than was used for the trim and fill procedure. See ‘Examples’ for an illustration of an ‘equal-random’ approach.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Duval, S. J., & Tweedie, R. L. (2000a). Trim and fill: A simple funnel-plot-based method of testing and adjusting for publication bias in meta-analysis. *Biometrics*, **56**(2), 455–463. <https://doi.org/10.1111/j.0006-341x.2000.00006-341x>.
- Duval, S. J., & Tweedie, R. L. (2000b). A nonparametric "trim and fill" method of accounting for publication bias in meta-analysis. *Journal of the American Statistical Association*, **95**(449), 89–98. <https://doi.org/10.1080/01621459.2000.10473905>
- Duval, S. J. (2005). The trim and fill method. In H. R. Rothstein, A. J. Sutton, & M. Borenstein (Eds.) *Publication bias in meta-analysis: Prevention, assessment, and adjustments* (pp. 127–144). Chichester, England: Wiley.
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[funnel](#) for a function to create funnel plots of the observed and augmented data.

[regtest](#) for the regression test, [ranktest](#) for the rank correlation test, [tes](#) for the test of excess significance, [fsn](#) to compute the fail-safe N (file drawer analysis), and [selmodel](#) for selection models.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### meta-analysis of the log risk ratios using an equal-effects model
res <- rma(yi, vi, data=dat, method="EE")
taf <- trimfill(res)
taf
funnel(taf, cex=1.2, legend=list(show="cis"))

### estimator "R0" also provides test of H0: no missing studies (on the chosen side)
taf <- trimfill(res, estimator="R0")
taf

### meta-analysis of the log risk ratios using a random-effects model
res <- rma(yi, vi, data=dat)
taf <- trimfill(res)
taf
funnel(taf, cex=1.2, legend=list(show="cis"))

### the examples above are equal-equal and random-random approaches

### illustration of an equal-random approach
res <- rma(yi, vi, data=dat, method="EE")
taf <- trimfill(res)
```

```
filled <- data.frame(yi = taf$yi, vi = taf$vi, fill = taf$fill)
filled
rma(yi, vi, data=filled)
```

update.rma

Model Updating for 'rma' Objects

Description

Function to update and (by default) refit "rma" models. It does this by extracting the call stored in the object, updating the call, and (by default) evaluating that call.

Usage

```
## S3 method for class 'rma'
update(object, formula., ..., evaluate=TRUE)
```

Arguments

object	an object of class "rma".
formula.	changes to the formula. See 'Details'.
...	additional arguments to the call, or arguments with changed values.
evaluate	logical to specify whether to evaluate the new call or just return the call.

Details

For objects of class "rma.uni", "rma.glmm", and "rma.mv", the formula. argument can be used to update the set of moderators included in the model (see 'Examples').

Value

If evaluate=TRUE the fitted object, otherwise the updated call.

Author(s)

Based on [update.default](#), with changes made by Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>) so that the formula updating works with the (somewhat non-standard) interface of the [rma.uni](#), [rma.glmm](#), and [rma.mv](#) functions.

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models which can be updated / refit.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit random-effects model (method="REML" is default)
res <- rma(yi, vi, data=dat, digits=3)
res

### fit mixed-effects model with two moderators (absolute latitude and publication year)
res <- update(res, ~ ablat + year)
res

### remove 'year' moderator
res <- update(res, ~ . - year)
res

### fit model with ML estimation
update(res, method="ML")

### example with rma.glmm()
res <- rma.glmm(measure="OR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg, digits=3)
res <- update(res, mods = ~ ablat)
res

### fit conditional model with approximate likelihood
update(res, model="CM.AL")
```

vcalc

Construct or Approximate the Variance-Covariance Matrix of Dependent Effect Size Estimates

Description

Function to construct or approximate the variance-covariance matrix of dependent effect size estimates, or more precisely, of their sampling errors (i.e., the V matrix in [rma.mv](#)).

Usage

```
vcalc(vi, cluster, subgroup, obs, type, time1, time2, grp1, grp2, w1, w2,
      data, rho, phi, rvars, checkpd=TRUE, nearpd=FALSE, sparse=FALSE, ...)
```

Arguments

vi	numeric vector to specify the sampling variances of the effect size estimates.
cluster	vector to specify the clustering variable (e.g., study).
subgroup	optional vector to specify different (independent) subgroups within clusters.
obs	optional vector to distinguish different estimates corresponding to the same construct or response/dependent variable.

type	optional vector to distinguish different types of constructs or response/dependent variables underlying the estimates.
time1	optional numeric vector to specify the time points when the estimates were obtained (in the first condition if the estimates represent contrasts between two conditions).
time2	optional numeric vector to specify the time points when the estimates were obtained in the second condition (only relevant when the estimates represent contrasts between two conditions).
grp1	optional vector to specify the group of the first condition when the estimates represent contrasts between two conditions.
grp2	optional vector to specify the group of the second condition when the estimates represent contrasts between two conditions.
w1	optional numeric vector to specify the size of the group (or more generally, the inverse-sampling variance weight) of the first condition when the estimates represent contrasts between two conditions.
w2	optional numeric vector to specify the size of the group (or more generally, the inverse-sampling variance weight) of the second condition when the estimates represent contrasts between two conditions.
data	optional data frame containing the variables given to the arguments above.
rho	argument to specify the correlation(s) of estimates measured concurrently. See ‘Details’.
phi	argument to specify the autocorrelation of estimates measured at different time points. See ‘Details’.
rvars	optional argument for specifying the variables that correspond to the correlation matrices of the studies (if this is specified, all arguments above except for cluster and subgroup are ignored). See ‘Details’.
checkpd	logical to specify whether to check that the variance-covariance matrices within clusters are positive definite (the default is TRUE). See ‘Note’.
nearpd	logical to specify whether the nearPD function from the Matrix package should be used on variance-covariance matrices that are not positive definite. See ‘Note’.
sparse	logical to specify whether the variance-covariance matrix should be returned as a sparse matrix (the default is FALSE).
...	other arguments.

Details

Standard meta-analytic models (such as those that can be fitted with the [rma.uni](#) function) assume that the effect size estimates (or more precisely, their sampling errors) are independent. This assumption is typically violated whenever multiple estimates are computed based on the same sample of subjects (or whatever the study units are) or if there is at least partial overlap of subjects that contribute information to the computation of multiple estimates.

The present function can be used to construct or approximate the variance-covariance matrix of the sampling errors of dependent effect size estimates for a wide variety of circumstances (this variance-covariance matrix is the so-called V matrix that may be needed as input for multilevel/multivariate

meta-analytic models as can be fitted with the `rma.mv` function; see also [here](#) for some recommendations on a general workflow for meta-analyses involving complex dependency structures).

Argument `cluster` is used to specify the clustering variable. Rows with the same value of this variable are allowed to be dependent, while rows with different values are assumed to be independent. Typically, `cluster` will be a study identifier.

Within the same cluster, there may be different subgroups with no overlap of subjects across subgroups. Argument `subgroup` can be used to distinguish such subgroups. Rows with the same value of this variable within a cluster are allowed to be dependent, while rows with different values are assumed to be independent even if they come from the same cluster. Therefore, from hereon, ‘cluster’ really refers to the combination of `cluster` and `subgroup`.

Multiple estimates belonging to the same cluster may be dependent due to a variety of reasons:

1. The same construct of interest (e.g., severity of depression) may have been measured using different scales or instruments within a study (e.g., using the Beck Depression Inventory (BDI) and the Hamilton Depression Rating Scale (HDRS)) based on which multiple effect size estimates can be computed for the same group of subjects (e.g., contrasting a treatment versus a control group with respect to each scale). In this case, we have multiple estimates that are different ‘observations’ of the effect with respect to the same type of construct.

Argument `obs` is then used to distinguish different effect size estimates corresponding to the same construct. If `obs` is specified, then argument `rho` must also be used to specify the degree of correlation among the sampling errors of the different estimates. Since this correlation is typically unknown, the correlation among the various scales (or a rough ‘guestimate’ thereof) can be used as a proxy (i.e., the (typical) correlation between BDI and HDRS measurements).

One can also pass an entire correlation matrix via `rho` to specify, for each possible pair of `obs` values, the corresponding correlation. The row/column names of the matrix must then correspond to the unique values of the `obs` variable.

2. Multiple types of constructs (or more generally, types of response/dependent variables) may have been measured in the same group of subjects (e.g., severity of depression as measured with the Beck Depression Inventory (BDI) and severity of anxiety as measured with the State-Trait Anxiety Inventory (STAI)). If this is of interest for a meta-analysis, effect size estimates can then be computed with respect to each ‘type’ of construct.

Argument `type` is then used to distinguish estimates corresponding to these different types of constructs. If `type` is specified, then argument `rho` must also be used to specify the degree of correlation among the sampling errors of estimates belonging to these different types. As above, the correlation among the various scales is typically used here as a proxy (i.e., the (typical) correlation between BDI and STAI measurements).

One can also pass an entire correlation matrix via `rho` to specify, for each possible pair of `type` values, the corresponding correlation. The row/column names of the matrix must then correspond to the unique values of the `type` variable.

3. If there are multiple types of constructs, multiple scales or instruments may also have been used (in at least some of the studies) to measure the same construct and hence there may again be multiple effect size estimates that are ‘observations’ of the same type of construct. Arguments `type` and `obs` should then be used together to specify the various construct types and observations thereof. In this case, argument `rho` must be a vector of two values, the first to specify the within-construct correlation and the second to specify the between-construct correlation.

One can also specify a list with two elements for *rho*, the first element being either a scalar or an entire correlation matrix for the within-construct correlation(s) and the second element being a scalar or an entire correlation matrix for the between-construct correlation(s). As above, any matrices specified must have row/column names corresponding to the unique values of the *obs* and/or *type* variables.

4. The same construct and scale may have been assessed/used multiple times, allowing the computation of multiple estimates for the same group of subjects at different time points (e.g., right after the end of a treatment, at a short-term follow-up, and at a long-term follow-up). Argument *time1* is then used to specify the time points when the estimates were obtained. Argument *phi* must then also be used to specify the autocorrelation among the sampling errors of two estimates that differ by one unit on the *time1* variable. As above, the autocorrelation of the measurements themselves can be used here as a proxy.

If multiple constructs and/or multiple scales have also been assessed at the various time points, then arguments *type* and/or *obs* (together with argument *rho*) should be used as needed to differentiate estimates corresponding to the different constructs and/or scales.

5. Many effect size measures (e.g., raw or standardized mean differences, log-transformed ratios of means, log risk/odds ratios and risk differences) reflect the difference between two conditions (i.e., a contrast). Within a study, there may be more than two conditions, allowing the computation of multiple such contrasts (e.g., treatment A versus a control condition and treatment B versus the same control condition) and hence corresponding effect size estimates. The reuse of information from the ‘shared’ condition (in this example, the control condition) then induces correlation among the estimates.

To account for this, arguments *grp1* and *grp2* should be used to specify (within each cluster) which two groups were compared in the computation of each estimate (e.g., in the example above, the coding could be *grp1*=c(2,3) and *grp2*=c(1,1); whether numbers or strings are used as identifiers is irrelevant).

The degree of correlation between two contrast-type effect size estimates that is induced by the use of a shared condition is a function of the size of the groups involved in the computation of the two estimates (or, more generally, the inverse-sampling variance weights of the condition-specific outcomes). By default, the group sizes (weights) are assumed to be identical across conditions, which implies a correlation of 0.5. If the group sizes (weights) are known, they can be specified via arguments *w1* and *w2* (in which case this information is used by the function to calculate a more accurate estimate of the correlation induced by the shared condition).

Moreover, a contrast-type effect size estimate can be based on a between- or a within-subjects design. When at least one or more of the contrast-type effect size estimates are based on a within-subjects design, then *time1* and *time2* should be used in combination with *grp1* and *grp2* to specify for each estimate the group(s) and time point(s) involved.

For example, *grp1*=c(2,3) and *grp2*=c(1,1) as above in combination with *time1*=c(1,1) and *time2*=c(1,1) would imply a between-subjects design involving three groups where two estimates were computed contrasting groups 2 and 3 versus group 1 at a single time point. On the other hand, *grp1*=c(1,1) and *grp2*=c(1,1) in combination with *time1*=c(2,3) and *time2*=c(1,1) would imply a within-subjects design where two estimates were computed contrasting time points 2 and 3 versus time point 1 in a single group. Argument *phi* is then used as above to specify the autocorrelation of the measurements within groups (i.e., for the within-subjects design above, it would be the autocorrelation between time points 2 and 1 or equivalently, between time points 3 and 2).

All of the arguments above can be specified together to account for a fairly wide variety of depen-

dency types.

Using the `rvars` Argument:

The function also provides an alternative approach for constructing the variance-covariance matrix using the `rvars` argument. Here, one must specify the names of the variables in the dataset that correspond to the correlation matrices of the studies. The variables should be specified as a vector (e.g., `c(var1, var2, var3)`) and do not need to be quoted.

In particular, let k_i denote the number of rows corresponding to the i th cluster. Then the values of the first k_i variables from `rvars` are used to construct the correlation matrix and, together with the sampling variances (specified via `vi`), the variance-covariance matrix. Say there are three studies, the first with two correlated estimates, the second with a single estimate, and the third with four correlated estimates. Then the data structure should look like this:

```
study yi vi r1 r2 r3 r4
=====
      1 . . 1 NA NA NA
      1 . . .6 1 NA NA
-----
      2 . . 1 NA NA NA
-----
      3 . . 1 NA NA NA
      3 . . .8 1 NA NA
      3 . . .5 .5 1 NA
      3 . . .5 .5 .8 1
=====
```

with `rvars = c(r1, r2, r3, r4)`. If the `rvars` variables are a consecutive set in the data frame (as above), then one can use the shorthand notation `rvars = c(r1:r4)`, so `r1` denotes the first and `r4` the last variable in the set. Note that only the lower triangular part of the submatrices defined by the `rvars` variables is used. Also, it is important that the rows in the dataset corresponding to a particular study are in consecutive order as shown above.

There must be as many variables specified via `rvars` as the number of rows in the *largest* cluster (in smaller clusters, the non-relevant variables can be set to NA; see above).

Value

A $k \times k$ variance-covariance matrix (given as a sparse matrix when `sparse=TRUE`), where k denotes the length of the `vi` variable (i.e., the number of rows in the dataset). When not given as a sparse matrix, the object has class `"vcovmat"`. See [methods.vcovmat](#) for some method functions for such objects.

Note

Depending on the data structure, the specified variables, and the specified values for `rho` and/or `phi`, it is possible that the constructed variance-covariance matrix is not positive definite within one or more clusters (this is checked when `checkpd=TRUE`, which is the default). If such non-positive definite submatrices are found, the reasons for this should be carefully checked since this might indicate misapplication of the function and/or the specification of implausible values for `rho` and/or `phi`.

When setting `nearpd=TRUE`, the `nearPD` function from the **Matrix** package is used on variance-covariance submatrices that are not positive definite. This should only be used cautiously and after understanding why these matrices are not positive definite.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>) with some tweaks to speed up the computations by James Pustejovsky (<pustejovsky@wisc.edu>, <https://jepusto.com>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

`escalc` for a function to compute the effect size estimates (and corresponding sampling variances) for which a variance-covariance matrix could be constructed.

`rcalc` for a function to construct the variance-covariance matrix of dependent correlation coefficients.

`rma.mv` for a model fitting function that can be used to meta-analyze dependent effect size estimates.

Examples

```
#####

### see help(dat.assink2016) for further details on this dataset

dat <- dat.assink2016
head(dat, 9)

### assume that the effect size estimates within studies are correlated with rho=0.6
V <- vcalc(vi, cluster=study, obs=esid, data=dat, rho=0.6)

### show part of V matrix for studies 1 and 2
round(V[dat$study %in% c(1,2), dat$study %in% c(1,2)], 4)

### or show as list of matrices
blsplit(V, dat$study, round, 4)[1:2]

### use a correlation of 0.7 for estimates corresponding to the same type of
### delinquent behavior and a correlation of 0.5 for estimates corresponding
### to different types of delinquent behavior
V <- vcalc(vi, cluster=study, type=delttype, obs=esid, data=dat, rho=c(0.7, 0.5))
blsplit(V, dat$study, round, 3)[16]

### examine the correlation matrix for study 16
blsplit(V, dat$study, cov2cor)[16]

#####
```

```

### see help(dat.ishak2007) for further details on this dataset

dat <- dat.ishak2007
head(dat, 5)

### create long format dataset
dat <- reshape(dat, direction="long", idvar="study", v.names=c("yi","vi"),
               varying=list(c(2,4,6,8), c(3,5,7,9)))
dat <- dat[order(study, time),]

### remove missing measurement occasions from dat
dat <- dat[!is.na(yi),]
rownames(dat) <- NULL

### show the data for the first 5 studies
head(dat, 8)

### construct the full (block diagonal) V matrix with an AR(1) structure
### assuming an autocorrelation of 0.97 as estimated by Ishak et al. (2007)
V <- vcalc(vi, cluster=study, time1=time, phi=0.97, data=dat)
V[1:8, 1:8]
cov2cor(V[1:8, 1:8])

### or show as a list of matrices
blsplit(V, dat$study)[1:5]
blsplit(V, dat$study, cov2cor)[1:5]

#####

### see help(dat.kalaian1996) for further details on this dataset

dat <- dat.kalaian1996
head(dat, 12)

### construct the variance-covariance matrix assuming rho = 0.66 for effect size
### estimates corresponding to the 'verbal' and 'math' outcome types
V <- vcalc(vi, cluster=study, type=outcome, data=dat, rho=0.66)
round(V[1:12,1:12], 4)

#####

### see help(dat.berkey1998) for further details on this dataset

dat <- dat.berkey1998

### variables v1i and v2i correspond to the 2x2 var-cov matrices of the studies;
### so use these variables to construct the V matrix (note: since v1i and v2i are
### var-cov matrices and not correlation matrices, set vi=1 for all rows)
V <- vcalc(vi=1, cluster=author, rvars=c(v1i, v2i), data=dat)
V
round(cov2cor(V), 2)

```

```

### or show as a list of matrices
blsplit(V, dat$author, function(x) round(cov2cor(x), 2))

### construct the variance-covariance matrix assuming rho = 0.4 for effect size
### estimates corresponding to the 'PD' and 'AL' outcome types
V <- vcalc(vi=vi, cluster=trial, type=outcome, data=dat, rho=0.4)
round(V,4)
cov2cor(V)

#####

### see help(dat.knapp2017) for further details on this dataset

dat <- dat.knapp2017
dat[-c(1:2)]

### create variable that indicates the task and difficulty combination as increasing integers
dat$task.diff <- unlist(lapply(split(dat, dat$study), function(x) {
  task.int <- as.integer(factor(x$task))
  diff.int <- as.integer(factor(x$difficulty))
  diff.int[is.na(diff.int)] <- 1
  paste0(task.int, ".", diff.int)}))

### construct correlation matrix for two tasks with four different difficulties where the
### correlation is 0.4 for different difficulties of the same task, 0.7 for the same
### difficulty of different tasks, and 0.28 for different difficulties of different tasks
R <- matrix(0.4, nrow=8, ncol=8)
R[5:8,1:4] <- R[1:4,5:8] <- 0.28
diag(R[1:4,5:8]) <- 0.7
diag(R[5:8,1:4]) <- 0.7
diag(R) <- 1
rownames(R) <- colnames(R) <- paste0(rep(1:2, each=4), ".", 1:4)
R

### construct an approximate V matrix accounting for the use of shared groups and
### for correlations among tasks/difficulties as specified in the R matrix above
V <- vcalc(vi, cluster=study, grp1=group1, grp2=group2, w1=n_sz, w2=n_hc,
  obs=task.diff, rho=R, data=dat)
Vs <- blsplit(V, dat$study)
cov2cor(Vs[[3]]) # study with multiple SZ groups and a single HC group
cov2cor(Vs[[6]]) # study with two task types and multiple difficulties
cov2cor(Vs[[12]]) # study with multiple difficulties for the same task
cov2cor(Vs[[24]]) # study with separate rows for males and females
cov2cor(Vs[[29]]) # study with separate rows for three genotypes

#####

```

Description

Function to extract various types of variance-covariance matrices from objects of class "rma". By default, the variance-covariance matrix of the fixed effects is returned.

Usage

```
## S3 method for class 'rma'
vcov(object, type="fixed", ...)
```

Arguments

object	an object of class "rma".
type	character string to specify the type of variance-covariance matrix to return: type="fixed" returns the variance-covariance matrix of the fixed effects (the default), type="obs" (or equivalently, type="marginal") returns the marginal variance-covariance matrix of the effect size estimates, type="fitted" returns the variance-covariance matrix of the fitted values, type="resid" returns the variance-covariance matrix of the residuals.
...	other arguments.

Details

Note that type="obs" currently only works for object of class "rma.uni" and "rma.mv".

For objects of class "rma.uni", the marginal variance-covariance matrix of the effect size estimates is a diagonal matrix with $\hat{\tau}^2 + v_i$ along the diagonal, where $\hat{\tau}^2$ is the estimated amount of (residual) heterogeneity (set to 0 in equal-effects models) and v_i is the sampling variance of the i th study.

For objects of class "rma.mv", the structure can be more complex and depends on the random effects included in the model.

Value

A matrix corresponding to the requested variance-covariance matrix.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which the various types of variance-covariance matrices can be extracted.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### var-cov matrix of the fixed effects (i.e., the model coefficients)
vcov(res)

### marginal var-cov matrix of the observed log risk ratios
round(vcov(res, type="marginal"), 3)

### var-cov matrix of the fitted values
round(vcov(res, type="fitted"), 3)

### var-cov matrix of the residuals
round(vcov(res, type="resid"), 3)
```

vec2mat

Convert a Vector into a Square Matrix

Description

Function to convert a vector into a square matrix by filling up the lower triangular part of the matrix.

Usage

```
vec2mat(x, diag=FALSE, corr=!diag, dimnames)
```

Arguments

x	a vector of the correct length.
diag	logical to specify whether the vector also contains the diagonal values of the lower triangular part of the matrix (the default is FALSE).
corr	logical to specify whether the diagonal of the matrix should be replaced with 1's (the default is to do this when diag=FALSE).
dimnames	optional vector of the correct length with the dimension names of the matrix.

Details

The values in x are filled into the lower triangular part of a square matrix with the appropriate dimensions (which are determined based on the length of x). If diag=TRUE, then x is assumed to also contain the diagonal values of the lower triangular part of the matrix. If corr=TRUE, then the diagonal of the matrix is replaced with 1's.

Value

A matrix.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

Examples

```
vec2mat(1:6, corr=FALSE)
vec2mat(seq(0.2, 0.7, by=0.1), corr=TRUE)
vec2mat(1:10, diag=TRUE)
vec2mat(1:6, corr=FALSE, dimnames=c("A","B","C","D"))
```

vif

*Variance Inflation Factors for 'rma' Objects***Description**

Function to compute (generalized) variance inflation factors (VIFs) for objects of class "rma".

Usage

```
vif(x, ...)

## S3 method for class 'rma'
vif(x, btt, att, table=FALSE, reestimate=FALSE, sim=FALSE, progbar=TRUE,
    seed=NULL, parallel="no", ncpus=1, cl, digits, ...)

## S3 method for class 'vif.rma'
print(x, digits=x$digits, ...)
```

Arguments

<code>x</code>	an object of class "rma" (for <code>vif</code>) or "vif.rma" (for <code>print</code>).
<code>btt</code>	optional vector of indices (or list thereof) to specify a set of coefficients for which a generalized variance inflation factor (GVIF) should be computed. Can also be a string to <code>grep</code> for.
<code>att</code>	optional vector of indices (or list thereof) to specify a set of scale coefficients for which a generalized variance inflation factor (GVIF) should be computed. Can also be a string to <code>grep</code> for. Only relevant for location-scale models (see rma.uni).
<code>table</code>	logical to specify whether the VIFs should be added to the model coefficient table (the default is <code>FALSE</code>). Only relevant when <code>btt</code> (or <code>att</code>) is not specified.
<code>reestimate</code>	logical to specify whether the model should be reestimated when removing moderator variables from the model for computing a (G)VIF (the default is <code>FALSE</code>).

<code>sim</code>	logical to specify whether the distribution of each (G)VIF under independence should be simulated (the default is FALSE). Can also be an integer to specify how many values to simulate (when <code>sim=TRUE</code> , the default is 1000).
<code>progbar</code>	logical to specify whether a progress bar should be shown when <code>sim=TRUE</code> (the default is TRUE).
<code>seed</code>	optional value to specify the seed of the random number generator when <code>sim=TRUE</code> (for reproducibility).
<code>parallel</code>	character string to specify whether parallel processing should be used (the default is "no"). For parallel processing, set to either "snow" or "multicore". See 'Note'.
<code>ncpus</code>	integer to specify the number of processes to use in the parallel processing.
<code>cl</code>	optional cluster to use if <code>parallel="snow"</code> . If unspecified, a cluster on the local machine is created for the duration of the call.
<code>digits</code>	optional integer to specify the number of decimal places to which the printed results should be rounded. If unspecified, the default is to take the value from the object.
<code>...</code>	other arguments.

Details

The function computes (generalized) variance inflation factors (VIFs) for meta-regression models. Hence, the model specified via argument `x` must include moderator variables (and more than one for this to be useful, as the VIF for a model with a single moderator variable will always be equal to 1).

VIFs for Individual Coefficients:

By default (i.e., if `btt` is not specified), VIFs are computed for the individual model coefficients. Let b_j denote the estimate of the j th model coefficient of a particular meta-regression model and $\text{Var}[b_j]$ its variance (i.e., the corresponding diagonal element from the matrix obtained with the `vcov` function). Moreover, let b'_j denote the estimate of the same model coefficient if the other moderator variables in the model had *not* been included in the model and $\text{Var}[b'_j]$ the corresponding variance. Then the VIF for the model coefficient is given by

$$\text{VIF}[b_j] = \frac{\text{Var}[b_j]}{\text{Var}[b'_j]},$$

which indicates the inflation in the variance of the estimated model coefficient due to potential collinearity of the j th moderator variable with the other moderator variables in the model. Taking the square root of a VIF gives the corresponding standard error inflation factor (SIF).

GVIFs for Sets of Coefficients:

If the model includes factors (coded in terms of multiple dummy variables) or other sets of moderator variables that belong together (e.g., for polynomials or cubic splines), one may want to examine how much the variance in all of the coefficients in the set is jointly impacted by collinearity with the other moderator variables in the model. For this, we can compute a generalized variance inflation factor (GVIF) (Fox & Monette, 1992) by setting the `btt` argument equal to the indices of those coefficients for which the GVIF should be computed. The square root of a GVIF indicates

the inflation in the confidence ellipse/(hyper)ellipsoid for the set of coefficients corresponding to the set due to collinearity. However, to make this value more directly comparable to SIFs (based on single coefficients), the function computes the generalized standard error inflation factor (GSIF) by raising the GVIF to the power of $1/(2m)$ (where m denotes the number of coefficients in the set). One can also specify a list of indices/strings, in which case GVIFs/GSIFs of all list elements will be computed. See ‘Examples’.

For location-scale models fitted with the `rma.uni` function, one can use the `att` argument in an analogous manner to specify the indices of the scale coefficients for which a GVIF should be computed.

Re-Estimating the Model:

The way the VIF is typically computed for a particular model coefficient (or a set thereof for a GVIF) makes use of some clever linear algebra to avoid having to re-estimate the model when removing the other moderator variables from the model. This speeds up the computations considerably. However, this assumes that the other moderator variables do not impact other aspects of the model, in particular the amount of residual heterogeneity (or, more generally, any variance/correlation components in a more complex model, such as those that can be fitted with the `rma.mv` function).

For a more accurate (but slower) computation of each (G)VIF, one can set `reestimate=TRUE`, in which case the model is refitted to account for the impact that removal of the other moderator variables has on all aspects of the model. Note that refitting may fail, in which case the corresponding (G)VIF will be missing.

Interpreting the Size of (G)VIFs:

A large VIF value suggests that the precision with which we can estimate a particular model coefficient (or a set thereof for a GVIF) is negatively impacted by multicollinearity among the moderator variables. However, there is no specific cutoff for determining whether a particular (G)VIF is ‘large’. Sometimes, values such as 5 or 10 have been suggested as rules of thumb, but such cutoffs are essentially arbitrary.

Simulating the Distribution of (G)VIFs Under Independence:

As a more principled approach, we can simulate the distribution of a particular (G)VIF under independence and then examine how extreme the actually observed (G)VIF value is under this distribution. The distribution is simulated by randomly reshuffling the columns of the model matrix (to break any dependence between the moderators) and recomputing the (G)VIF. When setting `sim=TRUE`, this is done 1000 times (but one can also set `sim` to an integer to specify how many (G)VIF values should be simulated).

The way the model matrix is reshuffled depends on how the model was fitted. When the model was specified as a formula via the `mods` argument and the data was supplied via the `data` argument, then each column of the data frame specified via `data` is reshuffled and the formula is evaluated within the reshuffled data (creating the corresponding reshuffled model matrix). This way, factor/character variables are properly reshuffled and derived terms (e.g., interactions, polynomials, splines) are correctly constructed. This is the recommended approach.

On the other hand, if the model matrix was directly supplied to the `mods` argument, then each column of the matrix is directly reshuffled. This is not recommended, since this approach cannot account for any inherent relationships between variables in the model matrix (e.g., an interaction term is the product of two variables and should not be reshuffled by itself).

Once the distribution of a (G)VIF under independence has been simulated, the proportion of simulated values that are smaller than the actually observed (G)VIF value is computed. If the proportion is close to 1, then this indicates that the actually observed (G)VIF value is extreme.

The general principle underlying the simulation approach is the same as that underlying Horn's parallel analysis (1965) for determining the number of components / factors to keep in a principal component / factor analysis.

Value

An object of class `"vif.rma"`. The object is a list containing the following components:

<code>vif</code>	a list of data frames with the (G)VIFs and (G)SIFs and some additional information.
<code>vifs</code>	a vector with the (G)VIFs.
<code>table</code>	the model coefficient table (only when <code>table=TRUE</code>).
<code>sim</code>	a matrix with the simulated (G)VIF values (only when <code>sim=TRUE</code>).
<code>prop</code>	a vector with the proportions of simulated values that are smaller than the actually observed (G)VIF values (only when <code>sim=TRUE</code>).
<code>...</code>	some additional elements/values.

When `x` was a location-scale model object and (G)VIFs can be computed for both the location and the scale coefficients, then the object is a list with elements `beta` and `alpha`, where each element is a `"vif.rma"` object as described above.

The results are formatted and printed with the `print` function. To format the results as a data frame, one can use the `as.data.frame` function. When `sim=TRUE`, the distribution of each (G)VIF can be plotted with the `plot` function.

Note

If the model fitted involved redundant predictors that were dropped from the model, then `sim=TRUE` cannot be used. In this case, one should remove any redundancies in the model fitted before using this method.

When using `sim=TRUE`, the model needs to be refitted (by default) 1000 times. When `sim=TRUE` is combined with `reestimate=TRUE`, then this value needs to be multiplied by the total number of (G)VIF values that are computed by the function. Hence, the combination of `sim=TRUE` with `reestimate=TRUE` is computationally expensive, especially for more complex models where model fitting can be slow.

When refitting the model fails, the simulated (G)VIF value(s) will be missing. It can also happen that one or multiple model coefficients become inestimable due to redundancies in the model matrix after the reshuffling. In this case, the corresponding simulated (G)VIF value(s) will be set to `Inf` (as that is the value of (G)VIFs in the limit as we approach perfect multicollinearity).

On machines with multiple cores, one can try to speed things up by delegating the model fitting to separate worker processes, that is, by setting `parallel="snow"` or `parallel="multicore"` and `ncpus` to some value larger than 1. Parallel processing makes use of the `parallel` package, using the `makePSOCKcluster` and `parLapply` functions when `parallel="snow"` or using `mclapply` when `parallel="multicore"` (the latter only works on Unix/Linux-alikes).

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

- Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics*. New York: Wiley.
- Fox, J., & Monette, G. (1992). Generalized collinearity diagnostics. *Journal of the American Statistical Association*, **87**(417), 178–183. <https://doi.org/10.2307/2290467>
- Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, **30**(2), 179–185. <https://doi.org/10.1007/BF02289447>
- Stewart, G. W. (1987). Collinearity and least squares regression. *Statistical Science*, **2**(1), 68–84. <https://doi.org/10.1214/ss/1177013439>
- Wax, Y. (1992). Collinearity diagnosis for a relative risk regression-analysis: An application to assessment of diet cancer relationship in epidemiologic studies. *Statistics in Medicine*, **11**(10), 1273–1287. <https://doi.org/10.1002/sim.4780111003>
- Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>
- Viechtbauer, W., & López-López, J. A. (2022). Location-scale models for meta-analysis. *Research Synthesis Methods*, **13**(6), 697–715. <https://doi.org/10.1002/jrsm.1562>

See Also

[rma.uni](#), [rma.glmm](#), and [rma.mv](#) for functions to fit models for which variance inflation factors can be computed.

[plot](#) for the plot method and [as.data.frame](#) for the method to format the results as a data frame.

Examples

```
### copy data from Bangert-Drowns et al. (2004) into 'dat'
dat <- dat.bangertdrowns2004

### fit mixed-effects meta-regression model
res <- rma(yi, vi, mods = ~ length + wic + feedback + info + pers + imag + meta, data=dat)

### get variance inflation factors
vif(res)

### use the simulation approach to analyze the size of the VIFs
## Not run:
vif(res, sim=TRUE, seed=1234)

## End(Not run)

### get variance inflation factors using the re-estimation approach
vif(res, reestimate=TRUE)

### show that VIFs are not influenced by scaling of the predictors
u <- scale # to standardize the predictors
```

```

res <- rma(yi, vi, mods = ~ u(length) + u(wic) + u(feedback) + u(info) +
          u(pers) + u(imag) + u(meta), data=dat)
vif(res, reestimate=TRUE)

### get full table
vif(res, reestimate=TRUE, table=TRUE)

#####

### an example where the VIFs are close to 1, but actually reflect considerable
### multicollinearity as can be seen based on the simulation approach
dat <- dat.mcdaniel1994
dat <- escalc(measure="ZCOR", ri=ri, ni=ni, data=dat)
res <- rma(yi, vi, mods = ~ factor(type) + factor(struct), data=dat)
res
vif(res)

### use the simulation approach to analyze the size of the VIFs
## Not run:
vifs <- vif(res, sim=TRUE, seed=1234)
vifs
plot(vifs, lwd=c(2,2), breaks=seq(1,2,by=0.0015), xlim=c(1,1.08))

## End(Not run)

### an example for a location-scale model
res <- rma(yi, vi, mods = ~ factor(type) + factor(struct),
          scale = ~ factor(type) + factor(struct), data=dat)
res
vif(res)

#####

### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit meta-regression model where one predictor (alloc) is a three-level factor
res <- rma(yi, vi, mods = ~ ablat + alloc + year, data=dat)

### get variance inflation factors for all individual coefficients
vif(res, table=TRUE)

### generalized variance inflation factor for the 'alloc' factor
vif(res, btt=3:4)

### can also specify a string to grep for
vif(res, btt="alloc")

### can also specify a list for the 'btt' argument (and use the simulation approach)
## Not run:
vif(res, btt=list(2,3:4,5), sim=TRUE, seed=1234)

## End(Not run)

```

Description

Functions to compute the weights given to the estimates during the model fitting for objects of class "rma.uni", "rma.mh", "rma.peto", and "rma.mv".

Usage

```
## S3 method for class 'rma.uni'
weights(object, type="diagonal", ...)
## S3 method for class 'rma.mh'
weights(object, type="diagonal", ...)
## S3 method for class 'rma.peto'
weights(object, type="diagonal", ...)
## S3 method for class 'rma.glmm'
weights(object, ...)
## S3 method for class 'rma.mv'
weights(object, type="diagonal", ...)
```

Arguments

object	an object of class "rma.uni", "rma.mh", "rma.peto", or "rma.mv". The method is not (yet) implemented for objects of class "rma.glmm".
type	character string to specify whether to return only the diagonal of the weight matrix ("diagonal") or the entire weight matrix ("matrix"). For "rma.mv", this can also be "rowsum" for 'row-sum weights' (for intercept-only models).
...	other arguments.

Value

Either a vector with the diagonal elements of the weight matrix or the entire weight matrix. When only the diagonal elements are returned, they are given in % (and they add up to 100%).

When the entire weight matrix is requested, this is always a diagonal matrix for objects of class "rma.uni", "rma.mh", "rma.peto".

For "rma.mv", the structure of the weight matrix depends on the model fitted (i.e., the random effects included and the variance-covariance matrix of the sampling errors) but is often more complex and not just diagonal.

For intercept-only "rma.mv" models, one can also take the sum over the rows in the weight matrix, which are actually the weights assigned to the estimates when estimating the model intercept. These weights can be obtained with type="rowsum" (as with type="diagonal", they are also given in %). See [here](#) for a discussion of this.

Author(s)

Wolfgang Viechtbauer (<wvb@metafor-project.org>, <https://www.metafor-project.org>).

References

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**(3), 1–48. <https://doi.org/10.18637/jss.v036.i03>

Viechtbauer, W. (2021). Model checking in meta-analysis. In C. H. Schmid, T. Stijnen, & I. R. White (Eds.), *Handbook of meta-analysis* (pp. 219–254). Boca Raton, FL: CRC Press. <https://doi.org/10.1201/9781315>

See Also

[rma.uni](#), [rma.mh](#), [rma.peto](#), and [rma.mv](#) for functions to fit models for which model fitting weights can be extracted.

[influence.rma.uni](#) and [influence.rma.mv](#) for other model diagnostics.

Examples

```
### calculate log risk ratios and corresponding sampling variances
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)

### fit mixed-effects model with absolute latitude and publication year as moderators
res <- rma(yi, vi, mods = ~ ablat + year, data=dat)

### extract the model fitting weights (in %)
weights(res)

### extract the weight matrix
round(weights(res, type="matrix"), 4)
```

Index

- * **aplot**
 - addpoly, 9
 - addpoly.default, 10
 - addpoly.predict.rma, 12
 - addpoly.rma, 14
- * **datagen**
 - escalc, 83
 - rcalc, 256
 - simulate.rma, 349
 - vcalc, 373
- * **documentation**
 - misc-models, 178
 - misc-options, 183
 - misc-recs, 189
- * **hplot**
 - baujat, 27
 - forest, 111
 - forest.cumul.rma, 112
 - forest.default, 116
 - forest.rma, 122
 - funnel, 141
 - labbe, 161
 - llplot, 167
 - plot.cumul.rma, 200
 - plot.gosh.rma, 202
 - plot.infl.rma.uni, 204
 - plot.permutest.rma.uni, 207
 - plot.rma, 210
 - plot.rma.uni.selmodel, 211
 - plot.vif.rma, 214
 - profile.rma, 241
 - qqnorm.rma, 246
 - radial, 249
 - regplot, 260
- * **htest**
 - fsn, 137
 - hc, 150
 - hettest, 152
 - ranktest, 254
 - regtest, 266
 - robust, 325
 - tes, 351
- * **manip**
 - bldiag, 29
 - blsplit, 31
 - cmicalc, 34
 - contrmat, 48
 - conv.2x2, 49
 - conv.delta, 55
 - conv.fivenum, 60
 - conv.wald, 66
 - dfround, 79
 - emmprep, 80
 - formatters, 133
 - mfopt, 177
 - replmiss, 271
 - to.long, 354
 - to.table, 357
 - to.wide, 360
 - transf, 362
 - vec2mat, 382
- * **methods**
 - cumul, 72
 - gosh, 147
 - leave1out, 164
 - reporter, 272
- * **misc**
 - misc-options, 183
 - misc-recs, 189
- * **models**
 - aggregate.escalc, 17
 - anova.rma, 21
 - blup, 32
 - coef.deltamethod, 36
 - coef.matreg, 37
 - coef.permutest.rma.uni, 39
 - coef.rma, 40
 - confint.rma, 42

- deltamethod, 75
- fitstats, 108
- fitted.rma, 110
- formula.rma, 136
- influence.rma.mv, 155
- influence.rma.uni, 158
- matreg, 169
- misc-models, 178
- model.matrix.rma, 192
- pairmat, 193
- permutest, 195
- predict.matreg, 216
- predict.rma, 218
- ranef, 252
- residuals.rma, 274
- rma.glmm, 278
- rma.mh, 288
- rma.mv, 293
- rma.peto, 307
- rma.uni, 311
- se, 330
- selmodel, 331
- trimfill, 369
- update.rma, 372
- vcov.rma, 380
- vif, 383
- weights.rma, 389
- * package**
 - metafor-package, 4
- * print**
 - methods.vcovmat, 176
 - print.anova.rma, 224
 - print.confint.rma, 225
 - print.deltamethod, 226
 - print.escalc, 227
 - print.fsn, 229
 - print.gosh.rma, 230
 - print.hc.rma.uni, 231
 - print.list.rma, 232
 - print.matreg, 233
 - print.permutest.rma.uni, 234
 - print.ranktest, 235
 - print.regtest, 236
 - print.rma, 237
- * utilities**
 - metafor.news, 175
 - .Random.seed, 349, 350
 - [.vcovmat (methods.vcovmat), 176
 - addpoly, 9, 119, 121, 128, 131, 222
 - addpoly.default, 9, 10
 - addpoly.predict.rma, 9, 12
 - addpoly.rma, 9, 14
 - aggregate, 102
 - aggregate (aggregate.escalc), 17
 - aggregate.escalc, 17
 - AIC, 37, 109, 171, 292, 303, 310, 319
 - AIC (fitstats), 108
 - AIC.matreg (coef.matreg), 37
 - anova, 109, 190, 193, 194, 224, 225, 240, 302, 303, 318, 327
 - anova (anova.rma), 21
 - anova.rma, 21
 - as.data.frame, 24, 25, 32, 45, 74, 160, 166, 217, 221, 253, 276, 386, 387
 - auglag, 321
 - baujat, 27, 292, 310, 318
 - BBoptim, 285, 304, 320, 342
 - bc.mean.sd, 62
 - BIC, 171, 292, 303, 310, 319
 - BIC (fitstats), 108
 - BIC.matreg (coef.matreg), 37
 - bldiag, 29, 31
 - blsplit, 30, 31
 - BLUE_s, 63
 - blup, 32, 111, 222, 253, 254, 276, 318
 - bobyqa, 284, 304, 320, 342
 - cmicalc, 34
 - coef, 77, 171, 198, 199, 219, 292, 303, 310, 319
 - coef (coef.rma), 40
 - coef.deltamethod, 36
 - coef.matreg, 37
 - coef.permutest.rma.uni, 39
 - coef.rma, 40
 - confint, 37, 42, 171, 187, 225, 226, 245, 302, 303, 318, 342
 - confint (confint.rma), 42
 - confint.matreg (coef.matreg), 37
 - confint.rma, 42
 - constrOptim, 321
 - constrOptim.nl, 321
 - contrmat, 48, 362
 - conv.2x2, 49, 91, 106
 - conv.delta, 55, 77, 106
 - conv.fivenum, 60, 86, 106

- conv.wald, [66](#), [91](#), [106](#)
- cooks.distance (influence.rma.uni), [158](#)
- cooks.distance.rma.mv
(influence.rma.mv), [155](#)
- cor.test, [255](#)
- cov2cor, [300](#)
- cumul, [72](#), [112–115](#), [187](#), [200–202](#), [292](#), [310](#),
[318](#)
- dat.aloe2013, [100](#)
- dat.assink2016, [190](#), [298](#)
- dat.bangertdrowns2004, [179](#)
- dat.bcg, [91](#)
- dat.berkey1998, [298](#)
- dat.bonett2010, [99](#)
- dat.bornmann2007, [297](#)
- dat.bourassa1996, [94](#)
- dat.collins1985a, [91](#), [287](#), [310](#)
- dat.collins1985b, [91](#), [310](#)
- dat.craft2003, [258](#), [298](#)
- dat.crede2010, [297](#)
- dat.crisafulli2020, [96](#)
- dat.curtis1998, [88](#)
- dat.dagostino1998, [298](#)
- dat.debruin2009, [96](#)
- dat.egger2001, [91](#)
- dat.fine1993, [298](#)
- dat.frank2008, [101](#)
- dat.gibson2002, [88](#), [92](#)
- dat.hannum2020, [96](#)
- dat.hart1999, [92](#)
- dat.hasselblad1998, [49](#), [298](#), [362](#)
- dat.hine1989, [91](#)
- dat.ishak2007, [298](#)
- dat.kalaian1996, [298](#)
- dat.kearon1998, [298](#)
- dat.knapp2017, [190](#), [298](#)
- dat.konstantopoulos2011, [297](#)
- dat.laopaiboon2015, [91](#)
- dat.lee2004, [91](#)
- dat.li2007, [91](#)
- dat.lim2014, [300](#)
- dat.linde2005, [91](#)
- dat.lopez2019, [49](#), [298](#), [362](#)
- dat.maire2019, [299](#)
- dat.mccurdy2020, [298](#)
- dat.mcdaniel1994, [93](#)
- dat.molloy2014, [93](#)
- dat.moura2021, [188](#), [300](#)
- dat.nielweise2007, [91](#), [287](#)
- dat.nielweise2008, [92](#), [287](#)
- dat.normand1999, [88](#)
- dat.obrien2003, [297](#), [299](#), [362](#)
- dat.pagliaro1992, [298](#), [362](#)
- dat.pritz1997, [96](#), [287](#)
- dat.senn2013, [49](#), [298](#), [362](#)
- dat.tannersmith2016, [190](#), [298](#)
- dat.yusuf1985, [91](#), [310](#)
- deltamethod, [36](#), [57](#), [75](#), [227](#)
- density, [203](#), [208](#), [214](#)
- derivative, [76](#)
- deviance, [171](#), [292](#), [303](#), [310](#), [319](#)
- deviance (fitstats), [108](#)
- df.residual (fitstats), [108](#)
- dfbetas (influence.rma.uni), [158](#)
- dfbetas.rma.mv (influence.rma.mv), [155](#)
- dfround, [79](#)
- emmeans, [80](#)
- emmprep, [80](#)
- escalc, [5](#), [8](#), [18](#), [20](#), [35](#), [51](#), [53](#), [56](#), [57](#), [61](#), [65](#),
[67](#), [71](#), [80](#), [83](#), [145](#), [162](#), [163](#), [168](#),
[227](#), [229](#), [268](#), [278–280](#), [283](#),
[289–291](#), [296](#), [307–309](#), [311–314](#),
[343](#), [344](#), [355–359](#), [378](#)
- factor, [282](#), [301](#), [315](#)
- family, [286](#)
- fitstats, [108](#), [284](#), [292](#), [303](#), [309](#), [318](#)
- fitted, [33](#), [193](#), [222](#), [253](#), [254](#), [303](#), [318](#)
- fitted (fitted.rma), [110](#)
- fitted.rma, [110](#)
- fmltp (formatters), [133](#)
- fmltp2 (formatters), [133](#)
- fmltt (formatters), [133](#)
- fmltx (formatters), [133](#)
- forest, [9](#), [11–16](#), [74](#), [111](#), [115](#), [121](#), [131](#), [211](#),
[273](#), [292](#), [310](#), [318](#)
- forest.cumul.rma, [112](#), [112](#)
- forest.default, [112](#), [114](#), [116](#), [131](#)
- forest.rma, [11](#), [112](#), [114](#), [121](#), [122](#)
- formatC, [79](#), [134](#), [135](#)
- formatters, [133](#)
- formula, [279](#), [281](#), [294](#), [296](#), [312](#), [315](#)
- formula (formula.rma), [136](#)
- formula.rma, [136](#)
- fsn, [137](#), [229](#), [230](#), [255](#), [270](#), [318](#), [354](#), [371](#)

- funnel, 56, 70, 141, 211, 273, 292, 310, 318, 370, 371
- getmfopt (mfopt), 177
- glm, 284, 285
- glmer, 284, 285
- glmmTMB, 285
- gosh, 147, 202, 204, 230, 231
- grad, 56
- grep, 21, 22, 193, 194, 196, 280, 282, 295, 301, 313, 315, 383
- hatvalues, 303, 319
- hatvalues (influence.rma.uni), 158
- hatvalues.rma.mv (influence.rma.mv), 155
- hc, 150, 231, 232, 318
- here, 6, 10, 15, 32, 37, 43, 67, 76, 85, 117, 123, 142, 170, 177, 178, 216, 219, 228, 238, 239, 247, 252, 261, 280, 290, 294–296, 298, 302, 308, 313, 314, 316, 326, 375
- hessian, 284, 285, 304, 321, 343
- hettest, 152
- hist, 203, 207, 208, 214
- hjk, 284, 304, 320, 342
- influence, 29, 149, 158, 205, 206, 303, 318
- influence (influence.rma.uni), 158
- influence.rma.mv, 155, 276, 278, 390
- influence.rma.uni, 158, 276, 278, 390
- integrate, 284, 285, 342
- interactive, 187
- labbe, 161, 292, 310, 318
- lbfgsb3c, 284, 304, 320, 342
- leave1out, 164, 187, 292, 310, 318
- legend, 144, 163, 209, 263
- llplot, 167
- logLik, 171, 292, 303, 310, 319
- logLik (fitstats), 108
- logLik.matreg (coef.matreg), 37
- mad, 319
- mads, 284, 304, 320, 342
- make.names, 48
- makePSOCKcluster, 149, 156, 244, 277, 386
- matreg, 38, 169, 217, 234, 302
- mclapply, 149, 156, 244, 277, 386
- metafor (metafor-package), 4
- metafor-package, 4, 175, 278, 288, 293, 307, 311, 314
- metafor.news, 175
- methods.escalc, 102
- methods.list.rma, 232
- methods.vcovmat, 176, 377
- mfopt, 177, 184, 186, 187
- misc-models, 178
- misc-options, 183
- misc-recs, 189
- misc_models (misc-models), 178
- misc_options (misc-options), 183
- misc_rec (misc-recs), 189
- mixed_model, 285
- mln.mean.sd, 62
- model.matrix, 219
- model.matrix (model.matrix.rma), 192
- model.matrix.rma, 192
- nearPD, 170, 172, 304, 374, 378
- newuoa, 284, 304, 320, 342
- nlm, 284, 304, 320, 342
- nlminb, 284, 304, 320, 342
- nloptr, 284, 285, 304, 320, 321, 342
- nmk, 284, 304, 320, 342
- nobs (fitstats), 108
- optim, 284, 285, 304, 320, 342
- optimParallel, 285, 304, 321, 342
- options, 121, 130, 177, 226, 233, 234, 238
- p.adjust, 22
- pairmat, 23, 193
- pairs, 203
- par, 143, 205, 251
- parallel, 149, 156, 188, 244, 277, 386
- parLapply, 149, 156, 188, 244, 277, 386
- parLapplyLB, 188
- permutest, 40, 189, 195, 207, 209, 234, 235, 318
- plot, 74, 149, 160, 198, 199, 292, 310, 318, 342, 386, 387
- plot.cumul.rma, 200
- plot.gosh.rma, 202
- plot.infl.rma.uni, 204
- plot.permutest.rma.uni, 207
- plot.profile.rma (profile.rma), 241
- plot.rma, 210
- plot.rma.uni.selmodel, 211

- plot.vif.rma, 214
- plotmath, 135
- pnorm, 69
- points, 28, 114, 118, 124, 143, 162, 201, 202, 205, 242, 247, 250, 261
- points.regplot (regplot), 260
- predict, 9, 23, 33, 111, 126, 190, 193, 194, 253, 254, 260, 262, 263, 303, 318, 327, 367
- predict (predict.rma), 218
- predict.emmGrid, 81
- predict.matreg, 38, 216
- predict.rma, 218
- print, 19, 24, 25, 32, 44, 45, 74, 77, 102, 106, 140, 149, 151, 166, 171, 198, 199, 217, 221, 253, 255, 268, 276, 284, 292, 303, 309, 317, 318, 342, 370
- print (print.rma), 237
- print.anova.rma, 223
- print.confint.matreg (coef.matreg), 37
- print.confint.rma, 225
- print.deltamethod, 226
- print.escalc, 227
- print.fsn, 229
- print.gosh.rma, 230
- print.hc.rma.uni, 231
- print.hetest (hettest), 152
- print.infl.rma.uni (influence.rma.uni), 158
- print.list.anova.rma (print.anova.rma), 224
- print.list.confint.rma (print.confint.rma), 225
- print.list.rma, 232
- print.matreg, 233
- print.permutest.rma.uni, 234
- print.profile.rma (profile.rma), 241
- print.ranktest, 235
- print.regtest, 236
- print.rma, 237
- print.rma.mv, 327
- print.rma.uni, 327
- print.summary.matreg (print.matreg), 233
- print.table, 176
- print.tes (tes), 351
- print.vcovmat (methods.vcovmat), 176
- print.vif.rma (vif), 383
- profile, 44, 47, 187, 188, 191, 302, 303, 305, 318, 342
- profile (profile.rma), 241
- profile.rma, 241
- qdr, 80
- qe.mean.sd, 62
- qnorm, 69
- qqnorm, 211, 292, 310, 318
- qqnorm (qqnorm.rma), 246
- qqnorm.rma, 246
- radial, 211, 249, 292, 310, 318
- ranef, 33, 252, 303, 318
- ranktest, 140, 146, 235, 236, 254, 270, 318, 354, 371
- rcalc, 93, 107, 172, 177, 190, 256, 296, 378
- Rcgmin, 285, 304, 321, 342
- regplot, 260, 303, 318
- regtest, 140, 146, 236, 237, 255, 266, 318, 354, 369, 371
- replmiss, 271
- reporter, 272
- resid (residuals.rma), 274
- residuals, 144, 154, 247, 292, 303, 310, 318
- residuals (residuals.rma), 274
- residuals.rma, 274
- rma, 5
- rma (rma.uni), 311
- rma.glmm, 7–9, 41, 46, 107, 109, 131, 137, 146, 162, 164, 169, 187–190, 193, 194, 239, 241, 251, 264, 278, 278, 293, 306, 310, 323, 331, 372, 381, 387
- rma.mh, 6, 8, 9, 28, 29, 41, 46, 107, 109, 112, 131, 146, 148, 149, 162, 164, 166, 187, 188, 241, 248, 251, 274, 278, 287, 288, 306, 310, 323, 331, 372, 381, 390
- rma.mv, 7–9, 18, 24, 25, 30, 41, 44, 46, 80, 81, 107, 109, 125, 128, 131, 137, 146, 156, 172, 187–191, 193, 194, 218–220, 241, 243, 245, 251, 254, 257, 258, 264, 274, 276–278, 287, 293, 293, 310, 323, 329, 331, 349, 350, 372, 373, 375, 378, 381, 385, 387, 390
- rma.peto, 6, 8, 9, 28, 29, 41, 46, 107, 109, 112, 131, 146, 148, 149, 162, 164, 166, 187, 188, 241, 248, 251, 274,

- 278, 287, 293, 306, 307, 323, 331, 372, 381, 390
- rma.uni, 5, 8, 9, 21–23, 25, 28, 29, 33, 40, 41, 43, 44, 46, 89, 107, 109, 112, 128, 131, 137–139, 145, 146, 148, 149, 151–153, 162, 164, 166, 169, 179, 180, 182, 187–191, 193, 194, 197, 199, 207, 209, 219–221, 240, 241, 243, 245, 248, 251, 254, 264, 274, 276, 278, 282, 287, 293, 301, 306, 310, 311, 329, 331, 332, 341, 343, 344, 349, 350, 370, 372, 374, 381, 383, 385, 387, 390
- robust, 190, 303, 318, 325
- rstandard, 33, 292, 303, 310, 318
- rstandard (residuals.rma), 274
- rstudent, 157, 159, 160, 187, 292, 303, 310, 318
- rstudent (residuals.rma), 274
- Rvmmmin, 285, 304, 321, 342
- se, 38, 171, 292, 303, 310, 319, 330
- selmodel, 44, 46, 140, 187, 191, 212, 213, 243, 245, 255, 270, 318, 331, 354, 371
- set.seed, 63
- setmfopt (mfopt), 177
- sigma, 171
- sigma.matreg (coef.matreg), 37
- simulate (simulate.rma), 349
- simulate.rma, 349
- solnp, 321
- subplex, 284, 304, 320, 342
- summary, 102, 106, 284, 292, 303, 309, 318
- summary (print.rma), 237
- summary.emmGrid, 81
- summary.escalc (print.escalc), 227
- summary.matreg (print.matreg), 233
- suppressPackageStartupMessages, 177
- supsmu, 247
- tempdir, 272
- tes, 140, 255, 270, 318, 351, 371
- text, 121, 130, 135
- to.long, 354, 359
- to.table, 357, 357
- to.wide, 49, 360
- transf, 11, 13, 15, 32, 43, 73, 114, 118, 119, 124, 126, 142, 144, 150, 162, 163, 165, 201, 216, 219, 228, 250, 252, 261, 362
- trimfill, 140, 143, 144, 146, 255, 270, 318, 354, 369
- ucminf, 284, 304, 320, 342
- unirroot, 44, 151, 196, 198, 199, 319
- uobyqa, 284, 304, 320, 342
- update (update.rma), 372
- update.default, 372
- update.rma, 372
- vcalc, 19, 31, 106, 177, 190, 296, 373
- vcov, 77, 171, 292, 303, 310, 319, 384
- vcov (vcov.rma), 380
- vcov.deltamethod (coef.deltamethod), 36
- vcov.matreg (coef.matreg), 37
- vcov.rma, 380
- vec2mat, 382
- vif, 214, 215, 303, 318, 383
- weights, 128, 157, 160, 303, 319
- weights (weights.rma), 389
- weights.rma, 389