Package 'dmetar'

August 10, 2020

Title Companion R Package For The Guide 'Doing Meta-Analysis in R'

Author Mathias Harrer, Pim Cuijpers, Toshi Furukawa, David Daniel Ebert

Version 0.0.9000

Description
This package serves as the companion R package for the guide 'Doing Meta-Analysis in R' by Mathias Harrer, Pim Cuijpers, Toshi Furukawa and David Daniel Ebert. The package contains complementary functions to facilitate the conduction of meta-analyses using the {meta}, {metafor}, {netmeta} and {gemtc} packages.

URL https://github.com/MathiasHarrer/dmetar

Depends R (>= 3.6)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports ggplot2 (>= 3.2.0), stringr (>= 1.4.0), gridExtra (>= 2.3), netmeta (>= 1.0.1), meta (>= 4.9.4), metafor (>= 2.1.0), ggrepel (>= 0.8.0), grid (>= 3.5.2), poibin (>= 1.3), MuMIn (>= 1.43.6), scales (>= 1.0.0), graphics (>= 3.5.2), stats (>= 3.5.2), fpc (>= 2.1.11.1), mclust (>= 5.4.6), utils (>= 3.5.2)

Suggests gemtc, robvis, rmarkdown, knitr, devtools, testthat

VignetteBuilder knitr

R topics documented:

`direct.evidence.plot` *Plot for direct evidence proportions in a network meta-analysis using* `netmeta`

Description

This function plots relevant measures quantifying the direct evidence proportion, mean path length and aggregated minimal parallelism of a frequentist network meta-analysis model generated by `netmeta`.

Usage

```
direct.evidence.plot(x, random=FALSE, comparison.label.size=2,
numeric.label.size=3, subplot.ratio=c(5, 1.3, 1.3))
```

Arguments

`x`	An object of class `netmeta` containing the results of a network meta-analysis using the `netmeta` function.
`random`	Logical. If set to `TRUE`, results for the random-effects model are displayed. If set to `FALSE`, results for the fixed-effect model are displayed. `FALSE` by default.
`comparison.label.size`	A numeric value for the size of comparison labels to be used in the plot. Default is 2.
`numeric.label.size`	A numeric value for the label size of numeric values to be used in the plot. Default is 3.
`subplot.ratio`	A numeric vector containing three numbers. Defines the width for each of the three subplots included in the plot (from left to right). Default is `c(5,1.3,1.3)`.

Details

The function generates a plot containing three subplots displaying relevant characteristics to evaluate the reliability of effect size estimates within a network meta-analysis model.

- **Direct Evidence Proportion**. This bar chart displays the proportion of direct evidence (orange) contained in each network estimate. It is of note that both direct and indirect evidence may contribute to the violation of the assumption of consistency underlying network meta-analysis models. Nevertheless, this plot allows to distinguish comparison estimates for which direct evidence was used, and to what extent, and comparisons which had to be inferred by indirect evidence alone.
- **Minimal Parallelism**. This bar chart displays the minimum number of independent paths contributing to the effect estimate on an aggregated level. Large values of parallelism can be interpreted as supporting the robustness of the estimate.
- **Mean Path Length**. This bar chart displays the mean path length, which characterizes the degree of indirectness of an estimate. Higher mean path lengths indicate less reliable estimates, given that more similarity assumptions have to be made when serially combining direct comparisons. Following König, Krahn and Binder (2013), comparisons with mean path lengths greater than two should be interpreted with caution. This threshold is displayed as a blue vertical line in the plot.

Value

- `data`: A `data.frame` containing columns for the proportion of direct and indirect evidence of each comparison (`proportion.direct` and `proportion.indirect`), the mean path length (`meanpath`) and the minimal parallelism (`minpar`) for each comparison.
- `plot`: The generated plot (if the function output was saved to an object).

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 11.1

König J., Krahn U., Binder H. (2013): Visualizing the flow of evidence in network meta-analysis and characterizing mixed treatment comparisons. *Statistics in Medicine, 32*, 5414–29

See Also

`netmeta`, `netmeasures`

Examples

```
# Load Senn2013 data from netmeta
suppressPackageStartupMessages(library(netmeta))
data(Senn2013)

# Conduct network meta-analysis (fixed-effects model)
nma = netmeta(TE, seTE, treat1, treat2, studlab,
              data=Senn2013, sm='MD', comb.random=FALSE)

# Generate the plot
dep = direct.evidence.plot(nma, random=FALSE, comparison.label.size = 1,
numeric.label.size=1, subplot.ratio=c(3,1,1))
dep
```

`dmetar` *dmetar: Companion R package for the guide 'Doing Meta-Analysis in R'*

Description

`dmetar` serves as the companion R package for the guide Doing Meta-Analysis in R by Mathias Harrer, Pim Cuijpers, Toshi Furukawa and David Daniel Ebert.

Details

The package contains complementary functions to facilitate the conduction of meta-analyses using the **meta**, **metafor**, **netmeta** and **gemtc** packages.

Author(s)

Mathias Harrer <mathias.harrer@fau.de>, David Daniel Ebert <david.daniel.ebert@gmail.com>

See Also

Useful links:

- https://github.com/MathiasHarrer/dmetar

`eggers.test` *Perform Egger's test of the intercept*

Description

This function performs Egger's test of the intercept for funnel plot asymmetry using an object of class `meta`.

Usage

```
eggers.test(x)
```

Arguments

`x`	An object of class `meta`, generated by the `metabin`, `metagen`, `metacont`, `metacor`, `metainc`, or `metaprop` function.

Details

Performs Egger's test (Egger et al., 1997) for funnel plot asymmetry. The `metabias` function is called internally. Egger's test may lack the statistical power to detect bias when the number of studies is small. Sterne et al. (2011) recommend to perform funnel plot asymmetry tests only when $k \geq 10$. A warning is therefore printed when the number of studies in the `meta` object is $k < 10$.

Value

Returns a list containing the following elements:

- `intercept`: The intercept (bias).
- `llci`: The lower bound of the 95% intercept confidence interval.
- `ulci`: The upper bound of the 95% intercept confidence interval.
- `t`: The t-statistic for the intercept test.
- `p`: The p-value for Egger's test.
- `meta.obj`: The meta-analysis object of class `meta` originally provided to the function.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 9.1

Egger M, Smith GD, Schneider M & Minder C (1997), Bias in meta-analysis detected by a simple, graphical test. *BMJ*, 315, 629–634.

Sterne, JAC et al. (2011), Recommendations for Examining and Interpreting Funnel Plot Asymmetry in Meta-Analyses of Randomised Controlled Trials. *BMJ* 343, 1, doi: 10.1136/bmj.d4002 .

See Also

metabias

Examples

```
# Create meta-analysis results using the 'metagen' function
suppressPackageStartupMessages(library(meta))
data(ThirdWave)
m = metagen(TE, seTE, studlab = paste(Author),
    data = ThirdWave, comb.random = FALSE, hakn=TRUE)

# Plug result into 'eggers.test' function
res.et <- eggers.test(m)

# Inspect the results
summary(res.et)

# Generate a funnel plot. This calls the 'funnel' function
# in 'meta' internally; additional parameters of this function can also
# be provided (see '?meta::funnel').
plot(res.et, bg = "lightblue")
```

---

find.outliers *Find Statistical Outliers in a Meta-Analysis*

Description

Searches for statistical outliers in meta-analysis results generated by meta functions or the rma.uni in the metafor package.

Usage

```
find.outliers(x, ...)
```

Arguments

x	Either (1) an object of class meta, generated by the metabin, metagen, metacont, metacor, metainc, metarate or metaprop function; or (2) and object of class rma.uni created with the rma.uni function in metafor.
...	Additional parameters for the rma.uni or update.meta function.

Details

This function searches for outlying studies in a meta-analysis results object. Studies are defined as outliers when their 95% confidence interval lies ouside the 95% confidence interval of the pooled effect.

When outliers are found, the function automatically recalculates the meta-analysis results, using the same settings as in the object provided in x, but excluding the detected outliers.

A forest plot of the meta-analysis with outliers removed can be generated directly by plugging the output of the function into the forest function.

Value

Returns the identified outliers and the meta-analysis results when the outliers are removed.

If the provided meta-analysis object is of class `meta`, the following objects are returned if the results of the function are saved to another object:

- `out.study.fixed`: A numeric vector containing the names of the outlying studies when assuming a fixed-effect model.
- `out.study.random`: A numeric vector containing the names of the outlying studies when assuming a random-effects model. The τ^2 estimator `method.tau` is inherited from `x`.
- `m.fixed`: An object of class `meta` containing the results of the meta-analysis with outliers removed (assuming a fixed-effect model).
- `m.random`: An object of class `meta` containing the results of the meta-analysis with outliers removed (assuming a random-effects model, and using the same `method.tau` as in the original analysis).

If the provided meta-analysis object is of class `rma.uni`, the following objects are returned if the results of the function are saved to another object:

- `out.study`: A numeric vector containing the names of the outlying studies.
- `m`: An object of class `rma.uni` containing the results of the meta-analysis with outliers removed (using the same settings as in the meta-analysis object provided).

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 6.2

See Also

`influence.rma.uni`, `metainf`, `baujat`

Examples

```
suppressPackageStartupMessages(library(meta))
suppressPackageStartupMessages(library(metafor))
suppressPackageStartupMessages(library(dmetar))

# Pool with meta
m1 <- metagen(TE, seTE, data = ThirdWave,
              studlab = ThirdWave$Author, comb.fixed = FALSE)

# Pool with metafor
m2 <- rma(yi = TE, sei = seTE, data = ThirdWave,
          slab = ThirdWave$Author, method = "PM")

# Find outliers
fo1 <- find.outliers(m1)
fo2 <- find.outliers(m2)

# Show summary
```

```
summary(fo1)
summary(fo2)

## Not run:
# Make forest plot
# Pass additional arguments from meta & metafor's forest function
forest(fo1, prediction = TRUE)
forest(fo2, cex = .8, col = "lightblue")

## End(Not run)
```

`forest.find.outliers` *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'find.outliers'
forest(x, ...)
```

Arguments

`x`	An object of class `find.outliers`.
`...`	Other arguments of the `meta.forest` or `metafor`'s codeforest function. Can be used for styling the generated forest plot.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical. When both `plot` and `forest` are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The `forest/plot` function allows additional arguments of the `meta.forest` or `metafor`'s codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

forest.subgroup.analysis.mixed.effects

Print, summary and plot methods for objects created using 'dmetar' functions

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'subgroup.analysis.mixed.effects'
forest(x, ...)
```

Arguments

`x`	An object of class `subgroup.analysis.mixed.effects`.
`...`	Other arguments of the `meta.forest` or `metafor`'s codeforest function. Can be used for styling the generated forest plot.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical. When both `plot` and `forest` are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The `forest`/`plot` function allows additional arguments of the `meta.forest` or `metafor`'s codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

gosh.diagnostics

Identify studies contributing to heterogeneity patterns found in GOSH plots

Description

This function uses three unsupervised learning learning algorithms (*k*-means, DBSCAN and Gaussian Mixture Models) to identify studies contributing to the heterogeneity-effect size patterns found in GOSH (graphic display of study heterogeneity) plots.

Usage

```
gosh.diagnostics(data, km = TRUE, db = TRUE, gmm = TRUE,
                 km.params = list(centers = 3,
                                  iter.max = 10, nstart = 1,
                                  algorithm = c("Hartigan-Wong",
                                  "Lloyd", "Forgy","MacQueen"),
                                  trace = FALSE),
                 db.params = list(eps = 0.15, MinPts = 5,
                                  method = c("hybrid", "raw", "dist")),
                 gmm.params = list(G = NULL, modelNames = NULL,
                                   prior = NULL, control = emControl(),
                                   initialization = list(hcPairs = NULL,
                                   subset = NULL,
                                   noise = NULL),
                                   Vinv = NULL,
                                   warn = mclust.options("warn"),
                                   x = NULL, verbose = FALSE),
                 seed = 123,
                 verbose = TRUE)
```

Arguments

data	An object of class gosh.rma created through the gosh function.
km	Logical. Should the *k*-Means algorithm be used to identify patterns in the GOSH plot matrix? TRUE by default.
db	Logical. Should the DBSCAN algorithm be used to identify patterns in the GOSH plot matrix? TRUE by default.
gmm	Logical. Should a bivariate Gaussian Mixture Model be used to identify patterns in the GOSH plot matrix? TRUE by default.
km.params	A list containing the parameters for the *k*-Means algorithm as implemented in kmeans. Run ?kmeans for further details.
db.params	A list containing the parameters for the DBSCAN algorithm as implemented in dbscan. Run ?fpc::dbscan for further details.
gmm.params	A list containing the parameters for the Gaussian Mixture Models as implemented in mclustBIC. Run ?mclust::mclustBIC for further details.
seed	Seed used for reproducibility. Default seed is 123.
verbose	Logical. Should a progress bar be printed in the console during clustering?

Details

GOSH Plots

GOSH (*graphic display of study heterogeneity*) plots were proposed by Olkin, Dahabreh and Trikalinos (2012) as a diagnostic plot to assess effect size heterogeneity. GOSH plots facilitate the detection of both (i) outliers and (ii) distinct homogeneous subgroups within the modeled data.

Data for the plots is generated by fitting a random-effects-model with the same specifications as in the meta-analysis to all $\mathcal{P}(k), \emptyset \notin \mathcal{P}(k), \forall 2^{k-1} \leq 10^6$ possible subsets of studies in an analysis. For $|\mathcal{P}(k)| > 10^6$, 1 million subsets are randomly sampled and used for model fitting when using the gosh function.

GOSH Plot Diagnostics

Although GOSH plots allow to detect heterogeneity patterns and distinct subgroups within the data, interpretation which studies contribute to a certain subgroup or pattern is often difficult or computationally intensive. To facilitate the detection of studies responsible for specific patterns within the GOSH plots, this function randomly samples 10^4 data points from the GOSH Plot data (to speed up computation). Of the data points, only the z-transformed I^2 and effect size value is used (as other heterogeneity metrics produced for the GOSH plot data using the `gosh` function are linear combinations of I^2). To this data, three clustering algorithms are applied.

- The first algorithm is *k*-Means clustering using the algorithm by Hartigan & Wong (1979) and $m_k = 3$ cluster centers by default. The functions uses the `kmeans` implementation to perform *k*-Means clustering.
- As *k*-Means does not perform well in the presence of distinct arbitrary subclusters and noise, the function also applies **DBSCAN** (*density reachability and connectivity clustering*; Schubert et al., 2017). The hyperparameters ϵ and $MinPts$ can be tuned for each analysis to maintain a reasonable amount of granularity while not producing too many subclusters. The function uses the `dbscan` implementation to perform the DBSCAN clustering.
- Lastly, as a clustering approach using a probabilistic model, Gaussian Mixture Models (GMM; Fraley & Raftery, 2002) are integrated in the function using an internal call to the `mclustBIC` implementation. Clustering hyperparameters can be tuned by providing a list of parameters of the `mclustBIC` function in the `mclust` package.

To assess which studies predominantly contribute to a detected cluster, the function calculates the cluster imbalance of a specific study using the difference between (i) the expected share of subsets containing a specific study if the cluster makeup was purely random (viz., representative for the full sample), and the (ii) actual share of subsets containing a specific study within a cluster. Cook's distance for each study is then calculated based on a linear intercept model to determine the leverage of a specific study for each cluster makeup. Studies with a leverage value three times above the mean in any of the generated clusters (for all used clustering algorithms) are returned as potentially influential cases and the GOSH plot is redrawn highlighting these specific studies.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Fraley C. and Raftery A. E. (2002) Model-based clustering, discriminant analysis and density estimation, *Journal of the American Statistical Association*, 97/458, pp. 611-631.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 28* (1). 100–108.

Olkin, I., Dahabreh, I. J., Trikalinos, T. A. (2012). GOSH–a Graphical Display of Study Heterogeneity. *Research Synthesis Methods 3*, (3). 214–23.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P. & Xu, X. (2017). DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems (TODS) 42*, (3). ACM: 19.

See Also

`InfluenceAnalysis`

Examples

```
# Example: load gosh data (created with metafor's 'gosh' function),
# then use function
## Not run:
data("m.gosh")
res <- gosh.diagnostics(m.gosh)

# Look at results
summary(res)

# Plot detected clusters
plot(res, which = "cluster")

# Plot outliers
plot(res, which = "outlier")

## End(Not run)
```

`InfluenceAnalysis` *Influence Diagnostics*

Description

Conducts an influence analysis of a meta-analysis generated by `meta` functions, and allows to produce influence diagnostic plots.

Usage

```
InfluenceAnalysis(x, random = FALSE, subplot.heights = c(30,18),
    subplot.widths = c(30,30), forest.lims = 'default',
    return.separate.plots = FALSE, text.scale = 1)
```

Arguments

`x`	An object of class `meta`, generated by the `metabin`, `metagen`, `metacont`, `metacor`, `metainc`, `metarate` or `metaprop` function.
`random`	Logical. Should the random-effects model be used to generate the influence diagnostics? Uses the `method.tau` specified in the `meta` object if one of `"DL"`, `"HE"`, `"SJ"`, `"ML"`, `"REML"`, `"EB"`, `"PM"`, `"HS"` or `"GENQ"` (to ensure compatibility with the `metafor` package). Otherwise, the DerSimonian-Laird (`"DL"`; DerSimonian & Laird, 1986) estimator is used. `FALSE` by default.
`subplot.heights`	Concatenated array of two numerics. Specifies the heights of the first (first number) and second (second number) row of the overall plot generated when plotting the results. Default is `c(30,18)`.
`subplot.widths`	Concatenated array of two numerics. Specifies the widths of the first (first number) and second (second number) column of the overall results plot generated when plotting the results. Default is `c(30,30)`.
`forest.lims`	Concatenated array of two numerics. Specifies the x-axis limits of the forest plots generated when plotting the results. Use `"default"` if standard settings should be used (this is the default).

`return.separate.plots`
: Logical. When plotted, should the influence plots be shown as separate plots in lieu of returning them in one overall plot?

`text.scale`
: Positive numeric. Scaling factor for the text geoms used when plotting the results. Values <1 shrink the text, while values >1 increase the text size. Default is 1.

Details

The function conducts an influence analysis using the "Leave-One-Out" paradigm internally and produces data for four influence diagnostics. Diagnostic plots can be produced by saving the output of the function to an object and plugging it into the `plot` function. These diagnostics may be used to determine which study or effect size may have an excessive influence on the overall results of a meta-analysis and/or contribute substantially to the between-study heterogeneity in an analysis. This may be used for outlier detection and to test the robustness of the overall results found in an analysis. Results for four diagnostics are calculated:

- **Baujat Plot**: Baujat et al. (2002) proposed a plot to evaluate heterogeneity patterns in a meta-analysis. The x-axis of the Baujat plot shows the overall heterogeneity contribution of each effect size while the y-axis shows the influence of each effect size on the pooled result. The `baujat` function is called internally to produce the results. Effect sizes or studies with high values on both the x and y-axis may be considered to be influential cases; effect sizes or studies with high heterogeneity contribution (x-axis) and low influence on the overall results can be outliers which might be deleted to reduce the amount of between-study heterogeneity.
- **Influence Characteristics**: Several influence analysis diagnostics proposed by Viechtbauer & Cheung (2010). Results are calculated by an internal call to `influence.rma.uni`. In the console output, potentially influential studies are marked with an asterisk (`*`). When plotted, effect sizes/studies determined to be influential cases using the "rules of thumb" described in Viechtbauer & Cheung (2010) are shown in red. For further details, see the documentation of the `influence.rma.uni` function.
- **Forest Plot for the Leave-One-Out Analysis, sorted by Effect Size**: This displays the effect size and I^2-heterogeneity when omitting one of the k studies each time. The plot is sorted by effect size to determine which studies or effect sizes particularly affect the overall effect size. Results are generated by an internal call to `metainf`.
- **Forest Plot for the Leave-One-Out Analysis, sorted by I^2**: see above; results are sorted by I^2 to determine the study for which exclusion results in the greatest reduction of heterogeneity.

Value

A `list` object of class `influence.analysis` containing the following objects is returned (if results are saved to a variable):

- `BaujatPlot`: The Baujat plot
- `InfluenceCharacteristics`: The Viechtbauer-Cheung influence characteristics plot
- `ForestEffectSize`: The forest plot sorted by effect size
- `ForestI2`: The forest plot sorted by between-study heterogeneity
- `Data`: A `data.frame` containing the data used for plotting.

Otherwise, the function prints out (1) the results of the Leave-One-Out Analysis (sorted by I^2), (2) the Viechtbauer-Cheung Influence Diagnostics and (3) Baujat Plot data (sorted by heterogeneity contribution), in this order. Plots can be produced manually by plugging a saved object of class `InfluenceAnalysis` generated by the function into the `plot` function. It is also possible to only

produce one specific plot by specifying the name of the plot as a `character` in the second argument of the `plot` call (see Examples).

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 6.3

DerSimonian R. & Laird N. (1986), Meta-analysis in clinical trials. *Controlled Clinical Trials, 7*, 177–188.

Viechtbauer, W., & Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research Synthesis Methods, 1*, 112–125.

See Also

`influence.rma.uni`, `metainf`, `baujat`

Examples

```
## Not run:
# Load 'ThirdWave' data
data(ThirdWave)

# Create 'meta' meta-analysis object
suppressPackageStartupMessages(library(meta))
meta = metagen(TE, seTE, studlab = paste(ThirdWave$Author), data=ThirdWave)

# Run influence analysis; specify to return separate plots when plotted
inf.an = InfluenceAnalysis(meta, return.separate.plots = TRUE)

# Show results in console
inf.an

# Generate all plots
plot(inf.an)

# For baujat plot
plot(inf.an, "baujat")

# For influence diagnostics plot
plot(inf.an, "influence")

# For forest plot sorted by effect size
plot(inf.an, "ES")

# For forest plot sorted by I-squared
plot(inf.an, "I2")
## End(Not run)
```

m.gosh *GOSH plot dataset*

Description

GOSH plot dataset

Usage

```
data("m.gosh")
```

Format

Dataset of class gosh.rma.

Author(s)

Mathias Harrer, David Daniel Ebert

Source

Data generated by fitting a random-effects meta-analysis with rma.uni using dmetars in-built ThirdWave dataset, which was then provided as input to the gosh function in metafor.

mlm.variance.distribution *Calculate I-squared and the variance distribution for multilevel meta-analysis models*

Description

This function calculates values of I^2 and the variance distribution for multilevel meta-analysis models fitted with rma.mv.

Usage

```
mlm.variance.distribution(x)
```

Arguments

x	An object of class rma.mv. Must be a multilevel model with two random effects (three-level meta-analysis model).

Details

This function estimates the distribution of variance in a three-level meta-analysis model (fitted with the rma.mv function). The share of variance attributable to sampling error, within and between-cluster heterogeneity is calculated, and an estimate of I^2 (total and for Level 2 and Level 3) is provided. The function uses the formula by Cheung (2014) to estimate the variance proportions attributable to each model component and to derive the I^2 estimates.

Value

Returns a plot summarizing the variance distribution and I^2 values, as well as a data frame for the results.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 12.

Cheung, M. W. L. (2014). Modeling dependent effect sizes with three-level meta-analyses: a structural equation modeling approach. *Psychological Methods, 19*(2), 211.

Examples

```
# Use dat.konstantopoulos2011 from the "metafor" package
library(metafor)

# Build Multilevel Model (Three Levels)
m = rma.mv(yi, vi, random = ~ 1 | district/school, data=dat.konstantopoulos2011)

# Calculate Variance Distribution
mlm.variance.distribution(m)
```

---

multimodel.inference *Perform multimodel inference with a meta-regression model*

Description

This function performs multimodel inference to evaluate the importance of predictors with a meta-analytical meta-regression model.

Usage

```
multimodel.inference(TE, seTE, data, predictors, method='REML', test='knha',
    eval.criterion='AICc', interaction=FALSE, seed=123)
```

Arguments

TE	The precalculated effect size for each study. Must be supplied as the name of the effect size column in the dataset (in quotation marks; e.g. TE = "effectsize").
seTE	The precalculated standard error of the effect size for each study. Must be supplied as the name of the standard error column in the dataset (in quotation marks; e.g. seTE = "se").
data	A data.frame containing columns for the effect size, standard error and meta-regression predictors of each study/effect.
predictors	A character vector specifying the predictors to be used for multimodel inference. Names of the predictors must be identical to the names of the columns in the data.frame supplied to data.

`method`	Meta-analysis model to use for pooling effect sizes. Use `'FE'` for the fixed-effect model. Different random-effect models are available: `"DL"`, `"HE"`, `"SJ"`, `"ML"`, `"REML"`, `"EB"`, `"HS"` or `"GENQ"`. If `'FE'` is used, the `test` argument is automatically set to `'z'`, as the Knapp-Hartung method is not meant to be used with fixed-effect models. Default is `'REML'`, and it is strongly advised to remain with this option to use a standard (mixed-effects) meta-regression model.
`test`	Method to use to compute test statistics and confidence intervals. Default is `'knha'` which uses the Knapp-Hartung (Knapp & Hartung, 2003) adjustment method. "Conventional" Wald-type tests and CIs are calculated by setting this argument to `'z'`. When `method='FE'`, this argument is set to `'z'` automatically as the Knapp-Hartung method was not intended to be used with fixed-effect models.
`eval.criterion`	Evaluation criterion to sort the multiple models by. Can be either `'AICc'` (default; corrected Akaike's Information Criterion), `'AIC'` (Akaike's Information Criterion) or `'BIC'` (Bayesian Information Criterion).
`interaction`	If set to `FALSE` (default), no interactions between predictors are considered. Setting this parameter to `TRUE` means that all interactions are modeled (interactions will only be modeled if the number of provided predictors is 4 or less).
`seed`	Set a seed for the function. Default seed is `123`.

Details

Multimodel methods differ from stepwise regression methods as they do not try to successively build the "best" single (meta-regression) model explaining most of the variance. Instead, in this procedure, all possible combinations of a predefined selection of predictors are modeled, and evaluated using a criterion such as Akaike's Information Criterion, which rewards simpler models. This enables a full examination of all possible models, and how they perform. A common finding using this procedure is that there are many different kinds of predictor combinations within a model which lead to a good fit. In multimodel inference, the estimated coefficients of predictors can then be synthesized across all possible models to infer how important certain predictors are overall.

Multimodel Inference can be a useful way to obtain a comprehensive look on which predictors are more or less important for predicting differences in effect sizes. Despite avoiding some of the problems of stepwise regression methods, it should be noted that this method should still be rather seen as exploratory, and may be used when there is no prior knowledge on how predictors are related to effect sizes in the research field under study.

The `multimodel.inference` function calls the `rma.uni` function internally, results of which are then fed forward to an adapted version of the `dredge` function internally for multimodel inference. Parts of the computations in this function are adapted from a vignette by Wolfgang Viechtbauer, which can be found here.

Value

Returns four tables and a plot:

- **Final Results (Summary Table)**: Displays the number of fitted models, model formula, method to calculate test statistics and confidence intervals, interactions, and evaluation criterion used.
- **Best 5 Models**: Displays the top five models in terms of the evaluation criterion used. Predictors are displayed as columns of the table, and models as rows. A number (weight) or + sign (for categorical predictors) indicates that a predictor/interaction term was used for the model, while empty cells indicate that the predictor was omitted in this model. Other metrics such

as the `weight`, evaluation metric `delta` compared to the best model, log-Likelihood (`logLik`) and degrees of freedom are also displayed.

- **Multimodel Inference Coefficients**: Displays the estimated coefficients and statistical significance of each regression term in the model.
- **Predictor Importance**: Displays the estimated importance of each model term. The table is sorted from highest to lowest. A common rule of thumb is to consider a predictor as important when its importance value is above 0.8.
- **Predictor Importance Plot**: A bar plot for the predictor importance data along with a reference line for the 0.8 value often used as a crude threshold to characterize a predictor as important.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 9.1

Knapp, G., & Hartung, J. (2003). Improved tests for a random effects meta-regression with a single covariate. *Statistics in Medicine, 22*, 2693–2710.

Viechtbauer, W. (2019). *Model Selection using the glmulti and MuMIn Packages*. Link. Last accessed 01-Aug-2019.

See Also

`dredge`

Examples

```
## Not run:
# Example 1: Perform multimodel inference with default settings
data('MVRegressionData')
library(metafor)
mmi = multimodel.inference(TE = 'yi', seTE = 'sei', data = MVRegressionData,
                           predictors = c('pubyear', 'quality',
                                          'reputation', 'continent'))
# Print summary
summary(mmi)

# Plot predictor importance
plot(mmi)

# Example 2: Model Interaction terms, set method to 'DL',
# change evaluation criterion to bic
multimodel.inference(TE = 'yi', seTE = 'sei', data = MVRegressionData,
                     predictors = c('pubyear', 'quality',
                                    'reputation', 'continent'),
                     method='DL', eval.criterion = 'BIC', interaction = TRUE)

# Example 3: Use only categorical predictors
data('ThirdWave')
multimodel.inference(TE = 'TE', seTE = 'seTE', data = ThirdWave,
                     predictors = colnames(ThirdWave)[4:7], interaction = FALSE)
```

```
## End(Not run)
```

`MVRegressionData` *Toy Dataset for Multivariate Meta-Regression*

Description

This is a toy dataset containing simulated effect size data of a fictitious meta-analysis examining the effect of various putative effect moderators. Effect size data is provided as the standardized mean difference (SMD) between the intervention and control group and its corresponding standard error for each study at post. Columns are named in after arguments of the `rma.uni` function to facilitate out-of-the-box usage.

Usage

```
data("MVRegressionData")
```

Format

A data.frame with 6 columns.

yi Numeric. The calculated standardized mean difference at post-test between the intervention and control group

sei Numeric. The standard error of the standardized mean difference

reputation Numeric. The mean-centered score signifying the "reputation" (for example, impact factor) of the journal the respective study was published in.

quality Numeric. The methodological quality of the study, rated from 0-10 (low to high).

pubyear Numeric. The z-standardized year of publication.

continent Numeric. The continent the study was conducted in.

Author(s)

Mathias Harrer, David Daniel Ebert

Source

Simulated data.

NetDataGemtc *Toy Dataset for Network Meta-Analyses using the gemtc package*

Description

This is a toy dataset containing simulated effect size data of a fictitious network meta-analysis examining the effect of psychotherapies. Effect size data is provided as the standardized mean difference (*SMD*) between the intervention and control group and its corresponding standard error for each study at post. The dataframe layout is optimized for out-of-the-box usage using the `data.re` argument of the `mtc.network` function.

Usage

```
data("NetDataGemtc")
```

Format

A data.frame with 4 columns.

study `character`. The name of the included study.

treatment `character`. The name of the treatment under study. Includes psychotherapies for the treatment of depression, "CBT" (Cognitive Behavioral Therapy), "PDT" (Psychodynamic Therapy), "IPT" (Interpersonal Therapy), "PST" (Problem-solving Therapy) and "SUP" (Supportive Counseling), and comparison conditions, "TAU" (Treatment as usual), "Placebo" (Placebo), and "WLC" (Waitlist control). Each treatment condition in a study is displayed in its own row of the dataframe.

diff `numeric`. The standardized mean difference of the comparison. The row in each study in which this variable is `NA` represents the comparison condition for the effect size displayed above.

std.err `numeric`. The standard error of the comparison. The row in each study in which this variable is `NA` represents the comparison condition for the standard error of the effect size displayed above.

Author(s)

Mathias Harrer, David Daniel Ebert

Source

Simulated data.

NetDataNetmeta *Toy Dataset for Network Meta-Analysis using the netmeta package*

Description

This is a toy dataset containing simulated effect size data of a fictitious network meta-analysis examining the effect of psychotherapies. Effect size data is provided as the standardized mean difference (SMD) between the intervention and control group and its corresponding standard error for each study at post. The dataframe layout is optimized for out-of-the-box usage using the `netmeta` function.

Usage

```
data("NetDataNetmeta")
```

Format

A data.frame with 5 columns.

studlab `character`. The name of the included study.

treat1 `character`. The name of the first treatment. Includes psychotherapies for the treatment of depression, "CBT" (Cognitive Behavioral Therapy), "PDT" (Psychodynamic Therapy), "IPT" (Interpersonal Therapy), "PST" (Problem-solving Therapy) and "SUP" (Supportive Counseling), and standard comparison conditions, "TAU" (Treatment as usual), "Placebo" (Placebo), and "WLC" (Waitlist control).

treat2 `character`. The name of the treatment the first treatment was compared to. Includes psychotherapies for the treatment of depression, "CBT" (Cognitive Behavioral Therapy), "PDT" (Psychodynamic Therapy), "IPT" (Interpersonal Therapy), "PST" (Problem-solving Therapy) and "SUP" (Supportive Counseling), and standard comparison conditions, "TAU" (Treatment as usual), "Placebo" (Placebo), and "WLC" (Waitlist control).

TE `numeric`. The standardized mean difference of the comparison.

seTE `numeric`. The standard error of the comparison.

Author(s)

Mathias Harrer, David Daniel Ebert

Source

Simulated data.

NNT *Calculate the number needed to treat (NNT)*

Description

This function calculates the number needed to treat (NTT) using event data or effect sizes (such as Cohen's d or Hedges' g).

Usage

```
NNT(d, CER, event.e, n.e, event.c, n.c, names, method)
```

Arguments

d	A single numeric or concatenated vector of numerics representing the effect size expressed as Cohen's d or Hedges' g. If this is the only parameter specified in the function, the method by Kraemer and Kupfer is used automatically to calculate NNTs.
CER	The control group event ratio. Furukawa's method (Furukawa & Leucht, 2011) to calculate NNTs from d requires that the assumed response ("event") ratio in the control group ($\frac{n_{responders}}{N_{total}}$) is specified. The CER can assume values from 0 to 1. If a value is specified for CER, Furukawa's method is used automatically. Argument method has to be set to "KraemerKupfer" to override this.
event.e	Single number or numeric vector. The number of (favourable) events in the experimental group.
n.e	Single number or numeric vector. The number participants in the experimental group.
event.c	Single number or numeric vector. The number of (favourable) events in the control group.
n.c	Single number or numeric vector. The number of participants in the control group.
names	Optional. Character vector of equal length as the vector supplied to d or event.e containing study/effect size labels.
method	The method to be used to calculate the NNT from d. Either "KraemerKupfer" for the method proposed by Kraemer and Kupfer (2006) or "Furukawa" for the Furukawa method (Furukawa & Leucht, 2011). Please note that the Furukawa's method can only be used when CER is specified.

Details

This function calculates the number needed to treat (NNT) from effect sizes (Cohen's d and Hedges' g) or, alternatively, from raw event data.

Two methods to calculate the NTT from d are implemented in this function.

- The method by **Kraemer and Kupfer** (2006), calculates NTT from the Area Under the Curve (AUC) defined as the probability that a patient in the treatment has an outcome preferable to one in the control. This method allows to calculate the NNT directly from d without any extra variables.

- The method by **Furukawa** calculates the NNT from d using a reasonable estimate of CER, in this context the response rate in the control group.

Furukawa's method has been shown to be superior in predicting the NNT compared to the Kraemer & Kupfer method (Furukawa & Leucht, 2011). If reasonable assumptions can be made concerning the CER, Furukawa's method should therefore be preferred.

When event data is used for the function, the CER and EER (experimental group event rate) is calculated internally, and the standard definition of the NTT, $\frac{1}{EER-CER}$, is used.

Please note that negative NNT values returned by the function refer to the number needed to harm (NNH), as the intervention is assumed to be inferior to the control group treatment based on the effect size data supplied to the function.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 9.1.

Furukawa, T. A., & Leucht, S. (2011). How to obtain NNT from Cohen's d: comparison of two methods. *PloS one, 6*(4), e19070.

Kraemer H.C., Kupfer D.J. (2006) Size of treatment effects and their importance to clinical research and practice. Biol. Psychiatry 59: 990–996.

See Also

`se.from.p`

Examples

```
# Example 1: Convert Cohen's d using the Kraemer & Kupfer method
d = c(-0.123, 0.234, 0.123, 1.234, 0.12)
NNT(d)

# Example 2: Convert Cohen's d using the Furukawa method
d = c(-0.123, 0.234, 0.123, 1.234, 0.12)
CER = c(0.42, 0.35, 0.26, 0.21, 0.23)
NNT(d, CER)

# Example 3: Convert event data
NNT(event.e = 10, event.c = 20, n.e = 200, n.c = 200)
```

---

pcurve *Perform a* p*-curve analysis*

Description

This function performs a p-curve analysis using a `meta` object or calculated effect size data.

Usage

```
pcurve(x, effect.estimation = FALSE, N, dmin = 0, dmax = 1)
```

Arguments

x	Either an object of class meta, generated by the metagen, metacont, metacor, metainc, or metabin function, or a dataframe containing the calculated effect size (named TE, log-transformed if based on a ratio), standard error (named seTE) and study label (named studlab) for each study.
effect.estimation	Logical. Should the true effect size underlying the p-curve be estimated? If set to TRUE, a vector containing the total sample size for each study must be provided for N. FALSE by default.
N	A numeric vector of same length as the number of effect sizes included in x specifying the total sample size N corresponding to each effect. Only needed if effect.estimation = TRUE.
dmin	If effect.estimation = TRUE: lower limit for the effect size (d) space in which the true effect size should be searched. Must be greater or equal to 0. Default is 0.
dmax	If effect.estimation = TRUE: upper limit for the effect size (d) space in which the true effect size should be searched. Must be greater than 0. Default is 1.

Details

P-curve Analysis

P-curve analysis (Simonsohn, Simmons & Nelson, 2014, 2015) has been proposed as a method to detect p-hacking and publication bias in meta-analyses.

P-Curve assumes that publication bias is not only generated because researchers do not publish non-significant results, but also because analysts "play" around with their data ("p-hacking"; e.g., selectively removing outliers, choosing different outcomes, controlling for different variables) until a non-significant finding becomes significant (i.e., $p < 0.05$).

The method assumes that for a specific research question, p-values smaller 0.05 of included studies should follow a right-skewed distribution if a true effect exists, even when the power in single studies was (relatively) low. Conversely, a left-skewed p-value distribution indicates the presence of p-hacking and absence of a true underlying effect. To control for "ambitious" p-hacking, P-curve also incorporates a "half-curve" test (Simonsohn, Simmons & Nelson, 2014, 2015).

Simonsohn et al. (2014) stress that p-curve analysis should only be used for test statistics which were actually of interest in the context of the included study, and that a detailed table documenting the reported results used in for the p-curve analysis should be created before communicating results (link).

Implementation in the function

To generate the p-curve and conduct the analysis, this function reuses parts of the *R* code underlying the P-curve App 4.052 (Simonsohn, 2017). The effect sizes included in the meta object or data.frame provided for x are transformed into z-values internally, which are then used to calculate p-values and conduct the Stouffer and Binomial test used for the p-curve analysis. Interpretations of the function concerning the presence or absence/inadequateness of evidential value are made according to the guidelines described by Simonsohn, Simmons and Nelson (2015):

- **Evidential value present**: The right-skewness test is significant for the half curve with $p < 0.05$ **or** the p-value of the right-skewness test is < 0.1 for both the half and full curve.

- **Evidential value absent or inadequate**: The flatness test is $p < 0.05$ for the full curve **or** the flatness test for the half curve and the binomial test are $p < 0.1$.

For effect size estimation, the `pcurve` function implements parts of the loss function presented in Simonsohn, Simmons and Nelson (2014b). The function generates a loss function for candidate effect sizes $\hat{d}$, using D-values in a Kolmogorov-Smirnov test as the metric of fit, and the value of $\hat{d}$ which minimizes D as the estimated true effect.

It is of note that a lack of robustness of p-curve analysis results has been noted for meta-analyses with substantial heterogeneity (van Aert, Wicherts, & van Assen, 2016). Following van Aert et al., adjusted effect size estimates should only be reported and interpreted for analyses with I^2 values below 50 percent. A warning message is therefore printed by the `pcurve` function when `x` is of class `meta` and the between-study heterogeneity of the meta-analysis is substantial (i.e., I^2 greater than 50 percent).

Value

Returns a plot and main results of the pcurve analysis:

- **P-curve plot**: A plot displaying the observed p-curve and significance results for the right-skewness and flatness test.
- **Number of studies**: The number of studies provided for the analysis, the number of significant p-values included in the analysis, and the number of studies with $p < 0.025$ used for the half-curve tests.
- **Test results**: The results for the right-skewness and flatness test, including the $p_{binomial}$ value, as well as the z and p value for the full and half-curve test.
- **Power Estimate**: The power estimate and 95% confidence interval.
- **Evidential value**: Two lines displaying if evidential value is present and/or absent/inadequate based on the results (using the guidelines by Simonsohn et al., 2015, see Details).
- **True effect estimate**: If `effect.estimation` is set to `TRUE`, the estimated true effect $\hat{d}$ is returned additionally.

If results are saved to a variable, a list of class `pcurve` containing the following objects is returned:

- `pcurveResults`: A data frame containing the results for the right-skewness and flatness test, including the $p_{binomial}$ value, as well as the z and p value for the full and half-curve test.
- `Power`: The power estimate and 95% confidence interval.
- `PlotData`: A data frame with the data used in the p-curve plot.
- `Input`: A data frame containing the provided effect sizes, calculated p-values and individual results for each included (significant) effect.
- `EvidencePresent`, `EvidenceAbsent`, `kInput`, `kAnalyzed`, `kp0.25`: Further results of the p-curve analysis, including the presence/absence of evidence interpretation, and number of provided/significant/$p < 0.025$ studies.
- `I2`: I^2-Heterogeneity of the studies provided as input (only when `x` is of class `meta`).
- `class.meta.object`: `class` of the original object provided in `x`.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 9.2.

Simonsohn, U., Nelson, L. D., & Simmons, J. P. (2014a). P-curve: a Key to the File-drawer. *Journal of Experimental Psychology, 143*(2), 534.

Simonsohn, U., Nelson, L. D. & Simmons, J. P. (2014b). P-Curve and Effect Size: Correcting for Publication Bias Using Only Significant Results. *Perspectives on Psychological Science 9*(6), 666–81.

Simonsohn, U., Nelson, L. D. & Simmons, J. P. (2015). Better P-Curves: Making P-Curve Analysis More Robust to Errors, Fraud, and Ambitious P-Hacking, a Reply to Ulrich and Miller (2015). *Journal of Experimental Psychology, 144*(6), 1146-1152.

Simonsohn, U. (2017). R code for the P-Curve App 4.052. http://p-curve.com/app4/pcurve_app4.052.r (Accessed 2019-08-16).

Van Aert, R. C., Wicherts, J. M., & van Assen, M. A. (2016). Conducting meta-analyses based on p values: Reservations and recommendations for applying *p*-uniform and *p*-curve. *Perspectives on Psychological Science, 11*(5), 713-729.

See Also

eggers.test

Examples

```
# Example 1: Use metagen object, do not estimate d
suppressPackageStartupMessages(library(meta))
data("ThirdWave")
meta1 = metagen(TE,seTE, studlab=ThirdWave$Author, data=ThirdWave)
pcurve(meta1)

# Example 2: Provide Ns, calculate d estimate
N = c(105, 161, 60, 37, 141, 82, 97, 61, 200, 79, 124, 25, 166, 59, 201,  95, 166, 144)
pcurve(meta1, effect.estimation = TRUE, N = N)

# Example 3: Use metacont object, calculate d estimate
data("amlodipine")
meta2 <- metacont(n.amlo, mean.amlo, sqrt(var.amlo),
                  n.plac, mean.plac, sqrt(var.plac),
                  data=amlodipine, studlab=study, sm="SMD")
N = amlodipine$n.amlo + amlodipine$n.plac
pcurve(meta2, effect.estimation = TRUE, N = N, dmin = 0, dmax = 1)

# Example 4: Construct x object from scratch
sim = data.frame("studlab" = c(paste("Study_", 1:18, sep = "")),
                 "TE" = c(0.561, 0.296, 0.648, 0.362, 0.770, 0.214, 0.476,
                          0.459, 0.343, 0.804, 0.357, 0.476, 0.638, 0.396, 0.497,
                          0.384, 0.568, 0.415),
                 "seTE" = c(0.338, 0.297, 0.264, 0.258, 0.279, 0.347, 0.271, 0.319,
                            0.232, 0.237, 0.385, 0.398, 0.342, 0.351, 0.296, 0.325,
                            0.322, 0.225))
pcurve(sim)
```

plot.direct.evidence.plot

Print, summary and plot methods for objects created using 'dmetar' functions

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'direct.evidence.plot'
plot(x, ...)
```

Arguments

`x`	An object of class `direct.evidence.plot`.
`...`	Other arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical. When both `plot` and `forest` are available for a class, outputs of both functions are identical (i.e., a forest plot is returned).

Author(s)

Mathias Harrer & David Daniel Ebert

plot.eggers.test

Print, summary and plot methods for objects created using 'dmetar' functions

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `eggers.test`, and `sucra`.

Usage

```
## S3 method for class 'eggers.test'
plot(x, ...)
```

Arguments

x	An object of class eggers.test.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

---

plot.find.outliers *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'find.outliers'
plot(x, ...)
```

Arguments

x	An object of class find.outliers.
...	Other arguments of the meta.forest or metafor's forest function. Can be used for styling the generated forest plot.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical. When both plot and forest are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The forest/plot function allows additional arguments of the meta.forest or metafor's forest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

plot.gosh.diagnostics *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `eggers.test`, `gosh.diagnostics`, and `sucra`.

Usage

```
## S3 method for class 'gosh.diagnostics'
plot(x, which = c("all", "cluster", "outlier"), ...)
```

Arguments

`x`	An object of class `gosh.diagnostics`.
`which`	Type of plot to display. Can be `"all"` (all plots), `"cluster"` (cluster plots only) or `"outlier"` (outlier plots only, if available).
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

plot.InfluenceAnalysis *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'InfluenceAnalysis'
plot(x, which = "all", ...)
```

Arguments

x	An object of class influence.analysis.
which	Which plot(s) should be generated? Can be either "all" (default), "baujat", "influence", "ES" (forest plot sorted by effect size) or "I2" (forest plot sorted by heterogeneity).
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

---

plot.multimodel.inference

Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'multimodel.inference'
plot(x, ...)
```

Arguments

x	An object of class multimodel.inference.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

plot.power.analysis *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `eggers.test`, and `sucra`.

Usage

```
## S3 method for class 'power.analysis'
plot(x, ...)
```

Arguments

`x`	An object of class `power.analysis`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

plot.power.analysis.subgroup *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'power.analysis.subgroup'
plot(x, ...)
```

Arguments

`x`	An object of class `power.analysis.subgroup`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

`plot.subgroup.analysis.mixed.effects`
Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'subgroup.analysis.mixed.effects'
plot(x, ...)
```

Arguments

`x`	An object of class `subgroup.analysis.mixed.effects`.
`...`	Other arguments of the `meta.forest` or `metafor`'s `forest` function. Can be used for styling the generated forest plot.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical. When both `plot` and `forest` are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The `forest/plot` function allows additional arguments of the `meta.forest` or `metafor`'s `forest` function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

plot.sucra *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'sucra'
plot(x, ...)
```

Arguments

`x`	An object of class `sucra`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

pool.groups *Pool the results of two treatment arms*

Description

This function allows to pool the mean, standard deviation and sample size of two experimental groups of a study. Results of two treatment arms may be pooled to mitigate the risk of a unit-of-analysis error and to avoid "double-counting" in meta-analyses in which studies with more than two experimental groups are included.

Usage

```
pool.groups(n1, n2, m1, m2, sd1, sd2)
```

Arguments

`n1`	Numeric vector or single number. The number of participants in arm 1.
`n2`	Numeric vector or single number. The number of participants in arm 2.
`m1`	Numeric vector or single number. The mean in arm 1.
`m2`	Numeric vector or single number. The mean in arm 2.
`sd1`	Numeric vector or single number. The standard deviation (SD) in arm 1.
`sd2`	Numeric vector or single number. The standard deviation (SD) in arm 2.

Details

Many randomized-controlled trials do not only include a single intervention and control group, but compare the effect of two or more interventions to a control group. It might be tempting in such a scenario to simply include all the comparisons between the intervention groups and control within a study into one meta-analysis. Yet, researchers should abstain from this practice, as this would mean that the control group is used twice for the meta-analysis, thus "double-counting" the participants in the control group. This results in a **unit-of-analysis error**, as the effect sizes are correlated, and thus not independent, but are treated as if they would stem from independent samples.

One way to deal with this is to synthesize the results of the intervention arms to obtain one single comparison to the control group. Despite its practical limitations (sometimes, this would mean synthesizing the results from extremely different types of interventions), this procedure does avoid the unit-of-analysis error problem.

To synthesize the pooled effect size data (pooled mean, standard deviation and N), the following formulae are used:

$$N_{pooled} = N_1 + N_2$$

$$M_{pooled} = \frac{N_1 M_1 + N_2 M_2}{N_1 + N_2}$$

$$SD_{pooled} = \sqrt{\frac{(N_1 - 1)SD_1^2 + (N_2 - 1)SD_2^2 + \frac{N_1 N_2}{N_1 + N_2}(M_1^2 + M_2^2 - 2M_1 M_2)}{N_1 + N_2 - 1}}$$

What should I do when a study has more than two intervention groups?

If a study has more than two intervention groups you want to synthesize (e.g. four arms, with three distinct intervention arms), you can pool the effect size data for the first two interventions, and then synthesize the pooled data you calculated with the data from the third group.

Value

Returns a `data.frame` containing the following columns:

- `Mpooled`: The pooled mean of both groups
- `SDpooled`: The pooled standard deviation of both groups
- `Npooled`: The pooled number of participants of both groups

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 13.9

See Also

`se.from.p`

Examples

```
pool.groups(n1 = 50, n2 = 56,
    m1 = 7.83, m2 = 8.32,
    sd1 = 3.52, sd2 = 2.25)
```

`power.analysis` *A priori power calculator*

Description

This function performs an *a priori* power estimation of a meta-analysis for different levels of assumed between-study heterogeneity.

Usage

```
power.analysis(d, OR, k, n1, n2, p = 0.05, heterogeneity = 'fixed')
```

Arguments

`d`	The hypothesized, or plausible overall effect size of a treatment/intervention under study compared to control, expressed as the standardized mean difference (*SMD*). Effect sizes must be positive numerics (i.e., expressed as positive effect sizes).
`OR`	The hypothesized, or plausible overall effect size of a treatment/intervention under study compared to control, expressed as the Odds Ratio (*OR*). If both `d` and `OR` are specified, results will only be computed for the value of `d`.
`k`	The expected number of studies to be included in the meta-analysis.
`n1`	The expected, or plausible mean sample size of the treatment group in the studies to be included in the meta-analysis.
`n2`	The expected, or plausible mean sample size of the control group in the studies to be included in the meta-analysis.
`p`	The alpha level to be used for the power computation. Default is $\alpha = 0.05$.
`heterogeneity`	Which level of between-study heterogeneity to assume for the meta-analysis. Can be either `"fixed"` for no heterogeneity/a fixed-effect model, `"low"` for low heterogeneity, `"moderate"` for moderate-sized heterogeneity or `"high"` for high levels of heterogeneity. Default is `"fixed"`.

Details

While researchers conducting primary studies can plan the size of their sample based on the effect size they want to find, the situation is a different in meta-analysis, where one can only work with the published material. However, researchers have some control over the number of studies they want to include in their meta-analysis (e.g., through more leniently or strictly defined inclusion criteria). Therefore, one can change the power to some extent by including more or less studies into the meta-analysis. Conventionally, a power of $1 - \beta = 0.8$ is deemed sufficient to detect an existing effect. There are four things one has to make assumptions about when assessing the power of a meta-analysis a priori.

- The number of included or includable studies
- The overall size of the studies we want to include (are the studies in the field rather small or large?)

- The effect size. This is particularly important, as assumptions have to be made about how big an effect size has to be to still be clinically meaningful. One study calculated that for interventions for depression, even effects as small as *SMD*=0.24 may still be meaningful for patients (Cuijpers et al. 2014). If the aim is to study negative effects of an intervention (e.g., death or symptom deterioration), even very small effect sizes are extremely important and should be detected.
- The heterogeneity of our studies' effect sizes, as this also affects the precision of the pooled estimate, and thus its potential to find significant effects.

The `power.analysis` function implements the formula by Borenstein et al. (2011) to calculate the power estimate. Odds Ratios are converted to `d` internally before the power is estimated, and are then reconverted.

Value

Returns a `list` with two elements:

- `Plot`: A plot showing the effect size (x), power (y), estimated power (red point) and estimated power for changing effect sizes (blue line). A dashed line at 80% power is also provided as a visual threshold for sufficient power.
- `Power`: The **estimated power** of the meta-analysis, expressed as a value between 0 and 1 (i.e., 0%-100%).

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 13

Cuijpers, P., Turner, E.H., Koole, S. L., Van Dijke, A., & Smit, F. (2014). What Is the Threshold for a Clinically Relevant Effect? The Case of Major Depressive Disorders. *Depression and Anxiety, 31*(5): 374–78.

Borenstein, M., Hedges, L.V., Higgins, J.P.T. and Rothstein, H.R. (2011). Introduction to Meta-Analysis. John Wiley & Sons.

See Also

`power.analysis.subgroup`

Examples

```
# Example 1: Using SMD and fixed-effect model (no heterogeneity)
power.analysis(d=0.124, k=10, n1=50, n2=50, heterogeneity = 'fixed')

# Example 2: Using OR and assuming moderate heterogeneity
pa = power.analysis(OR=0.77, k=12, n1=50, n2=50, heterogeneity = 'high')
summary(pa)

# Only show plot
plot(pa)
```

`power.analysis.subgroup` *A priori power calculator for subgroup contrasts*

Description

This function performs an *a priori* power estimation for a test for subgroup differences within a meta-analysis.

Usage

```
power.analysis.subgroup(TE1, TE2, seTE1, seTE2, sd1, sd2, var1, var2,
    two.tailed=TRUE)
```

Arguments

`TE1`	Pooled effect size (e.g., standardized mean difference, Hedges' g, log-Odds Ratio or other linear continuous effect size) of the first subgroup of studies.
`TE2`	Pooled effect size (e.g., standardized mean difference, Hedges' g, log-Odds Ratio or other linear continuous effect size) of the second subgroup of studies.
`seTE1`	Pooled standard error of the first subgroup of studies. Either `seTE1`/`seTE2`, `sd1`/`sd2`, or `var1`/`var2` must be provided.
`seTE2`	Pooled standard error of the second subgroup of studies. Either `seTE1`/`seTE2`, `sd1`/`sd2`, or `var1`/`var2` must be provided.
`sd1`	Pooled standard deviation of the first subgroup of studies. Either `seTE1`/`seTE2`, `sd1`/`sd2`, or `var1`/`var2` must be provided.
`sd2`	Pooled standard deviation of the second subgroup of studies. Either `seTE1`/`seTE2`, `sd1`/`sd2`, or `var1`/`var2` must be provided.
`var1`	Pooled variance of the first subgroup of studies. Either `seTE1`/`seTE2`, `sd1`/`sd2`, or `var1`/`var2` must be provided.
`var2`	Pooled variance of the second subgroup of studies. Either `seTE1`/`seTE2`, `sd1`/`sd2`, or `var1`/`var2` must be provided.
`two.tailed`	Logical. Should a two-tailed (`TRUE`) or one-tailed (`FALSE`) test ($\alpha = 0.05$) be assumed? Default is `TRUE`.

Details

This function provides an estimate of the power $1 - \beta$ of a subgroup contrast analysis provided the assumed effect sizes in each subgroup and their dispersion measures. The function implements the formulae described by Hedges and Pigott (2001).

Value

Returns a `list` with five elements:

- `Power`: The estimated power of the subgroup contrast, expressed as a value between 0 and 1 (i.e., 0%-100%).

- `Plot`: A plot showing the effect size difference (x), power (y), estimated power (red point) and estimated power for changing effect size differences (blue line). A dashed line at 80% power is also provided as a visual threshold for sufficient power.
- `Data`: A `data.frame` containing the data used to generate the plot in `Plot`.
- `Test`: The type of test used for the power calculations (`"one.tailed"` or `"two.tailed"`).
- `Gamma`: The analyzed effect size difference calculated from the inputs.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Hedges, L. V., & Pigott, T. D. (2001). The power of statistical tests in meta-analysis. *Psychological methods, 6*(3), 203.

See Also

`power.analysis`

Examples

```
# Example 1: using standard error and two-tailed test
power.analysis.subgroup(TE1=0.30, TE2=0.66, seTE1=0.13, seTE2=0.14)

# Example 2: using variance and one-tailed test
pasg = power.analysis.subgroup(TE1=-0.91, TE2=-1.22, var1 = 0.0023, var2 = 0.0078,
    two.tailed = FALSE)
summary(pasg)

# Only show plot
plot(pasg)
```

---

print.direct.evidence.plot

Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `direct.evidence.plot`, `gosh.diagnostics` and `sucra`.

Usage

```
## S3 method for class 'direct.evidence.plot'
print(x, ...)
```

Arguments

`x`	An object of class `direct.evidence.plot`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical. When both plot and forest are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The forest/plot function allows additional arguments of the meta.forest or metafor's codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

---

print.eggers.test *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, eggers.test, and sucra.

Usage

```
## S3 method for class 'eggers.test'
print(x, ...)
```

Arguments

x	An object of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, eggers.test, or sucra.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

print.find.outliers *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'find.outliers'
print(x, ...)
```

Arguments

`x`	An object of class `find.outliers`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical. When both `plot` and `forest` are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The `forest/plot` function allows additional arguments of the `meta.forest` or `metafor`'s codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

print.gosh.diagnostics *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `direct.evidence.plot`, `gosh.diagnostics` and `sucra`.

Usage

```
## S3 method for class 'gosh.diagnostics'
print(x, ...)
```

Arguments

x	An object of class gosh.diagnostics.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical. When both plot and forest are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The forest/plot function allows additional arguments of the meta.forest or metafor's codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

---

print.InfluenceAnalysis

Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'InfluenceAnalysis'
print(x, ...)
```

Arguments

x	An object of class InfluenceAnalysis.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

print.multimodel.inference

Print, summary and plot methods for objects created using 'dmetar' functions

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'multimodel.inference'
print(x, ...)
```

Arguments

`x`	An object of class `multimodel.inference`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

print.pcurve

Print, summary and plot methods for objects created using 'dmetar' functions

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'pcurve'
print(x, ...)
```

Arguments

`x`	An object of class `pcurve`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

---

print.power.analysis *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'power.analysis'
print(x, ...)
```

Arguments

x	An object of class power.analysis.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

---

print.power.analysis.subgroup *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'power.analysis.subgroup'
print(x, ...)
```

Arguments

x	An object of class power.analysis.subgroup.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

---

print.subgroup.analysis.mixed.effects

Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'subgroup.analysis.mixed.effects'
print(x, ...)
```

Arguments

x	An object of class subgroup.analysis.mixed.effects.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

print.sucra — *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `eggers.test`, and `sucra`.

Usage

```
## S3 method for class 'sucra'
print(x, ...)
```

Arguments

`x`	An object of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `eggers.test`, or `sucra`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

rob.summary — *Create a RevMan-style risk of bias summary chart*

Description

This function generates summary plots for study quality assessments using the Cochrance Risk of Bias Tool. Summary plots follow the style of RevMan Risk of Bias (RoB) summary charts.

Usage

```
rob.summary(data, name.high="High", name.unclear="Unclear",
    name.low="Low", studies, name.missing, table = FALSE)
```

Arguments

data	A data.frame containing a column for each risk of bias criterion, where rows represent each individual studies. The risk of bias assessment for each criterion in each study must be coded as a character string. Up to four codes can be used, referring to low risk of bias, unclear risk of bias, high risk of bias, or missing information. The string used to specify the categories must be specified in name.high, name.unclear, name.low and/or name.missing, unless defaults for those parameters are used.
name.high	Character specifying how the "high risk of bias" category was coded in data (e.g., name.high = "high"). Default is "High".
name.unclear	Character specifying how the "unclear risk of bias" category was coded in data (e.g., name.unclear = "unclear"). Default is "Unclear".
name.low	Character specifying how the "low risk of bias" category was coded in data (e.g., name.low = "low"). Default is "Low".
studies	A vector of the same length as the number of rows in data specifying the study labels for the risk of bias ratings. Only has to be specified when table = TRUE.
name.missing	Character specifying how missing information was coded in data (e.g., name.missing = "missing"). Default is "Missing". All ratings, including missing information, must be coded as strings, so using NA in data to signify missing information is not valid.
table	Should an additional RevMan style risk of bias table be produced? If set to TRUE, studies must be specified. FALSE by default.

Details

The function automatically removes separators like "-" or "." from column names/risk of bias criteria. To produce a "clean" plot, you may therefore separate words in the column names of the data data frame using these symbols (e.g. "Allocation_Concealment" to return "Allocation Concealment").

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 10

See Also

direct.evidence.plot

Examples

```
# Example 1: No missing information, only produce summary plot
data = data.frame(
       "study" = c("Higgins et al., 2011", "Borenstein et al., 2008", "Holm, 1971",
                   "Zajonc et al., 2005", "Viechtbauer, 2014"),
       "Allocation_concealment" = c("Low", "High", "High", "Unclear", "High"),
       "Randomization" = c("Low", "High", "Unclear", "Low", "High"),
       "Sequence_generation" = c("Low", "High", "Unclear", "Unclear", "High"),
```

```
       "ITT.Analyses" = c("Low", "High", "Unclear", "Unclear", "Unclear"),
       "Selective_outcome_reporting" = c("Low", "High", "High", "High", "Unclear")
       )
rob.summary(data)

# Example 2: Missing information, produce additional summary table
data2 = data.frame(
        "study" = c("Higgins et al., 2011", "Borenstein et al., 2008", "Holm, 1971",
                    "Zajonc et al., 2005", "Cuijpers, 2014"),
        "Allocation_concealment" = c("low", "high", "high", "uc", "high"),
        "Randomization" = c("low", "high", "miss", "low", "high"),
        "Sequence_generation" = c("low", "high", "uc", "uc", "high"),
        "ITT.Analyses" = c("low", "high", "uc", "uc", "uc"),
        "Selective_outcome_reporting" = c("low", "high", "high", "high", "uc")
       )
rob.summary(data2, name.high = "high", name.unclear = "uc", name.low = "low",
    name.missing = "miss", studies = data2$study, table = TRUE)
```

`se.from.p` *Calculate the standard error from the effect size and p-value*

Description

This function calculates the standard error of an effect size provided the exact p-value and (continuous) effect size according to the formula by Altman and Bland (2011).

Usage

```
se.from.p(effect.size, p, N, effect.size.type = 'difference',
    calculate.g = FALSE)
```

Arguments

`effect.size`	Numeric vector or single number. The effect size, such as the standardized mean difference, Hedges' g or other continuous effect size.
`p`	Numeric vector or single number. The exact p-value corresponding to the effect size.
`N`	Numeric vector or single number. The total number of samples used to calculate the effect size/p-value.
`effect.size.type`	The type of effect sizes provided in `effect.size`. For effect sizes based on differences (e.g., mean differences), this parameter has to be set to `"difference"`. For effect sizes based on ratios (e.g., risk ratio, odds ratio), this parameter has to be set to `"ratio"`.
`calculate.g`	Logical. Calculates the standardized mean difference corrected for small sample bias (Hedges' g). `FALSE` by default.

Details

This function calculates the standard error, standard deviation and 95% confidence interval of an effect size given the effect size and exact p-value. The function can be used for:

- effect sizes based on **differences** (e.g., mean differences) by setting `effect.size.type` to `"difference"`, or
- effect sizes based on **ratios** (e.g. risk ratios, odds ratios or hazard ratios) by setting `effect.size.type` to `"ratio"`. When ratios are used, the function returns the log-transformed effect sizes, standard error, standard deviation and confidence interval, which can be used for meta-analytic pooling using the `metagen` function, along with the original effect size and confidence interval.

Value

A dataframe containing the following columns:

- `(log)EffectSize`: The input effect size. Log-transformed if `effect.size.type` is `"ratio"`.
- `Hedges.g`: The calculated Hedges' g values (only if `calculate.g=TRUE`).
- `(log)StandardError`: The standard error (SE) for the effect size. Log-transformed if `effect.size.type` is `"ratio"`.
- `(log)LLCI` and `(log)ULCI`: The lower and upper 95% confidence interval of the effect size. Log-transformed if `effect.size.type="ratio"`.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Altman D.G. & Bland J.M. (2011) How to obtain the confidence interval of a *p* value. *BMJ 343*:d2090.

Examples

```
# Example 1: one single effect size
se.from.p(effect.size = 0.71, p = 0.013, N = 75,
    effect.size.type= "difference", calculate.g = TRUE)

# Example 2: vector of effect sizes (Odds Ratio)
effect.size = c(0.91, 1.01, 0.72, 0.43)
p = c(0.05, 0.031, 0.001, 0.09)
N = c(120, 86, 450, 123)
se.from.p(effect.size = effect.size, p = p, N = N,
    effect.size.type = "ratio")
```

`subgroup.analysis.mixed.effects` *Subgroup analysis using a mixed-effects model*

Description

This function performs a mixed-effects (random-effects model within subgroups, fixed-effect model between subgroups) subgroup analysis using `meta` objects.

Usage

```
subgroup.analysis.mixed.effects(x, subgroups, exclude = "none")
```

Arguments

`x`	An object of class `meta`, generated by the `metabin`, `metagen`, `metacont`, `metacor`, `metainc`, or `metaprop` function.
`subgroups`	A character vector of the same length as the number of studies within the meta-analysis, with a unique code for the subgroup each study belongs to. Must have the same order as the studies in the `meta` object.
`exclude`	Single string or concatenated array of strings. The name(s) of the subgroup levels to be excluded from the subgroup analysis. If `"none"` (default), all subgroup levels are used for the analysis.

Details

This function conducts a test for differences in effect sizes between subgroups of a meta-analysis. The function implements a mixed-effect model, in which the overall effect size for each subgroup is calculated using a random-effect model, and the test for subgroup differences is conducted using a fixed-effect model. The implementation follows the fixed-effects (plural) model described in Borenstein and Higgins (2013).

This model is appropriate for subgroup tests when the subgroup levels under study are assumed to be exhaustive for the characteristic at hand, and are not randomly chosen instances of a "population" of subgroup levels. For example, the fixed-effects (plural) model used in the function is valid when differences between studies published before and after a certain year are considered as a (binary) subgroup level. When subgroup levels can be assumed to be random samples from a distribution of subgroup levels, a random-effects model is more appropriate, and may be calculated using the `update.meta` function.

The function uses the study effect sizes `TE` and their standard error `seTE` of the provided `meta` object to perform the subgroup analyses. Specifications of the summary measure `sm` are inherited and used to backtransform log-transformed effect sizes to their original metrics if necessary.

Results can be inspected by plugging the function output into the `summary` function. Forest plots can be generated using `forest`. Additional arguments of the `forest.meta` function can be passed to the `forest` function for additional styling.

Value

Returns a `list` with five objects:

- within.subgroup.results: The pooled effect size for each subgroup and corresponding measures of heterogeneity (Q and I2). If the summary measure sm is defined as one of "RR", "RD", "OR", "ASD", "HR" or "IRR" in the meta object provided in x, the backtransformed (exponentiated) pooled effect for each subgroup effect size along with the 95% confidence interval is also provided.
- subgroup.analysis.results: The results for the Q-test for subgroup differences, its degrees of freedom df and *p*-value.
- m.random: An object of class meta containing the results of the random-effects model applied for pooling results in each subgroup in the first step.
- method.tau: The τ^2 estimator used for within-subgroup pooling (inherited from the meta object provided in x).
- k: The total number of included studies.

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 7.

Borenstein, M. & Higgins, J. P. T. (2013). Meta-Analysis and Subgroups. *Prevention Science, 14* (2): 134–43.

See Also

multimodel.inference

Examples

```
# Example 1: Hedges' g as effect size, precalculated effect sizes
suppressPackageStartupMessages(library(dmetar))
suppressPackageStartupMessages(library(meta))
data("ThirdWave")
ThirdWave = ThirdWave[c(1,2,3,5,9,18),]

m1 <- metagen(TE = TE,
              seTE = seTE,
              studlab = paste(ThirdWave$Author),
              data=ThirdWave,
              comb.fixed = FALSE,
              method.tau = "PM",
              sm = "SMD")

sgame1 = subgroup.analysis.mixed.effects(x = m1, subgroups = ThirdWave$TypeControlGroup)
summary(sgame1)

# Example 2: Hedges' g as effect size, raw effect data
suppressPackageStartupMessages(library(meta))
data(amlodipine)

# Create an arbitrary subgroup for illustration purposes
amlodipine$subgroup = rep(c("A","B"),4)
```

```
m2 <- metacont(n.amlo, mean.amlo, sqrt(var.amlo),
               n.plac, mean.plac, sqrt(var.plac),
               data=amlodipine, studlab=amlodipine$study,
               sm = "SMD")

sgame2 = subgroup.analysis.mixed.effects(x = m2, subgroups = amlodipine$subgroup)
summary(sgame2)

# Example 3: Risk ratio as effect size, binary outcome data, exlcude one level
suppressPackageStartupMessages(library(meta))
data(Olkin95)

# Create an arbitrary subgroup for illustration purposes
Olkin95$subgroup = c(rep(c("A","B"), 30), rep("C",10))

m3 <- metabin(event.e, n.e, event.c, n.c,
              data = Olkin95, studlab = Olkin95$author,
              method = "Inverse")

# Use shorthand
sgame3 = sgame(x = m3, subgroups = Olkin95$subgroup,
               exclude = "B")
summary(sgame3)


# Example 4: IRR as effect size, incidence data
suppressPackageStartupMessages(library(meta))
data(smoking)

# Create an arbitrary subgroup for illustration purposes
smoking$subgroup = c(rep(c("A"), 4), rep(c("B"), 3))

m4 <- metainc(d.smokers, py.smokers,
              d.nonsmokers, py.nonsmokers,
              data=smoking, studlab=study, sm="IRR")

sgame4 = subgroup.analysis.mixed.effects(x = m4, subgroups = smoking$subgroup)
summary(sgame4)

## Not run:
# Generate Forest Plot
# Additional arguments of the meta::forest.meta can be supplied
forest(sgame1, col.diamond = "darkgreen")
forest(sgame2)
forest(sgame3)
forest(sgame4)

## End(Not run)
```

---

sucra *Calculate the Surface Under the Cumulative Ranking score of from a network meta-analysis*

Description

This function calculates the SUCRA (Surface Under the Cumulative Ranking) score from a rank probability matrix or an object of class `mtc.rank.probability` generated by the `rank.probability` function.

Usage

```
sucra(x, lower.is.better = FALSE)
```

Arguments

`x`	An object of class `mtc.rank.probability` generated by the `rank.probability` function or a matrix/data.frame in which the rows correspond to the treatment, and columns to the probability of a specific treatment having this rank (see Details). Rownames of the matrix should contain the name of the specific treatment.
`lower.is.better`	Logical. Do lower (i.e., more negative) effect sizes mean that effects are higher? `FALSE` by default. Use the default when the provided matrix already contains the correct rank probability for each treatment, and values ought not to be inverted.

Details

The SUCRA score is a metric to evaluate which treatment in a network is likely to be the most efficacious in the context of network meta-analyses. The SUCRA score is calculated in the function using the formula described in Salanti, Ades and Ioannidis (2011):

$$SUCRA_j = \frac{\sum_{b=1}^{a-1} cum_{jb}}{a-1}$$

Where j is some treatment, a are all competing treatments, b are the $b = 1, 2, ..., a-1$ best treatments, and cum represents the cumulative probability of a treatment being among the b best treatments.

Other than an object of class `mtc.rank.probability` for argument `x`, the function can also be provided with a $m \times n$ matrix where m are rows corresponding to each treatment in the network meta-analysis, and the n columns correspond to each rank (1st, 2nd, etc.). Rank probabilities should be provided as a value from 0 to 1. Rownames of the matrix should correspond to the treatment names. Here is an example rank probability matrix for eight treatments:

.	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
CBT	0.000000	0.000000	0.000000	0.000000	0.000000	0.001275	0.087400	0.911325
IPT	0.000000	0.000000	0.000000	0.000000	0.000000	0.179400	0.745875	0.074725
PDT	0.000000	0.000000	0.000225	0.020300	0.978025	0.001450	0.000000	0.000000
PLA	0.002825	0.551175	0.262525	0.181550	0.001925	0.000000	0.000000	0.000000
PST	0.000000	0.000000	0.000000	0.000025	0.001450	0.817850	0.166725	0.013950
SUP	0.000000	0.216450	0.398700	0.383950	0.000900	0.000000	0.000000	0.000000
TAU	0.000375	0.229200	0.338525	0.414175	0.017700	0.000025	0.000000	0.000000
WLC	0.996800	0.003175	0.000025	0.000000	0.000000	0.000000	0.000000	0.000000

Author(s)

Mathias Harrer & David Daniel Ebert

References

Harrer, M., Cuijpers, P., Furukawa, T.A, & Ebert, D. D. (2019). *Doing Meta-Analysis in R: A Hands-on Guide*. DOI: 10.5281/zenodo.2551803. Chapter 11.2.

Salanti, G., Ades, A. E. & Ioannidis, J.P.A. (2011). Graphical Methods and Numerical Summaries for Presenting Results from Multiple-Treatment Meta-Analysis: An Overview and Tutorial. *Journal of Clinical Epidemiology, 64* (2): 163–71.

See Also

direct.evidence.plot

Examples

```
## Not run:
# Example1 : conduct NMA using gemtc, calculate SUCRAs
suppressPackageStartupMessages(library(gemtc))
suppressPackageStartupMessages(library(igraph))
data("NetDataGemtc")

network = suppressWarnings(mtc.network(data.re = NetDataGemtc))

plot(network, layout = layout.fruchterman.reingold)

model = mtc.model(network, linearModel = "fixed",
                  n.chain = 4,
                  likelihood = "normal",
                  link = "identity")

mcmc = mtc.run(model, n.adapt = 5000, n.iter = 100000, thin = 10)

rp = rank.probability(mcmc)

sucra = sucra(rp, lower.is.better = TRUE)
sucra
plot(sucra)
## End(Not run)



# Example 2: construct rank proabability matrix, then use sucra function
rp = rbind(CBT = c(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.001500, 0.088025, 0.910475),
       IPT = c(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.176975, 0.748300, 0.074725),
       PDT = c(0.000000, 0.000000, 0.000250, 0.021725, 0.976525, 0.001500, 0.000000, 0.000000),
       PLA = c(0.003350, 0.546075, 0.266125, 0.182125, 0.002325, 0.000000, 0.000000, 0.000000),
       PST = c(0.000000, 0.000000, 0.000000, 0.000000, 0.001500, 0.820025, 0.163675, 0.014800),
       SUP = c(0.000000, 0.217450, 0.403950, 0.378000, 0.000600, 0.000000, 0.000000, 0.000000),
       TAU = c(0.000225, 0.232900, 0.329675, 0.418150, 0.019050, 0.000000, 0.000000, 0.000000),
       WLC = c(0.996425, 0.003575, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000))

sucra(rp, lower.is.better = TRUE)
plot(sucra(rp, lower.is.better = TRUE))
```

summary.direct.evidence.plot *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'direct.evidence.plot'
summary(object, ...)
```

Arguments

`object`	An object of class `direct.evidence.plot`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical. When both `plot` and `forest` are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The `forest`/`plot` function allows additional arguments of the `meta.forest` or `metafor`'s codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

summary.eggers.test *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `eggers.test`, and `sucra`.

Usage

```
## S3 method for class 'eggers.test'
summary(object, ...)
```

Arguments

object	An object of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, eggers.test, or sucra.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

---

summary.find.outliers *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'find.outliers'
summary(object, ...)
```

Arguments

object	An object of class find.outliers.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical. When both plot and forest are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The forest/plot function allows additional arguments of the meta.forest or metafor's codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

`summary.gosh.diagnostics`

Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, `direct.evidence.plot`, `gosh.diagnostics` and `sucra`.

Usage

```
## S3 method for class 'gosh.diagnostics'
summary(object, ...)
```

Arguments

`object`	An object of class `gosh.diagnostics`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical. When both `plot` and `forest` are available for a class, outputs of both functions are identical (i.e., a forest plot is returned). The `forest/plot` function allows additional arguments of the `meta.forest` or `metafor`'s codeforest function (depending on the class of the meta-analysis object on which prior calculations are based on). These can be used for further styling of the forest plot.

Author(s)

Mathias Harrer & David Daniel Ebert

`summary.InfluenceAnalysis`

Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

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

Arguments

object	An object of class InfluenceAnalysis.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

---

summary.multimodel.inference

Print, summary and plot methods for objects created using 'dmetar' functions

---

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'multimodel.inference'
summary(object, ...)
```

Arguments

object	An object of class multimodel.inference.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

summary.pcurve *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

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

Arguments

object	An object of class pcurve.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

summary.power.analysis *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'power.analysis'
summary(object, ...)
```

Arguments

object	An object of class power.analysis.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

`summary.power.analysis.subgroup` *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class direct.evidence.plot, find.outliers, influence.analysis, multimodel.inference, pcurve, power.analysis, subgroup.analysis.mixed.effects, and sucra.

Usage

```
## S3 method for class 'power.analysis.subgroup'
summary(object, ...)
```

Arguments

object	An object of class power.analysis.subgroup.
...	Additional arguments.

Details

A total of four package-specific S3 methods are provided in dmetar: S3 methods for print, summary, plot and forest. Outputs from print and summary are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

summary.subgroup.analysis.mixed.effects *Print, summary and plot methods for objects created using 'dmetar' functions*

Description

Print, summary and plot S3 methods for objects of class `direct.evidence.plot`, `find.outliers`, `influence.analysis`, `multimodel.inference`, `pcurve`, `power.analysis`, `subgroup.analysis.mixed.effects`, and `sucra`.

Usage

```
## S3 method for class 'subgroup.analysis.mixed.effects'
summary(object, ...)
```

Arguments

`object`	An object of class `subgroup.analysis.mixed.effects`.
`...`	Additional arguments.

Details

A total of four package-specific S3 methods are provided in `dmetar`: S3 methods for `print`, `summary`, `plot` and `forest`. Outputs from `print` and `summary` are always identical.

Author(s)

Mathias Harrer & David Daniel Ebert

ThirdWave *'Third-Wave' cognitive behavioral interventions for perceived stress in college students dataset*

Description

This is a toy dataset containing pre-calculated effect size data of a meta-analysis on randomized controlled trials comparing the effectiveness of 'third-wave' CBT interventions for perceived stress in college students to inactive controls. Effect size data is provided as the standardized mean difference (SMD) between the intervention and control group and the corresponding standard error for each study at post. The dataset also contains columns for study characteristics which may serve as potential effect size moderators.

Usage

```
data(ThirdWave)
```

Format

A `data.frame` with 8 columns.

Author `character`. The study label containing the author(s) of the study.

TE `numeric`. The calculated standardized mean difference at post-test between the intervention and control group.

seTE `numeric`. The standard error of the standardized mean difference.

RiskOfBias `character`. The risk of bias rating according to the Cochrane Risk of Bias Tool.

TypeControlGroup `character`. The type of control group used in the study.

InterventionDuration `character`. The dichotomized duration of the intervention.

InterventionType `character`. The type of third-wave intervention rationale used.

ModeOfDelivery `character`. The mode of delivery used for the intervention.

Author(s)

Mathias Harrer, Eva-Maria Rathner, David Daniel Ebert

Source

Slightly changed dataset of a meta-analysis on third-wave CBT interventions for perceived stress in college students.

Index