Package 'robmed'

August 17, 2022

Type Package

Title (Robust) Mediation Analysis

Version 1.0.0

Date 2022-08-16

Depends R (>= 3.5.0), ggplot2 (>= 3.3.0), robustbase (>= 0.92-7)

Imports boot (>= 1.3-20), grid, methods, quantreg (>= 5.36), sn (>= 1.5-4), utils

Suggests knitr, testthat

Description Perform mediation analysis via a fast-and-robust bootstrap test, as well as various other methods. Details on the implementation and code examples can be found in Alfons, Ates, and Groenen (2022) <doi:10.18637/jss.v103.i13>.

License GPL (>= 2)

URL https://github.com/aalfons/robmed

BugReports https://github.com/aalfons/robmed/issues

LazyData yes

VignetteBuilder knitr

Author Andreas Alfons [aut, cre] (<https://orcid.org/0000-0002-2513-3788>), Nufer Y. Ates [ctb] (<https://orcid.org/0000-0003-4572-4101>, provided the BSG2014 data)

Maintainer Andreas Alfons <alfons@ese.eur.nl>

Encoding UTF-8

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-17 07:00:02 UTC

R topics documented:

`robmed-package` *(Robust) Mediation Analysis*

Description

Perform mediation analysis via a fast-and-robust bootstrap test, as well as various other methods. Details on the implementation and code examples can be found in Alfons, Ates, and Groenen (2022) <doi:10.18637/jss.v103.i13>.

Details

The DESCRIPTION file:

Package:	robmed
Type:	Package
Title:	(Robust) Mediation Analysis
Version:	1.0.0

Date:	2022-08-16
Depends:	R (>= 3.5.0), ggplot2 (>= 3.3.0), robustbase (>= 0.92-7)
Imports:	boot (>= 1.3-20), grid, methods, quantreg (>= 5.36), sn (>= 1.5-4), utils
Suggests:	knitr, testthat
Description:	Perform mediation analysis via a fast-and-robust bootstrap test, as well as various other methods. Details o
License:	GPL (>= 2)
URL:	https://github.com/aalfons/robmed
BugReports:	https://github.com/aalfons/robmed/issues
LazyData:	yes
VignetteBuilder:	knitr
Authors@R:	c(person("Andreas", "Alfons", email = "alfons@ese.eur.nl", role = c("aut", "cre"), comment = c(ORCID =
Author:	Andreas Alfons [aut, cre] (<https://orcid.org/0000-0002-2513-3788>), Nufer Y. Ates [ctb] (<https://orcid.o
Maintainer:	Andreas Alfons <alfons@ese.eur.nl>
Encoding:	UTF-8
RoxygenNote:	7.2.0

Index of help topics:

	weights
sim_mediation	Generate data from a fitted mediation model
summary.test_mediation	Summary of results from (robust) mediation analysis
test_mediation	(Robust) mediation analysis
weight_plot	Diagnostic plot of robust regression weights
weights.cov_Huber	Robustness weights of Huber M-estimation of location and scatter

Author(s)

Andreas Alfons [aut, cre] (<https://orcid.org/0000-0002-2513-3788>), Nufer Y. Ates [ctb] (<https://orcid.org/0000-0003-4572-4101>, provided the BSG2014 data)

Maintainer: Andreas Alfons <alfons@ese.eur.nl>

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) A Robust Bootstrap Test for Mediation Analysis. *Organizational Research Methods*, **25**(3), 591–617. doi:10.1177/1094428121999096.

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust mediation analysis: The R package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

---

boot_samples *Draw bootstrap samples*

Description

Draw bootstrap samples to be used for (fast-and-robust) bootstrap tests for mediation analysis. Note that this function is intended for use in simulation studies by experienced users.

Usage

```
boot_samples(n, R)
```

Arguments

n	an integer giving the number of observations in the original data set.
R	an integer giving the number of bootstrap samples to be generated.

Value

An object of class "boot_samples" with the following components:

indices	an integer matrix in which each column contains the indices of the corresponding bootstrap sample.
seed	the state of the random number generator before the bootstrap samples were drawn

Author(s)

Andreas Alfons

See Also

test_mediation()

Examples

```
# control parameters
n <- 100
a <- b <- c <- 0.4

# generate data
set.seed(20200309)
x <- rnorm(n)
m <- a * x + rnorm(n)
y <- b * m + c * x + rnorm(n)
simulated_data <- data.frame(x, y, m)

# perform boostrap tests
indices <- boot_samples(n, R = 5000)
robust_boot <- test_mediation(simulated_data,
                              x = "x", y = "y", m = "m",
                              robust = TRUE,
                              indices = indices)
summary(robust_boot)
ols_boot <- test_mediation(simulated_data,
                           x = "x", y = "y", m = "m",
                           robust = FALSE,
                           indices = indices)
summary(ols_boot)
```

BSG2014 *Business simulation game data*

Description

The data were collected from 354 senior business administration students during a business simulation game at a Western European University.

The game was played for a total of 12 rounds (i.e., two separate games of 6 rounds) as part of the capstone strategy class. Students were randomly assigned to teams of four, and surveyed in three waves: prior to the first game, in between the two games, and after the second game (with different variables being surveyed in the different waves).

The 354 students formed 92 teams, and the responses of individual students were aggregated to the team level. Leaving out teams with less than 50 percent response rate yields $n = 89$ teams. Only a small subset of the collected variables are included here.

Usage

```
data("BSG2014")
```

Format

A data frame with 89 observations on the following 13 variables.

`ProcessConflict` Based on Shah & Jehn (1993), the team members rated three items on the presence of conflict related to the process of working together, using a 5-point scale (1 = none, 5 = a lot). The individual responses were aggregated by taking the average across items and team members. Process conflict was measured in the second survey (between the two games).

`SharedExperience` As teams were randomly formed, no prior shared group experience is expected, and shared group experience and training is developed during the first game for the second game. Hence the team performance score on the first game is used as a proxy for the level of shared group experience and training. Those scores were computed through a mix of five objective performance measures: return on equity, earnings-per-share, stock price, credit rating, and image rating. The computation of the scores is handled by the simulation game software, and details can be found in Mathieu & Rapp (2009). The scores ranged from 57 to 111, and they were communicated to the teams only after the third survey.

`TaskConflict` Using the intra-group conflict scale of Jehn (1995), the team members rated four items on the presence of conflict regarding the work on a 5-point scale (1 = none, 5 = a lot). The individual responses were aggregated by taking the average across items and team members. Task conflict was measured in the second survey (between the two games).

`TeamCommitment` The team members indicated the extent to which they agree or disagree with four items on commitment to the team, which are based on Mowday, Steers & Porter (1979), using a 5-point scale (1 = strongly disagree, 5 = strongly agree). The individual responses were aggregated by taking the average across items and team members. Team commitment was measured in the third survey (after the second game).

`TeamPerformance` Following Hackman (1986), the team members indicated the extent to which they agree or disagree with four items on the team's functioning, using a 5-point scale (1 = strongly disagree, 5 = strongly agree). The individual responses were aggregated by taking the average across items and team members. Subjective team performance was measured in the third survey (after the second game).

`TMS` Transactive memory systems (TMS) are defined as shared systems that people in relationships develop for encoding, storing, and retrieving information about different substantive domains. TMS was operationalized with Lewis' (2003) 15-item scale that measures the three sub-dimensions of TMS (specialization, credibility, and coordination). For each item, the team members responded on a 5-point scale (1 = strongly disagree, 5 = strongly agree). Following Lewis (2003), the three sub dimensions were aggregated to form the TMS construct. That is, the individual responses were aggregated by taking the average across all 15 items and team members. TMS was measured in the second survey (between the two games).

`ValueDiversity` Using the short Schwartz's value survey (Lindeman & Verkasalo, 2005), the team members rated ten items on the importance of certain values (1 = not important, 10 = highly important). For each value item, the coefficient of variation of the individual responses across team members was computed, and the resulting coefficients of variation were averaged across the value items. Value diversity was measured in the first survey (before the first game).

`ProceduralJustice` Based on the intra-unit procedural justice climate scale of Li & Cropanzano (2009), the team members indicated the extent to which they agree or disagree with four items on a 5-point scale (1 = strongly disagree, 5 = strongly agree). The individual responses were aggregated by taking the average across items and team members. Procedural justice was measured in the third survey (after the second game).

`InteractionalJustice` Using the intra-unit interactional justice climate scale of Li & Cropanzano (2009), the team members indicated the extent to which they agree or disagree with four items on a 5-point scale (1 = strongly disagree, 5 = strongly agree). The individual responses were aggregated by taking the average across items and team members. Interactional justice was measured in the third survey (after the second game).

`SharedLeadership` Following Carson, Tesluk & Marrone (2007), every team member assessed each of their peers on the question of 'To what degree does your team rely on this individual for leadership?' using a 5-point scale (1 = not at all, 5 = to a very large extent). The leadership ratings were aggregated by taking the sum and dividing it by the number of pairwise relationships among team members. Shared leadership was measured in the second survey (between the two games).

`AgeDiversity` Following Harrison & Klein (2007), age diversity was operationalized by the coefficient of variation of the team members' ages.

`GenderDiversity` Gender diversity was measured with Blau's index, $1 - \sum_j p_j^2$, where p_j is the proportion of team members in the j-th category (Blau, 1977).

`TeamScore` The team performance scores on the second game were computed at the end of the simulation through a mix of five objective performance measures: return on equity, earnings-per-share, stock price, credit rating, and image rating. The computation of the scores is handled by the simulation game software, and details can be found in Mathieu & Rapp (2009). The scores ranged from 49 to 110, and they were communicated to the teams only after the third survey.

Source

The data were collected and provided by Nufer Y. Ates (https://orcid.org/0000-0003-4572-4101).

References

Blau, P.M. (1977) *Inequality and Heterogeneity: A Primitive Theory of Social Structure*. New York, NY: Free Press.

Carson, J.B., Tesluk, P.E. and Marrone, J.A. (2007) Shared Leadership in Teams: An Investigation of Antecedent Conditions and Performance. *Academy of Management Journal*, **50**(5), 1217–1234. doi:10.5465/amj.2007.20159921.

Hackman, J.R. (1986) The Psychology of Self-Management in Organizations. In Pallack, M.S and Perloff, R.O. (Eds.), *Psychology and Work: Productivity, Change, and Employment*, 89–136. Washington, DC: American Psychological Association.

Harrison, D.A. and Klein, K.J. (2007) What's the Difference? Diversity Constructs as Separation, Variety, or Disparity in Organizations. *Academy of Management Review*, **32**(4): 1199–1228. doi:10.5465/amr.2007.26586096.

Jehn, K.A. (1995) A Multimethod Examination of the Benefits and Detriments of Intragroup Conflict. *Administrative Science Quarterly*, **40**(2), 256–285. doi:10.2307/2393638.

Lewis, K. (2003) Measuring Transactive Memory Systems in the Field: Scale Development and Validation. *Journal of Applied Psychology*, **88**(4), 587–604. doi:10.1037/0021-9010.88.4.587.

Li, A. and Cropanzano, R. (2009) Fairness at the Group Level: Justice Climate and Intraunit Justice Climate. *Journal of Management*, **35**(3), 564–599. doi:10.1177/0149206308330557.

Lindeman, M. and Verkasalo, M. (2005) Measuring Values With the Short Schwartz's Value Survey. *Journal of Personality Assessment*, **85**(2), 170–178. doi:10.1207/s15327752jpa8502_09.

Mathieu, J.E. and Rapp, T.L. (2009). Laying the Foundation for Successful Team Performance Trajectories: The Roles of Team Charters and Performance Strategies. *Journal of Applied Psychology*, **94**(1), 90–103. doi:10.1037/a0013257.

Mowday, R.T., Steers, R.M. and Porter, L.W. (1979) The Measurement of Organizational Commitment. *Journal of Vocational Behavior*, **14**(2), 224–247. doi:10.1016/0001-8791(79)90072-1.

Shah, P.P. and Jehn, K.A. (1993) Do Friends Perform Better than Acquaintances? The Interaction of Friendship, Conflict, and Task. *Group Decision and Negotiation*, **2**(2), 149–165. doi:10.1007/bf01884769.

Examples

```
data("BSG2014")
summary(BSG2014)

# scatterplot matrix for the variables included in the
# illustrative mediation analysis
x <- "ValueDiversity"
y <- "TeamCommitment"
m <- "TaskConflict"
plot(BSG2014[, c(x, y, m)], pch = 21, bg = "black")
```

---

ci_plot *Dot plot with confidence intervals*

Description

Produce a dot plot with confidence intervals of selected effects from (robust) mediation analysis. In addition to confidence intervals, p-values of the selected effects can be plotted as well.

Usage

```
ci_plot(object, ...)

## Default S3 method:
ci_plot(object, parm = c("direct", "indirect"), ...)

## S3 method for class 'boot_test_mediation'
ci_plot(
  object,
  parm = c("direct", "indirect"),
  type = c("boot", "data"),
```

```
  p_value = FALSE,
  digits = 4L,
  ...
)

## S3 method for class 'sobel_test_mediation'
ci_plot(
  object,
  parm = c("direct", "indirect"),
  level = 0.95,
  p_value = FALSE,
  ...
)

## S3 method for class 'list'
ci_plot(
  object,
  parm = c("direct", "indirect"),
  type = c("boot", "data"),
  level = 0.95,
  p_value = FALSE,
  digits = 4L,
  ...
)

## S3 method for class 'setup_ci_plot'
ci_plot(object, ...)
```

Arguments

`object`	an object inheriting from class `"test_mediation"` containing results from (robust) mediation analysis, or a list of such objects.
`...`	additional arguments to be passed down.
`parm`	an integer, character or logical vector specifying which effects to include in the plot. In case of a character vector, possible values are `"a"`, `"b"`, `"d"` (only serial multiple mediator models), `"total"`, `"direct"`, and `"indirect"`. The default is to include the direct and the indirect effect(s).
`type`	a character string specifying which point estimates and confidence intervals to plot: those based on the bootstrap distribution (`"boot"`; the default), or those based on the original data (`"data"`). If `"boot"`, the confidence intervals of effects other than the indirect effect(s) are computed using a normal approximation (i.e., assuming a normal distribution of the corresponding effect with the standard deviation computed from the bootstrap replicates). If `"data"`, the confidence intervals of effects other than the indirect effect(s) are computed via statistical theory based on the original data (e.g., based on a t-distribution if the coefficients are estimated via regression). Note that this is only relevant for mediation analysis via a bootstrap test, where the confidence interval of the indirect

effect is always computed via a percentile-based method due to the asymmetry of its distribution.

`p_value` a logical indicating whether to include dot plots of the p-values in addition to those with confidence intervals. The default is `FALSE`.

`digits` an integer determining how many digits to compute for bootstrap p-values of the indirect effects (see `p_value()`). The default is to compute 4 digits after the comma. This is only relevant if `p_value = TRUE`.

`level` numeric; the confidence level of the confidence intervals from Sobel's test. The default is to include 95% confidence intervals. Note that this is not used for bootstrap tests, as those require to specify the confidence level already in `test_mediation()`.

Details

Methods first call `setup_ci_plot()` to extract all necessary information to produce the plot, then the `"setup_ci_plot"` method is called to produce the plot.

Value

An object of class `"ggplot"`.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

`test_mediation()`, `setup_ci_plot()`

`density_plot()`, `ellipse_plot()`, `weight_plot()`, `plot()`

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
robust_boot <- test_mediation(BSG2014,
                              x = "ValueDiversity",
                              y = "TeamCommitment",
                              m = "TaskConflict",
                              robust = TRUE)

# create plot for robust bootstrap test
ci_plot(robust_boot)
ci_plot(robust_boot, color = "#00BFC4")
```

```
# run OLS bootstrap test
ols_boot <- test_mediation(BSG2014,
                           x = "ValueDiversity",
                           y = "TeamCommitment",
                           m = "TaskConflict",
                           robust = FALSE)

# compare robust and OLS bootstrap tests
boot_list <- list("OLS bootstrap" = ols_boot,
                  "ROBMED" = robust_boot)
ci_plot(boot_list)

# the plot can be customized in the usual way
ci_plot(boot_list) +
  geom_hline(yintercept = 0, color = "darkgrey") +
  coord_flip() + theme_bw() +
  labs(title = "OLS bootstrap vs ROBMED")
```

`coef.test_mediation` *Coefficients in (robust) mediation analysis*

Description

Extract coefficients from models computed in (robust) mediation analysis.

Usage

```
## S3 method for class 'test_mediation'
coef(object, parm = NULL, ...)

## S3 method for class 'boot_test_mediation'
coef(object, parm = NULL, type = c("boot", "data"), ...)

## S3 method for class 'fit_mediation'
coef(object, parm = NULL, ...)
```

Arguments

`object`	an object inheriting from class `"test_mediation"` containing results from (robust) mediation analysis, or an object inheriting from class `"fit_mediation"` containing a (robust) mediation model fit.
`parm`	an integer, character or logical vector specifying the paths for which to extract coefficients, or `NULL` to extract all coefficients. In case of a character vector, possible values are `"a"`, `"b"`, `"d"` (only serial multiple mediator models), `"total"`, `"direct"`, and `"indirect"`.
`...`	additional arguments are currently ignored.

type — a character string specifying whether to extract the means of the bootstrap distribution ("boot"; the default), or the coefficient estimates based on the original data set ("data").

Value

A numeric vector containing the requested coefficients.

Author(s)

Andreas Alfons

See Also

test_mediation(), fit_mediation(), confint(), p_value()

Examples

```
data("BSG2014")

# fit robust mediation model and extract coefficients
fit <- fit_mediation(BSG2014,
                     x = "ValueDiversity",
                     y = "TeamCommitment",
                     m = "TaskConflict")
coef(fit)

# run fast-and-robust bootstrap test and extract coefficients
boot <- test_mediation(fit)
coef(boot, type = "data")  # from original sample
coef(boot, type = "boot")  # means of bootstrap replicates
```

---

confint.test_mediation

Confidence intervals from (robust) mediation analysis

Description

Extract or compute confidence intervals for effects in (robust) mediation analysis.

Usage

```
## S3 method for class 'boot_test_mediation'
confint(object, parm = NULL, level = NULL, type = c("boot", "data"), ...)

## S3 method for class 'sobel_test_mediation'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

object	an object inheriting from class "test_mediation" containing results from (robust) mediation analysis.
parm	an integer, character or logical vector specifying the paths for which to extract or compute confidence intervals, or NULL to extract or compute confidence intervals for all coefficients. In case of a character vector, possible values are "a", "b", "d" (only serial multiple mediator models), "total", "direct", and "indirect".
level	for the "boot_test_mediation" method, this is ignored and the confidence level of the bootstrap confidence interval for the indirect effect is used. For the other methods, the confidence level of the confidence intervals to be computed. The default is to compute 95% confidence intervals.
type	a character string specifying how to compute the confidence interval of the effects other than the indirect effect(s). Possible values are "boot" (the default) to compute bootstrap confidence intervals using the normal approximation (i.e., to assume a normal distribution of the corresponding effect with the standard deviation computed from the bootstrap replicates), or "data" to compute confidence intervals via statistical theory based on the original data (e.g., based on a t-distribution if the coefficients are estimated via regression). Note that this is only relevant for mediation analysis via a bootstrap test, where the confidence interval of the indirect effect is always computed via a percentile-based method due to the asymmetry of its distribution.
...	additional arguments are currently ignored.

Value

A numeric matrix containing the requested confidence intervals.

Author(s)

Andreas Alfons

See Also

test_mediation(), coef(), p_value(), boot.ci()

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
robust_boot <- test_mediation(BSG2014,
                              x = "ValueDiversity",
                              y = "TeamCommitment",
                              m = "TaskConflict",
                              robust = TRUE)
confint(robust_boot, type = "boot")
```

```
# run OLS bootstrap test
ols_boot <- test_mediation(BSG2014,
                           x = "ValueDiversity",
                           y = "TeamCommitment",
                           m = "TaskConflict",
                           robust = FALSE)
confint(ols_boot, type = "data")
```

`cov_control` *Tuning parameters for Huber M-estimation of location and scatter*

Description

Obtain a list with tuning paramters for `cov_Huber()`.

Usage

```
cov_control(prob = 0.95, max_iterations = 200, tol = 1e-07)
```

Arguments

`prob`	numeric; probability for the quantile of the χ^2 distribution to be used as cutoff point in the Huber weight function (defaults to 0.95).
`max_iterations`	an integer giving the maximum number of iterations in the iteratively reweighted algorithm.
`tol`	a small positive numeric value to be used to determine convergence of the iteratively reweighted algorithm.

Value

A list with components corresponding to the arguments.

Author(s)

Andreas Alfons

References

Huber, P.J. (1981) *Robust Statistics*. John Wiley & Sons.

See Also

`cov_Huber()`

Examples

```
data("BSG2014")

# run bootstrap test after winsorization
ctrl <- cov_control(prob = 0.95)
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict",
                       method = "covariance",
                       control = ctrl)
summary(boot)
```

---

cov_Huber *Huber M-estimator of location and scatter*

Description

Compute a Huber M-estimator of location and scatter, which is reasonably robust for a small number of variables.

Usage

```
cov_Huber(x, control = cov_control(...), ...)
```

Arguments

x	a numeric matrix or data frame.
control	a list of tuning parameters as generated by cov_control().
...	additional arguments can be used to specify tuning parameters directly instead of via control.

Details

An iterative reweighting algorithm is used to compute the Huber M-estimator. The Huber weight function thereby corresponds to a convex optimization problem, resulting in a unique solution.

Value

An object of class "cov_Huber" with the following components:

center	a numeric vector containing the location vector estimate.
cov	a numeric matrix containing the scatter matrix estimate.
prob	numeric; probability for the quantile of the χ^2 distribution used as cutoff point in the Huber weight function.
weights	a numeric vector containing the relative robustness weights for the observations.

tau	numeric; correction for Fisher consistency under multivariate normal distributions.
converged	a logical indicating whether the iterative reweighting algorithm converged.
iterations	an integer giving the number of iterations required to obtain the solution.

Author(s)

Andreas Alfons

References

Huber, P.J. (1981) *Robust Statistics*. John Wiley & Sons.

Zu, J. and Yuan, K.-H. (2010) Local Influence and Robust Procedures for Mediation Analysis. *Multivariate Behavioral Research*, **45**(1), 1–44. doi:10.1080/00273170903504695.

See Also

cov_control(), test_mediation(), fit_mediation()

Examples

```
data("BSG2014")

# define variables
x <- "ValueDiversity"
y <- "TeamCommitment"
m <- "TaskConflict"

# compute Huber M-estimator
cov_Huber(BSG2014[, c(x, y, m)])
```

---

cov_ML *Maximum likelihood estimator of mean vector and covariance matrix*

Description

Compute the maximum likelihood estimator of the mean vector and the covariance matrix.

Usage

```
cov_ML(x, ...)
```

Arguments

x	a numeric matrix or data frame.
...	additional arguments are currently ignored.

Value

An object of class "cov_ML" with the following components:

center	a numeric vector containing the mean vector estimate.
cov	a numeric matrix containing the covariance matrix estimate.
n	an integer giving the number of observations.

Author(s)

Andreas Alfons

References

Zu, J. and Yuan, K.-H. (2010) Local Influence and Robust Procedures for Mediation Analysis. *Multivariate Behavioral Research*, **45**(1), 1–44. doi:10.1080/00273170903504695.

See Also

test_mediation(), fit_mediation()

Examples

```
data("BSG2014")

# define variables
x <- "ValueDiversity"
y <- "TeamCommitment"
m <- "TaskConflict"

# compute Huber M-estimator
cov_ML(BSG2014[, c(x, y, m)])
```

---

density_plot *Density plot of the indirect effect(s)*

Description

Produce a density plot of the indirect effect(s) from (robust) mediation analysis. In addition to the density, a vertical line representing the point estimate and a shaded area representing the confidence interval are drawn.

Usage

```
density_plot(object, ...)

## Default S3 method:
density_plot(object, ...)

## S3 method for class 'sobel_test_mediation'
density_plot(object, grid = NULL, level = 0.95, ...)

## S3 method for class 'list'
density_plot(object, grid = NULL, level = 0.95, ...)

## S3 method for class 'setup_density_plot'
density_plot(object, ...)
```

Arguments

`object`	an object inheriting from class `"test_mediation"` containing results from (robust) mediation analysis, or a list of such objects.
`...`	additional arguments to be passed down.
`grid`	an optional numeric vector containing the values at which to evaluate the assumed normal density from Sobel's test. The default is to take 512 equally spaced points between the estimated indirect effect $\pm$ three times the standard error according to Sobel's formula.
`level`	numeric; the confidence level of the confidence intervals from Sobel's test. The default is to include 95% confidence intervals. Note that this is not used for bootstrap tests, as those require to specify the confidence level already in `test_mediation()`.

Details

Methods first call `setup_density_plot()` to extract all necessary information to produce the plot, then the `"setup_density_plot"` method is called to produce the plot.

Value

An object of class `"ggplot"`.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

test_mediation(), setup_density_plot()

ci_plot(), ellipse_plot(), weight_plot(), plot()

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
robust_boot <- test_mediation(BSG2014,
                              x = "ValueDiversity",
                              y = "TeamCommitment",
                              m = "TaskConflict",
                              robust = TRUE)

# create plot for robust bootstrap test
density_plot(robust_boot)
density_plot(robust_boot, color = "#00BFC4", fill = "#00BFC4")

# run OLS bootstrap test
ols_boot <- test_mediation(BSG2014,
                           x = "ValueDiversity",
                           y = "TeamCommitment",
                           m = "TaskConflict",
                           robust = FALSE)

# compare robust and OLS bootstrap tests
boot_list <- list("OLS bootstrap" = ols_boot,
                  "ROBMED" = robust_boot)
density_plot(boot_list)

# the plot can be customized in the usual way
density_plot(boot_list) + theme_bw() +
  labs(title = "OLS bootstrap vs ROBMED")
```

---

ellipse_plot *Diagnostic plot with a tolerance ellipse*

Description

Produce a scatter plot of two variables used in (robust) mediation analysis together with a tolerance ellipse. Exploiting the relationship between the regression coefficients and the covariance matrix, that tolerance ellipse illustrates how well the regression results represent the data. In addition, a line that visualizes the estimated regression coefficient is added when relevant.

Usage

```
ellipse_plot(object, ...)

## Default S3 method:
ellipse_plot(
  object,
  horizontal = NULL,
  vertical = NULL,
  partial = FALSE,
  level = 0.975,
  npoints = 100,
  ...
)

## S3 method for class 'setup_ellipse_plot'
ellipse_plot(object, ...)
```

Arguments

object	an object inheriting from class "fit_mediation" or "test_mediation" containing results from (robust) mediation analysis, or a list of such objects.
...	additional arguments to be passed down.
horizontal	a character string specifying the variable to be plotted on the horizontal axis. If the dependent variable is chosen for the vertical axis, a hypothsized mediator or an independent variable must be selected for the horizontal axis. If a hypothesized mediator is chosen for the vertical axis, an independent variable must be selected for the horizontal axis (in case of a serial multiple mediator model, a hypothesized mediator occurring earlier in the sequence is also allowed). The default is to plot the first independent variable on the horizontal axis.
vertical	a character string specifying the variable to be plotted on the vertical axis: the dependent variable or a hypothesized mediator. The default is to plot the first hypothesized mediator on the vertical axis.
partial	a logical indicating whether the vertical axis should display the observed values of the selected variable (FALSE), or the partial residuals with respect to the variable on the horizontal axis (TRUE). The latter allows to display the corresponding regression coefficient by a line.
level	numeric; the confidence level of the tolerance ellipse. It gives the percentage of observations that are expected to lie within the ellipse under the assumption of a normal distribution, and therefore it controls the size of the ellipse. The default is such that the ellipse is expected to contain 97.5% of the observations.
npoints	the number of grid points used to evaluate and draw the ellipse. The default is to use 100 grid points.

Details

A line to visualize the corresponding regression coefficient is added if partial = TRUE, or in case of a simple mediation model (without control variables) when the hypothesized mediator is plotted

on the vertical axis and the independent variable is plotted on the horizontal axis.

For robust estimation methods that return outlyingness weights for each data point, those weights are visualized by coloring the points on a grey scale. If a list of objects has been supplied and there are multiple objects from such robust methods, each method is placed in a separate panel.

Methods first call `setup_ellipse_plot()` to extract all necessary information to produce the plot, then the `"setup_ellipse_plot"` method is called to produce the plot.

Value

An object of class `"ggplot"`.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

`fit_mediation()`, `test_mediation()`, `setup_ellipse_plot()`

`ci_plot()`, `density_plot()`, `weight_plot()`, `plot()`

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
robust_boot <- test_mediation(BSG2014,
                              x = "ValueDiversity",
                              y = "TeamCommitment",
                              m = "TaskConflict",
                              robust = TRUE)

# create plot for robust bootstrap test
ellipse_plot(robust_boot)

# original data and partial residuals
ellipse_plot(robust_boot, horizontal = "TaskConflict",
             vertical = "TeamCommitment")
ellipse_plot(robust_boot, horizontal = "TaskConflict",
             vertical = "TeamCommitment", partial = TRUE)

# run OLS bootstrap test
ols_boot <- test_mediation(BSG2014,
                           x = "ValueDiversity",
                           y = "TeamCommitment",
                           m = "TaskConflict",
                           robust = FALSE)
```

```
# compare robust and OLS bootstrap tests
boot_list <- list("OLS bootstrap" = ols_boot,
                  "ROBMED" = robust_boot)
ellipse_plot(boot_list)

# the plot can be customized in the usual way
ellipse_plot(boot_list) + theme_bw() +
  labs(title = "OLS vs robust estimation")
```

`fit_mediation` *(Robustly) fit a mediation model*

Description

(Robustly) estimate the effects in a mediation model.

Usage

```
fit_mediation(object, ...)

## S3 method for class 'formula'
fit_mediation(formula, data, ...)

## Default S3 method:
fit_mediation(
  object,
  x,
  y,
  m,
  covariates = NULL,
  method = c("regression", "covariance"),
  robust = TRUE,
  family = "gaussian",
  model = c("parallel", "serial"),
  contrast = FALSE,
  fit_yx = TRUE,
  control = NULL,
  ...
)
```

Arguments

`object`	the first argument will determine the method of the generic function to be dispatched. For the default method, this should be a data frame containing the variables.

`...` additional arguments to be passed down. For the default method, this can be used to specify tuning parameters directly instead of via `control`.

`formula` an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Hypothesized mediator variables should be wrapped in a call to `m()` (see examples), and any optional control variables should be wrapped in a call to `covariates()`.

`data` for the `formula` method, a data frame containing the variables.

`x` a character, integer or logical vector specifying the columns of `object` containing the independent variables of interest.

`y` a character string, an integer or a logical vector specifying the column of `object` containing the dependent variable.

`m` a character, integer or logical vector specifying the columns of `object` containing the hypothesized mediator variables.

`covariates` optional; a character, integer or logical vector specifying the columns of `object` containing additional covariates to be used as control variables.

`method` a character string specifying the method of estimation. Possible values are `"regression"` (the default) to estimate the effects via regressions, or `"covariance"` to estimate the effects via the covariance matrix. Note that the effects are always estimated via regressions if more than one independent variable or hypothesized mediator is specified, or if control variables are supplied.

`robust` a logical indicating whether to robustly estimate the effects (defaults to `TRUE`). For estimation via regressions (`method = "regression"`), this can also be a character string, with `"MM"` specifying the MM-estimator of regression, and `"median"` specifying median regression.

`family` a character string specifying the error distribution to be used in maximum likelihood estimation of regression models. Possible values are `"gaussian"` for a normal distribution (the default), `skewnormal` for a skew-normal distribution, `"student"` for Student's t distribution, `"skewt"` for a skew-t distribution, or `"select"` to select among these four distributions via BIC (see 'Details'). This is only relevant if `method = "regression"` and `robust = FALSE`.

`model` a character string specifying the type of model in case of multiple mediators. Possible values are `"parallel"` (the default) for the parallel multiple mediator model, or `"serial"` for the serial multiple mediator model. This is only relevant for models with multiple hypothesized mediators, which are currently only implemented for estimation via regressions (`method = "regression"`).

`contrast` a logical indicating whether to compute pairwise contrasts of the indirect effects (defaults to `FALSE`). This can also be a character string, with `"estimates"` for computing the pairwise differences of the indirect effects, and `"absolute"` for computing the pairwise differences of the absolute values of the indirect effects. This is only relevant for models with multiple indirect effects, which are currently only implemented for estimation via regressions (`method = "regression"`). For models with multiple independent variables of interest and multiple hypothesized mediators, contrasts are only computed between indirect effects corresponding to the same independent variable.

`fit_yx` a logical indicating whether to fit the regression model `y ~ x + covariates` to estimate the total effect (the default is `TRUE`). This is only relevant if `method = "regression"` and `robust = FALSE`.

`control` a list of tuning parameters for the corresponding robust method. For robust regression (`method = "regression"`, and `robust = TRUE` or `robust = "MM"`), a list of tuning parameters for `lmrob()` as generated by `reg_control()`. For winsorized covariance matrix estimation (`method = "covariance"` and `robust = TRUE`), a list of tuning parameters for `cov_Huber()` as generated by `cov_control()`. No tuning parameters are necessary for median regression (`method = "regression"` and `robust = "median"`).

Details

With `method = "regression"`, and `robust = TRUE` or `robust = "MM"`, the effects are computed via the robust MM-estimator of regression from `lmrob()`. This is the default behavior.

With `method = "regression"` and `robust = "median"`, the effects are estimated via median regressions with `rq()`. Unlike the robust MM-regressions above, median regressions are not robust against outliers in the explanatory variables.

With `method = "regression"`, `robust = FALSE` and `family = "select"`, the error distribution to be used in maximum likelihood estimation of the regression models is selected via BIC. The following error distributions are included in the selection procedure: a normal distribution, a skew-normal distribution, Student's t distribution, and a skew-t distribution. Note that the parameters of those distributions are estimated as well. The skew-normal and skew-t distributions thereby use a centered parametrization such that the residuals are (approximately) centered around 0. Moreover, the skew-t distribution is only evaluated in the selection procedure if both the skew-normal and Student's t distribution yield an improvement in BIC over the normal distribution. Otherwise the estimation with a skew-t error distribution can be unstable. Furthermore, this saves a considerable amount of computation time in a bootstrap test, as estimation with those error distributions is orders of magnitude slower than any other implemented estimation procedure.

With `method = "covariance"` and `robust = TRUE`, the effects are estimated based on a Huber M-estimator of location and scatter. Note that this covariance-based approach is less robust than the approach based on robust MM-regressions described above.

Value

An object inheriting from class `"fit_mediation"` (class `"reg_fit_mediation"` if `method = "regression"` or `"cov_fit_mediation"` if `method = "covariance"`) with the following components:

`a` a numeric vector containing the point estimates of the effects of the independent variables on the proposed mediator variables.

`b` a numeric vector containing the point estimates of the direct effects of the proposed mediator variables on the dependent variable.

`d` in case of a serial multiple mediator model, a numeric vector containing the point estimates of the effects of proposed mediator variables on other mediator variables occurring later in the sequence (only `"reg_fit_mediation"` if applicable).

`total` a numeric vector containing the point estimates of the total effects of the independent variables on the dependent variable.

`direct` a numeric vector containing the point estimates of the direct effects of the independent variables on the dependent variable.

`indirect` a numeric vector containing the point estimates of the indirect effects.

`ab` for back-compatibility with versions <0.10.0, the point estimates of the indirect effects are also included here. **This component is deprecated and may be removed as soon as the next version.**

`fit_mx` an object of class `"lmrob"`, `"rq"`, `"lm"` or `"lmse"` containing the estimation results from the regression of the proposed mediator variable on the independent variables, or a list of such objects in case of more than one hypothesized mediator (only `"reg_fit_mediation"`).

`fit_ymx` an object of class `"lmrob"`, `"rq"`, `"lm"` or `"lmse"` containing the estimation results from the regression of the dependent variable on the proposed mediator and independent variables (only `"reg_fit_mediation"`).

`fit_yx` an object of class `"lm"` or `"lmse"` containing the estimation results from the regression of the dependent variable on the independent variables (only `"reg_fit_mediation"` if arguments `robust = FALSE` and `fit_yx = TRUE` were used).

`cov` an object of class `"cov_Huber"` or `"cov_ML"` containing the covariance matrix estimates (only `"cov_fit_mediation"`).

`x, y, m, covariates` character vectors specifying the respective variables used.

`data` a data frame containing the independent, dependent and proposed mediator variables, as well as covariates.

`robust` either a logical indicating whether the effects were estimated robustly, or one of the character strings `"MM"` and `"median"` specifying the type of robust regressions.

`model` a character string specifying the type of mediation model fitted: `"simple"` in case of one independent variable and one hypothesized mediator, `"multiple"` in case of multiple independent variables and one hypothesized mediator, `"parallel"` in case of parallel multiple mediators, or `"serial"` in case of serial multiple mediators (only `"reg_fit_mediation"`).

`contrast` either a logical indicating whether contrasts of the indirect effects were computed, or one of the character strings `"estimates"` and `"absolute"` specifying the type of contrasts of the indirect effects (only `"reg_fit_mediation"`).

`control` a list of tuning parameters used (if applicable).

Mediation models

The following mediation models are implemented. In the regression equations below, the i_j are intercepts and the e_j are random error terms.

- *Simple mediation model*: The mediatiom model in its simplest form is given by the equations

$$M = i_1 + aX + e_1,$$

$$Y = i_2 + bM + cX + e_2,$$

$$Y = i_3 + c'X + e_3,$$

where Y denotes the dependent variable, X the independent variable, and M the hypothesized mediator. The main parameter of interest is the product of coefficients ab, called the indirect effect. The coefficients c and c' are called the direct and total effect, respectively.

- *Parallel multiple mediator model*: The simple mediation model can be extended with multiple mediators $M_1, \ldots, M_k$ in the following way:

$$M_1 = i_1 + a_1 X + e_1,$$

$$\vdots$$

$$M_k = i_k + a_k X + e_k,$$

$$Y = i_{k+1} + b_1 M_1 + \ldots + b_k M_k + cX + e_{k+1},$$

$$Y = i_{k+2} + c' X + e_{k+2}.$$

  The main parameters of interest are the individual indirect effects $a_1 b_1, \ldots, a_k b_k$.

- *Serial multiple mediator model*: It differs from the parallel multiple mediator model in that it allows the hypothesized mediators $M_1, \ldots, M_k$ to influence each other in a sequential manner. It is given by the equations

$$M_1 = i_1 + a_1 X + e_1,$$

$$M_2 = i_1 + d_{21} M_1 + a_2 X + e_2,$$

$$\vdots$$

$$M_k = i_k + d_{k1} M_1 + \ldots + d_{k,k-1} M_{k-1} + a_k X + e_k,$$

$$Y = i_{k+1} + b_1 M_1 + \ldots + b_k M_k + cX + e_{k+1},$$

$$Y = i_{k+2} + c' X + e_{k+2}.$$

  The serial multiple mediator model quickly grows in complexity with increasing number of mediators due to the combinatorial increase in indirect paths through the mediators. It is therefore only implemented for two and three mediators to maintain a focus on easily interpretable models. For two serial mediators, the three indirect effects $a_1 b_1$, $a_2 b_2$, and $a_1 d_{21} b_2$ are the main parameters of interest. For three serial mediators, there are already seven indirect effects: $a_1 b_1$, $a_2 b_2$, $a_3 b_3$, $a_1 d_{21} b_2$, $a_1 d_{31} b_3$, $a_2 d_{32} b_3$, and $a_1 d_{21} d_{32} b_3$.

- *Multiple independent variables to be mediated*: The simple mediation model can also be extended by allowing multiple independent variables $X_1, \ldots, X_l$ instead of multiple mediators. It is defined by the equations

$$M = i_1 + a_1 X_1 + \ldots + a_l X_l + e_1,$$

$$Y = i_2 + bM + c_1 X_1 + \ldots + c_l X_l + e_2,$$

$$Y = i_3 + c_1' X_1 + \ldots + c_l' X_l + e_3.$$

  The indirect effects $a_1 b, \ldots, a_l b$ are the main parameters of interest. Note that an important special case of this model occurs when a categorical independent variable is represented by a group of dummy variables.

- *Control variables*: To isolate the effects of the independent variables of interest from other factors, control variables can be added to all regression equations of a mediation model. Note that that there is no intrinsic difference between independent variables of interest and control variables in terms of the model or its estimation. The difference is purely conceptual in nature: for the control variables, the estimates of the direct and indirect paths are not of particular interest to the researcher. Control variables can therefore be specified separately from the independent variables of interest. Only for the latter, results for the indirect effects are included in the output.
- *More complex models*: Some of the models described above can be combined, for instance parallel and serial multiple mediator models support multiple independent variables of interest.

Note

The default method takes a data frame its first argument so that it can easily be used with the pipe operator (R's built-in `|>` or **magrittr**'s `%>%`).

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) A Robust Bootstrap Test for Mediation Analysis. *Organizational Research Methods*, **25**(3), 591–617. doi:10.1177/1094428121999096.

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

Azzalini, A. and Arellano-Valle, R. B. (2013) Maximum Penalized Likelihood Estimation for Skew-Normal and Skew-t Distributions. *Journal of Statistical Planning and Inference*, **143**(2), 419–433. doi:10.1016/j.jspi.2012.06.022.

Yuan, Y. and MacKinnon, D.P. (2014) Robust Mediation Analysis Based on Median Regression. *Psychological Methods*, **19**(1), 1–20. doi:10.1037/a0033820.

Zu, J. and Yuan, K.-H. (2010) Local Influence and Robust Procedures for Mediation Analysis. *Multivariate Behavioral Research*, **45**(1), 1–44. doi:10.1080/00273170903504695.

See Also

`test_mediation()`

`lmrob()`, `lm()`, `cov_Huber()`, `cov_ML()`

Examples

```
data("BSG2014")

## seed to be used for the random number generator
seed <- 20211117

## simple mediation
# set seed of the random number generator
```

```
set.seed(seed)
# The results in Alfons et al. (2021) were obtained with an
# older version of the random number generator.  To reproduce
# those results, uncomment the two lines below.
# RNGversion("3.5.3")
# set.seed(20150601)
# perform mediation analysis
fit_simple <- fit_mediation(TeamCommitment ~
                              m(TaskConflict) +
                              ValueDiversity,
                            data = BSG2014)
boot_simple <- test_mediation(fit_simple)
summary(boot_simple)


## serial multiple mediators
# set seed of the random number generator
set.seed(seed)
# perform mediation analysis
fit_serial <- fit_mediation(TeamScore ~
                              serial_m(TaskConflict,
                                       TeamCommitment) +
                              ValueDiversity,
                            data = BSG2014)
boot_serial <- test_mediation(fit_serial)
summary(boot_serial)

## parallel multiple mediators and control variables
# set seed of the random number generator
set.seed(seed)
# perform mediation analysis
fit_parallel <- fit_mediation(TeamPerformance ~
                                parallel_m(ProceduralJustice,
                                           InteractionalJustice) +
                                SharedLeadership +
                                covariates(AgeDiversity,
                                           GenderDiversity),
                              data = BSG2014)
boot_parallel <- test_mediation(fit_parallel)
summary(boot_parallel)
```

---

m *Create an object of hypothesized mediators or control variables*

Description

`m()` and its wrappers `parallel_m()` and `serial_m()` create an object of hypothesized mediators, while `covariates()` creates an object of control variables. Usually, these are used in a formula specifying a mediation model.

Usage

```
m(..., .model = c("parallel", "serial"))

parallel_m(...)

serial_m(...)

covariates(...)
```

Arguments

...	variables are supplied as arguments, as usual separated by a comma.
.model	a character string specifying the type of model in case of multiple mediators. Possible values are "parallel" (the default) for the parallel multiple mediator model, or "serial" for the serial multiple mediator model.

Details

m() and covariates() are essentially wrappers for cbind() with a specific class prepended to the class(es) of the resulting object.

parallel_m() and serial_m() are wrappers for m() with the respective value for argument .model.

Value

m() returns an object inheriting from class "mediators" (with subclass "parallel_mediators" or "serial_mediators" as specified by argument .model), and covariates() returns an object of class "covariates". Typically, these inherit from class "matrix".

Author(s)

Andreas Alfons

See Also

fit_mediation(), test_mediation()

Examples

```
data("BSG2014")

# inside formula
fit_mediation(TeamCommitment ~ m(TaskConflict) + ValueDiversity,
              data = BSG2014)

# outside formula
mediator <- with(BSG2014, m(TaskConflict))
fit_mediation(TeamCommitment ~ mediator + ValueDiversity,
              data = BSG2014)
```

plot-methods *Plot (robust) mediation analysis results*

Description

Visualize results from (robust) mediation analysis.

Usage

```
## S3 method for class 'fit_mediation'
autoplot(object, which = c("ellipse", "weight"), ...)

## S3 method for class 'test_mediation'
autoplot(object, which = c("ci", "density", "ellipse", "weight"), ...)

## S3 method for class 'fit_mediation'
plot(x, which = c("ellipse", "weight"), ...)

## S3 method for class 'test_mediation'
plot(x, which = c("ci", "density", "ellipse", "weight"), ...)
```

Arguments

object, x	an object inheriting from class "fit_mediation" or "test_mediation" containing results from (robust) mediation analysis.
which	a character string specifying which plot to produce. Possible values are "ci" for a dot plot of selected effects together with confidence intervals (see ci_plot()), "density" for a density plot of the indirect effect(s) (see density_plot()), "ellipse" for a diagnostic plot of the data together with a tolerance ellipse (see ellipse_plot()), and "weight" for a diagnostic plot of robust regression weights (see weight_plot()).
...	additional arguments to be passed down.

Details

The "fit_mediation" method calls ellipse_plot() or weight_plot(), depending on the argument which.

The "test_mediation" method calls ci_plot(), density_plot(), ellipse_plot(), or weight_plot(), depending on the argument which.

Value

An object of class "ggplot".

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

`fit_mediation()`, `test_mediation()`

`ci_plot()`, `density_plot()`, `ellipse_plot()`, `weight_plot()`

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")

# create plots for robust bootstrap test
plot(boot, which = "ci")
plot(boot, which = "density")
plot(boot, which = "ellipse")
plot(boot, which = "weight")
```

---

p_value *p-Values from (robust) mediation analysis*

Description

Compute or extract the p-values for effects in (robust) mediation analysis.

Usage

```
p_value(object, ...)

## S3 method for class 'boot_test_mediation'
p_value(object, parm = NULL, type = c("boot", "data"), digits = 4L, ...)

## S3 method for class 'sobel_test_mediation'
p_value(object, parm = NULL, ...)
```

Arguments

`object`	an object inheriting from class `"test_mediation"` containing results from (robust) mediation analysis.
`...`	for the generic function, additional arguments to be passed down to methods. For the methods, additional arguments are currently ignored.
`parm`	an integer, character or logical vector specifying the paths for which to extract or compute p-values, or `NULL` to extract or compute p-values for all coefficients. In case of a character vector, possible values are `"a"`, `"b"`, `"d"` (only serial multiple mediator models), `"total"`, `"direct"`, and `"indirect"`.
`type`	a character string specifying how to compute the p-values of the effects other than the indirect effect(s). Possible values are `"boot"` (the default) to compute bootstrap p-values using the normal approximation (i.e., to assume a normal distribution of the corresponding effect with the standard deviation computed from the bootstrap replicates), or `"data"` to compute p-values via statistical theory based on the original data (e.g., based on a t-distribution if the coefficients are estimated via regression). Note that this is only relevant for mediation analysis via a bootstrap test, where the p-value of the indirect effect is always computed as described in 'Details'.
`digits`	an integer determining how many digits to compute for the p-values of the indirect effects (see 'Details'). The default is to compute 4 digits after the comma.

Details

For bootstrap tests, the p-value of the indirect effect is computed as the smallest significance level α for which the $(1 - \alpha) * 100\%$ confidence interval obtained from the bootstrapped distribution does not contain 0.

This is a simple implementation, where each digit after the comma is determined via a grid search. Hence computation time can be long if confidence intervals are computed via the bias-corrected and accelerated method (`"bca"`).

For Sobel tests, the p-value of the indirect effect is already stored in the object returned by `test_mediation()` and is simply extracted.

Value

A numeric vector containing the requested p-values.

Author(s)

Andreas Alfons

See Also

`test_mediation()`, `coef()`, `confint()`

Examples

```
data("BSG2014")


# BCa intervals are recommended, but take a while to run
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict",
                       type = "bca")
p_value(boot)
```

`reg_control` *Tuning parameters for MM-regression*

Description

Obtain a list with tuning paramters for `lmrob()`.

Usage

```
reg_control(efficiency = 0.85, max_iterations = 200, tol = 1e-07, seed = NULL)
```

Arguments

`efficiency`	a numeric value giving the desired efficiency (defaults to 0.85 for 85% efficiency).
`max_iterations`	an integer giving the maximum number of iterations in various parts of the algorithm.
`tol`	a small positive numeric value to be used to determine convergence in various parts of the algorithm.
`seed`	optional initial seed for the random number generator (see `.Random.seed`).

Value

A list of tuning parameters as returned by `lmrob.control()`.

Note

This is a simplified wrapper function for `lmrob.control()`, as the latter requires detailed knowledge of the MM-type regression algorithm. Currently only 95%, 90%, 85% (the default) and 80% efficiency are supported. For other values, please specify the corresponding tuning parameters in `lmrob.control()` directly.

Author(s)

Andreas Alfons

References

Salibian-Barrera, M. and Yohai, V.J. (2006) A Fast Algorithm for S-regression Estimates. *Journal of Computational and Graphical Statistics*, **15**(2), 414–427. doi:10.1198/106186006x113629.

Yohai, V.J. (1987) High Breakdown-Point and High Efficiency Estimates for Regression. *The Annals of Statistics*, **15**(20), 642–656. doi:10.1214/aos/1176350366.

See Also

`lmrob()`, `lmrob.control()`

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
ctrl <- reg_control(efficiency = 0.95)
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict",
                       control = ctrl)
summary(boot)
```

---

retest *Retest for mediation*

Description

Re-perform a test for the indirect effect(s) based on results from (robust) mediation analysis. This function is purely available for computational convenience if the analysis was accidentally run with the wrong parameter settings, as it avoids having to re-run the bootstrap procedure. It must not be abused for p-hacking.

Usage

```
retest(object, ...)

## S3 method for class 'boot_test_mediation'
retest(object, alternative, level, type, contrast, ...)

## S3 method for class 'sobel_test_mediation'
retest(object, alternative, order, ...)
```

Arguments

object	an object inheriting from class "test_mediation" containing results from (robust) mediation analysis.
...	additional arguments to be passed down to methods.
alternative	a character string specifying the alternative hypothesis in the test for the indirect effect. Possible values are "twosided", "less" or "greater".
level	numeric; the confidence level of the confidence interval in the bootstrap test.
type	a character string specifying the type of confidence interval to be computed in the bootstrap test. Possible values are "bca" for the bias-corrected and accelerated bootstrap, or "perc" for the percentile bootstrap.
contrast	a logical indicating whether to compute pairwise contrasts of the indirect effects. This can also be a character string, with "estimates" for computing the pairwise differences of the indirect effects (such that it is tested whether two indirect effects are equal), and "absolute" for computing the pairwise differences of the absolute values of the indirect effects (such that it is tested whether two indirect effects are equal in magnitude). This is only relevant for models with multiple indirect effects, which are currently only implemented for bootstrap tests and estimation via regressions.
order	a character string specifying the order of approximation of the standard error in Sobel's test. Possible values are "first" for a first-order approximation, and "second" for a second-order approximation.

Value

An object of the same class as object with updated test results (see test_mediation()).

Note

From version 0.9.0 onwards, the behavior of this function changed. For arguments that are not supplied, the corresponding values of object are now used as defaults.

Author(s)

Andreas Alfons

See Also

test_mediation()

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")
```

```
summary(boot)

# now compute 97.5% confidence interval
retest(boot, level = 0.975)
```

`setup_ci_plot` *Set up information for a dot plot with confidence intervals*

Description

Extract the relevant information for a dot plot with confidence intervals of selected effects from (robust) mediation analysis. Information on p-values of the selected effects can be included in addition to confidence intervals.

Usage

```
setup_ci_plot(object, ...)

## S3 method for class 'boot_test_mediation'
setup_ci_plot(
  object,
  parm = c("direct", "indirect"),
  type = c("boot", "data"),
  p_value = FALSE,
  digits = 4L,
  ...
)

## S3 method for class 'sobel_test_mediation'
setup_ci_plot(
  object,
  parm = c("direct", "indirect"),
  level = 0.95,
  p_value = FALSE,
  ...
)

## S3 method for class 'list'
setup_ci_plot(object, ...)
```

Arguments

`object`	an object inheriting from class "`test_mediation`" containing results from (robust) mediation analysis, or a list of such objects.
`...`	additional arguments to be passed down.

parm — an integer, character or logical vector specifying which effects to include in the plot. In case of a character vector, possible values are "a", "b", "d" (only serial multiple mediator models), "total", "direct", and "indirect". The default is to include the direct and the indirect effect(s).

type — a character string specifying which point estiamates and confidence intervals to plot: those based on the bootstrap distribution ("boot"; the default), or those based on the original data ("data"). If "boot", the confidence intervals of effects other than the indirect effect(s) are computed using a normal approximation (i.e., assuming a normal distribution of the corresponding effect with the standard deviation computed from the bootstrap replicates). If "data", the confidence intervals of effects other than the indirect effect(s) are computed via statistical theory based on the original data (e.g., based on a t-distribution if the coefficients are estimated via regression). Note that this is only relevant for mediation analysis via a bootstrap test, where the confidence interval of the indirect effect is always computed via a percentile-based method due to the asymmetry of its distribution.

p_value — a logical indicating whether to include information on the p-values in addition to the confidence intervals. The default is FALSE.

digits — an integer determining how many digits to compute for bootstrap p-values of the indirect effects (see p_value()). The default is to compute 4 digits after the comma. This is only relevant if p_value = TRUE.

level — numeric; the confidence level of the confidence intervals from Sobel's test. The default is to include 95% confidence intervals. Note that this is not used for bootstrap tests, as those require to specify the confidence level already in test_mediation().

Details

This function is used internally by ci_plot(). It may also be useful for users who want to produce a similar plot, but who want more control over what information to display or how to display that information.

Value

An object of class "setup_ci_plot" with the following components:

ci — a data frame consisting of column Effect indicating the different effects, column Estimate containing the point estimates, column Lower for the lower confidence limit, and column Upper for the upper confidence limit. If argument p_value = TRUE, there is an additional column Label which gives the default facet label for the confidence intervals. If a list of "test_mediation" objects has been supplied, there is also a column Method, which takes the names or indices of the list elements to indicate the different methods.

p_value — a data frame consisting of column Label which gives the default facet label for the p-values, column Effect indicating the different effects, and column Value containing the p-values. If a list of "test_mediation" objects has been supplied, there is also a column Method, which takes the names or indices of the

	list elements to indicate the different methods. This is only returned if argument `p_value = TRUE`.
`level`	numeric; the confidence level used for the confidence intervals of the indirect effect(s).
`have_methods`	a logical indicating whether a list of "`test_mediation`" objects has been supplied. If `TRUE`, the data frame in the `ci` component contains a column `Method`.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

`test_mediation()`, `ci_plot()`

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")

# set up information for plot
setup <- setup_ci_plot(boot, parm = "Indirect")

# plot only density and confidence interval
ggplot() +
  geom_hline(yintercept = 0, color = "darkgrey") +
  geom_pointrange(aes(x = "Robust bootstrap", y = Estimate,
                      ymin = Lower, ymax = Upper),
                  data = setup$ci) +
  labs(x = NULL, y = "Indirect effect")
```

---

setup_density_plot *Set up information for a density plot of the indirect effect(s)*

Description

Extract the relevant information for a density plot of the indirect effect(s) from results of (robust) mediation analysis.

Usage

```
setup_density_plot(object, ...)

## S3 method for class 'boot_test_mediation'
setup_density_plot(object, ...)

## S3 method for class 'sobel_test_mediation'
setup_density_plot(object, grid = NULL, level = 0.95, ...)

## S3 method for class 'list'
setup_density_plot(object, ...)
```

Arguments

object	an object inheriting from class "test_mediation" containing results from (robust) mediation analysis, or a list of such objects.
...	additional arguments to be passed down.
grid	an optional numeric vector containing the values at which to evaluate the assumed normal density from Sobel's test. The default is to take 512 equally spaced points between the estimated indirect effect $\pm$ three times the standard error according to Sobel's formula.
level	numeric; the confidence level of the confidence intervals from Sobel's test. The default is to include 95% confidence intervals. Note that this is not used for bootstrap tests, as those require to specify the confidence level already in test_mediation().

Details

This function is used internally by density_plot(). It may also be useful for users who want to produce a similar plot, but who want more control over what information to display or how to display that information.

Value

An object of class "setup_density_plot" with the following components:

density	a data frame containing the values of the indirect effect where the density is estimated (column Indirect), and the estimated density values (column Density). In case of a model with multiple indirect effects, there is a column Effect that indicates the different indirect effects. If a list of "test_mediation" objects has been supplied, there is also a column Method, which takes the names or indices of the list elements to indicate the different methods.
ci	a data frame consisting of column Estimate containing the point estimates, column Lower for the lower confidence limit, and column Upper for the upper confidence limit. In case of a model with multiple indirect effects, there is a column Effect that indicates the different indirect effects. If a list of "test_mediation" objects has been supplied, there is also a column Method,

which takes the names or indices of the list elements to indicate the different methods.

`test` a character string indicating whether the object contains results from a bootstrap test (`"boot"`) or a Sobel test (`"sobel"`), or a vector of such character strings if a list of `"test_mediation"` objects has been supplied.

`level` numeric; the confidence level used for the confidence intervals of the indirect effect(s).

`have_effects` a logical indicating whether the mediation model contains multiple indirect effects If `TRUE`, the data frames in the `density` and `ci` components contain a column `Effect`.

`have_methods` a logical indicating whether a list of `"test_mediation"` objects has been supplied. If `TRUE`, the data frames in the `density` and `ci` components contain a column `Method`.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

`test_mediation()`, `density_plot()`

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")

# set up information for plot
setup <- setup_density_plot(boot)

# plot only density and confidence interval
ggplot() +
  geom_density(aes(x = Indirect, y = Density), data = setup$density,
               stat = "identity") +
  geom_rect(aes(xmin = Lower, xmax = Upper,
                ymin = -Inf, ymax = Inf),
            data = setup$ci, color = NA, alpha = 0.2) +
  labs(x = "Indirect effect", y = "Bootstrap density")
```

`setup_ellipse_plot` *Set up a diagnostic plot with a tolerance ellipse*

Description

Extract the relevant information for a diagnostic plot with a tolerance ellipse from results of (robust) mediation analysis.

Usage

```
setup_ellipse_plot(object, ...)

## S3 method for class 'test_mediation'
setup_ellipse_plot(object, ...)

## S3 method for class 'reg_fit_mediation'
setup_ellipse_plot(
  object,
  horizontal = NULL,
  vertical = NULL,
  partial = FALSE,
  level = 0.975,
  npoints = 100,
  ...
)

## S3 method for class 'cov_fit_mediation'
setup_ellipse_plot(
  object,
  horizontal = NULL,
  vertical = NULL,
  partial = FALSE,
  level = 0.975,
  npoints = 100,
  ...
)

## S3 method for class 'list'
setup_ellipse_plot(object, ...)
```

Arguments

`object`	an object inheriting from class "`fit_mediation`" or "`test_mediation`" containing results from (robust) mediation analysis, or a list of such objects.
`...`	additional arguments to be passed down.

`horizontal` a character string specifying the variable to be plotted on the horizontal axis. If the dependent variable is chosen for the vertical axis, a hypothsized mediator or an independent variable must be selected for the horizontal axis. If a hypothesized mediator is chosen for the vertical axis, an independent variable must be selected for the horizontal axis (in case of a serial multiple mediator model, a hypothesized mediator occurring earlier in the sequence is also allowed). The default is to plot the first independent variable on the horizontal axis.

`vertical` a character string specifying the variable to be plotted on the vertical axis: the dependent variable or a hypothesized mediator. The default is to plot the first hypothesized mediator on the vertical axis.

`partial` a logical indicating whether to extract the observed values of the selected variable for the vertical axis (`FALSE`), or the partial residuals with respect to the variable on the horizontal axis (`TRUE`). The latter allows to display the corresponding regression coefficient by a line.

`level` numeric; the confidence level of the tolerance ellipse. It gives the percentage of observations that are expected to lie within the ellipse under the assumption of a normal distribution, and therefore it controls the size of the ellipse. The default is such that the ellipse is expected to contain 97.5% of the observations.

`npoints` the number of grid points used to evaluate the ellipse. The default is to use 100 grid points.

Details

This function is used internally by `ellipse_plot()`. It may also be useful for users who want to produce a similar plot, but who want more control over what information to display or how to display that information.

Value

An object of class `"setup_ellipse_plot"` with the following components:

`data` a data frame containing the coordinates of the data points to be plotted on the horizontal axis (column `x`) and the coordinates on the vertical axis (column `y`). For robust methods that assign outlyingness weights to each data point, those weights are given in column `Weight`. If a list of objects has been supplied and there are multiple objects from such robust methods, or if partial residuals are to be plotted on the vertical axis, there is also a column `Method`, which takes the names or indices of the list elements to indicate the different methods.

`ellipse` a data frame containing the coordinates of the tolerance ellipse on the horizontal axis (column `x`) and on the vertical axis (column `y`). If a list of objects has been supplied, there is also a column `Method`, which takes the names or indices of the list elements to indicate the different methods.

`line` a data frame with columns `intercept` and `slope` containing the intercept and slope, respectively, of the regression line to be plotted. If a list of objects has been supplied, there is also a column `Method`, which takes the names or indices of the list elements to indicate the different methods. This is only returned if

	partial = TRUE, or in case of a simple mediation model (without control variables) when the hypothesized mediator is plotted on the vertical axis and the independent variable is plotted on the horizontal axis.
horizontal	a character string giving the variable to be plotted on the horizontal axis.
vertical	a character string giving the variable to be plotted on the vertical axis
partial	a logical indicating whether the values to be plotted on the vertical axis correspond to the observed values of the selected variable (FALSE), or the partial residuals with respect to the variable on the horizontal axis (TRUE).
robust	a logical indicating whether the object contains results from a robust method, or a vector of such logicals if a list of objects has been supplied.
have_methods	a logical indicating whether a list of objects has been supplied.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

fit_mediation(), test_mediation(), ellipse_plot()

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")

# set up information for plot
setup <- setup_ellipse_plot(boot)

# plot only data and tolerance ellipse
ggplot() +
  geom_path(aes(x = x, y = y), data = setup$ellipse,
            color = "#00BFC4") +
  geom_point(aes(x = x, y = y, fill = Weight),
             data = setup$data, shape = 21) +
  scale_fill_gradient(limits = 0:1, low = "white",
                      high = "black") +
  labs(x = setup$horizontal, y = setup$vertical)
```

`setup_weight_plot` *Set up a diagnostic plot of robust regression weights*

Description

Extract the relevant information for a diagnostic plot of the regression weights from robust mediation analysis. This plot allows to easily detect deviations from normality assumptions such as skewness or heavy tails.

Usage

```
setup_weight_plot(object, ...)

## S3 method for class 'test_mediation'
setup_weight_plot(object, ...)

## S3 method for class 'reg_fit_mediation'
setup_weight_plot(object, outcome = NULL, npoints = 1000, ...)
```

Arguments

`object`	an object inheriting from class `"fit_mediation"` or `"test_mediation"` containing results from robust mediation analysis. Only mediation analysis objects fitted with the robust MM-estimator are supported.
`...`	additional arguments to be passed down.
`outcome`	a character vector specifying the outcome variables of the regressions to be included in the plot. This must be a subset of the hypothesized mediators and the dependent variable, or `NULL` (the default) to include all regressions of the mediation model.
`npoints`	the number of grid points used to evaluate the expected percentages. The default is to use 1000 grid points.

Details

This function is used internally by `weight_plot()`. It may also be useful for users who want to produce a similar plot, but who want more control over what information to display or how to display that information.

Value

An object of class `"setup_weight_plot"` with the following components:

`data`	a data frame containing the following information: the outcome variable of the regression (column `Outcome`; only if multiple regressions are to be included in the plot), whether the row corresponds to the negative or the positive tail of the residual distribution (column `Tail`), whether the row corresponds to the expected (under the normal distribution) or the empirical weights (column `Weights`),

the weight thresholds (column `Threshold`), and the corresponding percentage of observations that have a weight below this threshold (column `Percentage`).

`outcome` a character vector containing the outcome variables of the regressions to be included in the plot.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

`fit_mediation()`, `test_mediation()`, `weight_plot()`

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")

# set up information for plot
setup <- setup_weight_plot(boot)
# create diagnostic plot of robust regression weights
weight_plot(setup) +
  scale_color_manual("", values = c("black", "#00BFC4")) +
  theme(legend.position = "top")
```

`sim_mediation` *Generate data from a fitted mediation model*

Description

Generate data from a fitted mediation model, using the obtained coefficient estimates as the true model coefficients for data generation.

Usage

```
sim_mediation(object, n, ...)

## S3 method for class 'fit_mediation'
sim_mediation(
  object,
  n = NULL,
  explanatory = c("sim", "boot"),
  errors = c("sim", "boot"),
  num_discrete = 10,
  ...
)

## S3 method for class 'test_mediation'
sim_mediation(object, n = NULL, ...)

rmediation(n, object, ...)
```

Arguments

object	an object inheriting from class "fit_mediation" or "test_mediation" containing results from (robust) mediation analysis.
n	an integer giving the number of observations to be generated. If NULL (the default), the number of observations is taken from the data set used in the fitted mediation model from object.
...	additional arguments to be passed down.
explanatory	a character string specifying how to generate the explanatory variables (i.e., the independent variables and additional covariates). Possible values are "sim" to draw each explanatory variable independently from a certain distribution (the default), or "boot" to bootstrap the explanatory variables from the observed data (i.e., random sampling with replacement). See 'Details' for more information on how the data are generated.
errors	a character string specifying how to generate the error terms in the linear models for the mediators and the dependent variable. Possible values are "sim" to draw the error terms independently from the respective fitted model distribution (the default), or "boot" to bootstrap the error terms from the observed residuals in the respective fitted model (i.e., random sampling with replacement). See 'Details' for more information on how the data are generated.
num_discrete	integer; if the explanatory variables are drawn from distributions (explanatory = "sim"), variables that take num_discrete or fewer values are considered discrete (the default is 10). In that case, the corresponding variables are drawn from multinomial distributions with the relative frequencies from the observed data. This is only relevant if the mediation model was fitted via regressions and ignored if the mediation model was fitted via the covariance matrix, as the latter method assumes multivariate normality.

Details

The data generating process consists of three basic steps:

1. Generate the explanatory variables (i.e., the independent variables and additional covariates).
2. Generate the error terms of the different regression models.
3. Generate the mediators and the dependent variable from the respective regression models, using the coefficient estimates from the fitted mediation model as the true model coefficients.

If `explanatory = "sim"`, the explanatory variables are simulated as follows. For each variable, a regression on a constant term is performed, using the same estimator and assumed error distribution as in the fitted mediation model from `object`. Typically, the assumed error distribution is normal, but it can also be a skew-normal, t, or skew-t distribution, or a selection of the best-fitting error distribution. Using the obtained location estimate and parameter estimates of the assumed error distribution, values are drawn from this error distribution and added to the location estimate. It is important to note that all explanatory variables are simulated independently from each other, hence there are no correlations between the explanatory variables.

In order to generate correlated explanatory variables, it is recommended bootstrap the explanatory variables from the observed data by setting `explanatory = "boot"`.

If `errors = "sim"`, the error terms of the different regression models are drawn from the assumed error distribution in the fitted mediation model from `object`, using the respective parameter estimates. Typically, the assumed error distribution is normal, but it can also be a skew-normal, t, or skew-t distribution, or a selection of the best-fitting error distribution.

If `errors = "boot"`, bootstrapping the error terms from the observed residuals is done independently for the different regression models and, if also `explanatory = "boot"`, independently from bootstrapping the explanatory variables.

The `"boot_test_mediation"` method for results of a bootstrap test always uses the regression coefficient estimates obtained on the original data for data generation, not the bootstrap estimates. Keep in mind that all bootstrap estimates are the means of the respective bootstrap replicates. If the bootstrap estimates of the regression coefficients were used to generate the data, the true values of the indirect effects for the generated data (i.e., the products of the corresponding bootstrap coefficient estimates) would not be equal to the reported bootstrap estimates of the indirect effects in `object`, which could lead to confusion. For the estimates on the original data, it of course holds that the estimates of indirect effects are the products of the corresponding coefficient estimates.

Value

A data frame with `n` observations containing simulated data for the variables of the fitted mediation model.

Mediation models

The following mediation models are implemented. In the regression equations below, the i_j are intercepts and the e_j are random error terms.

- *Simple mediation model*: The mediatiom model in its simplest form is given by the equations

$$M = i_1 + aX + e_1,$$

$$Y = i_2 + bM + cX + e_2,$$
$$Y = i_3 + c'X + e_3,$$

where Y denotes the dependent variable, X the independent variable, and M the hypothesized mediator. The main parameter of interest is the product of coefficients ab, called the indirect effect. The coefficients c and c' are called the direct and total effect, respectively.

- *Parallel multiple mediator model*: The simple mediation model can be extended with multiple mediators $M_1, \ldots, M_k$ in the following way:

$$M_1 = i_1 + a_1X + e_1,$$
$$\vdots$$
$$M_k = i_k + a_kX + e_k,$$
$$Y = i_{k+1} + b_1M_1 + \ldots + b_kM_k + cX + e_{k+1},$$
$$Y = i_{k+2} + c'X + e_{k+2}.$$

  The main parameters of interest are the individual indirect effects $a_1b_1, \ldots, a_kb_k$.

- *Serial multiple mediator model*: It differs from the parallel multiple mediator model in that it allows the hypothesized mediators $M_1, \ldots, M_k$ to influence each other in a sequential manner. It is given by the equations

$$M_1 = i_1 + a_1X + e_1,$$
$$M_2 = i_1 + d_{21}M_1 + a_2X + e_2,$$
$$\vdots$$
$$M_k = i_k + d_{k1}M_1 + \ldots + d_{k,k-1}M_{k-1} + a_kX + e_k,$$
$$Y = i_{k+1} + b_1M_1 + \ldots + b_kM_k + cX + e_{k+1},$$
$$Y = i_{k+2} + c'X + e_{k+2}.$$

  The serial multiple mediator model quickly grows in complexity with increasing number of mediators due to the combinatorial increase in indirect paths through the mediators. It is therefore only implemented for two and three mediators to maintain a focus on easily interpretable models. For two serial mediators, the three indirect effects a_1b_1, a_2b_2, and $a_1d_{21}b_2$ are the main parameters of interest. For three serial mediators, there are already seven indirect effects: a_1b_1, a_2b_2, a_3b_3, $a_1d_{21}b_2$, $a_1d_{31}b_3$, $a_2d_{32}b_3$, and $a_1d_{21}d_{32}b_3$.

- *Multiple independent variables to be mediated*: The simple mediation model can also be extended by allowing multiple independent variables $X_1, \ldots, X_l$ instead of multiple mediators. It is defined by the equations

$$M = i_1 + a_1X_1 + \ldots + a_lX_l + e_1,$$
$$Y = i_2 + bM + c_1X_1 + \ldots + c_lX_l + e_2,$$
$$Y = i_3 + c_1'X_1 + \ldots + c_l'X_l + e_3.$$

  The indirect effects $a_1b, \ldots, a_lb$ are the main parameters of interest. Note that an important special case of this model occurs when a categorical independent variable is represented by a group of dummy variables.

- *Control variables*: To isolate the effects of the independent variables of interest from other factors, control variables can be added to all regression equations of a mediation model. Note that that there is no intrinsic difference between independent variables of interest and control variables in terms of the model or its estimation. The difference is purely conceptual in nature: for the control variables, the estimates of the direct and indirect paths are not of particular interest to the researcher. Control variables can therefore be specified separately from the independent variables of interest. Only for the latter, results for the indirect effects are included in the output.
- *More complex models*: Some of the models described above can be combined, for instance parallel and serial multiple mediator models support multiple independent variables of interest.

Note

Function `sim_mediation()` takes the object containing results from mediation analysis as its first argument so that it can easily be used with the pipe operator (R's built-in `|>` or **magrittr**'s `%>%`).

Function `rmediation()` is a wrapper conforming with the naming convention for functions that generate data, as well as the convention of those function to take the number of observations as the first argument.

Author(s)

Andreas Alfons

See Also

`fit_mediation()`, `test_mediation()`

Examples

```
data("BSG2014")

## simple mediation
# fit the mediation model
fit_simple <- fit_mediation(BSG2014,
                            x = "ValueDiversity",
                            y = "TeamCommitment",
                            m = "TaskConflict")
# simulate data from the fitted mediation model
sim_simple <- sim_mediation(fit_simple, n = 100)
head(sim_simple)

## serial multiple mediators
# fit the mediation model
fit_serial <- fit_mediation(BSG2014,
                            x = "ValueDiversity",
                            y = "TeamScore",
                            m = c("TaskConflict",
                                  "TeamCommitment"),
                            model = "serial")
# simulate data from the fitted mediation model
sim_serial <- sim_mediation(fit_serial, n = 100)
```

```
head(sim_serial)

## parallel multiple mediators and control variables
# fit the mediation model
fit_parallel <- fit_mediation(BSG2014,
                              x = "SharedLeadership",
                              y = "TeamPerformance",
                              m = c("ProceduralJustice",
                                    "InteractionalJustice"),
                              covariates = c("AgeDiversity",
                                             "GenderDiversity"),
                              model = "parallel")
# simulate data from the fitted mediation model
# (here the explanatory variables are bootstrapped
# to maintain the correlations between them)
sim_parallel <- sim_mediation(fit_parallel, n = 100,
                              explanatory = "boot")
head(sim_parallel)
```

`summary.test_mediation` *Summary of results from (robust) mediation analysis*

Description

Summarize results from (robust) mediation analysis for proper interpretation.

Usage

```
## S3 method for class 'boot_test_mediation'
summary(object, type = c("boot", "data"), plot = TRUE, ...)

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

Arguments

`object`	an object inheriting from class `"test_mediation"` containing results from (robust) mediation analysis.
`type`	a character string specifying how to summarize the effects other than the indirect effect(s). Possible values are `"boot"` (the default) to compute significance tests using the normal approximation of the bootstrap distribution (i.e., to assume a normal distribution of the corresponding effect with the standard deviation computed from the bootstrap replicates), or `"data"` to compute significance tests via statistical theory based on the original data (e.g., t-tests if the coefficients are estimated via regression). Note that this is only relevant for mediation analysis via a bootstrap test, where significance of the indirect effect is always assessed via a percentile-based confidence interval due to the asymmetry of its distribution.

plot	a logical indicating whether to include a diagnostic plot of robust regression weights (see weight_plot()). This is only used for mediation analysis objects fitted with the robust MM-estimator (see test_mediation()). Note that the diagnostic plot is only shown when the returned object is printed in order to maintain a clear separation between computing results and printing/plotting them.
...	additional arguments are currently ignored.

Value

An object of class "summary_test_mediation" with the following components:

object	the object passed to the summary method, which contains the results from testing the indirect effect(s).
summary	an object containing all necessary information to summarize the effects other than the indirect effect(s).
plot	if applicable, an object inheriting from class "ggplot" containing the diagnostic plot.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) A Robust Bootstrap Test for Mediation Analysis. *Organizational Research Methods*, **25**(3), 591–617. doi:10.1177/1094428121999096.

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

test_mediation(), weight_plot()

Examples

```
data("BSG2014")

## seed to be used for the random number generator
seed <- 20211117

## simple mediation
# set seed of the random number generator
set.seed(seed)
# The results in Alfons et al. (2021) were obtained with an
# older version of the random number generator.  To reproduce
# those results, uncomment the two lines below.
# RNGversion("3.5.3")
# set.seed(20150601)
# perform mediation analysis
```

```
boot_simple <- test_mediation(TeamCommitment ~
                                m(TaskConflict) +
                                  ValueDiversity,
                              data = BSG2014)
summary(boot_simple)
# the diagnostic plot is not shown when the summary is
# computed, only when the resulting object is printed
summary_simple <- summary(boot_simple)  # does not show plot
summary_simple                          # shows output and plot


## serial multiple mediators
# set seed of the random number generator
set.seed(seed)
# perform mediation analysis
boot_serial <- test_mediation(TeamScore ~
                                serial_m(TaskConflict,
                                         TeamCommitment) +
                                ValueDiversity,
                              data = BSG2014)
summary(boot_serial)

## parallel multiple mediators and control variables
# set seed of the random number generator
set.seed(seed)
# perform mediation analysis
boot_parallel <- test_mediation(TeamPerformance ~
                                  parallel_m(ProceduralJustice,
                                             InteractionalJustice) +
                                  SharedLeadership +
                                  covariates(AgeDiversity,
                                             GenderDiversity),
                                data = BSG2014)
summary(boot_parallel)
```

`test_mediation` *(Robust) mediation analysis*

Description

Perform (robust) mediation analysis via a (fast-and-robust) bootstrap test or Sobel's test.

Usage

```
test_mediation(object, ...)

## S3 method for class 'formula'
test_mediation(
```

```
  formula,
  data,
  test = c("boot", "sobel"),
  alternative = c("twosided", "less", "greater"),
  R = 5000,
  level = 0.95,
  type = c("bca", "perc"),
  order = c("first", "second"),
  method = c("regression", "covariance"),
  robust = TRUE,
  family = "gaussian",
  contrast = FALSE,
  fit_yx = TRUE,
  control = NULL,
  ...
)

## Default S3 method:
test_mediation(
  object,
  x,
  y,
  m,
  covariates = NULL,
  test = c("boot", "sobel"),
  alternative = c("twosided", "less", "greater"),
  R = 5000,
  level = 0.95,
  type = c("bca", "perc"),
  order = c("first", "second"),
  method = c("regression", "covariance"),
  robust = TRUE,
  family = "gaussian",
  model = c("parallel", "serial"),
  contrast = FALSE,
  fit_yx = TRUE,
  control = NULL,
  ...
)

## S3 method for class 'fit_mediation'
test_mediation(
  object,
  test = c("boot", "sobel"),
  alternative = c("twosided", "less", "greater"),
  R = 5000,
  level = 0.95,
  type = c("bca", "perc"),
```

```
  order = c("first", "second"),
  ...
)

robmed(..., test = "boot", method = "regression", robust = TRUE)
```

Arguments

object	the first argument will determine the method of the generic function to be dispatched. For the default method, this should be a data frame containing the variables. There is also a method for a mediation model fit as returned by `fit_mediation()`.
...	additional arguments to be passed down. For the bootstrap tests, those can be used to specify arguments of `boot()`, for example for parallel computing.
formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Hypothesized mediator variables should be wrapped in a call to `m()` (see examples), and any optional control variables should be wrapped in a call to `covariates()`.
data	for the `formula` method, a data frame containing the variables.
test	a character string specifying the test to be performed for the indirect effects. Possible values are `"boot"` (the default) for the bootstrap, or `"sobel"` for Sobel's test. Currently, Sobel's test is not implemented for models with multiple indirect effects.
alternative	a character string specifying the alternative hypothesis in the test for the indirect effects. Possible values are `"twosided"` (the default), `"less"` or `"greater"`.
R	an integer giving the number of bootstrap replicates. The default is to use 5000 bootstrap replicates.
level	numeric; the confidence level of the confidence interval in the bootstrap test. The default is to compute a 95% confidence interval.
type	a character string specifying the type of confidence interval to be computed in the bootstrap test. Possible values are `"bca"` (the default) for the bias-corrected and accelerated bootstrap, or `"perc"` for the percentile bootstrap.
order	a character string specifying the order of approximation of the standard error in Sobel's test. Possible values are `"first"` (the default) for a first-order approximation, and `"second"` for a second-order approximation.
method	a character string specifying the method of estimation for the mediation model. Possible values are `"regression"` (the default) to estimate the effects via regressions, or `"covariance"` to estimate the effects via the covariance matrix. Note that the effects are always estimated via regressions if more than one independent variable or hypothesized mediator is specified, or if control variables are supplied.
robust	a logical indicating whether to perform a robust test (defaults to `TRUE`). For estimation via regressions (`method = "regression"`), this can also be a character string, with `"MM"` specifying the MM-estimator of regression, and `"median"` specifying median regression.

`family` a character string specifying the error distribution to be used in maximum likelihood estimation of regression models. Possible values are `"gaussian"` for a normal distribution (the default), `skewnormal` for a skew-normal distribution, `"student"` for Student's t distribution, `"skewt"` for a skew-t distribution, or `"select"` to select among these four distributions via BIC (see `fit_mediation()` for details). This is only relevant if `method = "regression"` and `robust = FALSE`.

`contrast` a logical indicating whether to compute pairwise contrasts of the indirect effects (defaults to `FALSE`). This can also be a character string, with `"estimates"` for computing the pairwise differences of the indirect effects (such that it is tested whether two indirect effects are equal), and `"absolute"` for computing the pairwise differences of the absolute values of the indirect effects (such that it is tested whether two indirect effects are equal in magnitude). This is only relevant for models with multiple indirect effects, which are currently only implemented for estimation via regressions (`method = "regression"`). For models with multiple independent variables of interest and multiple hypothesized mediators, contrasts are only computed between indirect effects corresponding to the same independent variable.

`fit_yx` a logical indicating whether to fit the regression model `y ~ x + covariates` to estimate the total effect (the default is `TRUE`). This is only relevant if `method = "regression"` and `robust = FALSE`.

`control` a list of tuning parameters for the corresponding robust method. For robust regression (`method = "regression"`, and `robust = TRUE` or `robust = "MM"`), a list of tuning parameters for `lmrob()` as generated by `reg_control()`. For winsorized covariance matrix estimation (`method = "covariance"` and `robust = TRUE`), a list of tuning parameters for `cov_Huber()` as generated by `cov_control()`. No tuning parameters are necessary for median regression (`method = "regression"` and `robust = "median"`).

`x` a character, integer or logical vector specifying the columns of `object` containing the independent variables of interest.

`y` a character string, an integer or a logical vector specifying the column of `object` containing the dependent variable.

`m` a character, integer or logical vector specifying the columns of `object` containing the hypothesized mediator variables.

`covariates` optional; a character, integer or logical vector specifying the columns of `object` containing additional covariates to be used as control variables.

`model` a character string specifying the type of model in case of multiple mediators. Possible values are `"parallel"` (the default) for the parallel multiple mediator model, or `"serial"` for the serial multiple mediator model. This is only relevant for models with multiple hypothesized mediators, which are currently only implemented for estimation via regressions (`method = "regression"`).

Details

With `method = "regression"`, and `robust = TRUE` or `robust = "MM"`, the tests are based on robust regressions with the MM-estimator from `lmrob()`. The bootstrap test is thereby performed via the fast-and-robust bootstrap. This is the default behavior.

Note that the MM-estimator of regression implemented in `lmrob()` can be seen as weighted least squares estimator, where the weights are dependent on how much an observation is deviating from the rest. The trick for the fast-and-robust bootstrap is that on each bootstrap sample, first a weighted least squares estimator is computed (using those robustness weights from the original sample) followed by a linear correction of the coefficients. The purpose of this correction is to account for the additional uncertainty of obtaining the robustness weights.

With `method = "regression"` and `robust = "median"`, the tests are based on median regressions with `rq()`. Note that the bootstrap test is performed via the standard bootstrap, as the fast-and-robust bootstrap is not applicable. Unlike the robust regressions described above, median regressions are not robust against outliers in the explanatory variables, and the standard bootstrap can suffer from oversampling of outliers in the bootstrap samples.

With `method = "covariance"` and `robust = TRUE`, the tests are based on a Huber M-estimator of location and scatter. For the bootstrap test, the M-estimates are used to first clean the data via a transformation. Then the standard bootstrap is performed with the cleaned data. Note that this covariance-based approach is less robust than the approach based on robust regressions described above. Furthermore, the bootstrap does not account for the variability from cleaning the data.

`robmed()` is a wrapper function for performing robust mediation analysis via regressions and the fast-and-robust bootstrap.

Value

An object inheriting from class `"test_mediation"` (class `"boot_test_mediation"` if `test = "boot"` or `"sobel_test_mediation"` if `test = "sobel"`) with the following components:

`a`	a numeric vector containing the bootstrap point estimates of the effects of the independent variables on the proposed mediator variables (only `"boot_test_mediation"`).
`b`	a numeric vector containing the bootstrap point estimates of the direct effects of the proposed mediator variables on the dependent variable (only `"boot_test_mediation"`).
`d`	in case of a serial multiple mediator model, a numeric vector containing the bootstrap point estimates of the effects of proposed mediator variables on other mediator variables occurring later in the sequence (only `"boot_test_mediation"` if applicable.
`total`	a numeric vector containing the bootstrap point estimates of the total effects of the independent variables on the dependent variable (only `"boot_test_mediation"`).
`direct`	a numeric vector containing the bootstrap point estimates of the direct effects of the independent variables on the dependent variable (only `"boot_test_mediation"`).
`indirect`	a numeric vector containing the bootstrap point estimates of the indirect effects (only `"boot_test_mediation"`).
`ab`	for back-compatibility with versions <0.10.0, the bootstrap point estimates of the indirect effects are also included here (only `"boot_test_mediation"`). **This component is deprecated and may be removed as soon as the next version.**
`ci`	a numeric vector of length two or a matrix of two columns containing the bootstrap confidence intervals for the indirect effects (only `"boot_test_mediation"`).
`reps`	an object of class `"boot"` containing the bootstrap replicates (only `"boot_test_mediation"`). For regression model fits, bootstrap replicates of the coefficients in the individual regression models are stored.

`se`	numeric; the standard error of the indirect effect according to Sobel's formula (only `"sobel_test_mediation"`).
`statistic`	numeric; the test statistic for Sobel's test (only `"sobel_test_mediation"`).
`p_value`	numeric; the p-value from Sobel's test (only `"sobel_test_mediation"`).
`alternative`	a character string specifying the alternative hypothesis in the test for the indirect effects.
`R`	an integer giving the number of bootstrap replicates (only `"boot_test_mediation"`).
`level`	numeric; the confidence level of the bootstrap confidence interval (only `"boot_test_mediation"`).
`type`	a character string specifying the type of bootstrap confidence interval (only `"boot_test_mediation"`).
`fit`	an object inheriting from class `"fit_mediation"` containing the estimation results of the mediation model on the original data.

Mediation models

The following mediation models are implemented. In the regression equations below, the i_j are intercepts and the e_j are random error terms.

- *Simple mediation model*: The mediatiom model in its simplest form is given by the equations

$$M = i_1 + aX + e_1,$$

$$Y = i_2 + bM + cX + e_2,$$

$$Y = i_3 + c'X + e_3,$$

  where Y denotes the dependent variable, X the independent variable, and M the hypothesized mediator. The main parameter of interest is the product of coefficients ab, called the indirect effect. The coefficients c and c' are called the direct and total effect, respectively.

- *Parallel multiple mediator model*: The simple mediation model can be extended with multiple mediators $M_1, \ldots, M_k$ in the following way:

$$M_1 = i_1 + a_1X + e_1,$$

$$\vdots$$

$$M_k = i_k + a_kX + e_k,$$

$$Y = i_{k+1} + b_1M_1 + \ldots + b_kM_k + cX + e_{k+1},$$

$$Y = i_{k+2} + c'X + e_{k+2}.$$

  The main parameters of interest are the individual indirect effects $a_1b_1, \ldots, a_kb_k$.

- *Serial multiple mediator model*: It differs from the parallel multiple mediator model in that it allows the hypothesized mediators $M_1, \ldots, M_k$ to influence each other in a sequential manner. It is given by the equations

$$M_1 = i_1 + a_1X + e_1,$$

$$M_2 = i_1 + d_{21}M_1 + a_2X + e_2,$$

$$\vdots$$

$$M_k = i_k + d_{k1}M_1 + \ldots + d_{k,k-1}M_{k-1} + a_kX + e_k,$$

$$Y = i_{k+1} + b_1M_1 + \ldots + b_kM_k + cX + e_{k+1},$$

$$Y = i_{k+2} + c'X + e_{k+2}.$$

The serial multiple mediator model quickly grows in complexity with increasing number of mediators due to the combinatorial increase in indirect paths through the mediators. It is therefore only implemented for two and three mediators to maintain a focus on easily interpretable models. For two serial mediators, the three indirect effects a_1b_1, a_2b_2, and $a_1d_{21}b_2$ are the main parameters of interest. For three serial mediators, there are already seven indirect effects: a_1b_1, a_2b_2, a_3b_3, $a_1d_{21}b_2$, $a_1d_{31}b_3$, $a_2d_{32}b_3$, and $a_1d_{21}d_{32}b_3$.

- *Multiple independent variables to be mediated*: The simple mediation model can also be extended by allowing multiple independent variables $X_1, \ldots, X_l$ instead of multiple mediators. It is defined by the equations

$$M = i_1 + a_1X_1 + \ldots + a_lX_l + e_1,$$

$$Y = i_2 + bM + c_1X_1 + \ldots + c_lX_l + e_2,$$

$$Y = i_3 + c'_1X_1 + \ldots + c'_lX_l + e_3.$$

  The indirect effects $a_1b, \ldots, a_lb$ are the main parameters of interest. Note that an important special case of this model occurs when a categorical independent variable is represented by a group of dummy variables.

- *Control variables*: To isolate the effects of the independent variables of interest from other factors, control variables can be added to all regression equations of a mediation model. Note that that there is no intrinsic difference between independent variables of interest and control variables in terms of the model or its estimation. The difference is purely conceptual in nature: for the control variables, the estimates of the direct and indirect paths are not of particular interest to the researcher. Control variables can therefore be specified separately from the independent variables of interest. Only for the latter, results for the indirect effects are included in the output.

- *More complex models*: Some of the models described above can be combined, for instance parallel and serial multiple mediator models support multiple independent variables of interest.

Note

For the fast-and-robust bootstrap, the simpler correction of Salibian-Barrera & Van Aelst (2008) is used rather than the originally proposed correction of Salibian-Barrera & Zamar (2002).

The default method takes a data frame its first argument so that it can easily be used with the pipe operator (R's built-in `|>` or **magrittr**'s `%>%`).

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) A Robust Bootstrap Test for Mediation Analysis. *Organizational Research Methods*, **25**(3), 591–617. doi:10.1177/1094428121999096.

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

Azzalini, A. and Arellano-Valle, R. B. (2013) Maximum Penalized Likelihood Estimation for Skew-Normal and Skew-t Distributions. *Journal of Statistical Planning and Inference*, **143**(2), 419–433. doi:10.1016/j.jspi.2012.06.022.

Preacher, K.J. and Hayes, A.F. (2004) SPSS and SAS Procedures for Estimating Indirect Effects in Simple Mediation Models. *Behavior Research Methods, Instruments, & Computers*, **36**(4), 717–731. doi:10.3758/bf03206553.

Preacher, K.J. and Hayes, A.F. (2008) Asymptotic and Resampling Strategies for Assessing and Comparing Indirect Effects in Multiple Mediator Models. *Behavior Research Methods*, **40**(3), 879–891. doi:10.3758/brm.40.3.879.

Salibian-Barrera, M. and Van Aelst, S. (2008) Robust Model Selection Using Fast and Robust Bootstrap. *Computational Statistics & Data Analysis*, **52**(12), 5121–5135. doi:10.1016/j.csda.2008.05.007.

Salibian-Barrera, M. and Zamar, R. (2002) Bootstrapping Robust Estimates of Regression. *The Annals of Statistics*, **30**(2), 556–582. doi:10.1214/aos/1021379865.

Sobel, M.E. (1982) Asymptotic Confidence Intervals for Indirect Effects in Structural Equation Models. *Sociological Methodology*, **13**, 290–312. doi:10.2307/270723.

Yuan, Y. and MacKinnon, D.P. (2014) Robust Mediation Analysis Based on Median Regression. *Psychological Methods*, **19**(1), 1–20. doi:10.1037/a0033820.

Zu, J. and Yuan, K.-H. (2010) Local Influence and Robust Procedures for Mediation Analysis. *Multivariate Behavioral Research*, **45**(1), 1–44. doi:10.1080/00273170903504695.

See Also

`fit_mediation()`

`coef()`, `confint()` and `plot()` methods, `p_value()`

`boot()`, `lmrob()`, `lm()`, `cov_Huber()`, `cov_ML()`

Examples

```
data("BSG2014")

## seed to be used for the random number generator
seed <- 20211117

## simple mediation
# set seed of the random number generator
set.seed(seed)
# The results in Alfons et al. (2021) were obtained with an
# older version of the random number generator.  To reproduce
# those results, uncomment the two lines below.
# RNGversion("3.5.3")
# set.seed(20150601)
```

```
# perform mediation analysis
boot_simple <- test_mediation(TeamCommitment ~
                                m(TaskConflict) +
                                  ValueDiversity,
                              data = BSG2014)
summary(boot_simple)


## serial multiple mediators
# set seed of the random number generator
set.seed(seed)
# perform mediation analysis
boot_serial <- test_mediation(TeamScore ~
                                serial_m(TaskConflict,
                                         TeamCommitment) +
                                ValueDiversity,
                              data = BSG2014)
summary(boot_serial)

## parallel multiple mediators and control variables
# set seed of the random number generator
set.seed(seed)
# perform mediation analysis
boot_parallel <- test_mediation(TeamPerformance ~
                                  parallel_m(ProceduralJustice,
                                             InteractionalJustice) +
                                  SharedLeadership +
                                  covariates(AgeDiversity,
                                             GenderDiversity),
                                data = BSG2014)
summary(boot_parallel)
```

`weights.cov_Huber` *Robustness weights of Huber M-estimation of location and scatter*

Description

Extract (relative) robustness weights of a Huber M-estimate of location and scatter.

Usage

```
## S3 method for class 'cov_Huber'
weights(object, type = c("consistent", "relative"), ...)
```

Arguments

`object`	an object inheriting from class "`cov_Huber`" containing Huber M-estimates of location and scatter.

type	a character string specifying the type of robustness weights to be extracted. Possible values are "consistent" and "relative". The former can be used for a robust transformation of the data such that the covariance matrix of the transformed data is Fisher consistent. Observations that are not downweighted in general receive a weight larger than 1. The latter are useful for interpretation, as observations that are not downweighted receive a relative weight of 1.
...	additional arguments are currently ignored.

Value

A numeric vetor containing the requested robustness weights.

Author(s)

Andreas Alfons

References

Zu, J. and Yuan, K.-H. (2010) Local Influence and Robust Procedures for Mediation Analysis. *Multivariate Behavioral Research*, **45**(1), 1–44. doi:10.1080/00273170903504695.

See Also

cov_Huber()

Examples

```
data("BSG2014")

# define variables
x <- "ValueDiversity"
y <- "TeamCommitment"
m <- "TaskConflict"

# compute Huber M-estimator
S <- cov_Huber(BSG2014[, c(x, y, m)])
weights(S, type = "relative")
```

---

weight_plot *Diagnostic plot of robust regression weights*

Description

Produce a diagnostic plot of the regression weights from robust mediation analysis. This plot allows to easily detect deviations from normality assumptions such as skewness or heavy tails.

Usage

```
weight_plot(object, ...)

## Default S3 method:
weight_plot(object, outcome = NULL, npoints = 1000, ...)

## S3 method for class 'setup_weight_plot'
weight_plot(object, ...)
```

Arguments

object	an object inheriting from class "fit_mediation" or "test_mediation" containing results from robust mediation analysis. Only mediation analysis objects fitted with the robust MM-estimator are supported.
...	additional arguments to be passed down.
outcome	a character vector specifying the outcome variables of the regressions to be included in the plot. This must be a subset of the hypothesized mediators and the dependent variable, or NULL (the default) to include all regressions of the mediation model.
npoints	the number of grid points used to evaluate and draw the expected percentages. The default is to use 1000 grid points.

Details

The horizontal axis contains different weight thresholds, and the vertical axis displays the percentage of observations that have a weight below this threshold. For comparison, a reference line is drawn for the expected percentages under normally distributed errors. Observations with negative and positive residuals are shown separately to make it possible to distinguish between symmetric and asymmetric deviations from normality.

If the plot reveals more downweighted observations than expected, but roughly the same amounts in both tails, the residual distribution is symmetric but with heavy tails. If the plot shows that observations in one tail are downweighted more heavily than those in the other tail, the residual distribution is skewed.

Value

An object inheriting from class "ggplot".

Note

The current implementation is a slight hack of **ggplot2** and the **grid** graphics system in order to revert the horizontal axis only in the panels for observations with postive residuals. It is therefore not possible to change the horizontal axis with scale_x_continuous().

The implementation may change in the future if the required functionality becomes available in **ggplot2**.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2022) Robust Mediation Analysis: The R Package **robmed**. *Journal of Statistical Software*, **103**(13), 1–45. doi:10.18637/jss.v103.i13.

See Also

fit_mediation(), test_mediation(), setup_weight_plot()

ci_plot(), density_plot(), ellipse_plot(), plot()

Examples

```
data("BSG2014")

# run fast-and-robust bootstrap test
boot <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")

# create diagnostic plot of robust regression weights
weight_plot(boot) +
  scale_color_manual("", values = c("black", "#00BFC4")) +
  theme(legend.position = "top")

# plot only the regression model for the hypothesized mediator
weight_plot(boot, outcome = "TaskConflict") +
  scale_color_manual("", values = c("black", "#00BFC4")) +
  theme(legend.position = "top")
```

Index