Package 'robmed'

June 8, 2020

Type Package

Title (Robust) Mediation Analysis

Version 0.7.0

Date 2020-06-08

Depends R (>= 3.2.0), ggplot2 (>= 0.9.3), robustbase (>= 0.92-7)

Imports boot (>= 1.3-20), methods, quantreg (>= 5.36), shiny (>= 1.1.0), sn (>= 1.5-4)

Suggests MASS, dplyr, tidyr, testthat

Description Perform mediation analysis via a (fast and robust) bootstrap test.

License GPL (>= 2)

LazyData yes

LazyLoad yes

Author Andreas Alfons [aut, cre]

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

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-08 12:30:03 UTC

R topics documented:

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

Description

Perform mediation analysis via a (fast and robust) bootstrap test.

Details

The DESCRIPTION file:

Package:	robmed
Type:	Package
Title:	(Robust) Mediation Analysis
Version:	0.7.0
Date:	2020-06-08
Depends:	R (>= 3.2.0), ggplot2 (>= 0.9.3), robustbase (>= 0.92-7)
Imports:	boot (>= 1.3-20), methods, quantreg (>= 5.36), shiny (>= 1.1.0), sn (>= 1.5-4)
Suggests:	MASS, dplyr, tidyr, testthat
Description:	Perform mediation analysis via a (fast and robust) bootstrap test.
License:	GPL (>= 2)
LazyData:	yes
LazyLoad:	yes
Authors@R:	c(person("Andreas", "Alfons", email = "alfons@ese.eur.nl", role = c("aut", "cre")))
Author:	Andreas Alfons [aut, cre]
Maintainer:	Andreas Alfons <alfons@ese.eur.nl>
Encoding:	UTF-8
RoxygenNote:	7.1.0

Index of help topics:

Author(s)

Andreas Alfons [aut, cre]

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

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2018) A robust bootstrap test for mediation analysis. *ERIM Report Series in Management*, Erasmus Research Institute of Management. URL https://hdl.handle.net/1765/109594.

Examples

```
## compare bootstrap methods for mediation analysis on simulated data
## Not run:
run_shiny_app()

## End(Not run)

## reproduce empirical examples
## Not run:
demo("case1")
demo("case2")
demo("case3")

## End(Not run)
```

`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, it is not expected to be called by the user.

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 <- 0.2

# generate data
set.seed(20150326)
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)
standard_boot <- test_mediation(simulated_data,
                                x = "x", y = "y", m = "m",
                                robust = FALSE,
                                indices = indices)
summary(standard_boot)
robust_boot <- test_mediation(simulated_data,
                              x = "x", y = "y", m = "m",
                              robust = TRUE,
                              indices = indices)
summary(robust_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, during, and after the simulation 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 7 variables.

`ProcessConflict` Process conflict was measured with the three item scale of Jehn (1995) and responses were aggregated.

`SharedExperience` 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. Team performance in the first game, which was determined by objective performance measures, is a good proxy for the level of shared group experience and training.

`TaskConflict` Task conflict was operationalized with the intra-group conflict scale of Jehn (1995). Five items on the presence of conflict were rated on a 5-point Likert scale (1 = none, 5 = a lot) and aggregated.

`TeamCommitment` Team commitment was measured by four items based on Mowday, Steers & Porter (1979) and responses were aggregated.

`TeamPerformance` Team performance in the second game was measured subjectively by the team members' perceptions of the team's functioning. Hackman's (1986) Likert scale items were thereby used to operationalize team performance.

`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 (credibility, specialization and coordination). 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.

`ValueDiversity` Value diversity was operationalized with the short version of Schwartz's Value Survey (SVS) to measure team members' individual values (Lindeman & Verkasalo, 2005). The responses were aggregated with the average of the coefficient of variations of each value dimension among team members.

References

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.

Jehn, K.A. (1995) A Multi-Method Examination of the Benefits and Detriments of Intra-Group Conflict. *Administrative Science Quarterly*, **40**(2), 256–285.

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

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

Mowday, R.T., Steers, R.M. and Porter, L.W. (1979) The Measurement of Organizational Commitment. *Journal of Vocational Behavior*, **14**(2), 224–47.

Examples

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

```
## scatterplot matrices for three illustrative mediation analyses

# empirical case 1
x <- "SharedExperience"
y <- "TeamPerformance"
m <- "TMS"
plot(BSG2014[, c(x, y, m)], pch = 21, bg = "black")

# empirical case 2
x <- "ValueDiversity"
y <- "TeamCommitment"
m <- "TaskConflict"
plot(BSG2014[, c(x, y, m)], pch = 21, bg = "black")

# empirical case 3
x <- "ValueDiversity"
y <- "TeamPerformance"
m <- "ProcessConflict"
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 = NULL, ...)

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

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

```
## S3 method for class 'list'
ci_plot(
  object,
  parm = NULL,
  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	a character string specifying the effects to be included in the plot. 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 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

See Also

test_mediation(), setup_ci_plot()

density_plot(), ellipse_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 standard bootstrap test
standard_boot <- test_mediation(BSG2014,
                                x = "ValueDiversity",
                                y = "TeamCommitment",
                                m = "TaskConflict",
                                robust = FALSE)

# compare robust and standard tests
tests <- list(Standard = standard_boot, Robust = robust_boot)
ci_plot(tests)

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

`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 coefficients to be extracted, or `NULL` to extract all coefficients.
`...`	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 full 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
test <- test_mediation(fit)
coef(test, type = "data")  # from original sample
coef(test, 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 coefficients for which to extract or compute confidence intervals, or `NULL` to extract or compute confidence intervals for all coefficients.
`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 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 standard bootstrap test
standard_boot <- test_mediation(BSG2014,
                                x = "ValueDiversity",
                                y = "TeamCommitment",
                                m = "TaskConflict",
                                robust = FALSE)
confint(standard_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 fast and robust bootstrap test
ctrl <- cov_control(prob = 0.95)
test <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict",
                       method = "covariance",
                       control = ctrl)
summary(test)
```

`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.

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.

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 ± 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

See Also

`test_mediation()`, `setup_density_plot()`

`ci_plot()`, `ellipse_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 standard bootstrap test
standard_boot <- test_mediation(BSG2014,
                                x = "ValueDiversity",
                                y = "TeamCommitment",
                                m = "TaskConflict",
                                robust = FALSE)

# compare robust and standard tests
tests <- list(Standard = standard_boot, Robust = robust_boot)
```

```
density_plot(tests)

# the plot can be customized in the usual way
density_plot(tests) + theme_bw() +
  labs(title = "Standard vs robust bootstrap test")
```

`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 the independent variable must be selected for the horizontal axis. If a hypothesized mediator is chosen for the vertical axis, the independent variable must be selected for the horizontal axis. The default is to plot the 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

See Also

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

`ci_plot()`, `density_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 standard bootstrap test
standard_boot <- test_mediation(BSG2014,
                                x = "ValueDiversity",
                                y = "TeamCommitment",
                                m = "TaskConflict",
                                robust = FALSE)

# compare robust and standard tests
tests <- list(OLS = standard_boot, Robust = robust_boot)
ellipse_plot(tests)

# the plot can be customized in the usual way
ellipse_plot(tests) + 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",
  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 string, an integer or a logical vector specifying the column of object containing the independent variable.
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 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, "skew-t" 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.
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 Huberized 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 estimation procedure in package **robmed**.

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 effect of the independent variable on the proposed mediator variables.

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

`direct` numeric; the point estimate of the direct effect of the independent variable on the dependent variable.

`total` numeric; the point estimate of the total effect of the independent variable on the dependent variable.

`fit_mx` an object of class `"lmrob"`, `"lm"` or `"lmse"` containing the estimation results from the regression of the proposed mediator variable on the independent variable, 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"`, `"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 variable (only `"reg_fit_mediation"` and if `robust = FALSE`).

`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.

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

Note

The formula interface is still experimental and may change in future versions.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2018) A robust bootstrap test for mediation analysis. *ERIM Report Series in Management*, Erasmus Research Institute of Management. URL https://hdl.handle.net/1765/109594.

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.

Yuan, Y. and MacKinnon, D.P. (2014) Robust mediation analysis based on median regression. *Psychological Methods*, **19**(1), 1–20.

Zu, J. and Yuan, K.-H. (2010) Local influence and robust procedures for mediation analysis. *Multivariate Behavioral Research*, **45**(1), 1–44.

See Also

`test_mediation()`

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

Examples

```
data("BSG2014")

# to reproduce results in paper
RNGversion("3.5.3")
seed <- 20150601

# formula interface
set.seed(seed)
fit1 <- fit_mediation(TeamCommitment ~ m(TaskConflict) + ValueDiversity,
                      data = BSG2014)
test1 <- test_mediation(fit1)
summary(test1)

# default method
set.seed(seed)
fit2 <- fit_mediation(BSG2014,
                      x = "ValueDiversity",
                      y = "TeamCommitment",
                      m = "TaskConflict")
test2 <- test_mediation(fit2)
summary(test2)
```

---

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

Description

`m()` creates 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(...)

covariates(...)
```

Arguments

`...`	variables are supplied as arguments, as usual separated by a comma.

Details

These are essentially wrappers for `cbind()` with a specific class prepended to the class(es) of the resulting object.

Value

`m()` returns an object of class `"parallel_mediators"` and `covariates()` returns an object of class `"covariates"`. Typically, these inherit from class `"matrix"`.

Note

The formula interface is still experimental and may change in future versions.

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, ...)

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

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

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

Arguments

object, x	an object inheriting from class "fit_mediation" or "test_mediation" containing results from (robust) mediation analysis.
...	additional arguments to be passed down.
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()), or "ellipse" for a diagnostic plot of the data together with a tolerance ellipse (see ellipse_plot()).

Details

The "fit_mediation" method is a wrapper for ellipse_plot().

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

Value

An object of class "ggplot".

Author(s)

Andreas Alfons

See Also

fit_mediation(), test_mediation()

ci_plot(), density_plot(), ellipse_plot()

Examples

```
data("BSG2014")

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

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

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, digits = 4L, parm = NULL, type = c("boot", "data"), ...)

## 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.
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.
parm	an integer, character or logical vector specifying the coefficients for which to extract or compute p-values, or NULL to extract or compute p-values for all coefficients.
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 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'.

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.

Note

In version 0.7.0, functionality has been extended to be in line with the `coef()` and `confint()` methods. However, the old default behavior was kept for back-compatibility with previous versions. That is, if the new arguments `parm` and `type` are missing, only the p-values of the indirect effect will be returned, as in previous versions.

In a future version, possibly even the next version, the default behavior will change such that the p-values of all effects in the mediation model will be returned, similar to the behavior of `coef()` and `confint()`.

In addition, the `digits` argument of the `"boot_test_mediation"` method will be moved from the second position to the fourth. It was kept at the second position for back-compatibility, but it will be moved to the fourth position such that the function interface is in line with `coef()` and `confint()`. In the mean time, it is recommended to always use argument names in scripts.

Author(s)

Andreas Alfons

See Also

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

Examples

```
data("BSG2014")


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

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. (1987) A fast algorithm for S-regression estimates. *Journal of Computational and Graphical Statistics*, **15**(2), 414–427.

Yohai, V.J. (1987) High breakdown-point and high efficiency estimates for regression. *The Annals of Statistics*, **15**(20), 642–656.

See Also

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

Examples

```
data("BSG2014")

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

`retest` *Retest for mediation*

Description

Reperform a (fast and robust) bootstrap test or Sobel's test for the indirect effect(s) based on results from (robust) mediation analysis.

Usage

```
retest(object, ...)

## S3 method for class 'boot_test_mediation'
retest(
  object,
  alternative = c("twosided", "less", "greater"),
  level = 0.95,
  type = c("bca", "perc"),
  ...
)

## S3 method for class 'sobel_test_mediation'
retest(object, alternative = c("twosided", "less", "greater"), ...)
```

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"` (the default), `"less"` or `"greater"`.
`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.

Value

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

Author(s)

Andreas Alfons

See Also

test_mediation()

Examples

```
data("BSG2014")

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

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

---

run_shiny_app *Shiny app: simulation for mediation analysis with outliers*

Description

Compare various bootstrap methods for mediation analysis on simulated data.

Usage

```
run_shiny_app()
```

Details

The default settings follow the simulation design of Zu & Yuan (2010). You can adjust the total number of observations, the values of the coefficients in the mediation model, the number of outliers, as well as the expected distance of the outliers from the main point cloud.

As this simulation is just for illustration, the bootstrap procedures use only 1000 replicates. For each selected methods, the bootstrap distribution of the indirect effect is shown together with a shaded area representing the 95% confidence interval.

Author(s)

Andreas Alfons

See Also

test_mediation

Examples

```
## Not run:
run_shiny_app()

## End(Not run)
```

---

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 = NULL,
  type = c("boot", "data"),
  p_value = FALSE,
  digits = 4L,
  ...
)

## S3 method for class 'sobel_test_mediation'
setup_ci_plot(object, parm = NULL, 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	a character string specifying the effects to be included in the plot. 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 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

See Also

test_mediation(), ci_plot()

Examples

```
data("BSG2014")

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

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

# 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 ab), and the estimated density values (column Density). In case of a multiple mediator model, 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 multiple mediator model, 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 mediators. 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

See Also

test_mediation(), density_plot()

Examples

```
data("BSG2014")

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

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

# plot only density and confidence interval
ggplot() +
  geom_density(aes(x = ab, 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 the independent variable must be selected for the horizontal axis. If a hypothesized mediator is chosen for the vertical axis, the independent variable must be selected for the horizontal axis. The default is to plot the 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

See Also

fit_mediation(), test_mediation(), ellipse_plot()

Examples

```
data("BSG2014")

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

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

---

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"), ...)

## 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 "theory" 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.
...	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.
summary	an object containing all necessary information to summarize the effects other than the indirect effect.

Author(s)

Andreas Alfons

See Also

test_mediation

Examples

```
data("BSG2014")
test <- test_mediation(BSG2014,
                       x = "ValueDiversity",
                       y = "TeamCommitment",
                       m = "TaskConflict")
summary(test)
```

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"),
  method = c("regression", "covariance"),
  robust = TRUE,
  family = "gaussian",
  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"),
  method = c("regression", "covariance"),
  robust = TRUE,
  family = "gaussian",
  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"),
  ...
)

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 effect. Possible values are "boot" (the default) for the bootstrap, or "sobel" for Sobel's test. Currently, Sobel's test is not implemented for more than one hypothesized mediator variable.
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.
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 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, "skew-t" 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.
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 Huberized 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 string, an integer or a logical vector specifying the column of object containing the independent variable.
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.

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:

`ab`	a numeric vector containing the point estimates of the indirect effects.
`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 of the effects (only `"boot_test_mediation"`).
`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 for the direct effect and the total effect in the mediation model.

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 formula interface is still experimental and may change in future versions.

Author(s)

Andreas Alfons

References

Alfons, A., Ates, N.Y. and Groenen, P.J.F. (2018) A robust bootstrap test for mediation analysis. *ERIM Report Series in Management*, Erasmus Research Institute of Management. URL https://hdl.handle.net/1765/109594.

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.

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.

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.

Salibian-Barrera, M. and Van Aelst, S. (2008) Robust model selection using fast and robust bootstrap. *Computational Statistics & Data Analysis*, **52**(12), 5121–5135

Salibian-Barrera, M. and Zamar, R. (2002) Bootstrapping robust estimates of regression. *The Annals of Statistics*, **30**(2), 556–582.

Sobel, M.E. (1982) Asymptotic confidence intervals for indirect effects in structural equation models. *Sociological Methodology*, **13**, 290–312.

Yuan, Y. and MacKinnon, D.P. (2014) Robust mediation analysis based on median regression. *Psychological Methods*, **19**(1), 1–20.

Zu, J. and Yuan, K.-H. (2010) Local influence and robust procedures for mediation analysis. *Multivariate Behavioral Research*, **45**(1), 1–44.

See Also

`fit_mediation()`

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

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

Examples

```
data("BSG2014")

# to reproduce results in paper
RNGversion("3.5.3")
seed <- 20150601

# formula interface
set.seed(seed)
test1 <- test_mediation(TeamCommitment ~ m(TaskConflict) + ValueDiversity,
                        data = BSG2014)
summary(test1)

# default method
set.seed(seed)
test2 <- test_mediation(BSG2014,
                        x = "ValueDiversity",
                        y = "TeamCommitment",
                        m = "TaskConflict")
```

```
summary(test2)
```

`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.

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")
```

Index