Package 'semfindr'

April 8, 2024

Title Influential Cases in Structural Equation Modeling

Version 0.1.8

Description Sensitivity analysis in structural equation modeling using influence measures and diagnostic plots. Support leave-one-out casewise sensitivity analysis presented by Pek and MacCallum (2011) <doi:10.1080/00273171.2011.561068> and approximate casewise influence using scores and casewise likelihood.

URL https://sfcheung.github.io/semfindr/

BugReports https://github.com/sfcheung/semfindr/issues

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0), parallel, knitr, rmarkdown, modi, MASS

Imports lavaan, ggplot2, ggrepel, rlang, stats, methods, utils, Matrix

VignetteBuilder knitr

Config/testthat/parallel true

Config/testthat/edition 3

NeedsCompilation no

Author Shu Fai Cheung [aut, cre] (<https://orcid.org/0000-0002-9871-9448>), Mark Hok Chio Lai [aut] (<https://orcid.org/0000-0002-9196-7406>)

Maintainer Shu Fai Cheung <shufai.cheung@gmail.com>

Repository CRAN

Date/Publication 2024-04-08 13:30:03 UTC

R topics documented:

`approx_check` *Compatibility Check for the '_approx' Functions*

Description

Gets a 'lavaan' output and checks whether it is supported by the functions using the approximate approach.

Usage

```
approx_check(
  fit,
  print_messages = TRUE,
  multiple_group = FALSE,
  equality_constraint = FALSE
)
```

Arguments

`fit`	The output from `lavaan`, such as `lavaan::cfa()` and `lavaan::sem()`.
`print_messages`	Logical. If `TRUE`, will print messages about the check. If `FALSE`, the messages will be attached to the return value as an attribute. Default is `TRUE`.
`multiple_group`	Logical. Whether multiple-group models are supported. If yes, the check for multiple-groups models will be skipped. Default is `FALSE`.
`equality_constraint`	Logical. Whether models with equality constraints are supported. If yes, the check for equality constraints will be skipped. Default is `FALSE`.

Details

This function is not supposed to be used by users. It is called by functions such as `est_change_approx()` to see if the analysis passed to it is supported. If not, messages will be printed to indicate why.

Value

A single-element vector. If confirmed to be supported, will return 0. If not confirmed to be support but may still work, return 1. If confirmed to be not yet supported, will return a negative number, the value of this number without the negative sign is the number of tests failed.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

Examples

```
dat <- cfa_dat

mod <-
"
f1 =~ x4 + x5 + x6
"
dat_gp <- dat
dat$gp <- rep(c("gp1", "gp2"), length.out = nrow(dat_gp))

fit01 <- lavaan::sem(mod, dat)
# If supported, returns a zero
approx_check(fit01)
```

```
fit05 <- lavaan::cfa(mod, dat, group = "gp")
# If not supported, returns a negative number
approx_check(fit05)
```

---

cfa_dat *Sample Data: A CFA Model*

Description

A six-variable dataset with 100 cases.

Usage

```
cfa_dat
```

Format

A data frame with 100 rows and 6 variables:

x1 Indicator. Numeric.

x2 Indicator. Numeric.

x3 Indicator. Numeric.

x4 Indicator. Numeric.

x5 Indicator. Numeric.

x6 Indicator. Numeric.

Examples

```
library(lavaan)
data(cfa_dat)
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
"
fit <- cfa(mod, cfa_dat)
summary(fit)
```

cfa_dat2 *Sample Data: A CFA Model with an Influential Case*

Description

A six-variable dataset with 100 cases, with one influential case.

Usage

```
cfa_dat2
```

Format

A data frame with 100 rows and 7 variables:

case_id Case ID. Character.

x1 Indicator. Numeric.

x2 Indicator. Numeric.

x3 Indicator. Numeric.

x4 Indicator. Numeric.

x5 Indicator. Numeric.

x6 Indicator. Numeric.

Examples

```
library(lavaan)
data(cfa_dat2)
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
"
fit <- cfa(mod, cfa_dat2)
summary(fit)
inf_out <- influence_stat(fit)
gcd_plot(inf_out)
```

`cfa_dat_heywood` *Sample Data: A CFA Model with a Heywood Case*

Description

A six-variable dataset with 60 cases, with one case resulting in negative variance if not removed.

Usage

```
cfa_dat_heywood
```

Format

A data frame with 60 rows and 6 variables:

x1 Indicator. Numeric.

x2 Indicator. Numeric.

x3 Indicator. Numeric.

x4 Indicator. Numeric.

x5 Indicator. Numeric.

x6 Indicator. Numeric.

Examples

```
library(lavaan)
data(cfa_dat_heywood)
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
"
# The following will result in a warning
fit <- cfa(mod, cfa_dat_heywood)
# One variance is negative
parameterEstimates(fit, output = "text")
# Fit the model with the first case removed
fit_no_case_1 <- cfa(mod, cfa_dat_heywood[-1, ])
# Results admissible
parameterEstimates(fit_no_case_1, output = "text")
```

cfa_dat_mg *Sample Data: A Multiple-Group CFA Model with an Influential Case*

Description

A six-variable dataset with 100 cases.

Usage

```
cfa_dat_mg
```

Format

A data frame with 100 rows and 6 variables:

x1 Indicator. Numeric.

x2 Indicator. Numeric.

x3 Indicator. Numeric.

x4 Indicator. Numeric.

x5 Indicator. Numeric.

x6 Indicator. Numeric.

gp Group variable. Character. Either "GroupA" or "GroupB".

Examples

```
library(lavaan)
data(cfa_dat_mg)
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
"
fit1 <- cfa(mod, cfa_dat_mg, group = "gp")
fit2 <- cfa(mod, cfa_dat_mg, group = "gp",
            group.equal = "loadings")
fit3 <- cfa(mod, cfa_dat_mg, group = "gp",
            group.equal = c("loadings", "intercepts"))
lavTestLRT(fit1, fit2, fit3)
lavTestLRT(fit1, fit3)

# Drop the first case
cfa_dat_mgb <- cfa_dat_mg[-1, ]
fit1b <- cfa(mod, cfa_dat_mgb, group = "gp")
fit2b <- cfa(mod, cfa_dat_mgb, group = "gp",
             group.equal = "loadings")
fit3b <- cfa(mod, cfa_dat_mgb, group = "gp",
             group.equal = c("loadings", "intercepts"))
```

```
lavTestLRT(fit1b, fit2b, fit3b)
lavTestLRT(fit1b, fit3b)
```

`est_change` *Standardized Case Influence on Parameter Estimates*

Description

Gets a `lavaan_rerun()` output and computes the standardized changes in selected parameters for each case if included.

Usage

```
est_change(rerun_out, parameters = NULL)
```

Arguments

`rerun_out`	The output from `lavaan_rerun()`.
`parameters`	A character vector to specify the selected parameters. Each parameter is named as in `lavaan` syntax, e.g., `x ~ y` or `x ~~ y`, as appeared in the columns `lhs`, `op`, and `rhs` in the output of `lavaan::parameterEstimates()`. Supports specifying an operator to select all parameters with this operators: `~`, `~~`, `=~`, and `~1`. This vector can contain both parameter names and operators. More details can be found in the help of `pars_id()`. If omitted or `NULL`, the default, changes on all free parameters will be computed.

Details

For each case, `est_change()` computes the differences in the estimates of selected parameters with and without this case:

(Estimate with all case) - (Estimate without this case).

The differences are standardized by dividing the raw differences by their standard errors (Pek & MacCallum, 2011). This is a measure of the standardized influence of a case on the parameter estimates if it is included.

If the value of a case is positive, including the case increases an estimate.

If the value of a case is negative, including the case decreases an estimate.

If the analysis is not admissible or does not converge when a case is deleted, `NA`s will be turned for this case on the differences.

Unlike `est_change_raw()`, `est_change()` does not support computing the standardized changes of standardized estimates.

It will also compute generalized Cook's distance (*gCD*), proposed by Pek and MacCallum (2011) for structural equation modeling. Only the parameters selected (all free parameters, by default) will be used in computing *gCD*.

Since version 0.1.4.8, if (a) a model has one or more equality constraints, and (b) some selected parameters are linearly dependent or constrained to be equal due to the constraint(s), *gCD* will be computed by removing parameters such that the remaining parameters are not linearly dependent nor constrained to be equal. (Support for equality constraints and linearly dependent parameters available in 0.1.4.8 and later version).

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

An `est_change`-class object, which is matrix with the number of columns equals to the number of requested parameters plus one, the last column being the generalized Cook's distance. The number of rows equal to the number of cases. The row names are the case identification values used in `lavaan_rerun()`. The elements are the standardized difference. Please see Pek and MacCallum (2011), Equation 7. A print method is available for user-friendly output.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Pek, J., & MacCallum, R. (2011). Sensitivity analysis in structural equation models: Cases and their influence. *Multivariate Behavioral Research, 46*(2), 202-228. doi:10.1080/00273171.2011.561068

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
# Fit the model several times. Each time with one case removed.
# For illustration, do this only for four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(2, 4, 7, 9))
# Compute the standardized changes in parameter estimates
# if a case is included vs. if this case is excluded.
# That is, case influence on parameter estimates, standardized.
out <- est_change(fit_rerun)
# Case influence:
```

```
out
# Note that these are the differences divided by the standard errors
# The rightmost column, `gcd`, contains the
# generalized Cook's distances (Pek & MacCallum, 2011).
out[, "gcd", drop = FALSE]

# Compute the changes for the paths from iv1 and iv2 to m1
out2 <- est_change(fit_rerun, c("m1 ~ iv1", "m1 ~ iv2"))
# Case influence:
out2
# Note that only the changes in the selected parameters are included.
# The generalized Cook's distance is computed only from the selected
# parameter estimates.

# A CFA model

dat <- cfa_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f1 ~~ f2
"
# Fit the model
fit <- lavaan::cfa(mod, dat)

# Examine four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(2, 3, 5, 7))
# Compute the standardized changes in parameter estimates
# if a case is included vs. if a case is excluded.
# That is, case influence on parameter estimates, standardized.
# For free loadings only
out <- est_change(fit_rerun, parameters = "=~")
out

# A latent variable model

dat <- sem_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   a * f1
f3 ~   b * f2
ab := a * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)

# Examine four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
```

```
                          to_rerun = c(2, 3, 5, 7))
# Compute the changes in parameter estimates if a case is included
# vs. if a case is excluded.
# That is, standardized case influence on parameter estimates.
# For structural paths only
out <- est_change(fit_rerun, parameters = "~")
out
```

`est_change_approx` *Standardized Case Influence on Parameter Estimates (Approximate)*

Description

Gets a `lavaan::lavaan()` output and computes the approximate standardized changes in selected parameters for each case if included.

Usage

```
est_change_approx(
  fit,
  parameters = NULL,
  case_id = NULL,
  allow_inadmissible = FALSE,
  skip_all_checks = FALSE
)
```

Arguments

`fit`	The output from `lavaan::lavaan()` or its wrappers (e.g., `lavaan::cfa()` and `lavaan::sem()`).
`parameters`	A character vector to specify the selected parameters. Each parameter is named as in `lavaan` syntax, e.g., `x ~ y` or `x ~~ y`, as appeared in the columns `lhs`, `op`, and `rhs` in the output of `lavaan::parameterEstimates()`. Supports specifying an operator to select all parameters with these operators: `~`, `~~`, `=~`, and `~1`. This vector can contain both parameter names and operators. More details can be found in the help of `pars_id()`. If omitted or `NULL`, the default, changes on all free parameters will be computed.
`case_id`	If it is a character vector of length equals to the number of cases (the number of rows in the data in `fit`), then it is the vector of case identification values. If it is `NULL`, the default, then `case.idx` used by `lavaan` functions will be used as case identification values.
`allow_inadmissible`	If `TRUE`, accepts a fit object with inadmissible results (i.e., `post.check` from `lavaan::lavInspect()` is `FALSE`). Default is `FALSE`.
`skip_all_checks`	If `TRUE`, skips all checks and allows users to run this function on any object of the `lavaan` class. For users to experiment this and other functions on models not officially supported. Default is `FALSE`.

Details

For each case, `est_change_approx()` computes the approximate differences in the estimates of selected parameters with and without this case:

(Estimate with all case) - (Estimate without this case)

The differences are standardized by dividing the approximate raw differences by their standard errors. This is a measure of the standardized influence of a case on the parameter estimates if it is included.

If the value of a case is positive, including the case increases an estimate.

If the value of a case is negative, including the case decreases an estimate.

The model is not refitted. Therefore, the result is only an approximation of that of `est_change()`. However, this approximation is useful for identifying potentially influential cases when the sample size is very large or the model takes a long time to fit. This function can be used to identify potentially influential cases quickly and then select them to conduct the leave-one-out sensitivity analysis using `lavaan_rerun()` and `est_change()`.

This function also computes the approximate generalized Cook's distance (gCD). To avoid confusion, it is labelled `gcd_approx`.

For the technical details, please refer to the vignette on this approach: `vignette("casewise_scores", package = "semfindr")`

The approximate approach supports a model with equality constraints (available in 0.1.4.8 and later version).

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

An `est_change`-class object, which is matrix with the number of columns equals to the number of requested parameters plus one, the last column being the approximate generalized Cook's distance. The number of rows equal to the number of cases. The row names are the case identification values used in `lavaan_rerun()`. The elements are approximate standardized differences. A print method is available for user-friendly output.

Author(s)

Idea by Mark Hok Chio Lai https://orcid.org/0000-0002-9196-7406, implemented by Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
```

```
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Approximate standardized changes and gCD
out_approx <- est_change_approx(fit)
head(out_approx)

# Fit the model several times. Each time with one case removed.
# For illustration, do this only for the first 10 cases.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Compute the changes in chisq if a case is removed
out <- est_change(fit_rerun)
head(out)

# Compare the results
plot(out_approx[1:10, 1], out[, 1])
abline(a = 0, b = 1)
plot(out_approx[1:10, 2], out[, 2])
abline(a = 0, b = 1)
plot(out_approx[1:10, 3], out[, 3])
abline(a = 0, b = 1)
plot(out_approx[1:10, "gcd_approx"], out[, "gcd"])
abline(a = 0, b = 1)

# A CFA model

dat <- cfa_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f1 ~~ f2
"
# Fit the model
fit <- lavaan::cfa(mod, dat)
summary(fit)

# Approximate standardized changes and gCD
# Compute gCD only for free loadings
out_approx <- est_change_approx(fit,
                                parameters = "=~")
head(out_approx)

# A latent variable model

dat <- sem_dat
mod <-
```

```
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   a * f1
f3 ~   b * f2
ab := a * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Approximate standardized changes and gCD
# Compute gCD only for structural paths
out_approx <- est_change_approx(fit,
                                parameters = "~")
head(out_approx)
```

`est_change_plot` *Plots of Case Influence on Parameter Estimates*

Description

Gets the output of functions such as `est_change()` and `est_change_approx()` and plots case influence on selected parameters.

Usage

```
est_change_plot(
  change,
  parameters,
  cutoff_change = NULL,
  largest_change = 1,
  title = TRUE,
  point_aes = list(),
  vline_aes = list(),
  hline_aes = list(),
  cutoff_line_aes = list(),
  case_label_aes = list(),
  wrap_aes = list()
)

est_change_gcd_plot(
  change,
  parameters,
  cutoff_gcd = NULL,
```

```
  largest_gcd = 1,
  cutoff_change = NULL,
  largest_change = 1,
  title = TRUE,
  point_aes = list(),
  hline_aes = list(),
  cutoff_line_aes = list(),
  case_label_aes = list(),
  wrap_aes = list()
)
```

Arguments

change	The output from est_change(), est_change_raw(), est_change_approx(), or est_change_raw_approx().
parameters	If it is a character vector, it specifies the selected parameters. Each parameter is named as in lavaan syntax, e.g., x ~ y or x ~~ y, as appeared in the columns lhs, op, and rhs in the output of lavaan::parameterEstimates(). Supports specifying an operator to select all parameters with this operators: ~, ~~, =~, and ~1. This vector can contain both parameter names and operators. If it is a numeric vector, it specifies the columns to be used. If omitted or NULL, the default, changes on all parameters in change. will be used.
cutoff_change	Cases with absolute changes larger than this value will be labeled. Default is NULL. If NULL, no cutoff line will be drawn.
largest_change	The number of cases with the largest absolute changes to be labelled. Default is 1. If not an integer, it will be rounded to the nearest integer.
title	If TRUE, the default, a default title will be added to the plot. If it is a string, it will be used as the title. If FALSE, no title will be added to the plot.
point_aes	A named list of arguments to be passed to ggplot2::geom_point() to modify how to draw the points. Default is list() and internal default settings will be used.
vline_aes	A named list of arguments to be passed to ggplot2::geom_segment() to modify how to draw the line for each case in the index plot by est_change_plot(). Default is list() and internal default settings will be used.
hline_aes	A named list of arguments to be passed to ggplot2::geom_hline() to modify how to draw the horizontal line for zero case influence. Default is list() and internal default settings will be used.
cutoff_line_aes	A named list of arguments to be passed to ggplot2::geom_hline() in est_change_plot() or ggplot2::geom_vline() in est_change_gcd_plot() to modify how to draw the line for user cutoff value (cutoff_change or cutoff_gcd). Default is list() and internal default settings will be used.
case_label_aes	A named list of arguments to be passed to ggrepel::geom_label_repel() to modify how to draw the labels for cases marked (based on cutoff_change, cutoff_gcd, largest_change, or largest_gcd). Default is list() and internal default settings will be used.

wrap_aes	A named list of arguments to be passed to ggplot2::facet_wrap() to modify how the plots are organized. Default is list() and internal default settings will be used.
cutoff_gcd	Cases with generalized Cook's distance or approximate generalized Cook's distance larger than this value will be labeled. Default is NULL. If NULL, no cutoff line will be drawn.
largest_gcd	The number of cases with the largest generalized Cook's distance or approximate generalized Cook's distance to be labelled. Default is 1. If not an integer, it will be rounded to the nearest integer.

Details

The output of est_change(), est_change_raw(), est_change_approx(), and est_change_raw_approx() is simply a matrix. Therefore, these functions will work for any matrix provided. Row number will be used on the x-axis if applicable. However, case identification values will be used for labeling individual cases if they are stored as row names.

The default settings for the plots should be good enough for diagnostic purpose. If so desired, users can use the *_aes arguments to nearly fully customize all the major elements of the plots, as they would do for building a ggplot2 plot.

Value

A ggplot2 plot. Plotted by default. If assigned to a variable or called inside a function, it will not be plotted. Use plot() to plot it.

Functions

- est_change_plot(): Index plot of case influence on parameters.
- est_change_gcd_plot(): Plot case influence on parameter estimates against generalized Cook's distance.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Pek, J., & MacCallum, R. (2011). Sensitivity analysis in structural equation models: Cases and their influence. *Multivariate Behavioral Research, 46*(2), 202-228. doi:10.1080/00273171.2011.561068

See Also

est_change(), est_change_raw(), est_change_approx(), and est_change_raw_approx().

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Compute approximate case influence on parameters estimates
out <- est_change_approx(fit)

# Plot case influence for all regression coefficients
est_change_plot(out,
                parameters = "~",
                largest_change = 2)

# Plot case influence against approximated gCD for all
# regression coefficients
# Label top 5 cases with largest approximated gCD
est_change_gcd_plot(out,
                    parameters = "~",
                    largest_gcd = 5)

# Customize elements in a plot.
# For example, change the color and shape of the points.

est_change_plot(out,
                parameters = "~",
                largest_change = 2,
                point_aes = list(shape = 5,
                                 color = "red"))
```

`est_change_raw` *Case Influence on Parameter Estimates*

Description

Gets a `lavaan_rerun()` output and computes the changes in selected parameters for each case if included.

Usage

```
est_change_raw(
  rerun_out,
  parameters = NULL,
  standardized = FALSE,
  user_defined_label_full = FALSE
)
```

Arguments

rerun_out	The output from lavaan_rerun().
parameters	A character vector to specify the selected parameters. Each parameter is named as in lavaan syntax, e.g., x ~ y or x ~~ y, as appeared in the columns lhs, op, and rhs in the output of lavaan::parameterEstimates(). Supports specifying an operator to select all parameters with these operators: ~, ~~, =~, and ~1. This vector can contain both parameter names and operators. More details can be found in the help of pars_id(). If omitted or NULL, the default, changes on all free parameters will be computed.
standardized	If TRUE, the changes in the full standardized solution is returned (type = std.all in lavaan::standardizedSolution()). Otherwise, the changes in the unstandardized solution are returned. Default is FALSE.
user_defined_label_full	Logical. If TRUE, use the full labels for user-defined parameters (parameters created by :=), which include the definition. If FALSE, then only the label on the right-hand side of := will be used. Default is FALSE. In previous version, the full labels were used. Set to TRUE if backward compatibility is needed.

Details

For each case, est_change_raw() computes the differences in the estimates of selected parameters with and without this case:

(Estimate with all case) - (Estimate without this case).

The change is the raw change, either for the standardized or unstandardized solution. The change is *not* divided by standard error. This is a measure of the influence of a case on the parameter estimates if it is included.

If the value of a case is positive, including the case increases an estimate.

If the value of a case is negative, including the case decreases an estimate.

If the analysis is not admissible or did not converge when a case is deleted, NAs will be returned for this case on the differences.

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

An est_change-class object, which is matrix with the number of columns equals to the number of requested parameters, and the number of rows equals to the number of cases. The row names are

the case identification values used in lavaan_rerun(). The elements are the raw differences. A print method is available for user-friendly output.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Pek, J., & MacCallum, R. (2011). Sensitivity analysis in structural equation models: Cases and their influence. *Multivariate Behavioral Research, 46*(2), 202-228. doi:10.1080/00273171.2011.561068

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)

# Fit the model n times. Each time with one case is removed.
# For illustration, do this only for four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(3, 5, 7, 8))
# Compute the changes in parameter estimates if a case is included
# vs. if this case is excluded.
# That is, case influence on parameter estimates.
out <- est_change_raw(fit_rerun)
# Results excluding a case
out
# Note that these are the differences in parameter estimates.

# The parameter estimates from all cases
(coef_all <- coef(fit))
# The parameter estimates from manually deleting the third case
fit_no_3 <- lavaan::sem(mod, dat[-3, ])
(coef_no_3 <- coef(fit_no_3))
# The differences
coef_all - coef_no_3
# The first row of `est_change_raw(fit_rerun)`
round(out[1, ], 3)

# Compute only the changes of the paths from iv1 and iv2 to m1
```

```
out2 <- est_change_raw(fit_rerun, c("m1 ~ iv1", "m1 ~ iv2"))
# Results excluding a case
out2
# Note that only the changes in the selected paths are included.

# Use standardized = TRUE to compare the differences in standardized solution
out2_std <- est_change_raw(fit_rerun,
                           c("m1 ~ iv1", "m1 ~ iv2"),
                           standardized = TRUE)
out2_std
(est_std_all <- parameterEstimates(fit,
                standardized = TRUE)[1:2, c("lhs", "op", "rhs", "std.all")])
(est_std_no_1 <- parameterEstimates(fit_no_3,
                standardized = TRUE)[1:2, c("lhs", "op", "rhs", "std.all")])
# The differences
est_std_all$std.all - est_std_no_1$std.all
# The first row of `out2_std`
out2_std[1, ]

# A CFA model

dat <- cfa_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f1 ~~ f2
"
# Fit the model
fit <- lavaan::cfa(mod, dat)

# Examine four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(2, 3, 5, 7))
# Compute the changes in parameter estimates if a case is included
# vs. if this case is excluded.
# That is, case influence on parameter estimates.
# For free loadings only
out <- est_change_raw(fit_rerun, parameters = "=~")
out
# For standardized loadings only
out_std <- est_change_raw(fit_rerun, parameters = "=~",
                          standardized = TRUE)
out_std

# A latent variable model

dat <- sem_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
```

```
f2 ~   a * f1
f3 ~   b * f2
ab := a * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)

# Examine four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(2, 3, 5, 7))
# Compute the changes in parameter estimates if a case is included
# vs. if this case is excluded.
# That is, case influence on parameter estimates.
# For structural paths only
out <- est_change_raw(fit_rerun, parameters = "~")
out
# For standardized paths only
out_std <- est_change_raw(fit_rerun, parameters = "~",
                          standardized = TRUE)
out_std
```

`est_change_raw_approx` *Case Influence on Parameter Estimates (Approximate)*

Description

Gets a `lavaan::lavaan()` output and computes the approximate changes in selected parameters for each case if included.

Usage

```
est_change_raw_approx(
  fit,
  parameters = NULL,
  case_id = NULL,
  allow_inadmissible = FALSE,
  skip_all_checks = FALSE
)
```

Arguments

`fit`	The output from `lavaan::lavaan()` or its wrappers (e.g., `lavaan::cfa()` and `lavaan::sem()`).
`parameters`	A character vector to specify the selected parameters. Each parameter is named as in `lavaan` syntax, e.g., `x ~ y` or `x ~~ y`, as appeared in the columns `lhs`, `op`, and `rhs` in the output of `lavaan::parameterEstimates()`. Supports specifying an operator to select all parameters with these operators: `~`, `~~`, `=~`, and `~1`.

This vector can contain both parameter names and operators. More details can be found in the help of pars_id(). If omitted or NULL, the default, changes on all free parameters will be computed.

case_id
: If it is a character vector of length equals to the number of cases (the number of rows in the data in fit), then it is the vector of case identification values. If it is NULL, the default, then case.idx used by lavaan functions will be used as case identification values.

allow_inadmissible
: If TRUE, accepts a fit object with inadmissible results (i.e., post.check from lavaan::lavInspect() is FALSE). Default is FALSE.

skip_all_checks
: If TRUE, skips all checks and allows users to run this function on any object of lavaan class. For users to experiment this and other functions on models not officially supported. Default is FALSE.

Details

For each case, est_change_raw_approx() computes the approximate differences in the estimates of selected parameters with and without this case:

(Estimate with all case) - (Estimate without this case).

The change is the approximate raw change. The change is *not* divided by the standard error of an estimate (hence "raw" in the function name). This is a measure of the influence of a case on the parameter estimates if it is included.

If the value of a case is positive, including the case increases an estimate.

If the value of a case is negative, including the case decreases an estimate.

The model is not refitted. Therefore, the result is only an approximation of that of est_change_raw(). However, this approximation is useful for identifying potentially influential cases when the sample size is very large or the model takes a long time to fit. This function can be used to identify potentially influential cases quickly and then select them to conduct the leave-one-out sensitivity analysis using lavaan_rerun() and est_change_raw().

Unlike est_change_raw(), it does not yet support computing the changes for the standardized solution.

For the technical details, please refer to the vignette on this approach: vignette("casewise_scores", package = "semfindr")

The approximate approach supports a model with equality constraints (available in 0.1.4.8 and later version).

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

An est_change-class object, which is matrix with the number of columns equals to the number of requested parameters, and the number of rows equals to the number of cases. The row names are case identification values. The elements are the raw differences. A print method is available for user-friendly output.

Author(s)

Idea by Mark Hok Chio Lai https://orcid.org/0000-0002-9196-7406, implemented by Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
# Compute the approximate changes in parameter estimates if a case is included
# vs. if this case is excluded.
# That is, the approximate case influence on parameter estimates.
out_approx <- est_change_raw_approx(fit)
head(out_approx)
# Fit the model several times. Each time with one case removed.
# For illustration, do this only for 10 selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Compute the changes in parameter estimates if a case is included
# vs. if this case is excluded.
# That is, the case influence on the parameter estimates.
out <- est_change_raw(fit_rerun)
out
# Compare the results
plot(out_approx[1:10, 1], out[, 1])
abline(a = 0, b = 1)
plot(out_approx[1:10, 5], out[, 5])
abline(a = 0, b = 1)

# A CFA model
dat <- cfa_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f1 ~~ f2
"
# Fit the model
fit <- lavaan::cfa(mod, dat)
summary(fit)
```

```
# Compute the approximate changes in parameter estimates if a case is included
# vs. if this case is excluded.
# That is, approximate case influence on parameter estimates.
# Compute changes for free loadings only.
out_approx <- est_change_raw_approx(fit,
                                    parameters = "=~")
head(out_approx)

# A latent variable model
dat <- sem_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   a * f1
f3 ~   b * f2
ab := a * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
# Compute the approximate changes in parameter estimates if a case is included
# vs. if this case is excluded.
# That is, the approximate case influence on parameter estimates.
# Compute changes for structural paths only
out_approx <- est_change_raw_approx(fit,
                                    parameters = c("~"))
head(out_approx)
```

`fit_measures_change` *Case Influence on Fit Measures*

Description

Gets a `lavaan_rerun()` output and computes the changes in selected fit measures if a case is included.

Usage

```
fit_measures_change(
  rerun_out,
  fit_measures = c("chisq", "cfi", "rmsea", "tli"),
  baseline_model = NULL
)
```

Arguments

rerun_out	The output from lavaan_rerun().
fit_measures	The argument fit.measures used in lavaan::fitMeasures. Default is c("chisq", "cfi", "rmsea", "tli").
baseline_model	The argument baseline.model used in lavaan::fitMeasures. Default is NULL.

Details

For each case, fit_measures_change() computes the differences in selected fit measures with and without this case:

(Fit measure with all case) - (Fit measure without this case).

If the value of a case is positive, including the case increases an estimate.

If the value of a case is negative, including the case decreases an estimate.

Note that an increase is an improvement in fit for goodness of fit measures such as CFI and TLI, but a decrease is an improvement in fit for badness of fit measures such as RMSEA and model chi-square. This is a measure of the influence of a case on a fit measure if it is included.

If the analysis is not admissible or does not converge when a case is deleted, NAs will be turned for the differences of this case.

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

An fit_measures_change-class object, which is matrix with the number of columns equals to the number of requested fit measures, and the number of rows equals to the number of cases. The row names are the case identification values used in lavaan_rerun(). A print method is available for user-friendly output.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Pek, J., & MacCallum, R. (2011). Sensitivity analysis in structural equation models: Cases and their influence. *Multivariate Behavioral Research, 46*(2), 202-228. doi:10.1080/00273171.2011.561068

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
```

```
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
# Fit the model n times. Each time with one case removed.
# For illustration, do this only for four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Compute the changes in chisq if a case is included
# vs. if this case is removed.
# That is, case influence on model chi-squared.
out <- fit_measures_change(fit_rerun, fit_measures = "chisq")
# Results excluding a case, for the first few cases
head(out)
# Chi-square will all cases included.
(chisq_all <- fitMeasures(fit, c("chisq")))
# Chi-square with the first case removed
fit_01 <- lavaan::sem(mod, dat[-1, ])
(chisq_no_1 <- fitMeasures(fit_01, c("chisq")))
# Difference
chisq_all - chisq_no_1
# Compare to the result from the fit_measures_change
out[1, ]

# A CFA model

dat <- cfa_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f1 ~~ f2
"
# Fit the model
fit <- lavaan::cfa(mod, dat)

fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
out <- fit_measures_change(fit_rerun, fit_measures = "chisq")
head(out)
(chisq_all <- fitMeasures(fit, c("chisq")))
fit_01 <- lavaan::sem(mod, dat[-1, ])
(chisq_no_1 <- fitMeasures(fit_01, c("chisq")))
chisq_all - chisq_no_1
out[1, ]

# A latent variable model

dat <- sem_dat
mod <-
```

```
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   a * f1
f3 ~   b * f2
ab := a * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)

fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
out <- fit_measures_change(fit_rerun, fit_measures = "chisq")
head(out)
(chisq_all <- fitMeasures(fit, c("chisq")))
fit_01 <- lavaan::sem(mod, dat[-1, ])
(chisq_no_1 <- fitMeasures(fit_01, c("chisq")))
chisq_all - chisq_no_1
out[1, ]
```

`fit_measures_change_approx` *Case Influence on Fit Measures (Approximate)*

Description

Gets a `lavaan::lavaan()` output and computes the approximate change in selected fit measures if a case is included.

Usage

```
fit_measures_change_approx(
  fit,
  fit_measures = c("chisq", "cfi", "rmsea", "tli"),
  baseline_model = NULL,
  case_id = NULL,
  allow_inadmissible = FALSE,
  skip_all_checks = FALSE
)
```

Arguments

`fit`	The output from `lavaan::lavaan()` or its wrappers (e.g., `lavaan::cfa()` and `lavaan::sem()`).
`fit_measures`	The argument `fit.measures` used in lavaan::fitMeasures. Default is `c("chisq", "cfi", "rmsea", "tli")`. Currently, the approximate method supports only these four measures.

`baseline_model` The argument `baseline.model` used in lavaan::fitMeasures. Default is `NULL`.

`case_id` If it is a character vector of length equals to the number of cases (the number of rows in the data in `fit`), then it is the vector of case identification values. If it is `NULL`, the default, then `case.idx` used by `lavaan` functions will be used as case identification values.

`allow_inadmissible` If `TRUE`, accepts a fit object with inadmissible results (i.e., `post.check` from `lavaan::lavInspect()` is `FALSE`). Default is `FALSE`.

`skip_all_checks` If `TRUE`, skips all checks and allows users to run this function on any object of `lavaan` class. For users to experiment this and other functions on models not officially supported. Default is `FALSE`.

Details

For each case, `fit_measures_change_approx()` computes the approximate differences in selected fit measures with and without this case:

(Fit measure with all case) - (Fit measure without this case).

If the value of a case is positive, including the case increases an estimate.

If the value of a case is negative, including the case decreases an estimate.

Note that an increase is an improvement in fit for goodness of fit measures such as CFI and TLI, but a decrease is an improvement in fit for badness of fit measures such as RMSEA and model chi-square. This is a measure of the influence of a case on a fit measure if it is included.

The model is not refitted. Therefore, the result is only an approximation of that of `fit_measures_change()`. However, this approximation is useful for identifying potentially influential cases when the sample size is very large or the model takes a long time to fit. This function can be used to identify potentially influential cases quickly and then select them to conduct the leave-one-out sensitivity analysis using `lavaan_rerun()` and `fit_measures_change()`.

For the technical details, please refer to the vignette on this approach: `vignette("casewise_scores", package = "semfindr")`

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

An `fit_measures_change`-class object, which is matrix with the number of columns equals to the number of requested fit measures, and the number of rows equals to the number of cases. The row names are case identification values. A print method is available for user-friendly output.

Author(s)

Idea by Mark Hok Chio Lai https://orcid.org/0000-0002-9196-7406, implemented by Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Approximate changes
out_approx <- fit_measures_change_approx(fit, fit_measures = "chisq")
head(out_approx)
# Fit the model several times. Each time with one case removed.
# For illustration, do this only for four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:5)
# Compute the changes in chisq if a case is included
# vs. if this case is excluded.
# That is, case influence on model chi-squared.
out <- fit_measures_change(fit_rerun, fit_measures = "chisq")
# Case influence, for the first few cases
head(out)
# Compare the results
plot(out_approx[1:5, "chisq"], out)
abline(a = 0, b = 1)

# A CFA model

dat <- cfa_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f1 ~~ f2
"
# Fit the model
fit <- lavaan::cfa(mod, dat)

out_approx <- fit_measures_change_approx(fit, fit_measures = "chisq")
head(out_approx)

fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:5)
# Compute the changes in chisq if a case is included
```

```
# vs. if this case is excluded.
# That is, case influence on fit measures.
out <- fit_measures_change(fit_rerun, fit_measures = "chisq")
# Results excluding a case, for the first few cases
head(out)
# Compare the results
plot(out_approx[1:5, "chisq"], out)
abline(a = 0, b = 1)

# A latent variable model

dat <- sem_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   a * f1
f3 ~   b * f2
ab := a * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)

out_approx <- fit_measures_change_approx(fit, fit_measures = "chisq")
head(out_approx)

fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:5)
# Compute the changes in chisq if a case is excluded
# vs. if this case is included.
# That is, case influence on model chi-squared.
out <- fit_measures_change(fit_rerun, fit_measures = "chisq")
# Case influence, for the first few cases
head(out)
# Compare the results
plot(out_approx[1:5, "chisq"], out)
abline(a = 0, b = 1)
```

`implied_scores` *Implied Scores of Observed Outcome Variables*

Description

Gets a `lavaan::lavaan()` output and computes the implied scores of observed outcome variables.

Usage

```
implied_scores(fit, output = "matrix", skip_all_checks = FALSE)
```

Arguments

fit	The output from lavaan::lavaan(), such as lavaan::cfa() and lavaan::sem().
output	Output type. If "matrix", the default, the output will be combined to one matrix, with cases ordered as in the original dataset (after listwise deletion, if used). If "list", the a list of matrices is returned, even if the model has only one group.
skip_all_checks	If TRUE, skips all checks and allows users to run this function on any object of lavaan class. For users to experiment this and other functions on models not officially supported. Default is FALSE.

Details

The implied scores for each observed outcome variable (the y-variables or the endogenous variables) are simply computed in the same way the predicted scores in a linear regression model are computed.

Currently it supports only single-group and multiple-group path analysis models with only observed variables. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

A matrix of the implied scores if output is "matrix". If output is "list", a list of matrices of the implied scores.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

Examples

```
library(lavaan)
dat <- pa_dat
# For illustration, select only the first 50 cases
dat <- dat[1:50, ]
# The model
mod <-
"
m1 ~ iv1 + iv2
dv ~ m1
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Compute the implied scores for `m1` and `dv`
fit_implied_scores <- implied_scores(fit)
head(fit_implied_scores)
```

index_plot *Index Plot of an Arbitrary Statistic*

Description

A generic index plot function for plotting values of a column # in a matrix.

Usage

```
index_plot(
  object,
  column = NULL,
  plot_title = "Index Plot",
  x_label = NULL,
  cutoff_x_low = NULL,
  cutoff_x_high = NULL,
  largest_x = 1,
  absolute = FALSE,
  point_aes = list(),
  vline_aes = list(),
  hline_aes = list(),
  cutoff_line_aes = list(),
  case_label_aes = list()
)
```

Arguments

object	A matrix-like object, such as the output from influence_stat(), est_change(), est_change_raw(), and their counterparts for the approximate approach.
column	String. The column name of the values to be plotted.
plot_title	The title of the plot. Default is "Index Plot".
x_label	The Label for the vertical axis, for the value of column. Default is NULL. If NULL, then the label is changed to "Statistic" if absolute is FALSE, and "Absolute(Statistics)" if absolute is TRUE.
cutoff_x_low	Cases with values smaller than this value will be labeled. A cutoff line will be drawn at this value. Default is NULL. If NULL, no cutoff line will be drawn for this value.
cutoff_x_high	Cases with values larger than this value will be labeled. A cutoff line will be drawn at this value. Default is NULL. If NULL, no cutoff line will be drawn for this value.
largest_x	The number of cases with the largest absolute value on ‘column“ to be labelled. Default is 1. If not an integer, it will be rounded to the nearest integer.
absolute	Whether absolute values will be plotted. Useful when cases are to be compared on magnitude, ignoring sign. Default is FALSE.

point_aes A named list of arguments to be passed to ggplot2::geom_point() to modify how to draw the points. Default is list() and internal default settings will be used.

vline_aes A named list of arguments to be passed to ggplot2::geom_segment() to modify how to draw the line for each case in the index plot. Default is list() and internal default settings will be used.

hline_aes A named list of arguments to be passed to ggplot2::geom_hline() to modify how to draw the horizontal line for zero case influence. Default is list() and internal default settings will be used.

cutoff_line_aes A named list of arguments to be passed to ggplot2::geom_hline() to modify how to draw the line for user cutoff values. Default is list() and internal default settings will be used.

case_label_aes A named list of arguments to be passed to ggrepel::geom_label_repel() to modify how to draw the labels for cases marked (based on arguments such as cutoff_x_low or largest_x). Default is list() and internal default settings will be used.

Details

This index plot function is for plotting any measure of influence or extremeness in a matrix. It can be used for measures not supported with other functions.

Like functions such as gcd_plot() and est_change_plot(), it supports labelling cases based on the values on the selected measure (originaL values or absolute values).

Users can also plot cases based on the absolute values. This is useful when cases are to be compared on magnitude, ignoring the sign.

Value

A ggplot2 plot. Plotted by default. If assigned to a variable or called inside a function, it will not be plotted. Use plot() to plot it.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

See Also

influence_stat(), est_change(), est_change_raw()

Examples

```
library(lavaan)
dat <- pa_dat
# The model
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
```

```
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# --- Leave-One-Out Approach

# Fit the model n times. Each time with one case removed.
# For illustration, do this only for selected cases.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Get all default influence stats
out <- influence_stat(fit_rerun)

# Plot case influence on chi-square. Label the 3 cases with the influence.
index_plot(out, "chisq", largest_x = 3)

# Plot absolute case influence on chi-square.
index_plot(out, "chisq", absolute = TRUE)
```

`influence_plot` *Plots of Influence Measures*

Description

Gets an `influence_stat()` output and plots selected statistics.

Usage

```
gcd_plot(
  influence_out,
  cutoff_gcd = NULL,
  largest_gcd = 1,
  point_aes = list(),
  vline_aes = list(),
  cutoff_line_aes = list(),
  case_label_aes = list()
)

md_plot(
  influence_out,
  cutoff_md = FALSE,
  cutoff_md_qchisq = 0.975,
  largest_md = 1,
  point_aes = list(),
```

```
  vline_aes = list(),
  cutoff_line_aes = list(),
  case_label_aes = list()
)

gcd_gof_plot(
  influence_out,
  fit_measure,
  cutoff_gcd = NULL,
  cutoff_fit_measure = NULL,
  largest_gcd = 1,
  largest_fit_measure = 1,
  point_aes = list(),
  hline_aes = list(),
  cutoff_line_gcd_aes = list(),
  cutoff_line_fit_measures_aes = list(),
  case_label_aes = list()
)

gcd_gof_md_plot(
  influence_out,
  fit_measure,
  cutoff_md = FALSE,
  cutoff_fit_measure = NULL,
  circle_size = 2,
  cutoff_md_qchisq = 0.975,
  cutoff_gcd = NULL,
  largest_gcd = 1,
  largest_md = 1,
  largest_fit_measure = 1,
  point_aes = list(),
  hline_aes = list(),
  cutoff_line_md_aes = list(),
  cutoff_line_gcd_aes = list(),
  cutoff_line_fit_measures_aes = list(),
  case_label_aes = list()
)
```

Arguments

influence_out	The output from influence_stat().
cutoff_gcd	Cases with generalized Cook's distance or approximate generalized Cook's distance larger than this value will be labeled. Default is NULL. If NULL, no cutoff line will be drawn.
largest_gcd	The number of cases with the largest generalized Cook's distance or approximate generalized Cook's distance to be labelled. Default is 1. If not an integer, it will be rounded to the nearest integer.
point_aes	A named list of arguments to be passed to ggplot2::geom_point() to modify

how to draw the points. Default is `list()` and internal default settings will be used.

`vline_aes` A named list of arguments to be passed to `ggplot2::geom_segment()` to modify how to draw the line for each case in the index plot. Default is `list()` and internal default settings will be used.

`cutoff_line_aes`
A named list of arguments to be passed to `ggplot2::geom_vline()` or `ggplot2::geom_hline()` to modify how to draw the line for user cutoff value. Default is `list()` and internal default settings will be used.

`case_label_aes` A named list of arguments to be passed to `ggrepel::geom_label_repel()` to modify how to draw the labels for cases marked (based on arguments such as `cutoff_gcd` or `largest_gcd`). Default is `list()` and internal default settings will be used.

`cutoff_md` Cases with Mahalanobis distance larger than this value will be labeled. If it is `TRUE`, the (`cutoff_md_qchisq` x 100)th percentile of the chi-square distribution with the degrees of freedom equal to the number of variables will be used. Default is `FALSE`, no cutoff value.

`cutoff_md_qchisq`
This value multiplied by 100 is the percentile to be used for labeling case based on Mahalanobis distance. Default is .975.

`largest_md` The number of cases with the largest Mahalanobis distance to be labelled. Default is 1. If not an integer, it will be rounded to the nearest integer.

`fit_measure` The fit measure to be used in a plot. Use the name in the `lavaan::fitMeasures()` function. No default value.

`cutoff_fit_measure`
Cases with `fit_measure` larger than this cutoff in magnitude will be labeled. No default value and must be specified.

`largest_fit_measure`
The number of cases with the largest selected fit measure change in magnitude to be labelled. Default is

1. If not an integer, it will be rounded to the nearest integer.

`hline_aes` A named list of arguments to be passed to `ggplot2::geom_hline()` to modify how to draw the horizontal line for zero case influence. Default is `list()` and internal default settings will be used.

`cutoff_line_gcd_aes`
Similar to `cutoff_line_aes` but control the line for the cutoff value of *gCD*.

`cutoff_line_fit_measures_aes`
Similar to `cutoff_line_aes` but control the line for the cutoff value of the selected fit measure.

`circle_size` The size of the largest circle when the size of a circle is controlled by a statistic.

`cutoff_line_md_aes`
Similar to `cutoff_line_aes` but control the line for the cutoff value of the Mahalanobis distance.

Details

The output of `influence_stat()` is simply a matrix. Therefore, these functions will work for any matrix provided. Row number will be used on the x-axis if applicable. However, case identification values in the output from `influence_stat()` will be used for labeling individual cases.

The default settings for the plots should be good enough for diagnostic purpose. If so desired, users can use the `*_aes` arguments to nearly fully customize all the major elements of the plots, as they would do for building a ggplot2 plot.

Value

A ggplot2 plot. Plotted by default. If assigned to a variable or called inside a function, it will not be plotted. Use `plot()` to plot it.

Functions

- `gcd_plot()`: Index plot of generalized Cook's distance.
- `md_plot()`: Index plot of Mahalanobis distance.
- `gcd_gof_plot()`: Plot the case influence of the selected fit measure against generalized Cook's distance.
- `gcd_gof_md_plot()`: Bubble plot of the case influence of the selected fit measure against Mahalanobis distance, with the size of a bubble determined by generalized Cook's distance.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Pek, J., & MacCallum, R. (2011). Sensitivity analysis in structural equation models: Cases and their influence. *Multivariate Behavioral Research, 46*(2), 202-228. doi:10.1080/00273171.2011.561068

See Also

`influence_stat()`.

Examples

```
library(lavaan)
dat <- pa_dat
# The model
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
```

```
summary(fit)
# Fit the model n times. Each time with one case removed.
# For illustration, do this only for selected cases.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Get all default influence stats
out <- influence_stat(fit_rerun)
head(out)

# Plot generalized Cook's distance. Label the 3 cases with the largest distances.
gcd_plot(out, largest_gcd = 3)

# Plot Mahalanobis distance. Label the 3 cases with the largest distances.
md_plot(out, largest_md = 3)

# Plot case influence on model chi-square against generalized Cook's distance.
# Label the 3 cases with the largest absolute influence.
# Label the 3 cases with the largest generalized Cook's distance.
gcd_gof_plot(out, fit_measure = "chisq", largest_gcd = 3,
             largest_fit_measure = 3)

# Plot case influence on model chi-square against Mahalanobis distance.
# Size of bubble determined by generalized Cook's distance.
# Label the 3 cases with the largest absolute influence.
# Label the 3 cases with the largest Mahalanobis distance.
# Label the 3 cases with the largest generalized Cook's distance.

gcd_gof_md_plot(out, fit_measure = "chisq",
                     largest_gcd = 3,
                     largest_fit_measure = 3,
                     largest_md = 3,
                     circle_size = 10)

# Use the approximate method that does not require refitting the model.

# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
out <- influence_stat(fit)
head(out)

# Plot approximate generalized Cook's distance.
# Label the 3 cases with the largest values.
gcd_plot(out, largest_gcd = 3)

# Plot Mahalanobis distance.
# Label the 3 cases with the largest values.
md_plot(out, largest_md = 3)

# Plot approximate case influence on model chi-square against
# approximate generalized Cook's distance.
# Label the 3 cases with the largest absolute approximate case influence.
# Label the 3 cases with the largest approximate generalized Cook's distance.
```

```
gcd_gof_plot(out, fit_measure = "chisq", largest_gcd = 3,
             largest_fit_measure = 3)

# Plot approximate case influence on model chi-square against Mahalanobis distance.
# The size of a bubble determined by approximate generalized Cook's distance.
# Label the 3 cases with the largest absolute approximate case influence.
# Label the 3 cases with the largest Mahalanobis distance.
# Label the 3 cases with the largest approximate generalized Cook's distance.

gcd_gof_md_plot(out, fit_measure = "chisq",
                     largest_gcd = 3,
                     largest_fit_measure = 3,
                     largest_md = 3,
                     circle_size = 10)

# Customize elements in the plot.
# For example, change the color and shape of the points.

gcd_gof_plot(out, fit_measure = "chisq", largest_gcd = 3,
             largest_fit_measure = 3,
             point_aes = list(shape = 3, color = "red"))
```

`influence_stat` *Case Influence Measures*

Description

Gets a `lavaan_rerun()` output and computes the changes in selected parameters and fit measures for each case if included.

Usage

```
influence_stat(
  rerun_out,
  fit_measures = c("chisq", "cfi", "rmsea", "tli"),
  baseline_model = NULL,
  parameters = NULL,
  mahalanobis = TRUE,
  keep_fit = TRUE
)
```

Arguments

`rerun_out`	The output from `lavaan_rerun()`, or the output of `lavaan::lavaan()` or its wrappers (e.g., `lavaan::cfa()` and `lavaan::sem()`).
`fit_measures`	The argument `fit.measures` used in lavaan::fitMeasures. Default is `c("chisq", "cfi", "rmsea", "tli")`.

baseline_model	The argument baseline.model used in lavaan::fitMeasures. Default is NULL.
parameters	A character vector to specify the selected parameters. Each parameter is named as in lavaan syntax, e.g., x ~ y or x ~~ y, as appeared in the columns lhs, op, and rhs in the output of lavaan::parameterEstimates(). Supports specifying an operator to select all parameters with this operators: ~, ~~, =~, and ~1. This vector can contain both parameter names and operators. More details can be found in the help of pars_id(). If omitted or NULL, the default, changes on all free parameters will be computed.
mahalanobis	If TRUE, it will call mahalanobis_rerun() to compute the Mahalanobis distance. Default is TRUE.
keep_fit	If TRUE, it will keep the original lavaan output using the full sample as an attribute to the output. It can be used by other functions to extract necessary information. Default is TRUE.

Details

For each case, influence_stat() computes the differences in the estimates of selected parameters and fit measures with and without this case. Users can also request a measure of extremeness (only Mahalanobis distance is available for now).

If rerun_out is the output of lavaan_rerun(), it will use the leave-one-out approach. Measures are computed by est_change() and fit_measures_change().

If rerun_out is the output of lavaan::lavaan() or its wrappers (e.g., lavaan::cfa() or lavaan::sem()), it will use the approximate approach. Measures are computed by est_change_approx() and fit_measures_change_approx().

If Mahalanobis distance is requested, it is computed by mahalanobis_rerun().

Please refer to the help pages of the above functions on the technical details.

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

An influence_stat-class object, which is a matrix with the number of columns equals to the number of requested statistics, and the number of rows equals to the number of cases. The row names are the case identification values used in lavaan_rerun(). Please refer to the help pages of est_change() and fit_measures_change() (or est_change_approx() and fit_measures_change_approx() for details. This object has a print method for printing user-friendly output.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Pek, J., & MacCallum, R. (2011). Sensitivity analysis in structural equation models: Cases and their influence. *Multivariate Behavioral Research, 46*(2), 202-228. doi:10.1080/00273171.2011.561068

See Also

fit_measures_change(), est_change(), and mahalanobis_rerun().

Examples

```
library(lavaan)
dat <- pa_dat
# The model
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# --- Leave-One-Out Approach

# Fit the model n times. Each time with one case removed.
# For illustration, do this only for selected cases.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Get all default influence stats
out <- influence_stat(fit_rerun)
head(out)

# --- Approximate Approach

out_approx <- influence_stat(fit)
head(out_approx)
```

---

lavaan_rerun *Rerun a 'lavaan' Analysis Using the Leaving-One-Out Approach*

Description

Reruns a lavaan analysis several times, each time with one case removed.

Usage

```
lavaan_rerun(
  fit,
  case_id = NULL,
  to_rerun,
  md_top,
```

```
  resid_md_top,
  allow_inadmissible = FALSE,
  skip_all_checks = FALSE,
  parallel = FALSE,
  makeCluster_args = list(spec = getOption("cl.cores", 2)),
  rerun_method = c("lavaan", "update")
)
```

Arguments

fit	The output from lavaan::lavaan() or its wrappers (e.g., lavaan::cfa() and lavaan::sem()).
case_id	If it is a character vector of length equals to the number of cases (the number of rows in the data in fit), then it is the vector of case identification values. If it is NULL, the default, then case.idx used by lavaan functions will be used as case identification values. The case identification values will be used to name the list of *n* output.
to_rerun	The cases to be processed. If case_id is specified, this should be a subset of case_id. If case_id is not specified, then this should be a vector of integers indicating the rows to te processed, as appeared in the data in fit. to_rerun cannot be used together with md_top or resid_md_top.
md_top	The number of cases to be processed based on the Mahalanobis distance computed on all observed variables used in the model. The cases will be ranked from the largest to the smallest distance, and the top md_top case(s) will be processed. md_top cannot be used together with to_rerun or resid_md_top.
resid_md_top	The number of cases to be processed based on the Mahalanobis distance computed from the residuals of outcome variables. The cases will be ranked from the largest to the smallest distance, and the top resid_md_top case(s) will be processed. resid_md_top cannot be used together with to_rerun or md_top.
allow_inadmissible	If TRUE, accepts a fit object with inadmissible results (i.e., post.check from lavaan::lavInspect() is FALSE). Default is FALSE.
skip_all_checks	If TRUE, skips all checks and allow users to run this function on any object of lavaan class. For users to experiment this and other functions on models not officially supported. Default is FALSE.
parallel	Whether parallel will be used. If TRUE, will use functions in the parallel package to rerun the analysis. Currently, only support "snow" type clusters using local CPU cores. Default is FALSE.
makeCluster_args	A named list of arguments to be passed to parallel::makeCluster(). Default is list(spec = getOption("cl.cores", 2))). If only the number of cores need to be specified, use list(spec = x), where x is the number of cores to use.
rerun_method	How fit will be rerun. Default is "lavaan". An alternative method is "update". For internal use. If "lavaan" returns an error, try setting this argument to "update".

Details

`lavaan_rerun()` gets an `lavaan::lavaan()` output and reruns the analysis *n0* times, using the same arguments and options in the output, *n0* equals to the number of cases selected, by default all cases in the analysis. In each run, one case will be removed.

Optionally, users can rerun the analysis with only selected cases removed. These cases can be specified by case IDs, by Mahalanobis distance computed from all variables used in the model, or by Mahalanobis distance computed from the residuals (observed score - implied scores) of observed outcome variables. See the help on the arguments `to_rerun`, `md_top`, and `resid_md_top`.

It is not recommended to use Mahalanobis distance computed from all variables, especially for models with observed variables as predictors (Pek & MacCallum, 2011). Cases that are extreme on predictors may not be influential on the parameter estimates. Nevertheless, this distance is reported in some SEM programs and so this option is provided.

Mahalanobis distance based on residuals are supported for models with no latent factors. The implied scores are computed by `implied_scores()`.

If the sample size is large, it is recommended to use parallel processing. However, it is possible that parallel processing will fail. If this is the case, try to use serial processing, by simply removing the argument `parallel` or set it to `FALSE`.

Many other functions in semfindr use the output from `lavaan_rerun()`. Instead of running the *n* analyses every time, do this step once and then users can compute whatever influence statistics they want quickly.

If the analysis took a few minutes to run due to the large number of cases or the long processing time in fitting the model, it is recommended to save the output to an external file (e.g., by `base::saveRDS()`).

Supports both single-group and multiple-group models. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

A `lavaan_rerun`-class object, which is a list with the following elements:

- `rerun`: The *n* `lavaan` output objects.
- `fit`: The original output from `lavaan`.
- `post_check`: A list of length equals to *n*. Each analysis was checked by lavaan::lavTech`(x, "post.check")`, `x` being the `lavaan` results. The results of this test are stored in this list. If the value is `TRUE`, the estimation converged and the solution is admissible. If not `TRUE`, it is a warning message issued by `lavaan::lavTech()`.
- `converged`: A vector of length equals to *n*. Each analysis was checked by lavaan::lavTech`(x, "converged")`, `x` being the `lavaan` results. The results of this test are stored in this vector. If the value is `TRUE`, the estimation converged. If not `TRUE`, then the estimation failed to converge if the corresponding case is excluded.
- `call`: The call to `lavaan_rerun()`.
- `selected`: A numeric vector of the row numbers of cases selected in the analysis. Its length should be equal to the length of `rerun`.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

Examples

```
library(lavaan)
dat <- pa_dat
# For illustration, select only the first 50 cases
dat <- dat[1:50, ]
# The model
mod <-
"
m1 ~ iv1 + iv2
dv ~ m1
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Fit the model n times. Each time with one case removed.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE)

# Print the output for a brief description of the runs
fit_rerun

# Results excluding the first case
fitMeasures(fit_rerun$rerun[[1]], c("chisq", "cfi", "tli", "rmsea"))
# Results by manually excluding the first case
fit_01 <- lavaan::sem(mod, dat[-1, ])
fitMeasures(fit_01, c("chisq", "cfi", "tli", "rmsea"))
```

---

lavaan_rerun_check *Compatibility Check for 'lavaan_rerun'*

Description

Gets a 'lavaan' output and checks whether it is supported by `lavaan_rerun()`.

Usage

```
lavaan_rerun_check(fit, print_messages = TRUE)
```

Arguments

`fit`	The output from `lavaan`, such as `lavaan::cfa()` and `lavaan::sem()`.
`print_messages`	Logical. If `TRUE`, will print messages about the check. If `FALSE`, the messages will be attached to the return value as an attribute. Default is `TRUE`.

Details

This function is not supposed to be used by users. It is called by lavaan_rerun() to see if the analysis being passed to it is supported. If not, messages will be printed to indicate why.

Value

A single-element vector. If confirmed to be supported, will return 0. If not confirmed be support but may still work, return 1. If confirmed to be not yet supported, will return a negative number, the value of this number without the negative sign is the number of tests failed.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

Examples

```
dat <- cfa_dat

mod <-
"
f1 =~ x4 + x5 + x6
"
dat_gp <- dat
dat$gp <- rep(c("gp1", "gp2"), length.out = nrow(dat_gp))

fit01 <- lavaan::sem(mod, dat)
# If supported, returns a zero.
lavaan_rerun_check(fit01)

fit05 <- lavaan::cfa(mod, dat, group = "gp")
# If not supported, returns a negative number.
lavaan_rerun_check(fit05)
```

---

mahalanobis_predictors

Mahalanobis Distance On Observed Predictors

Description

Gets a lavaan_rerun() or lavaan::lavaan() output and computes the Mahalanobis distance for each case using only the observed predictors.

Usage

```
mahalanobis_predictors(
  fit,
  emNorm_arg = list(estimate.worst = FALSE, criterion = 1e-06)
)
```

Arguments

fit	It can be the output from lavaan, such as lavaan::cfa() and lavaan::sem(), or the output from lavaan_rerun().
emNorm_arg	No longer used. Kept for backward compatibility.

Details

For each case, mahalanobis_predictors() computes the Mahalanobis distance of each case on the observed predictors.

If there are no missing values, stats::mahalanobis() will be used to compute the Mahalanobis distance.

If there are missing values on the observed predictors, the means and variance-covariance matrices will be estimated by maximum likelihood using lavaan::lavCor(). The estimates will be passed to modi::MDmiss() to compute the Mahalanobis distance.

Supports both single-group and multiple-group models. For multiple-group models, the Mahalanobis distance for each case is computed using the means and covariance matrix of the group this case belongs to. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

A md_semfindr-class object, which is a one-column matrix (a column vector) of the Mahalanobis distance for each case. The number of rows equals to the number of cases in the data stored in the fit object. A print method is available for user-friendly output.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Béguin, C., & Hulliger, B. (2004). Multivariate outlier detection in incomplete survey data: The epidemic algorithm and transformed rank correlations. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 167(2), 275-294.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Science of India, 2*, 49-55.

Schafer, J.L. (1997) *Analysis of incomplete multivariate data*. Chapman & Hall/CRC Press.

Examples

```
library(lavaan)
dat <- pa_dat
# For illustration, select only the first 50 cases.
dat <- dat[1:50, ]
# The model
mod <-
"
m1 ~ a1 * iv1 +  a2 * iv2
dv ~ b * m1
```

```
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

md_predictors <- mahalanobis_predictors(fit)
md_predictors
```

`mahalanobis_rerun` *Mahalanobis Distance on All Observed Variables*

Description

Computes the Mahalanobis distance for each case on all observed variables in a model.

Usage

```
mahalanobis_rerun(
  fit,
  emNorm_arg = list(estimate.worst = FALSE, criterion = 1e-06)
)
```

Arguments

`fit`	It can be the output from `lavaan`, such as `lavaan::cfa()` and `lavaan::sem()`, or the output from `lavaan_rerun()`.
`emNorm_arg`	No longer used. Kept for backward compatibility.

Details

`mahalanobis_rerun()` gets a `lavaan_rerun()` or `lavaan::lavaan()` output and computes the Mahalanobis distance for each case on all observed variables.

If there are no missing values, `stats::mahalanobis()` will be used to compute the Mahalanobis distance.

If there are missing values on the observed predictors, the means and variance-covariance matrices will be estimated by maximum likelihood using `lavaan::lavCor()`. The estimates will be passed to `modi::MDmiss()` to compute the Mahalanobis distance.

Supports both single-group and multiple-group models. For multiple-group models, the Mahalanobis distance for each case is computed using the means and covariance matrix of the group this case belongs to. (Support for multiple-group models available in 0.1.4.8 and later version).

Value

A `md_semfindr`-class object, which is a one-column matrix (a column vector) of the Mahalanobis distance for each case. The row names are the case identification values used in `lavaan_rerun()`. A print method is available for user-friendly output.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Science of India, 2*, 49-55.

Examples

```
library(lavaan)
dat <- pa_dat
# The model
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
# Fit the model n times. Each time with one case removed.
# For illustration, do this only for selected cases.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Compute the Mahalanobis distance for each case
out <- mahalanobis_rerun(fit_rerun)
# Results excluding a case, for the first few cases
head(out)
# Compute the Mahalanobis distance using stats::mahalanobis()
md1 <- stats::mahalanobis(dat, colMeans(dat), stats::cov(dat))
# Compare the results
head(md1)

# A CFA model

dat <- cfa_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f1 ~~ f2
"
# Fit the model
fit <- lavaan::cfa(mod, dat)

fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
mahalanobis_rerun(fit_rerun)
```

```
# A latent variable model

dat <- sem_dat
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   a * f1
f3 ~   b * f2
ab := a * b
"
# Fit the model
fit <- lavaan::cfa(mod, dat)

fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
mahalanobis_rerun(fit_rerun)
```

---

pars_id *Convert Parameter Syntax to Position or Row Numbers in the Parameter Vector or Table*

Description

Converts a vector of lavaan syntax to the ids of parameters in the vector of free parameters or the row numbers in the parameter table.

Usage

```
pars_id(pars, fit, where = c("coef", "partable"), free_only = TRUE)
```

Arguments

pars	A character vector of parameters specified in lavaan syntax, e.g., "y ~ x" and f1 =~ x3. For multisample models, if only the parameters in some groups are needed, use the modifier for labeling parameters and use NA to denote parameters to be requested. E.g., f1 =~ c(NA, 0, NA, NA) * x2 denotes the loadings of x2 on f1 in the first, third, and fourth groups.
fit	A lavaan-class object. This object is used to determine the number of groups and the parameters in the model. Only parameters in pars that appear in this model will be considered.
where	Where the values are to be found. Can be "partable" (parameter table) or "coef" (coefficient vector). Default is "coef".
free_only	Whether only free parameters will be kept. Default is TRUE.

Details

It supports the following ways to specify the parameters to be included.

- lavaan syntax
  - For example, "y ~ x" denotes the regression coefficient regression y on x. It uses lavaan::lavaanify() to parse the syntax strings.
- Operator
  - For example, "~" denotes all regression coefficients.
  - It also supports :=, which can be used to select user-defined parameters.
- Label
  - For example, "ab" denotes all parameters with this labels defined in model syntax. It can be used to select user-defined parameters, such as "ab := a*b".

It is used by functions such as est_change().

Multisample model:

If a model has more than one group, a specification specified as in a single sample model denotes the same parameters in all group.

- For example, "f1 =~ x2" denotes the factor loading of x2 on f1 in all groups. "~~" denotes covariances and error covariances in all groups.

There are two ways to select parameters only in selected groups. First, the syntax to fix parameter values can be used, with NA denoting parameters to be selected.

- For example, "f2 =~ c(NA, 1, NA) * x5" selects the factor loadings of x5 on f2 in the first and third groups.

Users can also add ".grouplabel" to a specification, grouplabel being the group label of a group (the one appears in summary(), not the one of the form ".g2", "g3", etc.).

- For example, "f2 =~ x5.Alpha" denotes the factor loading of x5 on f2 in the group "Alpha".
- This method can be used for operators. For example, "=~.Alpha" denotes all factors loadings in the group "Alpha".

Though not recommended, users can use labels such as ".g2" and ".g3" to denote the parameter in a specific group. These are the labels appear in the output of some functions of lavaan. Although lavaan does not label the parameters in the first group by ".g1", this can still be used in pars_id().

- For example, "f2 =~ x5.g2" denotes the factor loading of x5 on f2 in the second group. "y ~ x.g1" denotes the regression coefficient from x to y in the first group.
- This method can also be used for operators. For example, "=~.g2" denotes all factors loadings in the second group.

However, this method is not as reliable as using grouplabel because the numbering of groups depends on the order they appear in the data set.

Value

A numeric vector of the ids. If where is "partable", the ids are row numbers. If where is "coef", the ids are the positions in the vector.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

Examples

```
dat <- sem_dat

library(lavaan)
sem_model <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   f1
f3 ~   f2
"

fit_ng <- sem(sem_model, dat)

pars <- c("f1 =~ x2", "f2 =~ x5", "f2 ~ f1")
tmp <- pars_id(pars, fit = fit_ng)
coef(fit_ng)[tmp]
tmp <- pars_id(pars, fit = fit_ng, where = "partable")
parameterTable(fit_ng)[tmp, ]

# Multiple-group models

dat <- sem_dat
set.seed(64264)
dat$gp <- sample(c("Alpha", "Beta", "Gamma"),
                 nrow(dat),
                 replace = TRUE)

library(lavaan)
sem_model <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   f1
f3 ~   f2
"

fit_ng <- sem(sem_model, dat)
fit_gp <- sem(sem_model, dat, group = "gp")

pars <- c("f1 =~ x2", "f2 =~ x5", "f2 ~ f1")
tmp <- pars_id(pars, fit = fit_ng)
coef(fit_ng)[tmp]
tmp <- pars_id(pars, fit = fit_ng, where = "partable")
parameterTable(fit_ng)[tmp, ]
```

```
pars <- c("f1 =~ x2", "f2 =~ c(NA, 1, NA) * x5")
tmp <- pars_id(pars, fit = fit_gp)
coef(fit_gp)[tmp]
tmp <- pars_id(pars, fit = fit_gp, where = "partable")
parameterTable(fit_gp)[tmp, ]

pars2 <- c("f1 =~ x2", "~~.Beta", "f2 =~ x5.Gamma")
tmp <- pars_id(pars2, fit = fit_gp)
coef(fit_gp)[tmp]
tmp <- pars_id(pars2, fit = fit_gp, where = "partable")
parameterTable(fit_gp)[tmp, ]
# Note that group 1 is "Beta", not "Alpha"
lavInspect(fit_gp, "group.label")
```

`pars_id_to_lorg` *Ids to "lhs-op-rhs-(group)"*

Description

Converts id numbers generated by `pars_id()` to values that can be used to extract the parameters from a source.

Usage

```
pars_id_to_lorg(pars_id, pars_source, type = c("free", "all"))
```

Arguments

`pars_id`	A vector of integers. Usually the output of pars_id.
`pars_source`	Can be the output of `lavaan::parameterEstimates()` or `lavaan::parameterTable()`, or a named vector of free parameters (e.g., the output of `coef()` applied to a `lavaan`-class object).
`type`	The meaning of the values in `pars_id`. If `"free"`, they are the position in the vector of free parameters (i.e., the output of of `coef()`). If "all", they are the row numbers in the parameter table (the output of `lavaan::parameterTable()`). If `pars_source` is the output of `lavaan::parameterEstimates()`, which does not indicate whether a parameter is free or fixed, this argument will be ignored.

Details

If the source is a parameter estimates table (i.e., the output of `lavaan::parameterEstimates()`, it returns a data frame with columns "lhs", "op", and "rhs". If "group" is present in the source, it also add a column "group". These columns can be used to uniquely identify the parameters specified by the ids.

If the source is a named vector of parameters (e.g., the output of `coef()`), it returns the names of parameters based on the ids.

Value

If `pars_source` is the output of `lavaan::parameterEstimates()` or `lavaan::parameterTable()`, it returns a subset of `pars_source`, keeping the rows of selected parameters and the columns `lhs`, `op`, `rhs`, and `group`. If `pars_source` is a named vector of free parameters, it returns a character vector containing the names of the selected parameters.

Examples

```
dat <- sem_dat
set.seed(64264)
library(lavaan)
sem_model <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~   f1
f3 ~   f2
"
fit_ng <- sem(sem_model, dat)

pars <- c("f1 =~ x2", "f2 =~ x5", "f2 ~ f1")
tmp <- pars_id(pars, fit = fit_ng)
pars_id_to_lorg(tmp, pars_source = coef(fit_ng))
tmp <- pars_id(pars, fit = fit_ng, where = "partable")
pars_id_to_lorg(tmp, pars_source = parameterEstimates(fit_ng))

# Multiple-group models

dat$gp <- sample(c("Alpha", "Beta", "Gamma"),
                 nrow(dat),
                 replace = TRUE)

fit_gp <- sem(sem_model, dat, group = "gp")

pars <- c("f1 =~ x2", "f2 =~ c(NA, 1, NA) * x5")
tmp <- pars_id(pars, fit = fit_gp)
pars_id_to_lorg(tmp, pars_source = coef(fit_gp))
tmp <- pars_id(pars, fit = fit_gp, where = "partable")
pars_id_to_lorg(tmp, pars_source = parameterEstimates(fit_gp))

parameterTable(fit_gp)[tmp, ]
pars2 <- c("f1 =~ x2", "~~.Beta", "f2 =~ x5.Gamma")
tmp <- pars_id(pars2, fit = fit_gp)
pars_id_to_lorg(tmp, pars_source = coef(fit_gp))
tmp <- pars_id(pars2, fit = fit_gp, where = "partable")
pars_id_to_lorg(tmp, pars_source = parameterEstimates(fit_gp))
# Note that group 1 is "Beta", not "Alpha"
lavInspect(fit_gp, "group.label")
```

pa_dat *Sample Data: A Path Model*

Description

A four-variable dataset with 100 cases.

Usage

```
pa_dat
```

Format

A data frame with 100 rows and 5 variables:

m1 Mediator. Numeric.

dv Outcome variable. Numeric.

iv1 Predictor. Numeric.

iv2 Predictor. Numeric.

Examples

```
library(lavaan)
data(pa_dat)
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
fit <- sem(mod, pa_dat)
summary(fit)
```

pa_dat2 *Sample Data: A Path Model with an Influential Case*

Description

A four-variable dataset with 100 cases, with one influential case.

Usage

```
pa_dat2
```

Format

A data frame with 100 rows and 5 variables:

case_id Case ID. Character.

iv1 Predictor. Numeric.

iv2 Predictor. Numeric.

m1 Mediator. Numeric.

dv Outcome variable. Numeric.

Examples

```
library(lavaan)
data(pa_dat2)
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
fit <- sem(mod, pa_dat2)
summary(fit)
inf_out <- influence_stat(fit)
gcd_plot(inf_out)
```

---

print.est_change *Print an 'est_change' Class Object*

Description

Print the content of an 'est_change'-class object.

Usage

```
## S3 method for class 'est_change'
print(x, digits = 3, first = 10, sort_by = c("gcd", "est"), ...)
```

Arguments

x	An 'est_change'-class object.
digits	The number of digits after the decimal. Default is 3.
first	Numeric. If not NULL, it prints only the first *k* cases, *k* equal to first. Default is 10.

sort_by — String. Should be "est", "gcd", or NULL. If the output was generated by est_change_raw() or est_change_raw_approx() and sort_by is not NULL, then each column is sorted individually, with case IDs inserted before each column. If the output was generated by est_change() or est_change_approx() and sort_by is not NULL, then sort_by determines how the cases are sorted. If by is "est", the cases are sorted as for the output of est_change_raw(). If by is "gcd", the default for the output of est_change() or est_change_approx(), then cases are sorted by generalized Cook's distance or approximate generalized Cook's distance, depending on which column is available.

... — Other arguments. They will be ignored.

Details

All the functions on case influence on parameter estimates, est_change(), est_change_approx(), est_change_raw(), and est_change_raw_approx(), return an est_change-class object. This method will print the output based on the type of changes and method used.

Value

x is returned invisibly. Called for its side effect.

See Also

est_change_raw(), est_change_raw_approx(), est_change(), est_change_approx()

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Approximate case influence
out <- est_change_approx(fit)
out
print(out, sort_by = "est")
out <- est_change_raw_approx(fit)
print(out, first = 3)
```

```
# Examine four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(2, 3, 5, 7))
est_change(fit_rerun)
est_change_raw(fit_rerun)
```

`print.fit_measures_change` *Print a 'fit_measures_change' Class Object*

Description

Print the content of a 'fit_measures_change'-class object.

Usage

```
## S3 method for class 'fit_measures_change'
print(
  x,
  digits = 3,
  first = 10,
  sort_by = NULL,
  decreasing = TRUE,
  absolute = TRUE,
  ...
)
```

Arguments

`x`	An 'fit_measures_change'-class object.
`digits`	The number of digits after the decimal. Default is 3.
`first`	Numeric. If not `NULL`, it prints only the first *k* cases, *k* equal to `first`. Default is 10.
`sort_by`	String. Default is `NULL` and the output is not sorted. If set to a column names of `x`, cases will sorted by this columns. The sorting is done on the absolute values if `absolute` is `TRUE`, and in decreasing order if `decreasing` is `TRUE`. If `decrease` is `FALSE`, the order is increasing. If `absolute` is `FALSE`, the sorting is done on the raw values.
`decreasing`	Logical. Whether cases, if sorted, is on decreasing order. Default is `TRUE`. See `sort_by`.
`absolute`	Logical. Whether cases, if sorted, are sorted on absolute values. Default is `TRUE`. See `sort_by`.
`...`	Other arguments. They will be ignored.

Details

All the functions on case influence on fit measures, `fit_measures_change()` and `fit_measures_change_approx()`, return an `fit_measures_change`-class object. This method will print the output, with the option to sort the cases.

Value

`x` is returned invisibly. Called for its side effect.

See Also

`fit_measures_change()`, `fit_measures_change_approx()`

Examples

```
library(lavaan)

# A path model

dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# Case influence
out <- fit_measures_change_approx(fit)
out
print(out, sort_by = "chisq", first = 5)

fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(2, 3, 5, 7))#'
out <- fit_measures_change(fit_rerun)
out
print(out, sort_by = "chisq", first = 5)
```

---

print.influence_stat *Print an 'influence_stat' Class Object*

Description

Print the content of an 'influence_stat'-class object.

Usage

```
## S3 method for class 'influence_stat'
print(
  x,
  digits = 3,
  what = c("parameters", "fit_measures", "mahalanobis"),
  first = 10,
  sort_parameters_by = c("gcd", "est"),
  sort_fit_measures_by = NULL,
  sort_mahalanobis = TRUE,
  sort_fit_measures_decreasing = TRUE,
  sort_fit_measures_on_absolute = TRUE,
  sort_mahalanobis_decreasing = TRUE,
  ...
)
```

Arguments

x	An 'influence_stat'-class object.
digits	The number of digits after the decimal. Default is 3.
what	A character vector of the results #' to be printed, can be one or more of the following: "parameters", "fit_measures", and "mahalanobis". Default is c("parameters", "fit_measures", "mahalanobis").
first	Numeric. If not NULL, it prints only the first k cases, k equal to first. Default is 10.
sort_parameters_by	String. If it is "est", the cases are sorted individually on each columns. If it is "gcd", the default, then cases are sorted by generalized Cook's distance or approximate generalized Cook's distance, depending on which column is available. If NULL, cases are not sorted.
sort_fit_measures_by	String. Default is NULL and the output of case influence on fit measures is not sorted. If set to a column names of case influence on fit measures , cases will sorted by these columns. The sorting is done on the absolute values if sort_fit_measures_on_absolute is TRUE, and in decreasing order if decreasing is TRUE. If decrease is FALSE, the order is increasing. If sort_fit_measures_on_absolute is FALSE, the sorting is done on the raw values.
sort_mahalanobis	Logical. If TRUE, the default, the cases in the output of Mahalanobis distance will be sorted based on Mahalanobis distance. The order is determined by sort_mahalanobis_decreasing.
sort_fit_measures_decreasing	Logical. Whether cases, if sorted on fit measures, are on decreasing order in the output of case influence on fit measures. Default is TRUE.
sort_fit_measures_on_absolute	Logical. Whether cases, if sorted on fit measures, are sorted on absolute values of fit measures. Default is TRUE. See sort_fit_measures_by.

sort_mahalanobis_decreasing
: Logical. Whether cases, if sorted on Mahalanobis distance, is on decreasing order. Default is TRUE.

...
: Optional arguments. Passed to other print methods, such as print.est_change(), print.fit_measures_change(), and print.md_semfindr().

Details

This method will print the output of influence_stat() in a user-friendly way. Users can select the set(s) of output, case influence on parameter estimates, case influence on fit measures, and Mahalanobis distance, to be printed. The corresponding print methods of est_change-class objects, fit_measures_change-class objects, and md_semfindr-class objects will be called.

Value

x is returned invisibly. Called for its side effect.

See Also

influence_stat(), print.est_change(), print.fit_measures_change(), print.md_semfindr()

Examples

```
library(lavaan)
dat <- pa_dat
# The model
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)

# --- Leave-One-Out Approach

# Fit the model n times. Each time with one case removed.
# For illustration, do this only for selected cases.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Get all default influence stats
out <- influence_stat(fit_rerun)
out
print(out, first = 4)
print(out, what = c("parameters", "fit_measures"))

# --- Approximate Approach

out_approx <- influence_stat(fit)
```

```
out_approx
print(out, first = 8)
print(out, what = c("parameters", "fit_measures"),
      sort_parameters_by = "est")
```

`print.lavaan_rerun` *Print Method for 'lavaan_rerun'*

Description

Prints the results of `lavaan_rerun()`.

Usage

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

Arguments

`x`	The output of `lavaan_rerun()`.
`...`	Other arguments. They will be ignored.

Value

`x` is returned invisibly. Called for its side effect.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

Examples

```
library(lavaan)
dat <- pa_dat
# For illustration only, select only the first 50 cases
dat <- dat[1:50, ]
# The model
mod <-
"
m1 ~ iv1 + iv2
dv ~ m1
"
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
# Fit the model n times. Each time with one case removed.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE)
fit_rerun
```

`print.md_semfindr` *Print a 'md_semfindr' Class Object*

Description

Print the content of a 'md_semfindr'-class object.

Usage

```
## S3 method for class 'md_semfindr'
print(x, digits = 3, first = 10, sort = TRUE, decreasing = TRUE, ...)
```

Arguments

`x`	An 'md_semfindr'-class object.
`digits`	The number of digits after the decimal. Default is 3.
`first`	Numeric. If not `NULL`, it prints only the first *k* cases, *k* equal to `first`. Default is 10.
`sort`	Logical. If `TRUE`, the default, the cases will be sorted based on Mahalanobis distance. The order is determined by `decreasing`.
`decreasing`	Logical. Whether cases, if sorted, is on decreasing order. Default is `TRUE`.
`...`	Other arguments. They will be ignored.

Details

The print method for the 'md_semfindr'-class object, returned by `mahalanobis_rerun()` or `mahalanobis_predictors()`. This method will print the output with the option to sort the cases.

Value

`x` is returned invisibly. Called for its side effect.

See Also

`mahalanobis_rerun()`, `mahalanobis_predictors()`

Examples

```
library(lavaan)
dat <- pa_dat
# The model
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
"
```

```
# Fit the model
fit <- lavaan::sem(mod, dat)
summary(fit)
# Fit the model n times. Each time with one case removed.
# For illustration, do this only for selected cases.
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = 1:10)
# Compute the Mahalanobis distance for each case
out <- mahalanobis_rerun(fit_rerun)
out
print(out, first = 3)
```

`sem_dat` *Sample Data: A Latent Variable Structural Model*

Description

A nine-variable dataset with 200 cases.

Usage

```
sem_dat
```

Format

A data frame with 200 rows and 9 variables:

x1 Indicator. Numeric.

x2 Indicator. Numeric.

x3 Indicator. Numeric.

x4 Indicator. Numeric.

x5 Indicator. Numeric.

x6 Indicator. Numeric.

x7 Indicator. Numeric.

x8 Indicator. Numeric.

x9 Indicator. Numeric.

Examples

```
library(lavaan)
data(sem_dat)
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
```

```
f2 ~ a * f1
f3 ~ b * f2
ab := a * b
"
fit <- sem(mod, sem_dat)
summary(fit)
```

`sem_dat2` *Sample Data: A Latent Variable Structural Model With an Influential Case*

Description

A ten-variable dataset with 200 cases, with one influential case.

Usage

```
sem_dat2
```

Format

A data frame with 200 rows and 10 variables:

case_id Case ID. Character.

x1 Indicator. Numeric.

x2 Indicator. Numeric.

x3 Indicator. Numeric.

x4 Indicator. Numeric.

x5 Indicator. Numeric.

x6 Indicator. Numeric.

x7 Indicator. Numeric.

x8 Indicator. Numeric.

x9 Indicator. Numeric.

Examples

```
library(lavaan)
data(sem_dat2)
mod <-
"
f1 =~  x1 + x2 + x3
f2 =~  x4 + x5 + x6
f3 =~  x7 + x8 + x9
f2 ~ a * f1
f3 ~ b * f2
ab := a * b
```

```
"
fit <- sem(mod, sem_dat2)
summary(fit)
inf_out <- influence_stat(fit)
gcd_plot(inf_out)
```

`user_change_raw` *Case Influence on User-Defined Statistics*

Description

Gets a `lavaan_rerun()` output and computes the changes in user-defined statistics for each case if included.

Usage

```
user_change_raw(rerun_out, user_function = NULL, ...)
```

Arguments

`rerun_out`	The output from `lavaan_rerun()`.
`user_function`	A function that accepts a `lavaan`-class object. This function is for computing user-defined statistics.
`...`	Optional arguments to be passed to `user_function`.

Details

For each case, `user_change_raw()` computes the differences in user-defined statistics with and without this case:

(User statistics with all case) - (User statistics without this case).

The change is the raw change. The change is *not* divided by standard error. This is a measure of the influence of a case on the use-defined statistics if it is included.

If the value of a case is positive, including the case increases a statistic.

If the value of a case is negative, including the case decreases a statistic.

The user-defined statistics are computed by a user-supplied function, `user_function`. It must return a named vector-like object (which can have only one value). The output needs to be named, even if it has only one value.

Value

An `est_change`-class object, which is matrix with the number of columns equals to the number of values returned by `user_function` when computed in one `lavaan`-class object, and the number of rows equals to the number of cases. The row names are the case identification values used in `lavaan_rerun()`. The elements are the raw differences. A print method is available for user-friendly output.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448.

References

Pek, J., & MacCallum, R. (2011). Sensitivity analysis in structural equation models: Cases and their influence. *Multivariate Behavioral Research, 46*(2), 202-228. doi:10.1080/00273171.2011.561068

Examples

```
# A path model

library(lavaan)
dat <- pa_dat
mod <-
"
m1 ~ a1 * iv1 + a2 * iv2
dv ~ b * m1
a1b := a1 * b
a2b := a2 * b
"
# Fit the model
fit <- sem(mod, dat)
summary(fit)
# Fit the model several times. Each time with one case removed.
# For illustration, do this only for four selected cases
fit_rerun <- lavaan_rerun(fit, parallel = FALSE,
                          to_rerun = c(2, 4, 7, 9))
# Get the R-squares
lavInspect(fit, what = "rsquare")
out <- user_change_raw(fit_rerun,
                       user_function = lavInspect,
                       what = "rsquare")
out

# Index plot
p <- index_plot(out,
                column = "dv",
                plot_title = "R-square: dv")
p
```

Index