Package 'semhelpinghands'

January 6, 2023

Title Helper Functions for Structural Equation Modeling

Version 0.1.6

Description An assortment of helper functions for doing structural equation modeling, mainly by 'lavaan' for now. Most of them are time-saving functions for common tasks in doing structural equation modeling and reading the output. This package is not for functions that implement advanced statistical procedures. It is a light-weight package for simple functions that do simple tasks conveniently, with as few dependencies as possible.

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

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

Depends R (>= 4.1.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/parallel true

Config/testthat/edition 3

Imports lavaan, boot

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-01-06 19:10:02 UTC

R topics documented:

`add_exo_cov` *Add Covariances Between Exogenous Variables*

Description

It generates the 'lavaan' model syntax for exogenous variables in a `lavaan` model.

Usage

```
add_exo_cov(model, FUN = "sem", print = TRUE)

auto_exo_cov(model, FUN = "sem", print = TRUE)
```

Arguments

`model`	The model syntax to which the covariances are to be added.
`FUN`	Name (as string) of the `lavaan` wrapper to be called. Normally should be `"sem"`, the default.
`print`	Logical. Whether the generated syntax should also be printed by `cat()`. Default is `TRUE`.

Details

The function `lavaan::sem()` usually will set covariances between "exogenous" variables free when `fixed.x = FALSE` ("exogenous" is defined here as variables that appear on the right hand side but not on the left hand side of the ~ operator'). However, if a covariance between the residual term of an endogenous variable and an exogenous variable is manually set to free, `lavaan::sem()` may not set the aforementioned covariances free. Users will need to free them manually, and there may be a lot of them in some models.

This function gets a model syntax and generates the syntax for these covariances. Users can then inspect it, modify it if necessary, and then copy and paste it to the model syntax.

Value

`add_exo_cov()` returns a one-element character vector of the syntax, with lines separated by "\n". The generated syntax is appended to the input model syntax.

`auto_exo_cov()` returns a one-element character vector of the generated syntax, with lines separated by "\n".

Functions

- `add_exo_cov()`: Add covariances between exogenous variables to the model syntax and than return the modified model syntax.
- `auto_exo_cov()`: Generate the model syntax for the covariances between exogenous variables.

Author(s)

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

Examples

```
library(lavaan)
set.seed(8976223)
n <- 100
x <- rnorm(n)
m <- .5 * x + rnorm(n, 0, sqrt(.4))
z <- rnorm(n)
y <- .4 * m + .3 * z * m + rnorm(n, 0, .5)
dat <- data.frame(x, m, z, y)
dat$zm <- dat$z * dat$m
mod0 <-
"
m ~ x
y ~ m + z + zm
m ~~ z + zm
"
fit <- sem(mod0, dat, fixed.x = FALSE)

# Add covariances. Also printed by default.
mod0_cov <- add_exo_cov(mod0)

# Fit the model
fit_cov <- sem(mod0_cov, dat, fixed.x = FALSE)

# Manually adding the covariances
mod1 <-
"
m ~ x
```

```
y ~ m + z + zm
m ~~ z + zm
z  ~~ zm + x
zm ~~ x
"
fit1 <- sem(mod1, dat, meanstructure = TRUE, fixed.x = FALSE)

# Compare the results

# No manual covariances
fit

# Automatically generated covariances
fit_cov

# Manually added covariances
fit1
```

---

add_sig *Add Significant Test Results*

Description

It inserts columns to denote whether a parameter is significant.

Usage

```
add_sig(object, ..., standardized = FALSE, na_str = "", use = "pvalue")
```

Arguments

object	A lavaan object or the output of `lavaan::parameterEstimates()` or `lavaan::standardizedSolution`. May also work on an `est_table`-class object returned by functions like `group_by_dvs()` but there is no guarantee.
...	Optional arguments to be passed to `lavaan::parameterEstimates()` or `lavaan::standardizedSolut`
standardized	Whether standardized solution is needed. If `TRUE`, `lavaan::standardizedSolution()` will be called. If `FALSE`, the default, `lavaan::parameterEstimates()` will be called. Ignored if a table if estimates is supplied.
na_str	The string to be used for parameters with no significant tests. For example, fixed parameters. Default is `""`.
use	A character vector of one or more strings. If `"pvalue"` is in the vector, *p*-values will be used. If `"ci"` is in the vector, confidence intervals appeared in `ci.lower` and `ci.upper` will be used. If `"boot.ci"` is in the vector and the columns `boot.ci.lower` and `boot.ci.upper` are available, these columns will be used. Note that `ci.lower` and `ci.upper` can also be bootstrap confidence intervals in some tables if `se = "boot"` is used.

Details

The function calls `lavaan::parameterEstimates()` or `lavaan::standardizedSolution()` and checks the columns `pvalue`, `ci.lower` and `ci.upper`, and/or `boot.ci.lower` and `boot.ci.upper` and then inserts columns to denote for each parameter estimate whether it is significant based on the requested criteria.

Value

The output of `lavaan::parameterEstimates()` or `lavaan::standardizedSolution()`, with one or two columns inserted after the parameter estimates to denote the significant test results.

Author(s)

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

See Also

`lavaan::parameterEstimates()` and `lavaan::standardizedSolution()`

Examples

```
library(lavaan)
set.seed(5478374)
n <- 50
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
dat <- data.frame(x = x, y = y, m = m)
model <-
'
m ~ a*x
y ~ b*m
ab := a*b
'
fit <- sem(model, data = dat, fixed.x = FALSE)

# Add "*" based on 'pvalue'
add_sig(fit)

# Add "*" for standardized solution
add_sig(fit, standardized = TRUE)

# Add "*" based on confidence interval
add_sig(fit, use = "ci")

# Add "*" for standardized solution based on confidence interval
add_sig(fit, standardized = TRUE, use = "ci")

# Add "*" for standardized solution based on confidence interval
# and 'pvalue'.
```

```
add_sig(fit, standardized = TRUE, use = c("ci", "pvalue"))

# Can also accept a parameter estimates table
est <- parameterEstimates(fit)
add_sig(est)

# So it can be used with some other functions in semhelpinghands
add_sig(filter_by(est, op = "~"))

# Piping can also be used
est |> filter_by(op = "~") |>
       add_sig()
```

`compare_estimators` *Refit a 'lavaan'-Model by Several Estimators*

Description

Refit a model in 'lavaan' by several lavaan-supported estimators

Usage

```
compare_estimators(object, estimators = NULL)

se_ratios(fit_list, reference = NULL)
```

Arguments

`object`	A lavaan object.
`estimators`	A character vector of the estimator supported by the `estimator` argument of `lavaan::lavaan()` and its wrappers, such as `lavaan::sem()` and `lavaan::cfa()`.
`fit_list`	The output of `compare_estimators()`.
`reference`	The name of the reference method (ratios will be equal to one). Must be one of the estimator used on `compare_estimators()`. If `NULL`, the first estimator will be used.

Details

The function simply uses `lapply()` and `update()` to rerun the analysis once for each of the estimator using `update(object, estimator = "x"`, `x` being the estimator.

The results can then be compared using `group_by_models()`.

Value

A list of `lavaan` outputs, each of them is an update of the original output using one of the estimators.

Functions

- `compare_estimators()`: Refit the model with different estimators.
- `se_ratios()`: A wrapper of `group_by_models()` that computes the ratios of standard errors of different methods to those of one method.

Author(s)

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

See Also

`group_by_models()`

Examples

```
library(lavaan)
set.seed(5478374)
n <- 50
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
dat <- data.frame(x = x, y = y, m = m)
model <-
'
m ~ a*x
y ~ b*m
ab := a*b
'

fit <- sem(model, data = dat, fixed.x = FALSE)

# Refit the model by three different estimators
fit_more <- compare_estimators(fit, estimator = c("GLS", "MLR", "ML"))

# Use group_by_models to compare the estimates
group_by_models(fit_more, col_names = c("est", "pvalue"))

# Use se_ratios to compare standard errors
se_ratios(fit_more, reference = "ML")
```

---

dvs_ivs *Sample Dataset: 3 Predictors and 3 Outcomes*

Description

A path model with three predictors and three outcomes, for illustration.

Usage

```
dvs_ivs
```

Format

A data frame with 100 rows and 7 variables:

y1 Outcome variable 1. Numeric.

y2 Outcome variable 2. Numeric.

y3 Outcome variable 3. Numeric.

x1 Predictor 1. Numeric.

x2 Predictor 2. Numeric.

x3 Predictor 3. Numeric.

gp Group variable: "gp1" or "gp2". String.

Examples

```
data(dvs_ivs)
library(lavaan)
mod <-
"
y1 ~ x1 + x2 + x3
y2 ~ x1 + x3
y3 ~ y2 + x2
"
fit <- sem(mod, dvs_ivs)
parameterEstimates(fit)
fit_gp <- sem(mod, dvs_ivs, group = "gp")
parameterEstimates(fit_gp)
```

---

filter_by *Filter a Parameter Estimates Table*

Description

Filter parameter estimates table and similar tables in lavaan by common fields such as op (operator).

Usage

```
filter_by(object, op = NULL, lhs = NULL, rhs = NULL, group = NULL, fit = NULL)
```

Arguments

object	The output of lavaan::parameterEstimates(), lavaan::standardizedSolution(), or a lavaan.data.frame object. May also work on an est_table-class object returned by functions like group_by_dvs() but there is no guarantee.
op	A character vector of the operators (op) for filtering. Common operators are "~", "~~", "=~", ":=", and "~1".
lhs	A character vector of names in the lhs column.
rhs	A character vector of names in the rhs column.
group	A vector of either the group numbers in the group column of the labels of the groups. If labels are supplied, the original fit object must be supplied for extracting the group labels.
fit	The original fit object. Usd when group is a vector of the group labels.

Details

This functions accepts the output of lavaan::parameterEstimates() and lavaan::standardizedSolution() and filter the rows by commonly used field.

Value

The filtered version of the input object.

Author(s)

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

Examples

```
library(lavaan)
set.seed(5478374)
n <- 50
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
dat <- data.frame(x = x, y = y, m = m)
model <-
'
m ~ a*x
y ~ b*m
ab := a*b
'
fit <- sem(model, data = dat, fixed.x = FALSE)

model_gp <-
'
m ~ c(a1, a2)*x
y ~ c(b1, b2)*m
a1b1 := a1*b1
```

```
a2b2 := a2*b2
'
dat$gp <- sample(c("gp1", "gp2"), n, replace = TRUE)
fit_gp <- sem(model_gp, dat, group = "gp", warn = FALSE)

est <- parameterEstimates(fit)
est_gp <- parameterEstimates(fit_gp)

filter_by(est, op = "~")

filter_by(est, op = "~", lhs = "y")

filter_by(est, rhs = c("m", "x"), op = "~")

filter_by(est_gp, group = 2)

# If the fit object is supplied, can filter
# by group label
filter_by(est_gp, group = "gp2", fit = fit_gp)
filter_by(est_gp, group = "gp2", fit = fit_gp, op = "~")

# Select user-defined parameters
filter_by(est_gp, op = ":=")

# Can be used with some other functions in semhelpinghands
# Piping can also be used
est_gp |> filter_by(op = "~", group = "gp2", fit = fit_gp) |>
          add_sig()
```

---

group_by_groups *Group Estimates By Groups*

Description

Groups parameter estimates or other information such as *p*-values into a table with groups as columns and parameters as rows.

Usage

```
group_by_groups(
  object,
  ...,
  col_names = "est",
  group_first = TRUE,
  group_labels = NULL,
  fit = NULL,
  use_standardizedSolution = FALSE
)
```

Arguments

object	A lavaan object or the output of lavaan::parameterEstimates() or lavaan::standardizedSolution
...	Optional arguments to be passed to lavaan::parameterEstimates(). Ignored if object is an output of lavaan::parameterEstimates() or lavaan::standardizedSolution().
col_names	A vector of the column names in the parameter estimate tables to be included. Default is "est".
group_first	If TRUE, the columns will be grouped by groups first and then by columns in the parameter estimates tables. Default is TRUE.
group_labels	A character vector of group labels. Will be assigned to group id = 1, 2, 3, etc. If not provided. will try to be retrieved from object if it is a lavaan::lavaan object.
fit	Optional. A lavaan::lavaan object. If object is a parameter estimates table and group_labels is NULL, it will try to retrieve the group labels from fit is supplied.
use_standardizedSolution	If TRUE and object is not an estimates table, then lavaan::standardizedSolution() will be used to generate the table. If FALSE, the default, then lavaan::parameterEstimates() will be used if necessary.

Value

A data-frame-like object of the class est_table.

Author(s)

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

Examples

```
library(lavaan)
set.seed(5478374)
n <- 100
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
city <- sample(c("City Alpha", "City Beta"), 100,
               replace = TRUE)
dat <- data.frame(x = x, y = y, m = m, city = city)
model <-
'
m ~ c(a1, a2)*x
y ~ c(b1, b2)*m
a1b1 := a1*b1
a2b2 := a2*b2
'
fit <- sem(model, data = dat, fixed.x = FALSE,
           group = "city")
(est <- parameterEstimates(fit))
```

```
# Group them by groups
group_by_groups(fit)

# Can also work on a parameter estimates table
# To have group labels, need to supply the fit object
group_by_groups(est, fit = fit)

# Can be used with some other functions in semhelpinghands
# when used on a parameter estimates table
group_by_groups(filter_by(est, op = "~"), fit = fit)

# Also support piping
est |> filter_by(op = "~") |>
       group_by_groups(fit = fit)
```

`group_by_models` *Group Estimates By Models*

Description

Groups parameter estimates or other information such as *p*-values into a table with models as columns.

Usage

```
group_by_models(
  object_list,
  ...,
  col_names = "est",
  group_first = FALSE,
  model_first = TRUE,
  use_standardizedSolution = FALSE
)
```

Arguments

`object_list`	A named list of lavaan objects, a named list of the output of `lavaan::parameterEstimates()`, or a named list of the output of `lavaan::standardizedSolution()`.
`...`	Optional arguments to be passed to `lavaan::parameterEstimates()`. Ignored if the elements in object_list are the results of `lavaan::parameterEstimates()` or `lavaan::standardizedSolution()`.
`col_names`	A vector of the column names in the parameter estimate tables to be included. Default is `"est"`.
`group_first`	If `TRUE`, the rows will be grouped by groups first and then by parameters. Ignored if the model has only one group. Default is `FALSE`.
`model_first`	If `TRUE`, the columns will be grouped by models first and then by columns in the parameter estimates tables. Default is `TRUE`.

use_standardizedSolution
: If TRUE and object_list is not a list of estimates tables, then lavaan::standardizedSolution() will be used to generate the table. If FALSE, the default, then lavaan::parameterEstimates() will be used if necessary.

Value

A data-frame-like object of the class est_table.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448 Inspired by the proposal Rönkkö posted in a GitHub https://github.com/simsem/semTools/issues/24#issue-235172313 for semTools. I want something simple for a quick overview and so I wrote this function.

Examples

```
library(lavaan)
set.seed(5478374)
n <- 50
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
dat <- data.frame(x = x, y = y, m = m)
model1 <-
'
m ~ a*x
y ~ b*m
ab := a*b
'
fit1 <- sem(model1, data = dat, fixed.x = FALSE)
model2 <-
'
m ~ a*x
y ~ b*m + x
ab := a*b
'
fit2 <- sem(model2, data = dat, fixed.x = FALSE)
parameterEstimates(fit1)
parameterEstimates(fit2)
group_by_models(list(no_direct = fit1,
                     direct = fit2),
                col_names = c("est", "pvalue"))
# Can also be used with some other functions in
# semhelpinghands
group_by_models(list(no_direct = fit1,
                     direct = fit2),
                col_names = c("est", "pvalue")) |>
                filter_by(op = "~")
```

group_estimates *Group Estimates By Dependent or Independent Variables*

Description

Groups parameter estimates or other information such as p-values into a table with dependent variables as columns and independent variables as rows, or a transpose of this table.

Usage

```
group_by_dvs(
  object,
  ...,
  col_name = "est",
  add_prefix = TRUE,
  group_first = FALSE,
  use_standardizedSolution = FALSE
)

group_by_ivs(
  object,
  ...,
  col_name = "est",
  add_prefix = TRUE,
  group_first = FALSE,
  use_standardizedSolution = FALSE
)
```

Arguments

object	A lavaan object or the output of lavaan::parameterEstimates() or lavaan::standardizedSolution
...	Optional arguments to be passed to lavaan::parameterEstimates(). Ignored if object is an output of lavaan::parameterEstimates() or lavaan::standardizedSolution().
col_name	The column name of information to be grouped. Default is "est". It accepts only one name.
add_prefix	If TRUE, the default, col_name will be added as prefix to the column names of the output.
group_first	If TRUE, the rows will be grouped by groups first and then by independent variables Ignored if the model has only one group. Default is FALSE.
use_standardizedSolution	If TRUE and object is not an estimates table, then lavaan::standardizedSolution() will be used to generate the table. If FALSE, the default, then lavaan::parameterEstimates() will be used if necessary.

Details

It gets a lavaan object or the output of `lavaan::parameterEstimates()` or `lavaan::standardizedSolution()` and group selected columns by "dependent" variables `group_by_dvs()` or by "independent" variables `group_by_ivs()`.

"Dependent" variables are defined as variables on the left hand side of the operator ~.

"Independent" variables are defined as variables on the right hand side of the operator ~.

Note that a variable can both be a "dependent" variable and an "independent" variable in a model.

Value

A data-frame-like object of the class `est_table`.

Author(s)

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

Examples

```
library(lavaan)
set.seed(5478374)
n <- 50
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
dat <- data.frame(x = x, y = y, m = m)
model <-
'
m ~ a*x
y ~ b*m
ab := a*b
'

fit <- sem(model, data = dat, fixed.x = FALSE)
parameterEstimates(fit)

# Group by DVs
group_by_dvs(fit)

# Group by IVs
group_by_ivs(fit)
```

print.est_table *Print an 'est_table' Object*

Description

Print method for an 'est_table' object

Usage

```
## S3 method for class 'est_table'
print(x, ..., nd = 3, empty_cells = "--", group_first = FALSE)
```

Arguments

x	Object of the class est_table.
...	Optional arguments to be passed to print() methods.
nd	The number of digits to be printed. Default is 3. (Scientific notation will never be used.)
empty_cells	String to be printed for empty cells or cells with no values. Default is "--".
group_first	Not used.

Value

x is returned invisibly. Called for its side effect.

Author(s)

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

record_history *Record the Minimization History*

Description

Record the minimization history when a model is fitted by lavaan::lavaan() or its wrappers (e.g., lavaan::sem() or lavaan::cfa()).

Usage

```
record_history(object)

## S3 method for class 'fit_history'
plot(x, params, last_n = -1, orientation = c("horizontal", "vertical"), ...)

## S3 method for class 'fit_history'
print(x, n_iterations = 10, digits = 3, ...)
```

Arguments

object	A lavaan object.
x	A fit_history class object, the output of record_history().
params	A character vector of the names of parameters to be plotted. Must be the names of one or more columns in x.
last_n	The lass n iterations to be plotted. Default is -1, plotting all iterations.
orientation	The orientation of the plot. Either "horizontal" (the default) or "vertical".
...	Optional arguments. To be passed to the print method of data frame.
n_iterations	The number of iterations to print. Default is 10, printing the first 10 iterations (or all iterations, if the number of iterations is less than 10).
digits	The number of digits to be displayed. Default is 3.

Details

It records the minimization history when a model is fitted by lavaan::lavaan() or its wrappers (e.g., lavaan::sem() or lavaan::cfa()). The recorded history can then be plotted or displayed, for visualizing how the estimates of free parameters is found.

It will refit the model by the update method of lavaan::lavaan, setting se = "none" and test = "standard" because they have no impact on the minimization process.

This and related functions are adapted from the package semunpack. The version in this package will be revised to be an advanced version intended for diagnostic purpose in real studies.

Value

A fit_history-class object with a plot method (plot.fit_history()).

Functions

- plot(fit_history): The plot method for the output of record_history().
- print(fit_history): The print method for the output of record_history().

Author(s)

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

Examples

```
# Adapted from the example for CFA in lavaan::cfa().
# Using only two of the factors
library(lavaan)
HS.model <-
'
visual  =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
'
fit <- cfa(HS.model, data = HolzingerSwineford1939)
```

```
# Refit the model with the history recorded
fit_h <- record_history(fit)
fit_h

# Plot the history for selected parameters
plot(fit_h, params = c("visual=~x2", "visual=~x3",
                       "visual~~textual"),
            last_n = 10)
plot(fit_h, params = c("visual=~x2", "visual=~x3",
                       "visual~~textual"),
            last_n = 10,
            orientation = "vertical")
```

`show_more_options` *Show More Major Options in an Output of 'lavaan'*

Description

Display the values of more major options in a model fitted by `lavaan::lavaan()` or its wrappers (e.g., lavaan::sem or `lavaan::cfa()`).

Usage

```
show_more_options(fit)

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

Arguments

`fit`	An output of `lavaan::lavaan()` or its wrappers (e.g., `lavaan::cfa()` and `lavaan::sem()`)
`x`	The output of `show_more_options()`.
`...`	Additional arguments. Ignored.

Details

It extracts the values of major options in the output of `lavaan::lavaan()` or its wrappers (e.g., lavaan::sem or `lavaan::cfa()`. Most of the values are also reported in the summary of a lavaan object. This function is used to show the values in one single table for a quick overview.

It checks the actual values, not the call used. This is useful for understanding how a prepackaged estimator such as `ML`, `MLM`, and `MLR` set other options. It supports the following options:

- Estimator (`estimator`)
- Standard error (`se`)

- Model chi-square test(s) (`test`)
- Missing data method (`missing`)
- Information matrix used for computing standard errors (`information`)
- Information matrix used for computing model chi-square (`information`)
- Whether the mean structure is included.

It is named `show_more_options()` to differentiate it from `show_options` in the `semunpack` package, which is intended for new users of lavaan. The code is adapted from `show_options` with more advanced options added.

Value

A `show_more_options`-class object with a print method that formats the output.

Methods (by generic)

- `print(show_more_options)`: The print method of the output of `show_more_options()`.

Author(s)

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

Examples

```
library(lavaan)

# From the help page of lavaan::cfa().

HS.model <- '
visual  =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed   =~ x7 + x8 + x9
'
fit <- cfa(HS.model, data = HolzingerSwineford1939)

tmp <- show_more_options(fit)
tmp

fit <- cfa(HS.model, data = HolzingerSwineford1939, estimator = "MLR")
show_more_options(fit)
fit <- cfa(HS.model, data = HolzingerSwineford1939, estimator = "MLM")
show_more_options(fit)
```

`simple_mediation` *Sample Dataset: Simple Mediation*

Description

A simple mediation model.

Usage

```
simple_mediation
```

Format

A data frame with 100 rows and 5 variables:

x Predictor. Numeric.

m Mediator. Numeric.

y Outcome variable. Numeric.

city Group variable: "City A" or "City B". String.

Examples

```
library(lavaan)
data(simple_mediation)
mod <-
"
m ~ a * x
y ~ b * m + x
ab := a * b
"
fit <- sem(mod, simple_mediation, fixed.x = FALSE)
parameterEstimates(fit)
mod_gp <-
"
m ~ c(a1, a2) * x
y ~ c(b1, b2) * m + x
a1b1 := a1 * b1
a2b2 := a2 * b2
ab_diff := a1b1 - a2b2
"
fit_gp <- sem(mod_gp, simple_mediation, fixed.x = FALSE, group = "city")
parameterEstimates(fit_gp)
```

sort_by *Sort a Parameter Estimates Table*

Description

Sort a parameter estimates table or a similar table in lavaan by common fields such as op (operator) and lhs (left- hand side).

Usage

```
sort_by(
  object,
  by = c("op", "lhs", "rhs"),
  op_priority = c("=~", "~", "~~", ":=", "~1", "|", "~*~"),
  number_rows = TRUE
)
```

Arguments

object	The output of lavaan::parameterEstimates(), lavaan::standardizedSolution(), or a lavaan.data.frame object. May also work on an est_table-class object returned by functions like group_by_dvs() but there is no guarantee.
by	A character vector of the columns for filtering. Default is c("op", "lhs", "rhs").
op_priority	How rows are sorted by op. Default is c("=~", "~", "~~", ":=", "~1", "|", "~*~"). Can set only a few of the operators, e.g., c("~", "~~"). Other operators will be placed to the end with orders not changed.
number_rows	Whether the row names will be set to row numbers after sorting *if* the row names of object is equal to row numbers. Default is TRUE.

Details

This functions accepts the output of lavaan::parameterEstimates() and lavaan::standardizedSolution() and filter the rows by commonly used field.

Value

The sorted version of the input object.

Author(s)

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

Examples

```
library(lavaan)
set.seed(5478374)
n <- 50
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
dat <- data.frame(x = x, y = y, m = m)
model1 <-
'
m ~ a*x
y ~ b*m
ab := a*b
'
fit1 <- sem(model1, data = dat, fixed.x = FALSE)
model2 <-
'
m ~ a*x
y ~ b*m + x
ab := a*b
'
fit2 <- sem(model2, data = dat, fixed.x = FALSE)
parameterEstimates(fit1)
parameterEstimates(fit2)
out <- group_by_models(list(no_direct = fit1,
                            direct = fit2),
                       col_names = c("est", "pvalue"))
out
sort_by(out)
sort_by(out, op_priority = c("~", ":="))
sort_by(out, by = c("op", "rhs"))
```

---

standardizedSolution_boot_ci

Bootstrap CIs for Standardized Solution

Description

It receives a lavaan::lavaan object fitted with bootstrapping standard errors requested and forms the confidence intervals for the standardized solution.

It works by calling `lavaan::standardizedSolution()` with the bootstrap estimates of free parameters in each bootstrap sample to compute the standardized estimates in each sample.

A more reliable way is to use function like `lavaan::bootstrapLavaan()`. Nevertheless, this simple function is good enough for some simple scenarios, and does not require repeating the bootstrapping step.

Usage

```
standardizedSolution_boot_ci(
  object,
  level = 0.95,
  type = "std.all",
  save_boot_est_std = TRUE,
  force_run = FALSE,
  boot_delta_ratio = FALSE,
  ...
)
```

Arguments

object	A lavaan object, fitted with 'se = "boot"'.
level	The level of confidence of the confidence intervals. Default is .95.
type	The type of standard estimates. The same argument of `lavaan::standardizedSolution()`, and support all values supported by `lavaan::standardizedSolution()`. Default is `"std.all"`.
save_boot_est_std	Whether the bootstrap estimates of the standardized solution are saved. If saved, they will be stored in the attribute `boot_est_std`. Default is `TRUE`.
force_run	If `TRUE`, will skip checks and run models without checking the estimates. For internal use. Default is `FALSE`.
boot_delta_ratio	The ratio of (a) the distance of the bootstrap confidence limit from the point estimate to (b) the distance of the delta-method limit from the point estimate.
...	Other arguments to be passed to `lavaan::standardizedSolution()`.

Value

The output of `lavaan::standardizedSolution()`, with bootstrap confidence intervals appended to the right.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448. Originally proposed in an issue at GitHub https://github.com/simsem/semTools/issues/101#issue-1021974657, inspired by a discussion at the Google group for lavaan https://groups.google.com/g/lavaan/c/qQBXSz5cd0o/m/R8YT5HxNAgAJ. `boot::boot.ci()` is used to form the percentile confidence intervals in this version.

See Also

`lavaan::standardizedSolution()`

Examples

```
library(lavaan)
set.seed(5478374)
n <- 50
x <- runif(n) - .5
m <- .40 * x + rnorm(n, 0, sqrt(1 - .40))
y <- .30 * m + rnorm(n, 0, sqrt(1 - .30))
dat <- data.frame(x = x, y = y, m = m)
model <-
'
m ~ a*x
y ~ b*m
ab := a*b
'

# Should set bootstrap to at least 2000 in real studies
fit <- sem(model, data = dat, fixed.x = FALSE,
           se = "boot",
           bootstrap = 100)
summary(fit)

standardizedSolution_boot_ci(fit)
```

Index