Package 'tidySEM'

February 19, 2021

Type Package

Date 2021-02-19

Title Tidy Structural Equation Modeling

Version 0.1.6

Description A tidy workflow for generating, estimating, reporting, and plotting structural equation models using 'lavaan' or 'Mplus'. Throughout this workflow, elements of syntax, results, and graphs are represented as 'tidy' data, making them easy to customize.

License GPL (>= 3)

URL https://cjvanlissa.github.io/tidySEM/

BugReports https://github.com/cjvanlissa/tidySEM/issues

Depends R (>= 3.2.0), stats, utils

Imports ggplot2, lavaan, MplusAutomation, igraph, psych

Suggests testthat, knitr, rmarkdown, dplyr, stringr, covr

VignetteBuilder knitr

Encoding UTF-8

LazyData yes

RoxygenNote 7.1.1

NeedsCompilation no

Author Caspar J. van Lissa [aut, cre] (<https://orcid.org/0000-0002-0808-5024>)

Maintainer Caspar J. van Lissa <c.j.vanlissa@uu.nl>

Repository CRAN

Date/Publication 2021-02-19 22:20:02 UTC

R topics documented:

`add_paths` *Add paths to an object of class 'tidy_sem'*

Description

Add paths to an object of class `tidy_sem`, or replace existing paths. The paths must be specified as `model.syntax`, and separated by commas.

Usage

```
add_paths(model, ...)
```

Arguments

`model`	An object of class `tidy_sem`.
`...`	Paths to add or substitute, specified in `lavaan{model.syntax}`, and separated by commas.

Details

Currently, only the lavaan{lavaan} commands ~, ~~, =~, and ~1 are parsed.

This function relies on lavaan model.syntax to convert syntax strings to lavaan parameter tables. By default, is uses the arguments int.ov.free = TRUE, int.lv.free = FALSE, auto.fix.first = TRUE, auto.fix.single = TRUE, auto.var = TRUE, auto.cov.lv.x = TRUE, auto.efa = TRUE, auto.th = TRUE, auto.delta = TRUE, auto.cov.y = TRUE, akin to sem and cfa.

Value

An object of class tidy_sem.

See Also

model.syntax

Examples

```
library(lavaan)
df <- iris[, 1:4]
names(df) <- paste0("x_", 1:4)
model <- tidy_sem(df)
model <- measurement(model)
model <- add_paths(model, x =~ a*x_1 + b*x_2 + a*x_3 + b*x_4)
res <- estimate_lavaan(model)
summary(res)
```

---

as_lavaan *Convert tidy_sem to 'lavaan' syntax*

Description

Final stage in the 'tidySEM' workflow for syntax generation: Convert the tidy_sem object to lavaan syntax in tabular format (see model.syntax).

Usage

```
as_lavaan(x, ...)
```

Arguments

x	An object of class tidy_sem
...	Additional parameters to be passed to and from functions.

Value

Character vector.

Examples

```
mod <- list(syntax = structure(list(lhs = "x", op = "~", rhs = "y",
                                    free = TRUE, value = "", label = "",
                                    category = "", aspect = ""),
            class = "data.frame", row.names = c(NA, -1L)))
class(mod) <- "tidy_sem"
as_lavaan(mod)
```

`as_mplus` *Convert tidy_sem to 'Mplus' syntax*

Description

Final stage in the 'tidySEM' workflow for syntax generation: Convert the `tidy_sem` object to 'Mplus' syntax.

Usage

```
as_mplus(x, ...)
```

Arguments

`x`	An object of class `tidy_sem`.
`...`	Additional parameters to be passed to and from functions.

Value

Character vector.

Examples

```
mod <- list(syntax = structure(list(lhs = "x", op = "~", rhs = "y",
                                    free = TRUE, value = "", label = "",
                                    category = "", aspect = ""),
            class = "data.frame", row.names = c(NA, -1L)))
class(mod) <- "tidy_sem"
as_mplus(mod)
```

conf_int *Format confidence intervals*

Description

Creates 'APA'-formatted confidence intervals, either from an object for which a method exists, or from the arguments lb and ub. When argument x is a numeric vector, it is also possible to construct a confidence interval using the standard error (se) and a percentile interval (ci).

Usage

```
conf_int(x, digits = 2, se = NULL, lb = NULL, ub = NULL, ci = 95)
```

Arguments

x	Optional. An object for which a method exists.
digits	Integer. The number of digits to round the confidence boundaries to.
se	Optional, numeric. Standard error of the parameters.
lb	Optional, numeric. Lower boundary of confidence intervals.
ub	Optional, numeric. Upper boundary of confidence intervals.
ci	Optional, numeric. What percentage CI to use (only used when computing CI from a numeric vector x, and the standard error se, based on a normal distribution).

Value

A character vector of formatted confidence intervals.

Author(s)

Caspar J. van Lissa

See Also

table_results est_sig

Other Reporting tools: est_sig(), table_results()

Examples

```
conf_int(x = c(1.325, 2.432), se = c(.05336, .00325))
```

cors *Generate syntax for correlations*

Description

Generate syntax for correlations between variables.

Usage

```
cors(x, ...)
```

Arguments

x	Object for which a method exists. If x is an object of class tidy_sem, then correlations between all observed and latent variables in the data dictionary of that object are computed, by default. If x is a character vector, all elements of the vector are used.
...	Optional additional character vectors of variables to be correlated. If x is an object of class tidy_sem, then up to two vectors can be provided. If x is a vector, then one more optional vector can be provided. When no additional vectors of variable names are provided, only the correlations among the elements of x are returned.

Value

An object of class tidy_sem.

Examples

```
dict <- tidy_sem(c("bfi_1", "bfi_2", "bfi_3", "bfi_4", "bfi_5"))
cors(dict, c("bfi_1", "bfi_2"))
```

create_scales *Create scale scores from observed variables*

Description

This function calculates mean or sum scores from a data.frame and a named list describing the items in each scale. It returns the scores, a scale descriptive table, and a scale correlation table. It relies on several functions from the psych package.

Usage

```
create_scales(
  x,
  keys.list,
  missing = TRUE,
  impute = "none",
  omega = NULL,
  digits = 2,
  ...
)

## S3 method for class 'tidy_sem'
create_scales(
  x,
  keys.list,
  missing = TRUE,
  impute = "none",
  omega = NULL,
  digits = 2,
  ...
)
```

Arguments

x	A data.frame containing all variables referenced in the keys.list, or an object of class tidy_sem.
keys.list	A named list, indicating which variables belong to which scale.
missing	Whether to use rows with partially missing values. Default: TRUE.
impute	Method for handling missing values, Default: 'none'. This default method uses all available data to calculate scale scores, which is acceptable for mean scales, but not for sum scales.
omega	Which of McDonald's omega coefficients to report. Default: NULL; valid options include: "omega_h", "omega.lim", "alpha", "omega.tot", "G6".
digits	Number of digits for rounding, Default: 2
...	Additional parameters to pass to and from functions.

Details

For scales with less than 3 items, Cronbach's alpha might not be suitable as an estimate of reliability. For such scales, the Spearman-Brown reliability coefficient for two-item scales is computed, as described in Eisinga, R., Grotenhuis, M. te, & Pelzer, B. (2012). The reliability of a two-item scale: Pearson, Cronbach, or Spearman-Brown? International Journal of Public Health, 58(4), 637–642. <doi:10.1007/s00038-012-0416-3>. These coefficients are marked with "(sb)".

Value

List with elements: $descriptives, $correlations, and $scores.

Examples

```
out <- create_scales(iris, keys.list = list(scalename =
            c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")))
out$descriptives
dict <- tidy_sem(iris, split = "\\.")
create_scales(dict)
```

`descriptives` *Describe a dataset*

Description

Provide descriptive statistics for a dataset.

Usage

```
descriptives(x, ...)
```

Arguments

`x`	An object for which a method exists.
`...`	Additional arguments.

Value

A `data.frame` with descriptive statistics for `x`.

Examples

```
descriptives(iris)
```

`dictionary` *Extract dictionary from tidy_sem*

Description

Provides access to the `dictionary` element of a `tidy_sem` object. This can be used to return or assign to the `dictionary` element.

Usage

```
dictionary(x)

dictionary(x) <- value
```

Arguments

x	Object of class tidy_sem.
value	A valid value for dictionary(x).

Value

data.frame

Examples

```
dict <- tidy_sem(iris, split = "\\.")
dictionary(dict)
```

---

edges *Extract edges from sem_graph*

Description

Provides access to the edges element of a sem_graph object. This can be used to return or assign to the edges element.

Usage

```
edges(x)

edges(x) <- value
```

Arguments

x	Object of class sem_graph.
value	A valid value for edges(x).

Value

data.frame

Examples

```
edg <- data.frame(from = "x", to = "y")
p <- prepare_graph(edges = edg, layout = get_layout("x", "y", rows = 1))
edges(p)
```

edit_graph *Edit graph elements*

Description

Evaluate an R expression within the environment of the elements of a sem_graph object, and return the modified sem_graph.

Usage

```
edit_graph(x, expr, element = "edges", ...)
```

Arguments

x	An object of class sem_graph.
expr	expression to evaluate.
element	Character. The element of the sem_graph to edit, defaults to "edges".
...	Arguments passed on to within.

Value

An object of class sem_graph.

Examples

```
p <- prepare_graph(layout = get_layout("x", rows = 1))
p <- edit_graph(p, {colour = "blue"}, element = "nodes")
plot(p)
```

estimate_lavaan *Estimate tidy_sem using 'lavaan'*

Description

This function is a wrapper for the lavaan estimating functions. By default, the wrapper uses sem, but users can also specify lavaan, cfa, or growth.

Usage

```
estimate_lavaan(x, func = "sem", ...)
```

Arguments

x	An object of class tidy_sem.
func	The lavaan modeling function to invoke, Default: 'sem'.
...	Additional parameters passed to the estimating function.

Value

An object of class lavaan.

Examples

```
library(lavaan)
model <- tidy_sem(iris, "\\.")
model <- measurement(model)
res <- estimate_lavaan(model)
summary(res)
```

`estimate_mplus` *Estimate tidy_sem using 'Mplus'*

Description

This function is a wrapper for the functions `mplusObject` and `mplusModeler`. Using this function requires 'Mplus' to be installed.

Usage

```
estimate_mplus(x, ...)
```

Arguments

`x`	An object of class `tidy_sem`.
`...`	Additional parameters passed to `mplusObject` and `mplusModeler`. These arguments are matched to the correct function by name. The arguments `rdata`, and `MODEL` cannot be edited, as they are determined from the `tidy_sem` object.

Details

The arguments `dataout`, `modelout`, and `run` are optional. If these are not specified, the model will be run in `tempdir`.

Value

An object of class `mplusObject`.

Examples

```
library(MplusAutomation)
model <- tidy_sem(iris, "\\.")
model <- measurement(model)
if(mplusAvailable() == 0){
  estimate_mplus(model)
}
```

`est_sig` *Add significance asterisks to object*

Description

Takes an object, and adds significance asterisks.

Usage

```
est_sig(x, digits = 2, sig = NULL)
```

Arguments

`x`	An object for which a method exists. This will be treated as numeric by the default method.
`digits`	Integer. The number of digits to round the estimate column to.
`sig`	Optional, a vector of p-values for the default method.

Value

A character vector of formatted estimates.

Author(s)

Caspar J. van Lissa

See Also

table_results

Other Reporting tools: `conf_int()`, `table_results()`

Examples

```
est_sig(c(.222, .3333), sig = c(.054, .045))
```

get_data *Extract data from tidy_sem*

Description

Provides access to the data element of a tidy_sem object. This can be used to return or assign to the data element.

Usage

```
get_data(x)

get_data(x) <- value
```

Arguments

x	Object of class tidy_sem.
value	A valid value for get_data(x).

Value

data.frame

Examples

```
dict <- tidy_sem(iris, split = "\\.")
get_data(dict)
```

get_edges *Extract edges from a SEM model object*

Description

Attempts to extract edges from a SEM model object, where edges are defined as regression paths and covariances between variables (nodes).

Usage

```
get_edges(x, label = "est_sig", ...)
```

Arguments

x	A model object of class mplusObject or lavaan.
label	Either a character, indicating which column to use for edge labels, or an expression. See Details. Defaults to "est_sig", which labels edges with the estimated value with significance asterisks, as obtained from table_results. See Details and examples for more information.
...	Additional parameters passed to table_results. For example, users can pass the digits argument to control the number of digits in the edge label, or pass the columns argument to retain auxiliary columns in the tidy_edges data.frame for further processing (see Examples).

Details

The function get_edges identifies all regression paths, latent variable definitions, and covariances in the model as edges. The output of table_results for those paths is used to label the edges.

Custom labels

One way to create custom edge labels is by passing an expression to label. When an expression is passed to label, it is evaluated in the context of a data.frame containing the results of a call to table_results on the x argument.

Another way to create custom labels is by requesting auxiliary variables using the columns argument (which is passed to table_results), and then using these columns to construct a new label. See examples.

Value

An object of class 'tidy_edges'

Examples

```
# Standard use
library(lavaan)
res <- sem("dist ~ speed", cars)
get_edges(res)

# Pass an expression to the 'label' argument for custom labels
get_edges(res, label = paste(est_sig, confint))

# Pass the argument 'columns' to table_results through '...' to retain
# auxiliary columns for further processing
edg <- get_edges(res, columns = c("est_sig", "confint"))
edg
edg <- within(edg, {label <- paste(est_sig, confint)})
edg
```

`get_layout.lavaan` *Generate graph layout*

Description

Generate a tidy_layout for a SEM graph.

Usage

```
## S3 method for class 'lavaan'
get_layout(x, ..., layout_algorithm = "layout_as_tree")

get_layout(x, ...)

## Default S3 method:
get_layout(x, ..., rows = NULL)
```

Arguments

`x`	An object for which a method exists; currently, methods exist for `character`, `lavaan`, and `mplus.model` objects.
`...`	Character arguments corresponding to layout elements. Use node names, empty strings (""), or NA values.
`layout_algorithm`	Optional argument for fit model objects. Character string, indicating which `igraph` layout algorithm to apply to position the nodes. Defaults to `"layout_as_tree"`; see details for more options.
`rows`	Numeric, indicating the number of rows of the graph.

Details

There are three ways to generate a layout:

1. Specify the layout in the call to `get_layout()` by providing node names and the number of rows to create a layout matrix. Empty strings (`""`) or `NA` can be used for empty cells. See Example 1.
2. Call `get_layout()` on a model object or `tidy_results` object. It will use the function `layout_as_tree`, or any other layout function from the `igraph` package, to generate a rudimentary layout. See Example 2.
3. Instead of using `get_layout()`, just use a `matrix` or `data.frame` with your layout. For example, specify the layout in a spreadsheet program, and load it into R (see Example 3). Or, copy the layout to the clipboard from your spreadsheet program, and load it from the clipboard (see Example 4)

The layout algorithms imported from `igraph` are: `c("layout_as_star","layout_as_tree","layout_in_circle","layo`
These can be used by specifying the optional argument `layout_algorithm = ""`.

Value

Object of class 'tidy_layout'

Examples

```
# Example 1
get_layout("c", NA,  "d",
           NA,  "e", NA, rows = 2)

# Example 2
library(lavaan)
fit <- cfa(' visual  =~ x1 + x2 + x3 ',
           data = HolzingerSwineford1939[1:50, ])
get_layout(fit)

## Not run:
# Example 3
# Here, we first write the layout to .csv, but you could create it in a
# spreadsheet program, and save the spreadsheet to .csv:
write.csv(matrix(c("c", "",  "d", "",  "e", ""), nrow = 2, byrow = TRUE),
          file = file.path(tempdir(), "example3.csv"), row.names = FALSE)
# Now, we load the .csv:
read.csv(file.path(tempdir(), "example3.csv"))

# Example 4
# For this example, make your layout in a spreadsheet program, select it, and
# copy to clipboard. Reading from the clipboard works differently in Windows
# and Mac. For this example, I used Microsoft Excel.
# On Windows, run:
read.table("clipboard", sep = "\t")
# On Mac, run:
read.table(pipe("pbpaste"), sep="\t")

## End(Not run)
```

`get_nodes` *Extract nodes from a SEM model object*

Description

Attempts to extract nodes from a SEM model object, where nodes are defined as observed or latent variables.

Usage

```
get_nodes(x, label = paste(name, est_sig, sep = "\n"), ...)
```

Arguments

x	A model object of class mplusObject or lavaan.
label	Either a character, indicating which column to use for node labels, or an expression. See Details. Defaults to paste(name,est_sig,sep = "\n", which gives the node name followed by the estimated value with significance asterisks.
...	Additional parameters passed to table_results. For example, users can pass the digits argument to control the number of digits in the node label, or pass the columns argument to retain auxiliary columns in the tidy_nodes data.frame for further processing (see Examples).

Details

The function get_nodes identifies all dependent and independent variables in the model as nodes. If a mean structure / intercepts are included in the model, the output of table_results for those means / intercepts is used to label the nodes.

Custom labels

One way to create custom node labels is by passing an expression to label, as in the default value of the argument. When an expression is passed to label, it is evaluated in the context of a data.frame containing the results of a call to table_results on the x argument, with an additional column labeled name, which contains the node names.

Another way to create custom labels is by requesting auxiliary variables using the columns argument (which is passed to table_results), and then using these columns to construct a new label. See examples.

Value

An object of class 'tidy_nodes'

Examples

```
# Standard use extracts node names and shape
# (rect for observed, oval for latent)
library(lavaan)
res <- sem("dist ~ speed", cars)
get_nodes(res)

# To label nodes with mean values, include meanstructure in the model
# Note that it is possible to pass the argument 'digits' to table_results
# through '...'
res <- sem("dist ~ speed", cars, meanstructure = TRUE)
get_nodes(res, digits = 3)

# Pass an expression to the 'label' argument for custom labels
get_nodes(res, label = paste0(name, " ", est_sig, "\n", confint))

# Pass the argument 'columns' to table_results through '...' to retain
# auxiliary columns for further processing
nod <- get_nodes(res, columns = c("est_sig", "confint"))
nod
```

```
nod <- within(nod, {label <- paste0(name, " ", est_sig, "\n", confint)})
nod
```

---

graph_sem *Render a graph*

Description

Render a graph based on a layout, and either nodes and edges, or a model object.

Usage

```
graph_sem(...)

## Default S3 method:
graph_sem(
  edges = NULL,
  layout = NULL,
  nodes = NULL,
  rect_width = 1.2,
  rect_height = 0.8,
  ellipses_width = 1,
  ellipses_height = 1,
  variance_diameter = 0.8,
  spacing_x = 2,
  spacing_y = 2,
  text_size = 4,
  curvature = 60,
  angle = NULL,
  fix_coord = FALSE,
  ...
)

## S3 method for class 'lavaan'
graph_sem(
  model,
  label = "est_sig",
  edges = get_edges(x = model, label = label),
  layout = get_layout(x = model),
  nodes = get_nodes(x = model, label = label),
  ...
)

## S3 method for class 'mplus.model'
graph_sem(
  model,
  label = "est_sig",
```

```
  edges = get_edges(x = model, label = label),
  layout = get_layout(x = model),
  nodes = get_nodes(x = model, label = label),
  ...
)
```

Arguments

...	Additional arguments passed to and from functions.
edges	Object of class 'tidy_edges', or a data.frame with (at least) the columns c("from","to"), and optionally, c("arrow","label","connect_from","connect_to","curvature").
layout	A matrix (or data.frame) that describes the layout; see get_layout.
nodes	Optional, object of class 'tidy_nodes', created with the get_nodes function, or a data.frame with (at least) the column c("name"), and optionally, c("shape","label"). If set to NULL (the default), nodes are inferred from the layout and edges arguments.
rect_width	Width of rectangles (used to display observed variables), Default: 1.2
rect_height	Height of rectangles (used to display observed variables), Default: 0.8
ellipses_width	Width of ellipses (used to display latent variables), Default: 1
ellipses_height	Height of ellipses (used to display latent variables), Default: 1
variance_diameter	Diameter of variance circles, Default: .8
spacing_x	Spacing between columns of the graph, Default: 1
spacing_y	Spacing between rows of the graph, Default: 1
text_size	Point size of text, Default: 4
curvature	Curvature of curved edges. The curve is a circle segment originating in a point that forms a triangle with the two connected points, with angles at the two connected points equal to curvature. To flip a curved edge, use a negative value for curvature. Default: 60
angle	Angle used to connect nodes by the top and bottom. Defaults to NULL, which means Euclidean distance is used to determine the shortest distance between node sides. A numeric value between 0-180 can be provided, where 0 means that only nodes with the same x-coordinates are connected top-to-bottom, and 180 means that all nodes are connected top-to-bottom.
fix_coord	Whether or not to fix the aspect ratio of the graph. Does not work with multigroup or multilevel models. Default: FALSE.
model	Instead of the edges argument, it is also possible to use the model argument and pass an object for which a method exists (e.g., mplus.model or lavaan).
label	Character, indicating which column to use for node labels. Nodes are labeled with mean values of the observed/latent variables they represent. Defaults to 'est_sig', which consists of the estimate value with significance asterisks.

Details

The default interface simply Runs the functions prepare_graph and plot. The alternative interface first runs get_nodes and get_edges on the model argument.

Value

Object of class 'sem_graph'

Examples

```
library(lavaan)
res <- sem("dist ~ speed", cars)
graph_sem(res)
```

---

measurement *Generate syntax for a measurement model*

Description

Generate syntax for a measurement model for latent variables. This function relies on add_paths to generate syntax.

Usage

```
measurement(x, ...)
```

Arguments

x	An object for which a method exists, including tidy_sem (generated using dictionary, or data.frame (for which dictionary will be run first).
...	Additional parameters passed to add_paths.

Value

An object of class tidy_sem.

Examples

```
dict <- tidy_sem(c("bfi_1", "bfi_2", "bfi_3", "bfi_4", "bfi_5"))
measurement(dict)
```

mplus_expand_names *Expand abbreviated Mplus variable names*

Description

Expand the Mplus syntax for abbreviating lists of variable names.

Usage

```
mplus_expand_names(x)
```

Arguments

x	Atomic character string containing the variable names section of an Mplus syntax file.

Value

Character vector of names.

Examples

```
mplus_expand_names("test1-test12")
```

nodes *Extract nodes from sem_graph*

Description

Provides access to the nodes element of a sem_graph object. This can be used to return or assign to the nodes element.

Usage

```
nodes(x)

nodes(x) <- value
```

Arguments

x	Object of class sem_graph.
value	A valid value for nodes(x).

Value

data.frame

Examples

```
edg <- data.frame(from = "x", to = "y")
p <- prepare_graph(edges = edg, layout = get_layout("x", "y", rows = 1))
nodes(p)
```

---

prepare_graph *Prepare graph data*

Description

Prepare an object of class sem_graph, containing data objects that can be rendered into a SEM graph. Using this function allows users to manually change the default graph specification before plotting it. Input consists of (at least) a layout, and either nodes and edges, or a model object.

Usage

```
prepare_graph(...)

## Default S3 method:
prepare_graph(
  edges = NULL,
  layout = NULL,
  nodes = NULL,
  rect_width = 1.2,
  rect_height = 0.8,
  ellipses_width = 1,
  ellipses_height = 1,
  variance_diameter = 0.8,
  spacing_x = 2,
  spacing_y = 2,
  text_size = 4,
  curvature = 60,
  angle = NULL,
  fix_coord = FALSE,
  ...
)

## S3 method for class 'lavaan'
prepare_graph(
  model,
  label = "est_sig",
  edges = get_edges(x = model, label = label),
  layout = get_layout(x = model),
  nodes = get_nodes(x = model, label = label),
  ...
)
```

```
## S3 method for class 'mplus.model'
prepare_graph(
  model,
  label = "est_sig",
  edges = get_edges(x = model, label = label),
  layout = get_layout(x = model),
  nodes = get_nodes(x = model, label = label),
  ...
)
```

Arguments

...	Additional arguments passed to and from functions.
edges	Object of class 'tidy_edges', or a `data.frame` with (at least) the columns `c("from","to")`, and optionally, `c("arrow","label","connect_from","connect_to","curvature")`.
layout	A matrix (or data.frame) that describes the layout; see `get_layout`.
nodes	Optional, object of class 'tidy_nodes', created with the `get_nodes` function, or a `data.frame` with (at least) the column `c("name")`, and optionally, `c("shape","label")`. If set to `NULL` (the default), nodes are inferred from the `layout` and `edges` arguments.
rect_width	Width of rectangles (used to display observed variables), Default: 1.2
rect_height	Height of rectangles (used to display observed variables), Default: 0.8
ellipses_width	Width of ellipses (used to display latent variables), Default: 1
ellipses_height	Height of ellipses (used to display latent variables), Default: 1
variance_diameter	Diameter of variance circles, Default: .8
spacing_x	Spacing between columns of the graph, Default: 1
spacing_y	Spacing between rows of the graph, Default: 1
text_size	Point size of text, Default: 4
curvature	Curvature of curved edges. The curve is a circle segment originating in a point that forms a triangle with the two connected points, with angles at the two connected points equal to `curvature`. To flip a curved edge, use a negative value for curvature. Default: 60
angle	Angle used to connect nodes by the top and bottom. Defaults to NULL, which means Euclidean distance is used to determine the shortest distance between node sides. A numeric value between 0-180 can be provided, where 0 means that only nodes with the same x-coordinates are connected top-to-bottom, and 180 means that all nodes are connected top-to-bottom.
fix_coord	Whether or not to fix the aspect ratio of the graph. Does not work with multigroup or multilevel models. Default: FALSE.
model	Instead of the edges argument, it is also possible to use the model argument and pass an object for which a method exists (e.g., `mplus.model` or `lavaan`).
label	Character, indicating which column to use for node labels. Nodes are labeled with mean values of the observed/latent variables they represent. Defaults to 'est_sig', which consists of the estimate value with significance asterisks.

Value

Object of class 'sem_graph'

Examples

```
library(lavaan)
res <- sem("dist ~ speed", cars)
prepare_graph(res)
```

`skew_kurtosis` *Calculate skew and kurtosis*

Description

Calculate skew and kurtosis, standard errors for both, and the estimates divided by two times the standard error. If this latter quantity exceeds an absolute value of 1, the skew/kurtosis is significant. With very large sample sizes, significant skew/kurtosis is common.

Usage

```
skew_kurtosis(x, verbose = FALSE, se = FALSE, ...)
```

Arguments

`x`	An object for which a method exists.
`verbose`	Logical. Whether or not to print messages to the console, Default: FALSE
`se`	Whether or not to return the standard errors, Default: FALSE
`...`	Additional arguments to pass to and from functions.

Value

A `matrix` of skew and kurtosis statistics for `x`.

Examples

```
skew_kurtosis(datasets::anscombe)
```

syntax *Extract syntax from tidy_sem*

Description

Provides access to the syntax element of a tidy_sem object. This can be used to return or assign to the syntax element.

Usage

```
syntax(x)

syntax(x) <- value
```

Arguments

x	Object of class tidy_sem.
value	A valid value for syntax(x).

Value

data.frame

Examples

```
dict <- tidy_sem(iris, split = "\\.")
dict <- add_paths(dict, Sepal.Width ~~ Sepal.Length)
syntax(dict)
```

table_cors *Extract correlation tables*

Description

Extracts a publication-ready covariance or correlation matrix from an object for which a method exists.

Usage

```
table_cors(x, value_column = "est_sig_std", digits = 2, ...)
```

Arguments

x	An object for which a method exists.
value_column	Character. Name of the column to use to propagate the matrix. Defaults to "est_sig_std", the standardized estimate with significance asterisks.
digits	Number of digits to round to when formatting values.
...	Additional arguments passed to and from methods.

Value

A Matrix or a list of matrices (in case there are between/within correlation matrices).

Author(s)

Caspar J. van Lissa

Examples

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

---

table_results *Print results table formatted for publication*

Description

Takes a model object, and formats it as a publication-ready table.

Usage

```
table_results(
  x,
  columns = c("label", "est_sig", "se", "pval", "confint", "group", "level"),
  digits = 2,
  ...
)
```

Arguments

x	A model object for which a method exists.
columns	A character vector of columns to retain from the results section. If this is set to NULL, all available columns are returned. Defaults to c("label","est_sig","se","pval","confint", These correspond to 1) the parameter label, 2) estimate column with significance asterisks appended (* <.05, ** < .01, *** < .001); 3) standard error, 4) p-value, 5) a formatted confidence interval, 6) grouping variable (if available), 7) level variable for multilevel models, if available.
digits	Number of digits to round to when formatting numeric columns.
...	Logical expressions used to filter the rows of results returned.

Value

A data.frame of formatted results.

Author(s)

Caspar J. van Lissa

See Also

Other Reporting tools: conf_int(), est_sig()

Examples

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

---

tidy_sem *Create a tidy_sem object*

Description

Create an object of class tidy_sem, which has the following elements:

- dictionary An overview of the variables in the tidy_sem object, and their assignment to scale/latent variables.
- data Optionally, the data.frame containing the data referenced in $dictionary.
- syntax Optionally, syntax defining a SEM-model by reference to the variables contained in $data.

Usage

```
tidy_sem(x, split = "_")
```

Arguments

x	An object for which a method exists, e.g., a vector of variable names, or a data.frame.
split	Character. Defining the regular expression used by strsplit to separate variable names into 1) the name of the scale/construct and 2) the number (or name) of the item.

Details

When tidy_sem is called on a character string or data.frame, it attempts to assign variables to superordinate scale/latent variables based on the variable name and the splitting character defined in the split argument. Thus, the function will assign the variable "scale_01" to a scale/latent variable called "scale" when split = "_". Alternatively, if the variable name is "construct.1", the split character "\." separates the "construct" name from item number "1". The character "." is escaped with a double backslash, because it is a special character in regular expressions.

Value

An object of class "tidy_sem"

Author(s)

Caspar J. van Lissa

Examples

```
tidy_sem(c("bfi_1", "bfi_2", "bfi_3", "bfi_4", "bfi_5",
"macqj_1", "macqj_2", "macqj_3", "macqj_4", "macqj_5", "macqj_6",
"macqj_7", "macqj_8", "macqj_9", "macqj_10", "macqj_11",
"macqj_12", "macqj_13", "macqj_14", "macqj_15", "macqj_16",
"macqj_17", "macqj_18", "macqj_19", "macqj_20", "macqj_21",
"macqr_1", "macqr_2", "macqr_3", "macqr_4", "macqr_5", "macqr_6",
"macqr_7", "macqr_8", "macqr_9", "macqr_10", "macqr_11",
"macqr_12", "macqr_13", "macqr_14", "macqr_15", "macqr_16",
"macqr_17", "macqr_18", "macqr_19", "macqr_20", "macqr_21", "sex"))
tidy_sem(c("bfi_1", "bfi_2", "bfi_3", "bfi_4", "bfi_5",
"mac_q_j_1", "mac_q_j_2", "mac_q_j_3", "mac_q_j_4", "mac_q_j_5", "mac_q_j_6",
"mac_q_j_7", "mac_q_j_8", "mac_q_j_9", "mac_q_j_10", "mac_q_j_11",
"mac_q_j_12", "mac_q_j_13", "mac_q_j_14", "mac_q_j_15", "mac_q_j_16",
"mac_q_j_17", "mac_q_j_18", "mac_q_j_19", "mac_q_j_20", "mac_q_j_21",
"mac_q_r_1", "mac_q_r_2", "mac_q_r_3", "mac_q_r_4", "mac_q_r_5", "mac_q_r_6",
"mac_q_r_7", "mac_q_r_8", "mac_q_r_9", "mac_q_r_10", "mac_q_r_11",
"mac_q_r_12", "mac_q_r_13", "mac_q_r_14", "mac_q_r_15", "mac_q_r_16",
"mac_q_r_17", "mac_q_r_18", "mac_q_r_19", "mac_q_r_20", "mac_q_r_21"))
```

Index