

Package ‘semptools’

July 20, 2026

Title Customizing Structural Equation Modelling Plots

Version 0.4.0

Description Most function focus on specific ways to customize a graph.
They use a 'qgraph' output as the first argument, and return a
modified 'qgraph' object. This allows the functions to be chained
by a pipe operator.

URL <https://sfcheung.github.io/semptools/>

BugReports <https://github.com/sfcheung/semptools/issues>

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports lavaan, rlang, semPlot

Suggests testthat (>= 2.1.0), knitr, rmarkdown, magrittr

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-07-20 15:10:02 UTC

Contents

add_object	3
add_rsqr	4

auto_factor_point_to	6
auto_indicator_order	7
auto_layout_mediation	8
cfa_example	12
change_node_label	12
get_edge_attribute	14
get_node_attribute	16
is_dv_residvar	17
keep_drop_nodes	18
lavaan_indicator_order	19
layout_matrix	20
mark_se	21
mark_sig	24
move_node	26
pa_example	28
pa_example_3covs	28
quick_sem_plot	29
rescale_layout	37
rotate_resid	38
safe_edge_label_position	40
safe_resid_position	42
sem_2nd_order_example	44
sem_example	45
set_cfa_layout	46
set_curve	48
set_edge_attribute	50
set_edge_color	52
set_edge_label_attributes	54
set_edge_line_type	59
set_edge_line_width	61
set_graph_attributes	63
set_node_attribute	65
set_node_border	67
set_node_color	69
set_node_label	70
set_node_size	72
set_sem_layout	74
to_list_of_lists	78

add_object

*Add a Fit Object to a 'qgraph' Object***Description**

Add a fit object (e.g., 'lavaan' output) to the a 'qgraph' object as an attribute.

Usage

```
add_object(semPaths_plot, object)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths() , or a similar qgraph::qgraph object modified by other semptools functions.
object	Should be the object, such as the output of lavaan::sem() or lavaan::cfa() , used by semPlot::semPaths() to generate semPaths_plot. Note that this function will not check whether the object is appropriate because there is no way to do so reliably.

Details

It adds an object to a [qgraph::qgraph](#) object as the attribute "semptools_fit_object", to be retrieved by other functions that need to access the original output used in [semPlot::semPaths\(\)](#) to create a diagram.

Value

The original [qgraph::qgraph](#) object set to semPaths_plot, with the attribute "semptools_fit_object" set to object.

See Also

[semPlot::semPaths\(\)](#)

Examples

```
library(lavaan)
library(semPlot)

mod <-
  'f1 =~ x01 + x02 + x03 + x06
  f2 =~ x04 + x05 + x06 + x07
  f3 =~ x08 + x09 + x10 + x03
  f4 =~ x11 + x12 + x13 + x14
  '
fit <- lavaan::cfa(mod, cfa_example)
p <- semPaths(fit,
```

```

      whatLabels = "est",
      sizeMan = 3.25,
      node.width = 1,
      edge.label.cex = .75,
      mar = c(10, 5, 10, 5),
      DoNotPlot = TRUE)
p <- add_object(p, fit)
attr(p, "semptools_fit_object")

```

add_rsq

*Add R-Squares to Endogenous Variables***Description**

Replace the residual variances of exogenous variables by their R-squares in a [qgraph::qgraph](#) object.

Usage

```

add_rsq(
  semPaths_plot,
  object = NULL,
  digits = 2L,
  rsq_string = "R2=",
  ests = NULL
)

```

Arguments

semPaths_plot	A qgraph object generated by semPaths , or a similar qgraph object modified by other semptools functions.
object	The object used by semPaths to generate the plot. Use the same argument name used in semPaths to make the meaning of this argument obvious. Currently only object of class lavaan is supported.
digits	Integer indicating number of decimal places for the R-squares. Default is 2L.
rsq_string	The string before the R-squares. Default is "R2=".
ests	A data.frame from the parameterEstimates function, or from other function with these columns: lhs, op, rhs, and est. The rows with op equal to r2 are used to find the R-squares. Only used when object is not specified.

Details

Modify a [qgraph::qgraph](#) object generated by [semPaths](#) by setting the labels of the residuals of endogenous variables to their R-squares.

Require either the original object used in the semPaths call, or a data frame with the R-square for each endogenous variable.

Currently supports only plots based on [lavaan](#) output.

Value

If the input is a `qgraph::qgraph` object, the function returns a `qgraph` based on the original one, with R-squares added. If the input is a list of `qgraph` objects, the function returns a list of the same length.

Examples

```
mod_pa <-
  'x1 ~~ x2
    x3 ~ x1 + x2
    x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[ , c("lhs", "op", "rhs",
                                         "est", "pvalue", "se")]

m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels = "est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)
p_pa2 <- add_rsq(p_pa, fit_pa)
plot(p_pa2)

mod_cfa <-
  'f1 =~ x01 + x02 + x03
    f2 =~ x04 + x05 + x06 + x07
    f3 =~ x08 + x09 + x10
    f4 =~ x11 + x12 + x13 + x14
  '

fit_cfa <- lavaan::sem(mod_cfa, cfa_example)
lavaan::parameterEstimates(fit_cfa)[ , c("lhs", "op", "rhs",
                                         "est", "pvalue", "se")]
p_cfa <- semPlot::semPaths(fit_cfa, whatLabels = "est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0)
# Place standard errors on a new line
p_cfa2 <- add_rsq(p_cfa, fit_cfa)
plot(p_cfa2)

mod_sem <-
  'f1 =~ x01 + x02 + x03
    f2 =~ x04 + x05 + x06 + x07
    f3 =~ x08 + x09 + x10
    f4 =~ x11 + x12 + x13 + x14
    f3 ~ f1 + f2
    f4 ~ f1 + f3
  '

# Can be used with mark_se() and mark_sig()
fit_sem <- lavaan::sem(mod_sem, sem_example)
lavaan::parameterEstimates(fit_sem)[ , c("lhs", "op", "rhs",
```

```

                                "est", "pvalue", "se")]
p_sem <- semPlot::semPaths(fit_sem, whatLabels = "est",
                           style = "ram",
                           nCharNodes = 0, nCharEdges = 0)
# Mark significance, and then add standard errors
p_sem2 <- mark_sig(p_sem, fit_sem)
p_sem3 <- mark_se(p_sem2, fit_sem, sep = "\n")
p_sem4 <- add_rsq(p_sem3, fit_sem)
plot(p_sem4)

```

auto_factor_point_to *Create a Matrix for 'factor_point_to'*

Description

Use a named vector or named arguments to create a matrix of the directions of indicators of factors.

Usage

```
auto_factor_point_to(factor_layout, ...)
```

Arguments

factor_layout	A layout matrix to be used in the layout argument, such as one created by layout_matrix() .
...	Additional arguments. If the first argument is not named, then it should be a named vector of directions, names being the names of the factors, and directions can be one of these values: "up", "down", "left", "right". Other arguments are ignored. If the arguments are named, then the names of the arguments are the names of the factors, and the argument values are the direction for the factors.

Details

A helper function to make it easier to create the matrix used by [set_sem_layout\(\)](#) to indicate where the indicators of each factor should be positioned.

It works in two modes. If the first argument is a named vector, such as `c(f1 = "up", f2 = "down")`, then this vector will be used to create the direction matrix.

If the arguments are named, such as `auto_factor_point_to(factor_layout, f1 = "up", f2 = "down")`, then the names are treated as the factor names, and the values of the arguments are treated as the directions.

The matrix created can then be used for the argument `factor_point_to` in [set_sem_layout\(\)](#).

Value

A character matrix of the same dimension as `factor_layout`. The cells of factor names are replaced by the directions to place their indicators.

See Also[set_sem_layout\(\)](#)**Examples**

```

factor_layout <- matrix(c("f1",  NA,  NA,
                          NA, "f3", "f4",
                          "f2",  NA,  NA), byrow = TRUE, 3, 3)
factor_point_to <- auto_factor_point_to(factor_layout,
                                       f1 = "left",
                                       f2 = "left",
                                       f3 = "down",
                                       f4 = "down")

factor_point_to

```

auto_indicator_order *Determine the Order of Indicators Automatically*

Description

Determine the order of indicators and match indicators and factors based on a plot from a 'qgraph' object.

Usage

```
auto_indicator_order(semPaths_plot, add_isolated_manifest = FALSE)
```

Arguments

semPaths_plot A [qgraph::qgraph](#) object generated by [semPlot::semPaths\(\)](#), or a similar [qgraph::qgraph](#) object modified by other [semptools](#) functions.

add_isolated_manifest Logical. Whether observed variables that are not indicators will be included in the output as "factors", each with one indicator (the observed variable).

Details

It inspects a [qgraph::qgraph](#) object and find variables that are the indicators of latent factors.

The output can be used in the argument `indicator_order` of [set_cfa_layout\(\)](#) and [set_sem_layout\(\)](#). It can also be modified, such as reordered, as necessary.

If the generated order is used, there is no need to call this function manually because [set_cfa_layout\(\)](#) and [set_sem_layout\(\)](#) will automatically call this function, if `indicator_order` is not set.

It assumes that observed variables are represented by squares (shape set to "square") and latent variables represented by circles or ovals (shape set to "circle").

An observed variable is considered as an indicator if there is an arrow pointing to it from a latent variable.

If an indicator loaded on more than one latent variable, it will only be matched to one of them, determined by the order of appearance in the internal storage.

It uses node names, not node labels, in generating the output.

Value

A named character vector. The values are the indicators identified. The names are the latent factors the indicators loaded on.

See Also

[set_sem_layout\(\)](#) and [set_cfa_layout\(\)](#).

Examples

```
library(lavaan)
library(semPlot)

mod <-
  'f1 =~ x01 + x02 + x03 + x06
    f2 =~ x04 + x05 + x06 + x07
    f3 =~ x08 + x09 + x10 + x03
    f4 =~ x11 + x12 + x13 + x14
  '

fit <- lavaan::cfa(mod, cfa_example)
p <- semPaths(fit,
  whatLabels = "est",
  sizeMan = 3.25,
  node.width = 1,
  edge.label.cex = .75,
  mar = c(10, 5, 10, 5),
  DoNotPlot = TRUE)

indicator_order <- auto_indicator_order(p)
indicator_order
p2 <- set_cfa_layout(p,
  indicator_order = indicator_order)

plot(p2)

# set_cfa_layout() will call auto_indicator_order()
# automatically if indicator_order is not set.
p3 <- set_cfa_layout(p)
plot(p3)
```

auto_layout_mediation *Set the Layout of a Mediation Model Automatically*

Description

Set the layout of variables in a mediation model in the typical left-to-right style automatically.

Usage

```
auto_layout_mediation(
  object,
  x = NULL,
  y = NULL,
  exclude = NULL,
  v_pos = c("middle", "lower", "upper"),
  v_preference = c("upper", "lower"),
  output = c("matrix", "xy"),
  update_plot = TRUE
)
```

Arguments

object	It can be the output of <code>lavaan::sem()</code> or <code>lavaan::lavaan()</code> , or a lavaan-class object. The model must have a beta matrix of the structural path. It can also be a qgraph object generated by <code>semPlot::semPaths()</code> . A beta` matrix will be reconstructed from <code>update_plot=TRUE</code> , then the function will be applied to all the objects.
x	The variables that will be treated as (pure) x variables: placed on the left of the plot, with no variables predicting them. If NULL, the default, the x variable(s) will be identified automatically.
y	The variables that will be treated as (pure) y variables: placed on the right of the plot, with no variables predicted by them. If NULL, the default, the y variable(s) will be identified automatically.
exclude	The variables to be omitted from the plot, typically the covariates ("control variables") in a model. If NULL, the default, all variables involved in the structural paths will be used in the plot. It is possible to exclude y-variables. However, excluding mediators is not allowed.
v_pos	How the mediators are to be positioned vertically in the first pass. If "middle", the function will try to position them close to the center of the plot. If "lower", it will try to position them to the lower part of the plot. If "upper", it will try to position them to the upper part of the plot.
v_preference	The preference in shifting the mediators upward ("upper") or downward ("lower") in the second pass to avoid blocking or overlapping with any paths in the models. It is used only when <code>v_pos</code> is "middle". If <code>v_pos</code> is "lower", then <code>v_preference</code> will be forced to be "lower". If <code>v_pos</code> is "upper", then <code>v_preference</code> will be forced to be "upper".
output	The format of the output, used if <code>update_plot</code> is FALSE. If "matrix", the output is a two-dimension character matrix with the names of the variables. If "xy", the output is a two-column matrix of the related x- and y-positions of each variables.
update_plot	Logical. Used if object is a qgraph object. If TRUE, the function returns a modified qgraph object with the new layout. If FALSE

Details

Typically, a path model with some x variables, some y variables, and some mediators are drawn from left to right. This function tries to generate the layout matrix automatically, meeting the following

requirements:

- The predictor(s), x variables(x), is/are placed to the left.
- The outcome variable(s), y variable(s), is/are placed to the right.
- The mediator(s) are positioned between x variable(s) and y variable(s) such that all paths point to the right. That is, no vertical path.
- The vertical position(s) of the mediator(s) will be adjusted such that no path passes through a mediator. That is, all paths are visible and not blocked by any mediator.

Multigroup Models or a List of Plots:

The function supports a multigroup model and a list of plots. However, the structural parts of all groups or plots must be identical: same variables, and same paths.

Value

If object is a lavaan-class object, or if update_plot is FALSE, it returns a two-dimension layout matrix of the position of the nodes, or a two-column matrix of the x-y positions of the nodes, depending on the argument output.

If object is a qgraph object and update_plot is TRUE, it returns a qgraph object with the the modified layout.

If object is a list of [qgraph::qgraph](#) objects and update_plot is TRUE, then it returns a list of processed [qgraph::qgraph](#) objects.

See Also

[set_sem_layout\(\)](#). The output of [auto_layout_mediation\(\)](#) can be used by [set_sem_layout\(\)](#).

Examples

```
library(lavaan)
library(semPlot)

# Create a dummy dataset
mod_pa <-
"
m11 ~ c1 + x1
m21 ~ c2 + m11
m2 ~ m11 + c3
m22 ~ m11 + c3
y ~ m2 + m21 + m22 + x1
"

fit <- lavaan::sem(
  mod_pa,
  do.fit = FALSE
)
dat <- simulateData(
  parameterTable(fit),
  sample.nobs = 500,
  seed = 1234
)
```

```

fit <- lavaan::sem(
  mod_pa,
  dat
)

# Set the layout
m <- auto_layout_mediation(
  fit,
  exclude = c("c1", "c2", "c3")
)
pm <- semPlotModel(fit) |> drop_nodes(c("c1", "c2", "c3"))
semPaths(
  pm,
  whatLabels = "est",
  layout = m
)

# v_pos = "lower"
m <- auto_layout_mediation(
  fit,
  exclude = c("c1", "c2", "c3"),
  v_pos = "lower"
)
pm <- semPlotModel(fit) |> drop_nodes(c("c1", "c2", "c3"))
p0 <- semPaths(
  pm,
  whatLabels = "est",
  layout = m
)

# v_pos = "upper"
m <- auto_layout_mediation(
  fit,
  exclude = c("c1", "c2", "c3"),
  v_pos = "upper"
)
pm <- semPlotModel(fit) |> drop_nodes(c("c1", "c2", "c3"))
p0 <- semPaths(
  pm,
  whatLabels = "est",
  layout = m
)

# Can modify a qgraph

pm <- semPlotModel(fit) |> drop_nodes(c("c1", "c2", "c3"))
p <- semPaths(
  pm,
  whatLabels = "est"
)
p2 <- auto_layout_mediation(p)
plot(p2)

```

cfa_example

Sample dataset pa_example

Description

A sample dataset for fitting a confirmatory factor analysis model.

Usage

```
cfa_example
```

Format

An object of class `data.frame` with 200 rows and 14 columns.

Details

Fourteen variables (x01 to x14), 200 cases.

Sample model to fit (in [lavaan::model.syntax](#) notation)

```
mod <-
'f1 =~ x01 + x02 + x03
 f2 =~ x04 + x05 + x06 + x07
 f3 =~ x08 + x09 + x10
 f4 =~ x11 + x12 + x13 + x14
'
```

change_node_label

Change node labels

Description

Change the labels of selected nodes.

Usage

```
change_node_label(
  semPaths_plot,
  label_list = NULL,
  label.cex,
  label.scale,
  label.prop,
  label.norm
)
```

Arguments

semPaths_plot	A <code>qgraph::qgraph</code> object generated by <code>semPlot::semPaths</code> , or a similar <code>qgraph</code> object modified by other <code>semtools</code> functions.
label_list	A list of named lists. Each named list should have two named values: <code>node</code> and <code>to</code> . The first part, <code>node</code> , is a character denoting the label to be changed. It should be as appeared in the <code>qgraph</code> . The second part, <code>to</code> , is the new label. Expression can be used in <code>to</code> . A named vector can also be used, with the names being the nodes to be changed, and the values the new labels.
label.cex	Identical to the same argument in <code>semPlot::semPaths()</code> . A number tha control the size of labels in the nodes. It has no default. If not set, then this option in the <code>semPaths_plot</code> will not be changed.
label.scale	Identical to the same argument in <code>semPlot::semPaths</code> . A logical value that determine whether labels will be scaled (resized) to the nodes they attach to. It has no default. If not set, then this option in the <code>semPaths_plot</code> will not be changed.
label.prop	Identical to the same argument in <code>semPlot::semPaths</code> . A numeric vector of length equal to the number of nodes. If <code>label.scale</code> is <code>TRUE</code> , this number is the proportion of the width of a node that its label will be scaled (resized) to. It has no default. If not set, then this option in the <code>semPaths_plot</code> will not be changed.
label.norm	Identical to the same argument in <code>semPlot::semPaths</code> . It must be a string. All labels as wide as or narrower than this string will have the same font size, while all labels wider than this string will be rescaled to have the same width as this string. It has no default. If not set, then this option in the <code>semPaths_plot</code> will not be changed.

Details

Modify a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the labels of selected nodes.

Value

A `qgraph::qgraph` based on the original one, with node attributes of selected nodes modified. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
library(semPlot)
library(lavaan)
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '
fit_pa <- sem(mod_pa, pa_example)
parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
```

```

      NA, "x3", "x4",
      "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)

my_label_list <- list(list(node = "x3", to = "mediator"),
  list(node = "x4", to = expression(gamma)))

p_pa2 <- change_node_label(p_pa, my_label_list)
plot(p_pa2)

```

get_edge_attribute	<i>Get the Attribute Values of Selected Edges</i>
--------------------	---

Description

Get the values of an arbitrary attribute of selected edges.

Usage

```

get_edge_attribute(
  semPaths_plot,
  edges = NULL,
  attribute_name = NULL,
  check_direction = TRUE
)

```

Arguments

- | | |
|-----------------|---|
| semPaths_plot | A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. Unlike some other functions, this function does <i>not</i> support a list of qgraph::qgraph objects. If semPaths_plot is a list of qgraph::qgraph objects, only the first object will be processed. Use functions such as lapply() manually to get the attribute values from a list of qgraph::qgraph objects. |
| edges | A character vector of the edges. Each edge is denoted by lavaan model syntax, such as $y \sim x$ or $x_1 \sim x_2$. If NULL, then the values of all edges will be retrieved. |
| attribute_name | The name of the attribute from which the values will be retrieved. |
| check_direction | If FALSE, the direction of an edge is ignored. For example, both $y \sim x$ and $x \sim y$ will affect $y \sim x$, $x \sim y$, and $y \sim x$. Useful when we want to identify an edge regardless of its direction and whether it is directional. |

Details

From a `qgraph::qgraph()` object generated by `semPlot::semPaths()`, get the values of an attribute of selected edges.

This function is designed to be a general one that gets the values of the attributes named by the user.

Setting the value of values:

This argument can be set in three ways.

For a named vector, the name of an element should be the path as specified by `lavaan::model.syntax` or as appeared in `lavaan::parameterEstimates()`.

For example, if the attributes to be changed are the colors of selected edges, to change the color of the path regressing y on x, the name should be "y ~ x". To change the color of the covariance between x1 and x2, the name should be "x1 ~~ x2". Therefore, `c("y ~ x1" = "red", "x1 ~~ x2" = "blue")` changes the colors of the path from x1 to y and the covariance between x1 and x2 to "red" and "blue", respectively.

The order of the two nodes *may* matter for covariances. Therefore, if the attribute of a covariance is not changed, try switching the order of the two nodes.

For a list of named lists, each named list should have three named values: `from`, `to`, and `new_value`. The attribute of the edge from `from` to `to` will be set to `new_value`.

The second approach is no longer recommended, though kept for backward compatibility.

The last approach is setting values to a one-element vector with *no name*. All edges in plot will then have the selected attributes set to this value.

Value

A named vector of the values.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~  x1 + x2
  x4 ~  x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1",  NA,  NA,
              NA, "x3", "x4",
              "x2",  NA,  NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)

get_edge_attribute(
  p_pa,
  edges = c("x2 ~~ x1", "x4 ~ x1"),
  attribute_name = "labels")

get_edge_attribute(
```

```
p_pa,
edges = c("x2 ~~ x1", "x4 ~ x1"),
attribute_name = "color")
```

get_node_attribute	<i>Get the Attribute Values of Selected Nodes</i>
--------------------	---

Description

Get the values of an arbitrary attributes of selected nodes.

Usage

```
get_node_attribute(
  semPaths_plot,
  nodes = NULL,
  attribute_name = NULL,
  check_nodes = TRUE
)
```

Arguments

- | | |
|----------------|---|
| semPaths_plot | A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. Unlike some other functions, this function does <i>not</i> support a list of qgraph::qgraph objects. If semPaths_plot is a list of qgraph::qgraph objects, only the first object will be processed. Use functions such as lapply() manually to get the attribute values from a list of qgraph::qgraph objects. |
| nodes | A character vector of the names of nodes. If NULL, then the values of all nodes will be retrieved. |
| attribute_name | The name of the attribute from which the values will be retrieved. |
| check_nodes | Logical. If TRUE and at least one node specified in values are not in semPaths_plot. |

Details

From a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#), and get the values of an attribute of selected nodes.

This function is designed to be a general one that gets the values of the attributes named by the user.

Value

A named vector of the values.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

get_node_attribute(
  p_pa,
  nodes = c("x1", "x3"),
  attribute_name = "color"
)
get_node_attribute(
  p_pa,
  attribute_name = "label.cex"
)
```

is_dv_residvar

Identify dependent Variable residual variance

Description

Check which parameters in a lavaan output are the residual variance of a dependent variable.

Usage

```
is_dv_residvar(lavaan_out)
```

Arguments

lavaan_out A [lavaan::lavaan](#) object.

Details

Check which parameters in a lavaan output are the variance of a dependent variable. Indicators of a latent variable will be excluded.

Value

A boolean vector with length equal to the number of rows in the lavaan output.

Examples

```

mod <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod, pa_example)
is_dv_residvar(fit_pa)

mod <-
  'f1 =~ x01 + x02 + x03
  f2 =~ x04 + x05 + x06 + x07
  f3 =~ x08 + x09 + x10
  f4 =~ x11 + x12 + x13 + x14
  '

fit_cfa <- lavaan::cfa(mod, cfa_example)
is_dv_residvar(fit_cfa)

mod <-
  'f1 =~ x01 + x02 + x03
  f2 =~ x04 + x05 + x06 + x07
  f3 =~ x08 + x09 + x10
  f4 =~ x11 + x12 + x13 + x14
  f3 ~ f1 + f2
  f4 ~ f1 + f3
  '

fit_sem <- lavaan::sem(mod, sem_example)
is_dv_residvar(fit_sem)

```

keep_drop_nodes	<i>Keep or drop nodes</i>
-----------------	---------------------------

Description

Keep or drop nodes from an `semPlotModel` object.

Usage

```
drop_nodes(object, nodes)
```

```
keep_nodes(object, nodes)
```

Arguments

<code>object</code>	An an <code>semPlot::semPlotModel</code> generated by <code>semPlot::semPlotModel()</code> .
<code>nodes</code>	A character vector of the nodes to be kept or removed.

Details

These functions can be used to edit the nodes in an `semPlot::semPlotModel` generated by `semPlot::semPlotModel()`. The edited object can then be passed to `semPlot::semPaths()` to generate a path diagram.

Use `keep_nodes()` to specify the nodes to be kept. All other nodes will be removed.

Use `drop_nodes()` to specify the nodes to be dropped. All other nodes will be kept.

Value

An object of the class `semPlot::semPlotModel`.

Examples

```
mod_pa <-
  'x1 ~~ x2
    x3 ~ x1 + x2
    x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
pm_pa <- semPlot::semPlotModel(fit_pa)
semPlot::semPaths(pm_pa, whatLabels = "est",
                  style = "ram",
                  nCharNodes = 0, nCharEdges = 0,
                  layout = m)
pm_pa2 <- drop_nodes(pm_pa, c("x3"))
semPlot::semPaths(pm_pa2, whatLabels = "est",
                  style = "ram",
                  nCharNodes = 0, nCharEdges = 0,
                  layout = m)
pm_pa3 <- keep_nodes(pm_pa, c("x1", "x3", "x4"))
semPlot::semPaths(pm_pa3, whatLabels = "est",
                  style = "ram",
                  nCharNodes = 0, nCharEdges = 0,
                  layout = m)
```

lavaan_indicator_order

Determine the Order of Indicators Using a 'lavaan' Model Syntax

Description

Determine the order of indicators and match indicators and factors based on a 'lavaan' model syntax.

Usage

```
lavaan_indicator_order(model_syntax)
```

Arguments

`model_syntax` A string that should be a model specified in lavaan model syntax. Only the factor structure (operator `=~`) in the model will be used.

Details

It generates a named vector for the argument `indicator_order` of `set_cfa_layout()` and `set_sem_layout()` using a lavaan model syntax.

A variable is considered an indicator if it is on the right-hand side of the operator `=~`.

If an indicator loaded on more than one latent variable, it will only be matched to one of them, determined by the order of appearance in the internal storage.

Value

A named character vector. The values are the indicators in the model syntax. The names are the latent factors the indicators loaded on.

See Also

`set_sem_layout()` and `set_cfa_layout()`.

Examples

```
mod <-
  'f1 =~ x01 + x02 + x03 + x06
   f4 =~ x11 + x12 + x13 + x14
   f2 =~ x04 + x05 + x06 + x07
   f3 =~ x08 + x09 + x10 + x03
  '
lavaan_indicator_order(mod)

mod <-
  'f1 =~ x01 + x02 + x03 + x06
   f3 =~ x08 + x09 + x10 + x03
   f2 =~ x04 + x05 + x06 + x07
   f4 =~ x11 + x12 + x13 + x14
   f3 ~ f1 + f2
   f4 ~ f3
  '
lavaan_indicator_order(mod)
```

layout_matrix

Create the layout matrix for semPaths

Description

Create the layout matrix from a list of coordinates for semPaths.

Usage

```
layout_matrix(...)
```

Arguments

... Each node in the matrix is specified by this form: `name = c(x, y)`. The name is the node label, and the vector is the position of the node. The first element is the x position, and the second element is the y position, measured from the top left corner. The size of the grid is determined automatically. For a grid of n rows and m columns, the top left cell is specified by `c(1, 1)`, and the bottom right cell is specified by `c(n, m)`.

Details

The layout argument in `semPlot::semPaths()` accepts a matrix with node labels as the elements, and NA for empty cells. This function allows user to create the matrix using a list of coordinates for the node labels.

Note that `semPlot::semPaths()` only treat a matrix as a layout matrix as a matrix of grid if it has three or more columns. Therefore, if the resulting matrix has only one or two columns, it will be converted to a matrix with three columns, by inserting a column of NAs for a two-column matrix, and inserting the resulting matrix between two columns of NAs for a one-column matrix.

Value

A layout matrix for the layout argument of `semPlot::semPaths()`.

Examples

```
# Suppose this is the layout to be created:
m0 <- matrix(c("x1", NA, NA, NA,
               "x2", "x3", NA, NA,
               NA, "x4", NA, "x5"), byrow = TRUE, 3, 4)

# This call will create the same matrix.
m1 <- layout_matrix(x1 = c(1, 1),
                   x2 = c(2, 1),
                   x3 = c(2, 2),
                   x4 = c(3, 2),
                   x5 = c(3, 4))

#The two matrices should be identical.
m0 == m1
```

mark_se

Add Standard Error/Confidence Interval Estimates to Parameter Estimates (Edge Labels)

Description

Add standard error or confidence interval estimates, in parentheses, to parameter estimates (edge labels) in a `qgraph::qgraph` object.

Usage

```
mark_se(
  semPaths_plot,
  object = NULL,
  sep = " ",
  digits = 2L,
  ests = NULL,
  std_type = FALSE
)

mark_ci(
  semPaths_plot,
  object = NULL,
  sep = " ",
  digits = 2L,
  ests = NULL,
  std_type = FALSE
)
```

Arguments

<code>semPaths_plot</code>	A qgraph object generated by semPaths , or a similar qgraph object modified by other semtools functions.
<code>object</code>	The object used by <code>semPaths</code> to generate the plot. Use the same argument name used in semPaths to make the meaning of this argument obvious. Currently only object of class <code>lavaan</code> is supported.
<code>sep</code>	A character string to separate the coefficient and the standard error (in parentheses). Default to " " (one space). Use "\n" to enforce a line break.
<code>digits</code>	Integer indicating number of decimal places for the appended standard errors. Default is 2L.
<code>ests</code>	A data.frame from the parameterEstimates function, or from other function with these columns: <code>lhs</code> , <code>op</code> , <code>rhs</code> , <code>se</code> (for SE), and <code>ci.lower</code> and <code>ci.upper</code> (for CI). Only used when <code>object</code> is not specified.
<code>std_type</code>	If standardized solution is used in the plot, set this either to the type of standardization (e.g., "std.all") or to TRUE. It will be passed to lavaan::standardizedSolution() to compute the standard errors for the standardized solution. Used only if standard errors are not supplied directly through <code>ests</code> .

Details

Modify a [qgraph::qgraph](#) object generated by [semPaths](#) (currently in parentheses) to the labels. Require either the original object used in the `semPaths` call, or a data frame with the standard error (or confidence interval) for each parameter. The latter option is for standard errors/confidence interval not computed by `lavaan` but by other functions.

Currently supports only plots based on [lavaan](#) output.

This function is a variant of, and can be combined with, the [mark_sig](#) function.


```

p_sem <- semPlot::semPaths(fit_sem, whatLabels = "est",
                           style = "ram",
                           nCharNodes = 0, nCharEdges = 0)
# Mark significance, and then add standard errors
p_sem2 <- mark_sig(p_sem, fit_sem)
p_sem3 <- mark_se(p_sem2, fit_sem, sep = "\n")
plot(p_sem3)

# Add confidence intervals
p_sem4 <- mark_ci(p_sem, fit_sem, sep = "\n")
plot(p_sem4)

```

mark_sig

*Mark Parameter Estimates (Edge Labels) Based on p-Value***Description**

Mark parameter estimates (edge labels) based on p-value.

Usage

```

mark_sig(
  semPaths_plot,
  object = NULL,
  alphas = c(`*` = 0.05, `**` = 0.01, `***` = 0.001),
  ests = NULL,
  std_type = FALSE,
  ests_r2 = NULL,
  r2_prefix = "R2="
)

```

Arguments

- | | |
|---------------|---|
| semPaths_plot | A qgraph::qgraph object generated by semPaths, or a similar qgraph object modified by other semptools functions. |
| object | The object used by semPaths to generate the plot. Use the same argument name used in semPaths to make the meaning of this argument obvious. Currently only object of class lavaan is supported. |
| alphas | A named numeric vector. Each element is the cutoff (level of significance), and the name of it is the symbol to be used if p-value is less than this cutoff. The default is c("" = .05, "" = .01 , "" = .001). |
| ests | A data.frame from the parameterEstimates function, or from other function with these columns: lhs, op, rhs, and pvalue. Only used when object is not specified. |

std_type	If standardized solution is used in the plot, set this either to the type of standardization (e.g., "std.all") or to TRUE. It will be passed to <code>lavaan::standardizedSolution()</code> to compute the p -values for the standardized solution. Used only if p -values are not supplied directly through <code>ests</code> .
ests_r2	A data.frame with these columns: lhs and pvalue, storing the p -values for R-squares of the variables listed on lhs. If provided, the p -values in this data.frame will override those in object or <code>ests</code> , if any. Used only when R-squares are present in the <code>semPaths_plot</code> .
r2_prefix	The prefix used to identify R-squares in <code>semPaths_plot</code> . Default is "R2=".

Details

Modify a `qgraph::qgraph` object generated by `semPaths` and add marks (currently asterisk, "*") to the labels based on their p -values. Require either the original object used in the `semPaths` call, or a data frame with the p -values for each parameter. The latter option is for p -values not computed by `lavaan` but by other functions.

Currently supports only plots based on `lavaan` output.

R-squares:

Normally, `lavaan` does not compute p -values for R-squares. If R-squares are detected in the plot (based on `r2_prefix`), they will not be marked, except for the following conditions:

- Users set `ests` to a data frame with R-squares (`op = "r2"`) as well as their p -values (under `pvalue`) computed by some methods.
- Users set `ests_r2` to a data frame storing only the p -values for R-squares computed by some methods (and let the function find the p -values for other parameters from object).

Value

A `qgraph::qgraph` based on the original one, with marks appended to edge labels based on their p -values.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '
fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)
p_pa2 <- mark_sig(p_pa, fit_pa)
plot(p_pa2)
```

```

mod_cfa <-
'f1 =~ x01 + x02 + x03
  f2 =~ x04 + x05 + x06 + x07
  f3 =~ x08 + x09 + x10
  f4 =~ x11 + x12 + x13 + x14
,

fit_cfa <- lavaan::sem(mod_cfa, cfa_example)
lavaan::parameterEstimates(fit_cfa)[, c("lhs", "op", "rhs", "est", "pvalue")]
p_cfa <- semPlot::semPaths(fit_cfa, whatLabels="est",
                           style = "ram",
                           nCharNodes = 0, nCharEdges = 0)
p_cfa2 <- mark_sig(p_cfa, fit_cfa)
plot(p_cfa2)

mod_sem <-
'f1 =~ x01 + x02 + x03
  f2 =~ x04 + x05 + x06 + x07
  f3 =~ x08 + x09 + x10
  f4 =~ x11 + x12 + x13 + x14
  f3 ~ f1 + f2
  f4 ~ f1 + f3
,

fit_sem <- lavaan::sem(mod_sem, sem_example)
lavaan::parameterEstimates(fit_sem)[, c("lhs", "op", "rhs", "est", "pvalue")]
p_sem <- semPlot::semPaths(fit_sem, whatLabels="est",
                           style = "ram",
                           nCharNodes = 0, nCharEdges = 0)
p_sem2 <- mark_sig(p_sem, fit_sem)
plot(p_sem2)

```

move_node

Move Nodes in a Plot

Description

Move the nodes in an plot generated by `semPlot::semPaths()`.

Usage

```
move_node(semPaths_plot, move_by = NULL, check_nodes = TRUE)
```

Arguments

`semPaths_plot` A `qgraph::qgraph` object generated by `semPlot::semPaths`, or a similar `qgraph` object modified by other `semptools` functions. It can also be a list of `qgraph::qgraph` objects, probably though not necessarily from a multigroup model. If it is a list of `qgraph::qgraph` objects, then the function will be applied to all the objects.

- move_by** A named list. The names are the names of nodes to be moved. Each element of the list can be either a named numeric vector or an unnamed vector of two numbers. If a named vector, *x* is the horizontal change in the position (e.g., .25 means moving to the right by .25, and -.5 means moving to the left by .5), and *y* is the vertical change (e.g., .5 means moving up by .5, and -.1 means moving down by .1). If unnamed, then the vector must have two numbers, the first for the horizontal move (*x*) and the second for the vertical move. The value is interpreted based on the unit of the plot, usually from -1 to 1 for *x* and *y*, with 0 being the center of the plot.
- check_nodes** Logical. If TRUE and at least one node specified in *values* are not in *semPaths_plot*.

Details

Modify a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#) and move selected nodes.

The change assumes that the layout is defined with 0 as the center, -1 to 1 from left to right, and from bottom to top.

Value

A [qgraph::qgraph](#) based on the original one, with the selected nodes moved. If *semPaths_plot* is a list of [qgraph::qgraph](#) objects, then a list of processed [qgraph::qgraph](#) objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)

p_changed <- move_node(p_pa,
  list(x3 = c(x = -.25, y = -.25),
       x4 = c(y = .5)))

plot(p_changed)
```

pa_example	Sample dataset pa_example
------------	---------------------------

Description

A sample dataset for fitting a path analysis model.

Usage

pa_example

Format

An object of class `data.frame` with 100 rows and 4 columns.

Details

Four variables (x1 to x4), 100 cases.
Sample model to fit (in `lavaan::model.syntax` notation)

```
mod <-  
'x1 ~~ x2  
  x3 ~ x1 + x2  
  x4 ~ x1 + x3  
'
```

pa_example_3covs	Sample dataset pa_example_3covs
------------------	---------------------------------

Description

A sample dataset for fitting a path analysis model, with three control variables.

Usage

pa_example_3covs

Format

An object of class `data.frame` with 100 rows and 7 columns.

Details

Four variables (x1 to x4), and three control variables (cov1, cov2, cov3), 100 cases.

Sample model to fit (in `lavaan::model.syntax` notation)

```
mod <-
'
x3 ~ x1 + x2 + cov1 +cov2 + cov3
x4 ~ x1 + x3 + cov1 +cov2 + cov3
'
```

quick_sem_plot	<i>Quick Plots of Common Models</i>
----------------	-------------------------------------

Description

Simple-to-use functions for generating plots of common models.

Usage

```
quick_parallel_mediation(
  object,
  x,
  m,
  y,
  mediators_position = c("center", "top", "bottom"),
  what = "path",
  whatLabels = "est",
  style = c("lisrel", "ram"),
  nCharNodes = 0,
  nCharEdges = 0,
  sizeMan = NULL,
  sizeLat = NULL,
  edge.label.cex = NULL,
  intercepts = FALSE,
  ...,
  plot_now = TRUE,
  do_mark_se = TRUE,
  do_mark_sig = TRUE,
  do_rotate_resid = TRUE,
  do_add_rsqr = TRUE,
  add_notes = FALSE,
  notes = NULL
)

q_parallel(
  object,
```

```

x,
m,
y,
mediators_position = c("center", "top", "bottom"),
what = "path",
whatLabels = "est",
style = c("lisrel", "ram"),
nCharNodes = 0,
nCharEdges = 0,
sizeMan = NULL,
sizeLat = NULL,
edge.label.cex = NULL,
intercepts = FALSE,
...,
plot_now = TRUE,
do_mark_se = TRUE,
do_mark_sig = TRUE,
do_rotate_resid = TRUE,
do_add_rsqr = TRUE,
add_notes = FALSE,
notes = NULL
)

```

```

quick_serial_mediation(
  object,
  x,
  m,
  y,
  mediators_position = c("top", "bottom"),
  what = "path",
  whatLabels = "est",
  style = c("lisrel", "ram"),
  nCharNodes = 0,
  nCharEdges = 0,
  sizeMan = NULL,
  sizeLat = NULL,
  edge.label.cex = NULL,
  intercepts = FALSE,
  ...,
  plot_now = TRUE,
  do_mark_se = TRUE,
  do_mark_sig = TRUE,
  do_rotate_resid = TRUE,
  do_add_rsqr = TRUE,
  add_notes = FALSE,
  notes = NULL
)

```

```
q_serial(  
  object,  
  x,  
  m,  
  y,  
  mediators_position = c("top", "bottom"),  
  what = "path",  
  whatLabels = "est",  
  style = c("lisrel", "ram"),  
  nCharNodes = 0,  
  nCharEdges = 0,  
  sizeMan = NULL,  
  sizeLat = NULL,  
  edge.label.cex = NULL,  
  intercepts = FALSE,  
  ...,  
  plot_now = TRUE,  
  do_mark_se = TRUE,  
  do_mark_sig = TRUE,  
  do_rotate_resid = TRUE,  
  do_add_rsqr = TRUE,  
  add_notes = FALSE,  
  notes = NULL  
)
```

```
quick_simple_mediation(  
  object,  
  x,  
  m,  
  y,  
  mediators_position = c("top", "bottom"),  
  what = "path",  
  whatLabels = "est",  
  style = c("lisrel", "ram"),  
  nCharNodes = 0,  
  nCharEdges = 0,  
  sizeMan = NULL,  
  sizeLat = NULL,  
  edge.label.cex = NULL,  
  intercepts = FALSE,  
  ...,  
  plot_now = TRUE,  
  do_mark_se = TRUE,  
  do_mark_sig = TRUE,  
  do_rotate_resid = TRUE,  
  do_add_rsqr = TRUE,  
  add_notes = FALSE,  
  notes = NULL
```

```

)

q_simple(
  object,
  x,
  m,
  y,
  mediators_position = c("top", "bottom"),
  what = "path",
  whatLabels = "est",
  style = c("lisrel", "ram"),
  nCharNodes = 0,
  nCharEdges = 0,
  sizeMan = NULL,
  sizeLat = NULL,
  edge.label.cex = NULL,
  intercepts = FALSE,
  ...,
  plot_now = TRUE,
  do_mark_se = TRUE,
  do_mark_sig = TRUE,
  do_rotate_resid = TRUE,
  do_add_rsqa = TRUE,
  add_notes = FALSE,
  notes = NULL
)

```

Arguments

<code>object</code>	A lavaan object, such as the output of <code>lavaan::sem()</code> .
<code>x</code>	The name of the x variable. Must have exactly one x variable.
<code>m</code>	The name(s) of the m variable(s). The allowed number of m variable(s) depends of the model to be drawn.
<code>y</code>	The name of the y variable. Must have exactly one y variable.
<code>mediators_position</code>	For a simple mediation model, it can be either "top" or "bottom". For a parallel or serial mediation model, it can be "top", "bottom", or "center".
<code>what</code>	The same argument of <code>semPlot::semPaths()</code> . What the edges (arrows) denote. Default is "path", the paths drawn with equal width.
<code>whatLabels</code>	The same argument of <code>semPlot::semPaths()</code> . What the edge labels represent. Default is "est", the parameter estimates. Can be set to "std" to print standardized coefficients.
<code>style</code>	The same argument of <code>semPlot::semPaths()</code> . How residual variances are drawn. Can be "lisrel", the default, or "ram".
<code>nCharNodes</code>	The same argument of <code>semPlot::semPaths()</code> . Default is 0, to disable abbreviation of the variable names.

nCharEdges	The same argument of <code>semPlot::semPaths()</code> . Default is 0, to disable abbreviation of the edge labels.
sizeMan	The same argument of <code>semPlot::semPaths()</code> . The size of the observed variables. Default is NULL and the actual size determined internally based on the number of mediators.
sizeLat	The same argument of <code>semPlot::semPaths()</code> . The size of the latent variables. Default is NULL and the actual size determined internally based on the number of mediators.
edge.label.cex	The same argument of <code>semPlot::semPaths()</code> . The size of the edge labels (parameter estimates). Default is NULL and the actual size determined internally based on the number of mediators.
intercepts	The same argument of <code>semPlot::semPaths()</code> , determining whether intercepts will be plotted. Default is FALSE, different from <code>semPlot::semPaths()</code> . It should not be set to TRUE, but included as an argument for possible features to be added in the future.
...	Other arguments to be passed to <code>semPlot::semPaths()</code> .
plot_now	Logical. If TRUE, the default, the plot will be plotted immediately. Set it to FALSE if the plot will be further processed before being plotted.
do_mark_se	Logical. If TRUE, the default, standard errors will be added by <code>mark_se()</code> . The lavaan standard errors will be used if the standardized coefficients are requested (whatLabels set to std).
do_mark_sig	Logical. If TRUE, the default, significance test results will be marked by asterisks using <code>mark_sig()</code> .
do_rotate_resid	Logical. If TRUE, the default, Error variances, or R-squares if <code>do_add_rsq</code> is TRUE, will be rotated based on the layout.
do_add_rsq	Logical. If TRUE, the default, Error variances will be replaced by R-squares, added by <code>add_rsq()</code> .
add_notes	Logical. If TRUE and <code>plot_now</code> is also TRUE, a note will be added by <code>text()</code> , using notes. Customization is limited. Do not use for now. Commonly used notes will be added in the future.
notes	A string, the notes to be printed if <code>add_notes</code> is TRUE.

Details

A collection of functions for generating the plots of common models. They are designed to need as few arguments as possible to have a plot that should need minimal postprocessing.

Currently, functions are available for these plots:

- Simple mediation models (a model with only one mediator): `q_simple()` or `quick_simple_mediation()`.
- Parallel mediation models (a model with one or more paths between two variables, each path with only one mediator): `q_parallel()` or `quick_parallel_mediation()`.
- Serial mediation models (a model with one main path between two variables, with one or more mediators along the path): `q_serial()` or `quick_serial_mediation()`.

For these three functions, if the default settings are desired, users only need to supply the lavaan output, and specify:

- The x variable (predictor) to be included.
- The m variable(s) to be included, which is a character vector if the model has more than one mediator.
- The y variable to be included.

These variables can be observed variables or latent factors. Indicators of latent variables will not be drawn (unless they are listed in x, m, or y).

The layout is determined by the argument `mediators_position`, with two or more preset layouts for each model.

By default, the following will be added to the plot:

- Asterisks included to denote the significance test results (implemented by `mark_sig()`).
- Standard errors included for free parameters (implemented by `mark_se()`).
- R-squares are drawn in place of error variances for the m variable(s) and y variable (implemented by `add_rsq()`).

These options can be turned off if so desired.

Unlike other function in `semptools`, these functions are usually used to plot a model immediately. Therefore, the resulting plot will be plotted by default. Turned this off by setting `plot_now` to `FALSE` (analogous to setting `DoNotPlot` to `FALSE` when calling `semPlot::semPaths()`).

Although the plot is designed to be ready-to-plot, it can be further processed by other `semptools` functions if necessary, just like the plot of `semPlot::semPaths()`.

Variables to be drawn:

For readability, it is common for researchers to omit some variables when drawing a model.

For example:

$$m \sim x + c1 + c2$$

$$y \sim m + x + c1 + c2$$

If `c1` and `c2` are control variables, researchers want to draw only these\ paths

$$m \sim x$$

$$y \sim m + x$$

The quick plot functions can be used for this purpose. Only selected variables will be included in the plots.

Researchers may also want to draw several plots, one for each pair of the predictor (the x-variable) and the outcome variable (the y-variable).

For example,

$$m \sim x + c1 + c2$$

$$y1 \sim m + x + c1 + c2$$

$$y2 \sim m + x + c1 + c2$$

For this model, in addition to excluding the control variables, researchers may want to generate two diagrams, one for `y1`:

$$m \sim x$$

$y1 \sim m + x$

and the other for y2:

$m \sim x$

$y2 \sim m + x$

Note that all the functions will not check the models. The specification of x , m , and y are assumed to be valid for the fitted models.

Value

A `qgraph::qgraph` generated by `semPlot::semPaths()` and customized by other `semtools` functions is returned invisibly. Called for its side effect.

Examples

```
library(lavaan)
library(semPlot)

# ---- Parallel Mediation Model

mod_parallel <-
  'x04 ~ x01
  x05 ~ x01
  x06 ~ x01
  x07 ~ x01
  x10 ~ x04 + x05 + x06 + x07 + x01'

fit_parallel <- lavaan::sem(mod_parallel,
                           sem_example)

q_parallel(fit_parallel,
           x = "x01",
           m = c("x04", "x05", "x06", "x07"),
           y = "x10")

q_parallel(fit_parallel,
           x = "x01",
           m = c("x04", "x05", "x06", "x07"),
           y = "x10",
           mediators_position = "top")

q_parallel(fit_parallel,
           x = "x01",
           m = c("x04", "x05", "x06", "x07"),
           y = "x10",
           mediators_position = "bottom")

# Suppress some elements for readability

q_parallel(fit_parallel,
           x = "x01",
           m = c("x04", "x05", "x06", "x07"),
           y = "x10",
```

```

        mediators_position = "bottom",
        do_mark_se = FALSE)

# ---- Serial Mediation Model

mod_serial <-
  'x04 ~ x01
  x05 ~ x04 + x01
  x06 ~ x04 + x05 + x01
  x07 ~ x04 + x05 + x06 + x01
  x08 ~ x04 + x05 + x06 + x07 + x01'
fit_serial <- lavaan::sem(mod_serial,
                          sem_example)

q_serial(fit_serial,
         x = "x01",
         m = c("x04", "x05", "x06", "x07"),
         y = "x08")

q_serial(fit_serial,
         x = "x01",
         m = c("x04", "x05", "x06", "x07"),
         y = "x08",
         mediators_position = "bottom")

# Suppress some elements for readability

q_serial(fit_serial,
         x = "x01",
         m = c("x04", "x05", "x06", "x07"),
         y = "x08",
         mediators_position = "bottom",
         do_mark_se = FALSE)

# ---- Simple Mediation Model: With Control Variables

mod_pa <-
  'x3 ~ x1 + x2
  x4 ~ x3 + x1 + x2'
fit_pa <- lavaan::sem(mod_pa,
                     pa_example)

mod_sem <-
  'f1 =~ x01 + x02 + x03
  f2 =~ x04 + x05 + x06 + x07
  f3 =~ x08 + x09 + x10
  f4 =~ x11 + x12 + x13 + x14
  f3 ~ f1 + f2
  f4 ~ f1 + f3'
fit_sem <- lavaan::sem(mod_sem,
                      sem_example)

q_simple(fit_pa,
        x = "x1",
        m = "x3",

```

```

        y = "x4")

# Drawing latent factors only

q_simple(fit_sem,
        x = "f1",
        m = "f3",
        y = "f4",
        whatLabels = "std",
        mediators_position = "bottom")

```

rescale_layout

*Rescale the Layout***Description**

Rescale the layout of a qgraph object, such as the output of semptools functions that modify the output of [semPlot::semPaths\(\)](#).

Usage

```
rescale_layout(semPaths_plot, x_min = -1, x_max = 1, y_min = -1, y_max = 1)
```

Arguments

semPaths_plot A [qgraph::qgraph](#) object generated by [semPlot::semPaths](#), or a similar qgraph object modified by other [semptools](#) functions. It can also be a list of [qgraph::qgraph](#) objects, probably though not necessarily from a multigroup model. If it is a list of [qgraph::qgraph](#) objects, then the function will be applied to all the objects.

x_min, x_max, y_min, y_max

The ranges of x-coordinates and y-coordinates after rescaling. Default is -1 for x_min and y_min, and 1 for x_max and y_max. Change them to enlarge or shrink the plot.

Details

The plot generated by some functions, such as [set_sem_layout\(\)](#), may have the area underused for some model. This function rescale the layout matrix, just like what the rescale argument of [semPlot::semPaths\(\)](#) does.

Value

A [qgraph::qgraph](#) based on the original one, with the layout matrix rescaled. If semPaths_plot is a list of [qgraph::qgraph](#) objects, then a list of processed [qgraph::qgraph](#) objects will be returned.

Examples

```

library(lavaan)
library(semPlot)
mod <-
  'f1 =~ x01 + x02 + x03
  f3 =~ x08 + x09 + x10
  f4 =~ x11 + x12 + x13 + x14
  f3 ~ f1 + x04
  f4 ~ f3 + x05'
fit_sem <- sem(mod, sem_example)
p <- semPaths(fit_sem, whatLabels="est",
  sizeMan = 5,
  nCharNodes = 0,
  nCharEdges = 0,
  edge.width = 0.8,
  node.width = 0.7,
  edge.label.cex = 0.6,
  mar = c(10, 10, 10, 10),
  DoNotPlot = TRUE)

plot(p)

indicator_order <- c("x04", "x05", "x01", "x02", "x03",
  "x11", "x12", "x13", "x14", "x08", "x09", "x10")
indicator_factor <- c("x04", "x05", "f1", "f1", "f1",
  "f4", "f4", "f4", "f4", "f3", "f3", "f3")
factor_layout <- matrix(c("f1", "f3", "f4",
  "x04", "x05", NA), byrow = TRUE, 2, 3)
factor_point_to <- matrix(c("left", "up", "right",
  NA, NA, NA), byrow = TRUE, 2, 3)
p2 <- set_sem_layout(p,
  indicator_order = indicator_order,
  indicator_factor = indicator_factor,
  factor_layout = factor_layout,
  factor_point_to = factor_point_to)
# The original plot with too much unused area
plot(p2)
rect(-1, -1, 1, 1)
rect(-1.5, -1.5, 1.5, 1.5)

# Expand the plot
p3 <- p2
p3 <- rescale_layout(p3)

plot(p3)
rect(-1, -1, 1, 1)
rect(-1.5, -1.5, 1.5, 1.5)

```

Description

Rotate the residuals of selected nodes.

Usage

```
rotate_resid(semPaths_plot, rotate_resid_list = NULL, check_nodes = TRUE)
```

Arguments

- semPaths_plot** A [qgraph::qgraph](#) object generated by [semPlot::semPaths](#), or a similar qgraph object modified by other [semptools](#) functions. It can also be a list of [qgraph::qgraph](#) objects, probably though not necessarily from a multigroup model. If it is a list of [qgraph::qgraph](#) objects, then the function will be applied to all the objects.
- rotate_resid_list** A named vector or a list of named list. For a named vector, the name of an element is the node for which its residual is to be rotated, and the value is the degree to rotate. The 12 o'clock position is zero degree. Positive degree denotes clockwise rotation, and negative degree denotes anticlockwise rotation. For example, `c(x3 = 45, x4 = -45)` means rotating the residual of x3 45 degrees clockwise, and rotating the residual of x4 45 degrees anticlockwise. For a list of named lists, each named list should have two named values: `node` and `rotate`. The position of the residual of node will be placed at `rotate`, in degree. For example, `list(list(node = "x3", rotate = 45), list(node = "x4", rotate = -45))` is equivalent to `c(x3 = 45, x4 = -45)`.
- check_nodes** Logical. If TRUE and at least one node specified in values are not in `semPaths_plot`.

Details

Modify a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#) and rotate the residuals of selected nodes. Currently only supports "ram" and similar styles of [semPlot::semPaths](#).

Value

A [qgraph::qgraph](#) object based on the original one, with `loopRotation` attributes of selected nodes modified. If `semPaths_plot` is a list of [qgraph::qgraph](#) objects, then a list of processed [qgraph::qgraph](#) objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
```

```

p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)

my_rotate_resid_vector <- c(x3 = 45, x4 = -45)

p_pa2v <- rotate_resid(p_pa, my_rotate_resid_vector)
plot(p_pa2v)

my_rotate_resid_list <- list(list(node = "x3", rotate = 45),
  list(node = "x4", rotate = -45))

p_pa2l <- rotate_resid(p_pa, my_rotate_resid_list)
plot(p_pa2l)

```

safe_edge_label_position

Adjust Edge Label Positions to Avoid Overlapping Labels

Description

Move the edge labels away from path intersections.

Usage

```

safe_edge_label_position(
  object,
  layout = NULL,
  default_pos = 0.5,
  tolerance = 0.05,
  update_plot = TRUE
)

```

Arguments

object	It can be the output of <code>lavaan::sem()</code> or <code>lavaan::lavaan()</code> , or a lavaan-class object. The model must have a beta matrix of the structural path. It can also be a qgraph object generated by <code>semPlot::semPaths()</code> . It can also be a list of <code>qgraph::qgraph</code> objects, probably though not necessarily from a multigroup model. If it is a list of <code>qgraph::qgraph</code> objects and <code>update_plot</code> is TRUE, then the function will be applied to all the objects.
layout	A layout matrix. Required if object is a lavaan-class object. Ignored if object is a qgraph object.
default_pos	Used if object is a lavaan-class object. The default position of an edge label. If this position is "safe" (not on the intersection between paths), it will be used. Ignored if object is a qgraph object.

tolerance	If the distance between a position and an intersection is greater than this distance, then a position is considered safe and will not be adjusted.
update_plot	Logical. Used on if object is a qgraph object. If TRUE, the function returns a modified qgraph object. If FALSE, the function returns a named vector of the new positions.

Details

This function identify all intersection points between two paths in a model, and set the position of an edge label to the mid-point of a line segment between an intersection point and the another intersection point or the origin/destination of a path.

This function is intended for having a "likely" readable graph with as little user-intervention as possible. If precise control of the edge label positions is desired, use [set_edge_label_position\(\)](#).

Multigroup Models or a List of Plots:

The function supports a multigroup model and a list of plots. However, the structural parts of all groups or plots must be identical: same variables, and same paths.

Value

If object is a lavaan-class object, it returns a named numeric vector of edge positions to be used by [set_edge_label_position\(\)](#). If object is a qgraph object and update_plot is TRUE, it returns a qgraph object with the adjusted edge label positions. Otherwise, it returns a named vector of the position to be adjusted, as for a lavaan-class object. If object is a list of [qgraph::qgraph](#) objects and update_plot is TRUE, then it returns a list of processed [qgraph::qgraph](#) objects.

See Also

[set_edge_label_position\(\)](#) on setting the positions of edge labels.

Examples

```
library(lavaan)
library(semPlot)
# Create a dummy dataset
mod_pa <-
"
m11 ~ c1 + x1
m12 ~ c2 + m11 + m21
m21 ~ c1 + x1
m22 ~ c1 + m21 + m11
y ~ m12 + m22 + x1
"
fit <- lavaan::sem(
  mod_pa,
  do.fit = FALSE
)
dat <- simulateData(
  parameterTable(fit),
  sample.nobs = 500,
  seed = 1234
```

```

    )
fit <- lavaan::sem(
  mod_pa,
  dat
)
# Set the layout
m <- auto_layout_mediation(
  fit,
  exclude = c("c1", "c2", "c3")
)
pos_new <- safe_edge_label_position(
  fit,
  layout = m
)
pos_new
pm <- semPlotModel(fit) |> drop_nodes(c("c1", "c2"))
p <- semPaths(
  pm,
  whatLabels = "est",
  layout = m,
  DoNotPlot = TRUE
)
# Three labels overlap with each other
plot(p)
# Update the plot
p_safe <- p |> safe_edge_label_position()
# Three labels do not overlap in this plot
plot(p_safe)
# Set the position manually
p_safe2 <- p |>
  set_edge_label_position(pos_new)
plot(p_safe2)

```

safe_resid_position *Adjust Residual Positions*

Description

Rotate the residuals (or R-squares) to avoid overlapping with paths.

Usage

```

safe_resid_position(
  object,
  layout,
  default_angle = 0,
  style = c("1200", "geometry"),
  update_plot = TRUE
)

```

Arguments

object	It can be the output of <code>lavaan::sem()</code> or <code>lavaan::lavaan()</code> , or a lavaan-class object. The model must have a beta matrix of the structural path. It can also be a qgraph object generated by <code>semPlot::semPaths()</code> . It can also be a list of <code>qgraph::qgraph</code> objects, probably though not necessarily from a multigroup model. If it is a list of <code>qgraph::qgraph</code> objects and <code>update_plot</code> is TRUE, then the function will be applied to all the objects.
layout	A layout matrix. Required if object is a lavaan-class object. Ignored if object is a qgraph object.
default_angle	Used if object is a lavaan-class object. The default position of a residual, defined in the same way angle is defined for <code>rotate_resid()</code> . Ignored if object is a qgraph object.
style	The convention for the angles. If "1200", the default, the convention of <code>rotate_resid()</code> is used: top (12 o'clock) is 0, clockwise angle is positive and counterclockwise angle is negative. if "geometry", then the convention in geometry is used: right is 0, counterclockwise is positive, and clockwise is negative.
update_plot	Logical. Used on if object is a qgraph object. If TRUE, the function returns a modified qgraph object. If FALSE, the function returns a named vector of the new positions.

Details

This function identify all directed paths connected to a node, and find the largest arc with no directed paths. The residual (or R-square) is then set to the mid-point of this arc.

This function is intended for having a "likely" readable graph with as little user-intervention as possible. If precise control of the positions is desired, use `rotate_resid()`.

Only directed paths (single-headed arrows) will be considered. Bidirectional paths such as covariances are not taken into account.

Multigroup Models or a List of Plots:

The function supports a multigroup model and a list of plots. However, the structural parts of all groups or plots must be identical: same variables, and same paths.

Value

If object is a lavaan-class object, it returns a named numeric vector of residual angles to be used by `rotate_resid()`. If object is a qgraph object and `update_plot` is TRUE, it returns a qgraph object with the residuals's angles adjusted. Otherwise, it returns a named vector of the angles, as for a lavaan-class object. If object is a list of `qgraph::qgraph` objects and `update_plot` is TRUE, then it returns a list of processed `qgraph::qgraph` objects.

See Also

`rotate_resid()` on rotating a residual.

Examples

```

library(lavaan)
library(semPlot)
# Create a dummy dataset
mod_pa <-
"
m11 ~ x1
m21 ~ m11
m2 ~ m11
m22 ~ m11
y ~ m2 + m21 + m22 + x1
"

fit <- lavaan::sem(
  mod_pa,
  do.fit = FALSE
)
dat <- simulateData(
  parameterTable(fit),
  sample.nobs = 500,
  seed = 1234
)
fit <- lavaan::sem(
  mod_pa,
  dat
)
# Set the layout
m <- auto_layout_mediation(
  fit
)
p <- semPaths(
  fit,
  whatLabels = "est",
  layout = m,
  DoNotPlot = TRUE
) |>
  safe_edge_label_position()
plot(p)
# Update the plot
p_safe <- p |> safe_resid_position()
plot(p_safe)
# Set the position manually
pos_new <- safe_resid_position(p,
  update_plot = FALSE)
pos_new
p_safe2 <- p |>
  rotate_resid(pos_new)
plot(p_safe2)

```

Description

A sample dataset for fitting a latent variable model with two 2nd-order factors.

Usage

sem_2nd_order_example

Format

An object of class data.frame with 500 rows and 21 columns.

Details

Twenty one variables (x01 to x21), 500 cases.
Sample model to fit (in lavaan::model.syntax notation)

```
mod <-  
'f1 =~ x01 + x02 + x03  
f2 =~ x04 + x05 + x06 + x07  
f3 =~ x08 + x09 + x10  
f4 =~ x11 + x12 + x13 + x14  
f5 =~ x15 + x16 + x17 + x18  
f6 =~ x19 + x20 + x21  
f21 =~ 1*f1 + f3 + f4  
f22 =~ 1*f2 + f5 + f6  
f22 ~ f21  
,
```

sem_example	<i>Sample dataset sem_example</i>
-------------	-----------------------------------

Description

A sample dataset for fitting a latent variable model.

Usage

sem_example

Format

An object of class data.frame with 200 rows and 14 columns.

Details

Fourteen variables (x01 to x14), 100 cases.

Sample model to fit (in `lavaan::model.syntax` notation)

```
mod <-
'f1 =~ x01 + x02 + x03
 f2 =~ x04 + x05 + x06 + x07
 f3 =~ x08 + x09 + x10
 f4 =~ x11 + x12 + x13 + x14
 f3 ~ f1 + f2
 f4 ~ f1 + f3
'
```

set_cfa_layout

Configure the layout of factors of a CFA graph by semPaths

Description

Configure the layout of factors and adjust other aspects of a CFA graph by `semPaths`.

Usage

```
set_cfa_layout(
  semPaths_plot,
  indicator_order = NULL,
  indicator_factor = NULL,
  fcov_curve = 0.4,
  loading_position = 0.5,
  point_to = "down"
)
```

Arguments

`semPaths_plot` A `qgraph::qgraph` object generated by `semPaths`, or a similar `qgraph` object modified by other `semptools` functions. It can also be a list of `qgraph::qgraph` objects, probably though not necessarily from a multigroup model. If it is a list of `qgraph::qgraph` objects, then the function will be applied to all the objects.

`indicator_order`

A string vector of the indicators. The order of the names is the order of the indicators in the graph, when they are drawn on the bottom of the graph. The indicators should be grouped by the factors on which they load on. For example, if x1, x2, x4 load on f2, and x3, x5, x6 load on f1, then vector should be either `c("x1", "x2", "x4", "x3", "x5", "x6")` or `c("x3", "x5", "x6", "x1", "x2", "x4")`. Indicators within a group can be ordered in any way. If it is a named vector, its names will be used for the argument `indicator_factor`. If it is `NULL` (default), `auto_indicator_order()` will be called to determine the indicator order automatically.

indicator_factor	A string vector of the same length of the indicator order, storing the name of the factor for which each of the indicator in indicator_factor loads on. For example, if x1, x2, x4 load on f2, and x3, x5, x6 load on f1, and indicator_order is c("x3", "x5", "x6", "x1", "x2", "x4"), then indicator_factor should be c("f2", "f2", "f2", "f1", "f1", "f1"). If NULL (default) and indicator_order is a named vector (supplied by users or generated by auto_indicator_order()), then it will be set to the names of indicator_order.
fcov_curve	A number used to set the curvature of the inter-factor covariances. Default is .4.
loading_position	The positions of all factor loadings. Default is .5, on the middle of the arrows. Larger the number, closer the loadings to the indicators. Smaller the number, closer the loadings to the factors.
point_to	Can be "down", "left", "up", or "right". Specify the direction that the factors "point" to the indicators. Default is "down".

Details

Modify a [qgraph::qgraph](#) object generated by semPaths based on a confirmatory factor analysis model.

Value

A [qgraph::qgraph](#) based on the original one, with various aspects of the model modified. If semPaths_plot is a list of [qgraph::qgraph](#) objects, then a list of processed [qgraph::qgraph](#) objects will be returned.

Examples

```
library(lavaan)
library(semPlot)
mod <-
'f1 =~ x01 + x02 + x03
 f2 =~ x04 + x05 + x06 + x07
 f3 =~ x08 + x09 + x10
 f4 =~ x11 + x12 + x13 + x14
'
fit_cfa <- lavaan::sem(mod, cfa_example)
lavaan::parameterEstimates(fit_cfa)[, c("lhs", "op", "rhs", "est", "pvalue")]
p <- semPaths(fit_cfa, whatLabels="est",
  sizeMan = 2.5,
  nCharNodes = 0, nCharEdges = 0,
  edge.width = 0.8, node.width = 0.7,
  edge.label.cex = 0.6,
  style = "ram",
  mar = c(10,10,10,10))
indicator_order <- c("x04", "x05", "x06", "x07", "x01", "x02", "x03", "x11",
  "x12", "x13", "x14", "x08", "x09", "x10")
indicator_factor <- c("f2", "f2", "f2", "f2", "f1", "f1", "f1", "f4",
  "f4", "f4", "f4", "f3", "f3", "f3")
p2 <- set_cfa_layout(p, indicator_order,
  indicator_factor,
```

```

                                fcov_curve = 1.5,
                                loading_position = .8)
plot(p2)

# Use a named vector for indicator_order
indicator_order2 <- c(f2 = "x04", f2 = "x05", f2 = "x06", f2 = "x07",
                     f1 = "x01", f1 = "x02", f1 = "x03",
                     f4 = "x11", f4 = "x12", f4 = "x13", f4 = "x14",
                     f3 = "x08", f3 = "x09", f3 = "x10")
p2 <- set_cfa_layout(p,
                    indicator_order = indicator_order2,
                    fcov_curve = 1.5,
                    loading_position = .8)
plot(p2)

# Use automatically generated indicator_order and indicator_factor
p2 <- set_cfa_layout(p,
                    fcov_curve = 1.5,
                    loading_position = .8)
plot(p2)

p2 <- set_cfa_layout(p, indicator_order,
                    indicator_factor,
                    fcov_curve = 1.5,
                    loading_position = .8,
                    point_to = "left")
plot(p2)
p2 <- set_cfa_layout(p, indicator_order,
                    indicator_factor,
                    fcov_curve = 1.5,
                    loading_position = .8,
                    point_to = "up")
plot(p2)
p2 <- set_cfa_layout(p, indicator_order,
                    indicator_factor,
                    fcov_curve = 1.5,
                    loading_position = .8,
                    point_to = "right")
plot(p2)

```

set_curve

Bend or Straighten Selected edges

Description

Set the curve attributes of selected edges.

Usage

```
set_curve(
  semPaths_plot,
  curve_list = NULL,
  how = c("value", "ratio"),
  check_direction = TRUE
)
```

Arguments

- | | |
|-----------------|--|
| semPaths_plot | A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. It can also be a list of qgraph::qgraph objects, probably though not necessarily from a multigroup model. If it is a list of qgraph::qgraph objects, then the function will be applied to all the objects. |
| curve_list | A named vector or a list of named list. For a named vector, the name of an element should be the path as specified by lavaan::model.syntax or as appeared in lavaan::parameterEstimates() . For example, to change the curve attribute of the path regressing y on x, the name should be "y ~ x". To change the curve attribute of the covariance between x1 and x2, the name should be "x1 ~~ x2". For example, c("y ~ x1" = -3, "x1 ~~ x2" = 2) change the curve attributes of the path from x1 to y and the covariance between x1 and x2 to -3 and 2, respectively. The order of the two nodes <i>may</i> matter for covariances. Therefore, if the curve of a covariance is not changed, try switching the order of the two nodes. For a list of named lists, each named list should have three named values: from, to, and new_curve. The curve attribute of the edge from from to to will be set to new_curve. |
| how | How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value. |
| check_direction | If FALSE, the direction of an edge is ignored. For example, both y ~ x and x ~ y will affect y ~ x, x ~ y, and y ~~ x. Useful when we want to change an edge regardless of its direction and whether it is directional. |

Details

Modified a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#) and change the curve attributes of selected edges.

If how is set to "ratio", then the new curve value is equal to the old curve value multiplied by the supplied value. This mode lets users increase or decrease the curvature of a curve without knowing the original value.

Value

A [qgraph::qgraph](#) based on the original one, with curve attributes for selected edges changed. If semPaths_plot is a list of [qgraph::qgraph](#) objects, then a list of processed [qgraph::qgraph](#) objects will be returned.

Examples

```

mod_pa <-
  'x1 ~~ x2
    x3 ~ x1 + x2
    x4 ~ x1 + x3
  ,

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

my_curve_vector <- c("x2 ~~ x1" = -1,
                    "x4 ~ x1" = 1)

p_pa2v <- set_curve(p_pa, my_curve_vector)
plot(p_pa2v)

my_curve_list <- list(list(from = "x1", to = "x2", new_curve = -1),
                     list(from = "x1", to = "x4", new_curve = 1))

p_pa2l <- set_curve(p_pa, my_curve_list)
plot(p_pa2l)

```

set_edge_attribute	<i>Set the Attributes of Selected Edges</i>
--------------------	---

Description

Set arbitrary attributes of selected edges.

Usage

```

set_edge_attribute(
  semPaths_plot,
  values = NULL,
  attribute_name = NULL,
  how = c("value", "ratio"),
  check_direction = TRUE
)

```

Arguments

semPaths_plot	A <code>qgraph::qgraph</code> object generated by <code>semPlot::semPaths</code> , or a similar <code>qgraph</code> object modified by other <code>semptools</code> functions. It can also be a list of <code>qgraph::qgraph</code> objects, probably though not necessarily from a multigroup model. If it is a list of <code>qgraph::qgraph</code> objects, then the function will be applied to all the objects.
values	A named vector or a list of named list. See the Details section on how to set this argument.
attribute_name	The name of the attribute to be changed.
how	How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value.
check_direction	If FALSE, the direction of an edge is ignored. For example, both $y \sim x$ and $x \sim y$ will affect $y \sim x$, $x \sim y$, and $y \sim\sim x$. Useful when we want to change an edge regardless of its direction and whether it is directional.

Details

Modify a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the selected attributes of selected edges.

This function is designed to be a general one that changes the attributes named by the user. The user needs to make sure that the attribute actually exists, and the values are valid for the named attribute.

Setting the value of values:

This argument can be set in three ways.

For a named vector, the name of an element should be the path as specified by `lavaan::model.syntax` or as appeared in `lavaan::parameterEstimates()`.

For example, if the attributes to be changed are the colors of selected edges, to change the color of the path regressing y on x , the name should be " $y \sim x$ ". To change the color of the covariance between x_1 and x_2 , the name should be " $x_1 \sim\sim x_2$ ". Therefore, `c("y ~ x1" = "red", "x1 ~ x2" = "blue")` changes the colors of the path from x_1 to y and the covariance between x_1 and x_2 to "red" and "blue", respectively.

The order of the two nodes *may* matter for covariances. Therefore, if the attribute of a covariance is not changed, try switching the order of the two nodes.

For a list of named lists, each named list should have three named values: `from`, `to`, and `new_value`. The attribute of the edge from `from` to `to` will be set to `new_value`.

The second approach is no longer recommended, though kept for backward compatibility.

The last approach is setting values to a one-element vector with *no name*. All edges in plot will then have the selected attributes set to this value.

Value

A `qgraph::qgraph` based on the original one, with the selected attributes of selected edges changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```

mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

my_values_vector <- c("x2 ~~ x1" = "red",
                     "x4 ~ x1" = "blue")

p_pa2v <- set_edge_attribute(p_pa,
                             values = my_values_vector,
                             attribute_name = "color")

plot(p_pa2v)

my_values_list <- list(list(from = "x1", to = "x2", new_value = "red"),
                      list(from = "x1", to = "x4", new_value = "blue"))

p_pa2l <- set_edge_attribute(p_pa,
                             values = my_values_list,
                             attribute_name = "color")

plot(p_pa2l)

```

set_edge_color

Set the Colors of Selected Edges

Description

Set the colors of selected edges.

Usage

```
set_edge_color(semPaths_plot, color_list = NULL, check_direction = TRUE)
```

Arguments

semPaths_plot A [qgraph::qgraph](#) object generated by [semPlot::semPaths](#), or a similar qgraph object modified by other [semptools](#) functions. It can also be a list of [qgraph::qgraph](#)

	objects, probably though not necessarily from a multigroup model. If it is a list of <code>qgraph::qgraph</code> objects, then the function will be applied to all the objects.
color_list	A named vector or a list of named list. See the Details section on how to set this argument.
check_direction	If FALSE, the direction of an edge is ignored. For example, both $y \sim x$ and $x \sim y$ will affect $y \sim x$, $x \sim y$, and $y \sim x$. Useful when we want to change an edge regardless of its direction and whether it is directional.

Details

Modified a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the colors of selected edges.

Setting the value of color_list:

This argument can be set in two ways.

For a named vector, the name of an element should be the path as specified by `lavaan::model.syntax` or as appeared in `lavaan::parameterEstimates()`.

For example, to change the color of the path regressing y on x, the name should be "y ~ x". To change the color of the covariance between x1 and x2, the name should be "x1 ~~ x2". Therefore, `c("y ~ x1" = "red", "x1 ~~ x2" = "blue")` changes the colors of the path from x1 to y and the covariance between x1 and x2 to "red" and "blue", respectively.

The order of the two nodes *may* matter for covariances. Therefore, if the attribute of a covariance is not changed, try switching the order of the two nodes.

For a list of named lists, each named list should have three named values: from, to, and new_color. The attribute of the edge from from to to will be set to new_color.

The second approach is no longer recommended, though kept for backward compatibility.

Value

A `qgraph::qgraph` based on the original one, with colors for selected edges changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)
```

```

my_color_vector <- c("x2 ~~ x1" = "red",
                    "x4 ~ x1" = "blue")

p_pa2v <- set_edge_color(p_pa, my_color_vector)
plot(p_pa2v)

my_color_list <- list(list(from = "x1", to = "x2", new_color = "red"),
                     list(from = "x1", to = "x4", new_color = "blue"))

p_pa2l <- set_edge_color(p_pa, my_color_list)
plot(p_pa2l)

```

set_edge_label_attributes

Set Edge Label Attributes of Selected Edges

Description

Set the attributes of selected edges, such as the label position.

Usage

```

set_edge_label_position(
  semPaths_plot,
  position_list = NULL,
  check_direction = TRUE
)

set_edge_label(semPaths_plot, label_list = NULL, check_direction = TRUE)

set_edge_label_bg(semPaths_plot, label_bg_list = NULL, check_direction = TRUE)

set_edge_label_size(
  semPaths_plot,
  label_size_list = NULL,
  how = c("ratio", "value"),
  check_direction = TRUE
)

```

Arguments

semPaths_plot A [qgraph::qgraph](#) object generated by [semPlot::semPaths](#), or a similar qgraph object modified by other [semptools](#) functions. It can also be a list of [qgraph::qgraph](#) objects, probably though not necessarily from a multigroup model. If it is a list of [qgraph::qgraph](#) objects, then the function will be applied to all the objects.

position_list	The new position. See 'Details' on how to set this argument. If a list of named list is used, the element for new position should be named new_position.
check_direction	If FALSE, the direction of an edge is ignored. For example, both $y \sim x$ and $x \sim y$ will affect $y \sim x$, $x \sim y$, and $y \sim x$. Useful when we want to change an edge regardless of its direction and whether it is directional.
label_list	The new label. See 'Details' on how to set this argument. If a list of named list is used, the element for new label should be named new_label.
label_bg_list	The new background color. See 'Details' on how to set this argument. If a list of named list is used, the element for new background color should be named new_label_bg.
label_size_list	How the label sizes are to be changed. See 'Details' on how to set this argument. If a list of named list is used, the element for new background color should be named new_label_size.
how	How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value.

Details

Modify a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the attributes of the edge label of selected edges.

Currently, the following attributes are supported:

- Edge label position.
- Edge label content.
- Edge label background color.
- Edge label size.

How to specify the values:

There are three approach to specify the value.

First, the values can be a named vector of the values, the recommended approach. The name of an element should be the path as specified by `lavaan::model.syntax` or as appeared in `lavaan::parameterEstimates()`. For example, to change position of the edge label of the path regressing y on x, the name should be "y ~ x".

Second, the values can be specified by a list of named lists, each named list should have three named values: from, to, and the new value (name depends on the attribute). The value of the edge label of the edge from from to to will be set to this new value. For example, for `set_edge_label_position()`, `list(list(from = "x1", to = "y", new_position = .2), list(from = "x2", to = "y", new_position = .7))` is equivalent to the named vector above.

The second approach is no longer recommended. It was supported for backward compatibility.

Third, the value can be a single value, which will be used to set the attribute of all edges.

Setting Edge Position:

When setting the edge position, the value is the position. The mid-point of the edge is 0.5. The closer the value to 1, the closer the label to the left-hand-side node (y in this example). The closer the value to 0, the closer the label to the right-hand-side node (x in this example). For example, `c("y ~ x1" = .2, "y ~ x2" = .7)` moves the path coefficient from x1 to y closer to x, and the path coefficient from x2 to y closer to y.

Setting Label Size:

The function `set_edge_label_size()` works in two modes. With `how = "ratio"`, the new label size is the original size multiplied by the supplied value. For example, if the value is 2, the size of a label is doubled. With `how = "value"`, the new size is set to be equal to the supplied value. For example, if the value is 2, the size of a label is set to 2.

label positions for selected edges changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
    x3 ~ x1 + x2
    x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

# ==== set_edge_label_position ====

my_position_vector <- c(
  "x3 ~ x2" = .25,
  "x4 ~ x1" = .75
)
p_pa2v <- set_edge_label_position(
  p_pa,
  my_position_vector
)
plot(p_pa2v)

# This approach is no longer recommended
my_position_list <- list(
  list(
    from = "x2",
    to = "x3",
    new_position = .25
  ),
  list(
    from = "x1",
```

```

                                to = "x4",
                                new_position = .75)
                                )
p_pa2l <- set_edge_label_position(
    p_pa,
    my_position_list
)
plot(p_pa2l)

# ==== set_edge_label ====

my_label_vector <- c(
    "x3 ~ x2" = "a",
    "x4 ~ x1" = "b"
)
p_pa3v <- set_edge_label(
    p_pa,
    my_label_vector
)
plot(p_pa3v)

# This approach is no longer recommended
my_label_list <- list(
    list(
        from = "x2",
        to = "x3",
        new_label = "a"
    ),
    list(
        from = "x1",
        to = "x4",
        new_position = "a"
    )
)
p_pa3l <- set_edge_label(
    p_pa,
    my_label_list
)
plot(p_pa3l)

# ==== set_edge_label_bg ====

my_label_bg_vector <- c(
    "x3 ~ x2" = "red",
    "x4 ~ x1" = "#00FF00"
)
p_pa4v <- set_edge_label_bg(
    p_pa,
    my_label_bg_vector
)
plot(p_pa4v)

```

```

# This approach is no longer recommended
my_label_bg_list <- list(
  list(
    from = "x2",
    to = "x3",
    new_label_bg = "red"
  ),
  list(
    from = "x1",
    to = "x4",
    new_label_bg = "#00FF00")
)
p_pa4l <- set_edge_label_bg(
  p_pa,
  my_label_bg_list
)
plot(p_pa4l)

# ==== set_edge_label_size ====

my_label_size_vector <- c(
  "x3 ~ x2" = 2,
  "x4 ~ x1" = 3
)
p_pa5v <- set_edge_label_size(
  p_pa,
  my_label_size_vector
)
plot(p_pa5v)

# This approach is no longer recommended
my_label_size_list <- list(
  list(
    from = "x2",
    to = "x3",
    new_label_size = 2
  ),
  list(
    from = "x1",
    to = "x4",
    new_label_size = 3)
)
p_pa5l <- set_edge_label_size(
  p_pa,
  my_label_size_list
)
plot(p_pa5l)

```

set_edge_line_type	<i>Set the Line Types of Selected Edges</i>
--------------------	---

Description

Set the line types of selected edges.

Usage

```
set_edge_line_type(
  semPaths_plot,
  line_type_list = NULL,
  check_direction = TRUE
)
```

Arguments

- semPaths_plot** A [qgraph::qgraph](#) object generated by [semPlot::semPaths](#), or a similar qgraph object modified by other [semptools](#) functions. It can also be a list of [qgraph::qgraph](#) objects, probably though not necessarily from a multigroup model. If it is a list of [qgraph::qgraph](#) objects, then the function will be applied to all the objects.
- line_type_list** A named vector or a list of named list. See the Details section on how to set this argument.
- check_direction** If FALSE, the direction of an edge is ignored. For example, both $y \sim x$ and $x \sim y$ will affect $y \sim x$, $x \sim y$, and $y \sim x$. Useful when we want to change an edge regardless of its direction and whether it is directional.

Details

Modified a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#) and change the widths of selected edges.

Setting the value of line_type_list:

This argument can be set in three ways.

For a named vector, the name of an element should be the path as specified by [lavaan::model.syntax](#) or as appeared in [lavaan::parameterEstimates\(\)](#).

For example, to change the line type of the path regressing y on x, the name should be "y ~ x". To change the line type of the covariance between x1 and x2, the name should be "x1 ~~ x2". Therefore, `c("y ~ x1" = "dotted", "x1 ~~ x2" = "dashed")` changes the line types of the path from x1 to y and the covariance between x1 and x2 to "dotted" and "dashed", respectively.

The order of the two nodes *may* matter for covariances. Therefore, if the attribute of a covariance is not changed, try switching the order of the two nodes.

For a list of named lists, each named list should have three named values: from, to, and new_line_type. The attribute of the edge from from to to will be set to new_line_type.

The second approach is no longer recommended, though kept for backward compatibility.

The third approach is one single *unnamed* value. This value will be applied to all edges.

Supported Line Type:

The supported line types are the same line types supported by the `lty` argument of `par()`: "blank" (0), "solid" (1), "dashed" (2), "dotted" (3), "dotdash" (4), "longdash" (5), and "twodash" (6). The new line type can be specified in either strings or integers, but not both.

Value

A `qgraph::qgraph` based on the original one, with widths for selected edges changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
    x3 ~ x1 + x2
    x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

my_line_type_vector <- c(
  "x2 ~~ x1" = "dotted",
  "x4 ~ x1" = "dashed"
)

p_pa2v <- set_edge_line_type(
  p_pa,
  my_line_type_vector)
plot(p_pa2v)

my_line_type_list <- list(
  list(
    from = "x1",
    to = "x2",
    new_line_type = "dotted"),
  list(
    from = "x1",
    to = "x4",
    new_line_type = "dashed")
)

p_pa2l <- set_edge_line_type(
  p_pa,
  my_line_type_list
)
```

```
plot(p_pa2l)
```

set_edge_line_width *Set the Line Widths of Selected Edges*

Description

Set the line widths of selected edges.

Usage

```
set_edge_line_width(
  semPaths_plot,
  line_width_list = NULL,
  how = c("ratio", "value"),
  check_direction = TRUE
)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. It can also be a list of qgraph::qgraph objects, probably though not necessarily from a multigroup model. If it is a list of qgraph::qgraph objects, then the function will be applied to all the objects.
line_width_list	A named vector or a list of named list. See the Details section on how to set this argument.
how	How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value.
check_direction	If FALSE, the direction of an edge is ignored. For example, both $y \sim x$ and $x \sim y$ will affect $y \sim x$, $x \sim y$, and $y \sim x$. Useful when we want to change an edge regardless of its direction and whether it is directional.

Details

Modified a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#) and change the widths of selected edges.

Setting the value of line_width_list:

This argument can be set in three ways.

For a named vector, the name of an element should be the path as specified by [lavaan::model.syntax](#) or as appeared in [lavaan::parameterEstimates\(\)](#).

For example, to change the line width of the path regressing y on x, the name should be "y ~ x". To change the line type of the covariance between x1 and x2, the name should be "x1 ~~ x2".

Therefore, `c("y ~ x1" = 2, "x1 ~~ x2" = 3)` changes the line widths of the path from `x1` to `y` and the covariance between `x1` and `x2` by 2 and 3, respectively.

The order of the two nodes *may* matter for covariances. Therefore, if the attribute of a covariance is not changed, try switching the order of the two nodes.

For a list of named lists, each named list should have three named values: `from`, `to`, and `new_line_width`. The attribute of the edge from `from` to `to` will be set to `new_line_width`.

The second approach is no longer recommended, though kept for backward compatibility.

The third approach is one single *unnamed* value. This value will be applied to all edges.

Value

A `qgraph::qgraph` based on the original one, with widths for selected edges changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

my_line_width_vector <- c(
  "x2 ~~ x1" = 2,
  "x4 ~ x1" = 3
)

p_pa2v <- set_edge_line_width(
  p_pa,
  my_line_width_vector)
plot(p_pa2v)

my_line_width_list <- list(
  list(
    from = "x1",
    to = "x2",
    new_line_width = 2),
  list(
    from = "x1",
    to = "x4",
    new_line_width = 3)
)
```

```
p_pa2l <- set_edge_line_width(
  p_pa,
  my_line_width_list
)
plot(p_pa2l)
```

set_graph_attributes *Set Selected Attributes of a Plot*

Description

Set some attributes of a plot.

Usage

```
set_graph_margins(
  semPaths_plot,
  bottom,
  left,
  top,
  right,
  how = c("ratio", "value")
)

node_labels_equal_scale(semPaths_plot, equal_scale)

set_node_labels_equal_scale(semPaths_plot)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. It can also be a list of qgraph::qgraph objects, probably though not necessarily from a multigroup model. If it is a list of qgraph::qgraph objects, then the function will be applied to all the objects.
bottom, left, right, top	The values for the corresponding sides. Please refer to 'Details' on how these values are used. Their default values depend on how: Equal to 1 if how = "ratio", and equal to the original values if how = "value".
how	How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value.
equal_scale	Logical. Set the value label.scale.equal of the plot. If not supplied, it will not change the existing value.

Details

Functions that modify a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the widths of selected attributes.

For numerical attributes, there are two possible modes.

If `how = "ratio"`, then the new value is equal to the old value multiplied by the user value. For example, if the original margins are `c(2, 3, 4, 5)` and the user value is 2, then the new margins are `c(4, 6, 8, 10)`. If the user values are `c(3, 1, 1, 1)`, then the new margins are `c(6, 3, 4, 5)`.

If `how = "value"`, then the new value is equal to the user value. For example, if the original margins are `c(2, 3, 4, 5)` and the user value is 2, then the new margins are `c(2, 2, 2, 2)`. If the user values are `c(1, 3, 5, 7)`, then the new margins are `c(1, 3, 5, 7)`.

The function `set_graph_margins()` is for modifying the margins of a generated plot. It can be useful for increasing or decreasing one or more margins of a plot without regenerating it.

The function `node_labels_equal_scale()` is for setting the value `label.scale.equal` of a plot. By default, `label.scale.equal` may be `FALSE`, and the labels of nodes may be of different font sizes. Use this function, with `equal_scale` set to `TRUE`, will make all node labels have the same font size.

The function `set_node_labels_equal_scale()` is just a wrapper of `node_labels_equal_scale()` with `equal_scale` set to `TRUE`.

Examples

```
mod_pa <-
  'x1 ~~ x2
    x3 ~ x1 + x2
    x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)

# ==== set_graph_margins ====

p2 <- set_graph_margins(p_pa, bottom = 2)
plot(p2)

p3 <- set_graph_margins(p_pa, bottom = 2, how = "value")
plot(p3)

# ==== node_labels_equal_scale ====
```

```

dat <- pa_example
colnames(dat)[1] <- "Mediator"
mod_pa2 <-
  'x1 ~~ x2
    Mediator ~ x1 + x2
    x4 ~ x1 + Mediator
  '
fit_pa2 <- lavaan::sem(mod_pa2, dat)
m2 <- matrix(c("x1", NA, NA,
               NA, "Mediator", "x4",
               "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa2 <- semPlot::semPaths(fit_pa2, whatLabels="est",
                           style = "ram",
                           nCharNodes = 0, nCharEdges = 0,
                           sizeMan = 10,
                           layout = m2)

p4 <- node_labels_equal_scale(p_pa2, TRUE)
plot(p4)

p5 <- set_node_labels_equal_scale(p_pa2)
plot(p5)

```

set_node_attribute	<i>Set the Attributes of Selected Nodes</i>
--------------------	---

Description

Set arbitrary attributes of selected nodes

Usage

```

set_node_attribute(
  semPaths_plot,
  values = NULL,
  attribute_name = NULL,
  how = c("value", "ratio"),
  check_nodes = TRUE
)

```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. It can also be a list of qgraph::qgraph objects, probably though not necessarily from a multigroup model. If it is a list of qgraph::qgraph objects, then the function will be applied to all the objects.
values	A named vector or a list of named list. See the Details section on how to set this argument.

attribute_name	The name of the attribute to be changed.
how	How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value.
check_nodes	Logical. If TRUE and at least one node specified in values are not in semPaths_plot.

Details

Modify a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the selected attributes of selected nodes.

This function is designed to be a general one that changes the attributes named by the user. The user needs to make sure that the attribute actually exists, and the values are valid for the named attribute.

Setting the value of values:

This argument can be set in three ways.

For a named vector, the name of an element should be the nodes for which their attributes are to be changed. The names need to be the *displayed names* if plotted, which may be different from the names in mode.

For example, if the attributes to be changed are the colors of selected nodes, to change the color of x is to be changed, the name should be "x". Therefore, `c("y" = "red", "x" = "red")` changes the colors of the nodes y and x to "red" and "blue", respectively.

For a list of named lists, each named list should have two named values: `node` and `new_value`. The attribute of node will be set to `new_value`.

The second approach is no longer recommended, though kept for backward compatibility.

The last approach is setting values to a one-element vector with *no name*. All nodes in plot will then have the selected attributes set to this value.

Value

A `qgraph::qgraph` based on the original one, with the selected attributes of selected nodes changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '
fit_pa <- lavaan::sem(mod_pa, pa_example)
lavaan::parameterEstimates(fit_pa)[, c("lhs", "op", "rhs", "est", "pvalue")]
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)
```

```

my_color_vector <- c(x3 = "red", x4 = "blue")

p_pa2v <- set_node_attribute(p_pa, my_color_vector, attribute_name = "color")
plot(p_pa2v)

my_color_list <- list(list(node = "x3", new_value = "green"),
                      list(node = "x4", new_value = "red"))

p_pa2l <- set_node_attribute(p_pa, my_color_list, attribute_name = "color")
plot(p_pa2l)

```

set_node_border	<i>Set the Attributes of Node Borders</i>
-----------------	---

Description

Set the attributes of the borders of selected nodes.

Usage

```

set_node_border_color(semPaths_plot, border_color = NULL, check_nodes = TRUE)

set_node_border_width(
  semPaths_plot,
  border_width = NULL,
  how = c("ratio", "value"),
  check_nodes = TRUE
)

```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. It can also be a list of qgraph::qgraph objects, probably though not necessarily from a multigroup model. If it is a list of qgraph::qgraph objects, then the function will be applied to all the objects.
border_color	A named vector or a list of named list. See the Details section on how to set this argument.
check_nodes	Logical. If TRUE and at least one node specified in values are not in semPaths_plot.
border_width	A named vector or a list of named list. See the Details section on how to set this argument.
how	How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value.

Details

Modify a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the attributes of the borders of selected nodes.

Setting the value of `border_color`:

and `border_width`

This argument can be set in three ways.

For a named vector, the name of an element should be the nodes for which the attributes are to be changed. The names need to be the *displayed names* if plotted, which may be different from the names in the model.

For example, to change the color of `x`, the name should be `"x"`. Therefore, `c("y" = "red", "x" = "blue")` changes the colors of the nodes `y` and `x` to `"red"` and `"blue"`, respectively.

For a list of named lists, each named list should have two named values: `node`, plus `new_border_color` or `new_border_width`.

The second approach is no longer recommended, though kept for backward compatibility.

The last approach is setting `border_color` or `border_width` to a one-element vector with *no name*. All nodes in plot will then have the corresponding attributes set to this value.

Note on Border Width:

There are also two modes for changing border width. If `how = "ratio"`, then the new border width is equal to the original border width multiplied by the supplied value. For example, if the supplied value is 2, and the original border width is 1.5, the new border width is 3.

If `how = "value"`, then the new border width is set to the user supplied value. For example, if the supplied value is 2, then the new border width is 2, regardless of the original border width.

Value

A `qgraph::qgraph` based on the original one, with the corresponding attribute of of selected nodes changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

my_border_color_vector <- c(x3 = "red", x4 = "blue")
```

```

p_pa2v <- set_node_border_color(p_pa, my_border_color_vector)
plot(p_pa2v)

p_pa2l <- set_node_border_color(p_pa, "green")
plot(p_pa2l)

my_border_width_vector <- c(x3 = 2, x4 = 3)

p_pa3v <- set_node_border_width(p_pa, my_border_width_vector)
plot(p_pa3v)

p_pa3l <- set_node_border_width(p_pa, 3)
plot(p_pa3l)

```

set_node_color	<i>Set the Colors of Selected Nodes</i>
----------------	---

Description

Set the colors of selected nodes.

Usage

```
set_node_color(semPaths_plot, colors = NULL, check_nodes = TRUE)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semptools functions. It can also be a list of qgraph::qgraph objects, probably though not necessarily from a multigroup model. If it is a list of qgraph::qgraph objects, then the function will be applied to all the objects.
colors	A named vector or a list of named list. See the Details section on how to set this argument.
check_nodes	Logical. If TRUE and at least one node specified in values are not in semPaths_plot.

Details

Modify a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#) and change the colors of selected nodes.

Setting the value of color_list:

This argument can be set in three ways.

For a named vector, the name of an element should be the nodes for which the colors are to be changed. The names need to be the *displayed names* if plotted, which may be different from the names in the model.

For example, to change the color of x, the name should be "x". Therefore, `c("y" = "red", "x" = "red")` changes the colors of the nodes y and x to "red" and "blue", respectively.

For a list of named lists, each named list should have two named values: `node` and `new_color`. The color of node will be set to `new_color`.

The second approach is no longer recommended, though kept for backward compatibility.

The last approach is setting `color_list` to a one-element vector with *no name*. All nodes in plot will then have the colors set to this value.

Value

A `qgraph::qgraph` based on the original one, with the colors of selected nodes changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
    x3 ~ x1 + x2
    x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

my_color_vector <- c(x3 = "red", x4 = "blue")

p_pa2v <- set_node_color(p_pa, my_color_vector)
plot(p_pa2v)

p_pa2l <- set_node_color(p_pa, "green")
plot(p_pa2l)
```

set_node_label	<i>Set Label Attributes of Selected Nodes</i>
----------------	---

Description

Set the attributes of labels of selected nodes.

Usage

```
set_node_label_size(semPaths_plot, values = NULL)
```

```
set_node_label_color(semPaths_plot, values = NULL)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths , or a similar qgraph object modified by other semtools functions. It can also be a list of qgraph::qgraph objects, probably though not necessarily from a multigroup model. If it is a list of qgraph::qgraph objects, then the function will be applied to all the objects.
values	A named vector or a list of named list. See the Details section on how to set this argument.

Details

Modify a [qgraph::qgraph](#) object generated by [semPlot::semPaths](#) and change the label attributes of selected nodes.

Attributes Supported:

The function [set_node_label_size\(\)](#) set the attribute `label.cex` of the plot, corresponding to the argument `label.cex` of [semPlot::semPaths\(\)](#). It controls the size of the label of a node.

The function [set_node_label_color\(\)](#) set the attribute `label.color` of the plot, corresponding to the argument `label.color` of [semPlot::semPaths\(\)](#). It controls the color of the label of a node.

Setting the Label-Related Attributes:

These arguments can be set in three ways.

For a named vector, the name of an element should be the nodes for which the attributes are to be changed. The names need to be the *displayed names* if plotted, which may be different from the names in the model.

For example, to change the label-related attribute of `x`, the name should be `"x"`. Therefore, `c("y" = 2, "x" = 3)` changes the selected attributes of the nodes `y` and `x` to 2 and 3, respectively.

For a list of named lists, each named list should have two named values: `node` and `new_value`. The size-related attribute of `node` will be changed based on `new_value`.

The second approach is no longer recommended, though kept for backward compatibility.

The last approach is setting the size-related attribute to a one-element vector with *no name*. All nodes in plot will then have their sizes changed based on this value.

Value

A [qgraph::qgraph](#) based on the original one, with the sizes of selected nodes changed. If `semPaths_plot` is a list of [qgraph::qgraph](#) objects, then a list of processed [qgraph::qgraph](#) objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '
fit_pa <- lavaan::sem(mod_pa, pa_example)
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
```

```

      "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
  style = "ram",
  nCharNodes = 0, nCharEdges = 0,
  layout = m)

p_pa2v <- set_node_label_size(p_pa, c(x1 = 5, x2 = 10))
plot(p_pa2v)

p_pa2l <- set_node_label_color(p_pa, "red")
plot(p_pa2l)

```

set_node_size	<i>Set the Sizes of Selected Nodes</i>
---------------	--

Description

Set the sizes of selected nodes.

Usage

```

set_node_size(
  semPaths_plot,
  values = NULL,
  how = c("ratio", "value"),
  check_nodes = TRUE
)

set_node_width(
  semPaths_plot,
  values = NULL,
  how = c("ratio", "value"),
  check_nodes = TRUE
)

set_node_height(
  semPaths_plot,
  values = NULL,
  how = c("ratio", "value"),
  check_nodes = TRUE
)

set_node_shape(semPaths_plot, values = NULL, check_nodes = TRUE)

```

Arguments

semPaths_plot	A <code>qgraph::qgraph</code> object generated by <code>semPlot::semPaths</code> , or a similar <code>qgraph</code> object modified by other <code>semtools</code> functions. It can also be a list of <code>qgraph::qgraph</code> objects, probably though not necessarily from a multigroup model. If it is a list of <code>qgraph::qgraph</code> objects, then the function will be applied to all the objects.
values	A named vector or a list of named list. See the Details section on how to set this argument.
how	How the width will be changed. If "ratio", then the new width is the original width multiplied by the supplied value. If "value", then the new width is set to the supplied value.
check_nodes	Logical. If TRUE and at least one node specified in values are not in semPaths_plot.

Details

Modify a `qgraph::qgraph` object generated by `semPlot::semPaths` and change the sizes of selected nodes.

Setting the Size-Related Attributes:

These arguments can be set in three ways.

For a named vector, the name of an element should be the nodes for which the attributes are to be changed. The names need to be the *displayed names* if plotted, which may be different from the names in the model.

For example, to change the size-related attribute of x, the name should be "x". Therefore, `c("y" = 2, "x" = 3)` changes the selected attributes of the nodes y and x to 2 and 3, respectively.

For a list of named lists, each named list should have two named values: `node` and `new_value`. The size-related attribute of node will be changed based on `new_value`.

The second approach is no longer recommended, though kept for backward compatibility.

The last approach is setting the size-related attribute to a one-element vector with *no name*. All nodes in plot will then have their sizes changed based on this value.

Shape, Width, and Height:

How width and height are used depends on shape. The supported values of shape depends on `qgraph`. Common values are "square", "rectangle", "circle", "rectangle", and "ellipse". For "square" and "circle", only the value of width is used to determine the size of the node.

Note on Changing a Value:

There are also two modes for changing an attribute. If `how = "ratio"`, then the new value, such as size, is equal to the original value multiplied by the supplied value. For example, if the supplied value is 2, and the original size is 5, the new size is 10.

If `how = "value"`, then the new value is set to the user supplied value. For example, if the supplied value is 15, then the new value is 15, regardless of the original value.

Value

A `qgraph::qgraph` based on the original one, with the sizes of selected nodes changed. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```
mod_pa <-
  'x1 ~~ x2
  x3 ~ x1 + x2
  x4 ~ x1 + x3
  '

fit_pa <- lavaan::sem(mod_pa, pa_example)
m <- matrix(c("x1", NA, NA,
              NA, "x3", "x4",
              "x2", NA, NA), byrow = TRUE, 3, 3)
p_pa <- semPlot::semPaths(fit_pa, whatLabels="est",
                          style = "ram",
                          nCharNodes = 0, nCharEdges = 0,
                          layout = m)

p_pa2v <- set_node_size(
  p_pa, c(x1 = 5, x2 = 10),
  how = "value"
)
plot(p_pa2v)

p_pa2v2 <- set_node_size(
  p_pa, c(x1 = 0.5, x2 = 2),
  how = "ratio"
)
plot(p_pa2v2)

p_pa2l <- set_node_shape(p_pa, "circle")
plot(p_pa2l)
```

set_sem_layout

Configure the layout of factors of an SEM graph by [semPlot::semPaths](#)

Description

Configure the layout of factors and adjust other aspects of an SEM graph by [semPlot::semPaths](#).

Usage

```
set_sem_layout(
  semPaths_plot,
  indicator_order = NULL,
  indicator_factor = NULL,
  factor_layout = NULL,
  factor_point_to = NULL,
  indicator_push = NULL,
  indicator_spread = NULL,
  loading_position = 0.5
)
```

Arguments

- semPaths_plot** A `qgraph::qgraph` object generated by `semPaths`, or a similar `qgraph` object modified by other `semptools` functions. It can also be a list of `qgraph::qgraph` objects, probably though not necessarily from a multigroup model. If it is a list of `qgraph::qgraph` objects, then the function will be applied to all the objects.
- indicator_order** A string vector of the indicators. The order of the names is the order of the indicators in the graph, when they are drawn on the bottom of the graph. The indicators should be grouped by the factors on which they load on. For example, if x1, x2, x4 load on f2, and x3, x5, x6 load on f1, then vector should be either `c("x1", "x2", "x4", "x3", "x5", "x6")` or `c("x3", "x5", "x6", "x1", "x2", "x4")`. Indicators within a group can be ordered in any way. If it is a named vector, its names will be used for the argument `indicator_factor`. If it is `NULL` (default), `auto_indicator_order()` will be called to determine the indicator order automatically.
- indicator_factor** A string vector of the same length of the indicator order, storing the name of the factor for which each of the indicator in `indicator_order` loads on. For example, if x1, x2, x4 load on f2, and x3, x5, x6 load on f1, and `indicator_order` is `c("x3", "x5", "x6", "x1", "x2", "x4")`, then `indicator_factor` should be `c("f2", "f2", "f2", "f1", "f1", "f1")`. If `NULL` (default) and `indicator_order` is a named vector (supplied by users or generated by `auto_indicator_order()`), then it will be set to the names of `indicator_order`.
- factor_layout** A matrix of arbitrary size. This matrix will serve as a grid for users to specify where each latent factor should be placed approximately on the graph. Each cell should contain `NA` or the name of a latent factor. The locations of all latent factors must be explicitly specified by this matrix.
- factor_point_to** Can be a named character vector with names being the names of factors, or a matrix of the same size as `factor_layout`. If it is a matrix, this matrix specifies where the indicators of each factor are positioned. Each cell should contain `NA` or one of these strings: "down", "left", "up", or "right". This is the direction that the corresponding latent factor (specified in `factor_layout`) points to its indicators. If it is a named character vector, the values must be the directions, and the names the factors. This vector will be converted internally by `auto_factor_point_to()` to create the matrix of direction.
- indicator_push** (Optional) This argument is used to adjust the positions of the indicators of selected latent factors. It can be named vector or a list of named lists. For a named vector, The name is the factor of which the indicators will be "pushed", and the value is how "hard" the push is: the multiplier to the distance from the factor to the indicators. If this value is 1, then there is no change. If this value is greater than 1, then the indicators are pushed away from the latent factor. If this value is less than 1, then the indicators are pulled toward the latent factor. For example, to push the indicators of f3 away from f3, and pull the indicators of f4 toward f4, the argument can be set to `c(f3 = 1.5, f4 = .5)`. For a list of named list, each named list has two named elements: `node`, the name of a latent factor, and `push`, how the positions of its indicators will be adjusted. For example, to

have the same effect as the vector above, the list is `list(list(node = "f3", push = 1.5), list(node = "f4", push = .5))`.

indicator_spread

(Optional) This argument is used to adjust the distance between indicators of selected latent factors. It can be a named vector or a list of named lists. For a named vector, the name is the factor of which the indicators will be spread out. The value is the multiplier to the distance between neighboring indicators. If this value is equal to 1, there is no change. Larger than one, the indicators will be "spread" away from each other. Less than one, the indicators will be placed closer to each others. For example, to spread the indicators of f1 and f4 farther away from each other, this argument can be set to `c(f1 = 2, f4 = 1.5)`, with the indicators of f1 being spread out more than those of f4. For a list of named list, each named list has two named elements: `node`, the name of a latent factor, and `spread`, how the distance between indicators will be adjusted. For example, to have the same effect as the vector above, the argument can be set to `list(list(node = "f1", spread = 2), list(node = "f4", spread = 1.5))`.

loading_position

(Optional) Default is .5. This is used adjust the position of the loadings. If this is one single number, it will be used to set the positions of all loadings. If it is .5, the loadings are placed on the center of the arrows. Larger the number, closer the loadings to the indicators. Smaller the number, closer to the latent factors. This argument also accepts a named vector or a list of named lists, allowing users to specify the positions of loadings for each factor separately. For a named vector, in each element, the name is the factor whose loadings will be moved. The value is the positions of its loadings. The default is .50. We only need to specify the positions for factors to be changed from .50 to other values. For example, move the loadings of f2 closer to the indicators and those of f4 close to the f4, this argument can be set to `c(f2 = .7, f4 = .3)`. For a list of named list, each named list should have two named elements: `node`, the name of the latent factor, and `position`, the positions of all loadings of this factors. To have the same effect as the vector above, this list can be used: `list(list(node = "f2", position = .7), list(node = "f4", position = .3))`.

Details

Modify a `qgraph::qgraph` object generated by `semPaths` based on an SEM model with latent factors. Since version 0.2.9.5, this function natively supports observed exogenous variable. If a variable is listed in both `indicator_order` and `indicator_factor`, as if it is both a factor and an indicator, this function will assume that it is an observed exogenous variable. It will be positioned as a factor according to `factor_layout`, but no indicators will be drawn.

For versions older than 0.2.9.5, an observed exogenous variable needs to be specified as an one-indicator factor in the model specification for this function to work.

Value

A `qgraph::qgraph` based on the original one, with various aspects of the model modified. If `semPaths_plot` is a list of `qgraph::qgraph` objects, then a list of processed `qgraph::qgraph` objects will be returned.

Examples

```

library(lavaan)
library(semPlot)
mod <-
'f1 =~ x01 + x02 + x03
f2 =~ x04 + x05 + x06 + x07
f3 =~ x08 + x09 + x10
f4 =~ x11 + x12 + x13 + x14
f3 ~ f1 + f2
f4 ~ f1 + f3
'

fit_sem <- lavaan::sem(mod, sem_example)
lavaan::parameterEstimates(fit_sem)[, c("lhs", "op", "rhs", "est", "pvalue")]
p <- semPaths(fit_sem, whatLabels="est",
  sizeMan = 5,
  nCharNodes = 0, nCharEdges = 0,
  edge.width = 0.8, node.width = 0.7,
  edge.label.cex = 0.6,
  style = "ram",
  mar = c(10,10,10,10))
indicator_order <- c("x04", "x05", "x06", "x07", "x01", "x02", "x03",
  "x11", "x12", "x13", "x14", "x08", "x09", "x10")
indicator_factor <- c("f2", "f2", "f2", "f2", "f1", "f1", "f1",
  "f4", "f4", "f4", "f4", "f3", "f3", "f3")
factor_layout <- matrix(c("f1", NA, NA,
  NA, "f3", "f4",
  "f2", NA, NA), byrow = TRUE, 3, 3)
factor_point_to <- matrix(c("left", NA, NA,
  NA, "down", "down",
  "left", NA, NA), byrow = TRUE, 3, 3)
indicator_push <- c(f3 = 2, f4 = 1.5)
indicator_spread <- c(f1 = 2, f2 = 2)
loading_position <- c(f1 = .5, f2 = .8, f3 = .8)
# Pipe operator can be used if desired
p2 <- set_sem_layout(p,
  indicator_order = indicator_order,
  indicator_factor = indicator_factor,
  factor_layout = factor_layout,
  factor_point_to = factor_point_to,
  indicator_push = indicator_push,
  indicator_spread = indicator_spread,
  loading_position = loading_position)
p2 <- set_curve(p2, c("f2 ~ f1" = -1,
  "f4 ~ f1" = 1.5))
p2 <- mark_sig(p2, fit_sem)
p2 <- mark_se(p2, fit_sem, sep = "\n")
plot(p2)

# Use a named vector for indicator_order
indicator_order2 <- c(f2 = "x04", f2 = "x05", f2 = "x06", f2 = "x07",
  f1 = "x01", f1 = "x02", f1 = "x03",
  f4 = "x11", f4 = "x12", f4 = "x13", f4 = "x14",

```

```

      f3 = "x08", f3 = "x09", f3 = "x10")
p2 <- set_sem_layout(p,
  indicator_order = indicator_order2,
  factor_layout = factor_layout,
  factor_point_to = factor_point_to,
  indicator_push = indicator_push,
  indicator_spread = indicator_spread,
  loading_position = loading_position)

plot(p2)

# Use automatically generated indicator_order and indicator_factor
p2 <- set_sem_layout(p,
  factor_layout = factor_layout,
  factor_point_to = factor_point_to,
  indicator_push = indicator_push,
  indicator_spread = indicator_spread,
  loading_position = loading_position)

plot(p2)

# Use named character vector for factor_point_to
directions <- c(f1 = "left",
  f2 = "left",
  f3 = "down",
  f4 = "down")
p2v2 <- set_sem_layout(p,
  indicator_order = indicator_order,
  indicator_factor = indicator_factor,
  factor_layout = factor_layout,
  factor_point_to = directions,
  indicator_push = indicator_push,
  indicator_spread = indicator_spread,
  loading_position = loading_position)
p2v2 <- set_curve(p2v2, c("f2 ~ f1" = -1,
  "f4 ~ f1" = 1.5))
p2v2 <- mark_sig(p2v2, fit_sem)
p2v2 <- mark_se(p2v2, fit_sem, sep = "\n")
plot(p2v2)

#Lists of named list which are equivalent to the vectors above:
#indicator_push <- list(list(node = "f3", push = 2),
#  list(node = "f4", push = 1.5))
#indicator_spread <- list(list(node = "f1", spread = 2),
#  list(node = "f2", spread = 2))
#loading_position <- list(list(node = "f1", position = .5),
#  list(node = "f2", position = .8),
#  list(node = "f3", position = .8))

```

Description

Convert a named vector to a list of lists, to be used by various functions in [semptools](#).

Usage

```
to_list_of_lists(input, name1 = NULL, name2 = NULL, name3 = NULL)
```

Arguments

input	A named vector
name1	The name for the first element in the list-in-list. Default is NULL.
name2	The name for the second element in the list-in-list. Default is NULL.
name3	The name for the third element in the list-in-list. Default is NULL. If this argument is not NULL, the names of the vector elements will be split using lavaan syntax (by calling lavaan::lavParseModelString()), and the right-hand side (rhs) and left-hand side (lhs) of each element will be assigned to name1 and name2, respectively.

Details

This function is not to be used by users, but to be used internally by other functions of [semptools](#).

Value

A list of lists.

Examples

```
x <- c("x1 ~~ x2" = -1, "x4 ~ x1" = 1)
to_list_of_lists(x, name1 = "from", name2 = "to", name3 = "new_curve")
#list(list(from = "x1", to = "x2", new_curve = -1),
#      list(from = "x1", to = "x4", new_curve = 1))

y <- c(x1 = 0, x2 = 180, x3 = 140, x4 = 140)
to_list_of_lists(y, name1 = "node", name2 = "rotate")
#list(list(node = "x1", rotate = 0),
#      list(node = "x2", rotate = 180),
#      list(node = "x3", rotate = 140),
#      list(node = "x4", rotate = 140))
```

Index

* **datasets**
 cfa_example, 12
 pa_example, 28
 pa_example_3covs, 28
 sem_2nd_order_example, 44
 sem_example, 45

add_object, 3
add_rsqr, 4
add_rsqr(), 33, 34
auto_factor_point_to, 6
auto_factor_point_to(), 75
auto_indicator_order, 7
auto_indicator_order(), 46, 47, 75
auto_layout_mediation, 8
auto_layout_mediation(), 10

cfa_example, 12
change_node_label, 12

drop_nodes (keep_drop_nodes), 18
drop_nodes(), 19

get_edge_attribute, 14
get_node_attribute, 16

is_dv_residvar, 17

keep_drop_nodes, 18
keep_nodes (keep_drop_nodes), 18
keep_nodes(), 19

lapply(), 14, 16
lavaan, 4, 22
lavaan::cfa(), 3
lavaan::lavaan, 17
lavaan::lavaan(), 9, 40, 43
lavaan::lavParseModelString(), 79
lavaan::model.syntax, 12, 15, 28, 29, 45, 46, 49, 51, 53, 55, 59, 61

lavaan::parameterEstimates(), 15, 49, 51, 53, 55, 59, 61
lavaan::sem(), 3, 9, 32, 40, 43
lavaan::standardizedSolution(), 22, 25
lavaan_indicator_order, 19
layout_matrix, 20
layout_matrix(), 6

mark_ci (mark_se), 21
mark_se, 21
mark_se(), 33, 34
mark_sig, 22, 24
mark_sig(), 33, 34
move_node, 26

node_labels_equal_scale
 (set_graph_attributes), 63
node_labels_equal_scale(), 64

pa_example, 28
pa_example_3covs, 28
par(), 60
parameterEstimates, 4, 22, 24

q_parallel (quick_sem_plot), 29
q_parallel(), 33
q_serial (quick_sem_plot), 29
q_serial(), 33
q_simple (quick_sem_plot), 29
q_simple(), 33
qgraph::qgraph, 3–5, 7, 10, 13, 14, 16, 21–27, 35, 37, 39–41, 43, 46, 47, 49, 51–56, 59–71, 73, 75, 76
qgraph::qgraph(), 15
quick_parallel_mediation
 (quick_sem_plot), 29
quick_parallel_mediation(), 33
quick_sem_plot, 29
quick_serial_mediation
 (quick_sem_plot), 29

quick_serial_mediation(), 33
 quick_simple_mediation
 (quick_sem_plot), 29
 quick_simple_mediation(), 33

 rescale_layout, 37
 rotate_resid, 38
 rotate_resid(), 43

 safe_edge_label_position, 40
 safe_resid_position, 42
 sem_2nd_order_example, 44
 sem_example, 45
 semPaths, 4, 22
 semPlot::semPaths, 13, 14, 16, 26, 27, 37,
 39, 49, 51–55, 59, 61, 63–69, 71, 73,
 74
 semPlot::semPaths(), 3, 7, 9, 13, 15, 19, 21,
 26, 32–35, 37, 40, 43, 71
 semPlot::semPlotModel, 18, 19
 semPlot::semPlotModel(), 18, 19
 semptools, 3, 4, 7, 13, 14, 16, 22, 24, 26, 37,
 39, 46, 49, 51, 52, 54, 59, 61, 63, 65,
 67, 69, 71, 73, 75, 79
 set_cfa_layout, 46
 set_cfa_layout(), 7, 8, 20
 set_curve, 48
 set_edge_attribute, 50
 set_edge_color, 52
 set_edge_label
 (set_edge_label_attributes), 54
 set_edge_label_attributes, 54
 set_edge_label_bg
 (set_edge_label_attributes), 54
 set_edge_label_position
 (set_edge_label_attributes), 54
 set_edge_label_position(), 41, 55
 set_edge_label_size
 (set_edge_label_attributes), 54
 set_edge_label_size(), 56
 set_edge_line_type, 59
 set_edge_line_width, 61
 set_graph_attributes, 63
 set_graph_margins
 (set_graph_attributes), 63
 set_graph_margins(), 64
 set_node_attribute, 65
 set_node_border, 67
 set_node_border_color
 (set_node_border), 67
 set_node_border_width
 (set_node_border), 67
 set_node_color, 69
 set_node_height(set_node_size), 72
 set_node_label, 70
 set_node_label_color(set_node_label),
 70
 set_node_label_color(), 71
 set_node_label_size(set_node_label), 70
 set_node_label_size(), 71
 set_node_labels_equal_scale
 (set_graph_attributes), 63
 set_node_labels_equal_scale(), 64
 set_node_shape(set_node_size), 72
 set_node_size, 72
 set_node_width(set_node_size), 72
 set_sem_layout, 74
 set_sem_layout(), 6–8, 10, 20, 37

 text(), 33
 to_list_of_lists, 78