Package 'semptools'

May 9, 2026

Title Customizing Structural Equation Modelling Plots

Version 0.3.3

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

RoxygenNote 7.3.3

VignetteBuilder knitr

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 2026-03-17 12:20:02 UTC

Contents

`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, 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`	Argument description.
`...`	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 lavaan::sem() or lavaan::lavaan(), 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 semPlot::semPaths(). A 'beta" matrix will be reconstructed from the graph.
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 whenv_pos is "middle". If v_pos is "lower", then v_preference will be forced to be "lower". If v_posis"upper", then v_preferencewill be forced to be"upper".
output	The format of the output, used if update_plot 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 relatived 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.

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.

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 qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
`label_list`	A list of named lists. Each named list should have two named values: `node` and `to`. The first part, `node`, is a character denoting the label to be changed. It should be as appeared in the qgraph. The second part, `to`, is the new label. Expression can be used in `to`. 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 `semPlot::semPaths()`. A number tha control the size of labels in the nodes. It has no default. If not set, then this option in the `semPaths_plot` will not be changed.

`label.scale` Identical to the same argument in semPlot::semPaths. A logical value that determine whether labels wil be scaled (resized) to the nodes they attach to. It has no default. If not set, then this option in the `semPaths_plot` will not be changed.

`label.prop` Identical to the same argument in semPlot::semPaths. A numeric vector of length equal to the number of nodes. If `label.scale` is `TRUE`, 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 `semPaths_plot` will not be changed.

`label.norm` Identical to the same argument in semPlot::semPaths. 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 `semPaths_plot` 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.

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)
```

`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 *Keep or drop nodes*

Description

Keep or drop nodes from an semPlotModel object.

Usage

```
drop_nodes(object, nodes)

keep_nodes(object, nodes)
```

Arguments

object	An an semPlot::semPlotModel generated by semPlot::semPlotModel().
nodes	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.

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

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.
sep	A character string to separate the coefficient and the standard error (in parentheses). Default to " " (one space). Use "\n" to enforce a line break.
digits	Integer indicating number of decimal places for the appended standard errors. Default is 2L.
ests	A data.frame from the parameterEstimates function, or from other function with these columns: lhs, op, rhs, se (for SE), and ci.lower and ci.upper (for CI). 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 lavaan::standardizedSolution() to compute the standard errors for the standardized solution. Used only if standard errors are not supplied directly through ests.

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.

Value

If the input is a qgraph::qgraph object, the function returns a qgraph based on the original one, with standard error or confidence interval estimates appended. 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 <- mark_se(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 <- mark_se(p_cfa, fit_cfa, sep = "\n")
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", "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")
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,
  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 lavaan::standardizedSolution() to compute the *p*-values for the standardized solution. Used only if *p*-values are not supplied directly through ests.
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 ests, if any. Used only when R-squares are present in the semPaths_plot.
r2_prefix	The prefix used to identify R-squares in semPaths_plot. 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)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
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.

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.

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 *Quick Plots of Common Models*

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_rsq = 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_rsq = 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_rsq = 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_rsq = 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_rsq = 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_rsq = TRUE,
  add_notes = FALSE,
  notes = NULL
)
```

Arguments

object	A lavaan object, such as the output of lavaan::sem().
x	The name of the x variable. Must have exactly one x variable.

`m` The name(s) of the `m` variable(s). The allowed number of `m` variable(s) depends of the model to be drawn.

`y` The name of the `y` variable. Must have exactly one `y` variable.

`mediators_position`
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"`.

`what` The same argument of `semPlot::semPaths()`. What the edges (arrows) denote. Default is `"path"`, the paths drawn with equal width.

`whatLabels` The same argument of `semPlot::semPaths()`. What the edge labels represent. Default is `"est"`, the parameter estimates. Can be set to `"std"` to print standardized coefficients.

`style` The same argument of `semPlot::semPaths()`. How residual variances are drawn. Can be `"lisrel"`, the default, or `"ram"`.

`nCharNodes` The same argument of `semPlot::semPaths()`. Default is 0, to disable abbreviation of the variable names.

`nCharEdges` The same argument of `semPlot::semPaths()`. Default is 0, to disable abbreviation of the edge labels.

`sizeMan` The same argument of `semPlot::semPaths()`. 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 `semPlot::semPaths()`. 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 `semPlot::semPaths()`. 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 `semPlot::semPaths()`, determining whether intercepts will be plotted. Default is `FALSE`, different from `semPlot::semPaths()`. 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 to `semPlot::semPaths()`.

`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 `mark_se()`. 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 `mark_sig()`.

`do_rotate_resid`
Logical. If `TRUE`, the default, Error variances, or R-squares if `do_add_rsq` 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 `add_rsq()`.

add_notes	Logical. If TRUE and plot_now is also TRUE, a note will be added by text(), 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 add_notes 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, withe 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 ~ x + c1 + c2

y ~ m + x + c1 + c2

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

m ~ x

y ~ 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 ~ x + c1 + c2

y1 ~ m + x + c1 + c2

y2 ~ 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 ~ x

y1 ~ m + x

and the other for y2:

m ~ x

y2 ~ 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 semptools 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.
`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.

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)
```

`rotate_resid` *Rotate the residuals of selected nodes*

Description

Rotate the residuals of selected nodes.

Usage

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

Arguments

`semPaths_plot`	A qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
`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)`.

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.

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 lavaan::sem() or lavaan::lavaan(), 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 semPlot::semPaths().
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().

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.

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 lavaan::sem() or lavaan::lavaan(), 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 semPlot::semPaths().
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 rotate_resid(). Ignored if object is a qgraph object.
style	The convention for the angles. If "1200", the default, the convention of rotate_resid() 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.

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.

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 positon manually
pos_new <- safe_resid_position(p,
                               update_plot = FALSE)
pos_new
p_safe2 <- p |>
            rotate_resid(pos_new)
plot(p_safe2)
```

`sem_2nd_order_example` *Sample dataset sem_2nd_order_example*

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` *Sample dataset sem_example*

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.
`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.

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)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
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 *may* 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.

Details

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

Value

A qgraph::qgraph based on the original one, with curve attributes for selected edges changed.

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` *Set the Attributes of Selected Edges*

Description

Set arbitrary attributes of selected edges.

Usage

```
set_edge_attribute(semPaths_plot, values = NULL, attribute_name = NULL)
```

Arguments

`semPaths_plot`	A qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
`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.

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 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, 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.

Value

A qgraph::qgraph based on the original one, with the selected attributes of selected edges changed.

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)
```

Arguments

`semPaths_plot`	A qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
`color_list`	A named vector or a list of named list. See the Details section on how to set this argument.

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.

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_position` *Set the positions of edge labels of selected edges*

Description

Set the positions of edge labels of selected edges.

Usage

```
set_edge_label_position(semPaths_plot, position_list = NULL)
```

Arguments

`semPaths_plot`	A qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
`position_list`	A named vector or a list of named lists. 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 position of the edge label of the path regressing `y` on `x`, the name should be `"y ~ x"`. 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 close 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`. For a list of named lists, each named list should have three named values: `from`, `to`, and `new_position`. The edge label position of the edge from `from` to `to` will be set to `new_position`. For example, `list(list(from = "x1", to = "y", new_position = .2), list(from = "x2", to = "y", new_position = .7))` is equivalent to the named vector above.

Details

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

Value

A qgraph::qgraph based on the original one, with edge label positions for selected edges changed.

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

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_node_attribute *Set the Attributes of Selected Nodes*

Description

Set arbitrary attributes of selected nodes

Usage

```
set_node_attribute(semPaths_plot, values = NULL, attribute_name = NULL)
```

Arguments

semPaths_plot	A qgraph::qgraph object generated by semPlot::semPaths, or a similar qgraph object modified by other semptools functions.
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.

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 two 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 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.

Value

A qgraph::qgraph based on the original one, with the selected attributes of selected nodes changed.

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_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.
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`.
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 the values must be the directions, and the names the 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.

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))
```

`to_list_of_lists` *Convert a named vector to a list of lists*

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. Defaultis `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