Package 'semptools'

October 15, 2023

Title Customizing Structural Equation Modelling Plots

Version 0.2.10

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 (>= 3.6.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports lavaan, rlang, semPlot

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

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-10-15 15:00:02 UTC

R topics documented:

add_object 2
auto_factor_point_to 3
auto_indicator_order 4
cfa_example 6

change_node_label 7
is_dv_residvar 8
keep_drop_nodes 9
lavaan_indicator_order 11
layout_matrix 12
mark_se 13
mark_sig 15
pa_example 16
pa_example_3covs 17
rotate_resid 18
sem_2nd_order_example 19
sem_example 20
set_cfa_layout 20
set_curve 23
set_edge_label_position 24
set_sem_layout 25
to_list_of_lists 30

Index **31**

`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")
```

---

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)
```

`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 Estimates to Parameter Estimates (Edge Labels)*

Description

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

Usage

```
mark_se(semPaths_plot, object, sep = " ", digits = 2L, 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.
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. Only used when object is not specified.

Details

Modify a qgraph::qgraph object generated by semPaths (currently in parentheses) to the labels. Require the original object used in the semPaths call.

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 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)
```

`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)
)
```

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).

Details

Modify a qgraph::qgraph object generated by semPaths and add marks (currently asterisk, "*") to the labels based on their p-values. Require the original object used in the semPaths call.

Currently supports only plots based on lavaan output.

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)
```

`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
'
```

`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)
```

`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_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_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

∗ **datasets**
  cfa_example, 6
  pa_example, 16
  pa_example_3covs, 17
  sem_2nd_order_example, 19
  sem_example, 20

add_object, 2
auto_factor_point_to, 3
auto_factor_point_to(), *26*
auto_indicator_order, 4
auto_indicator_order(), *21*, *26*

cfa_example, 6
change_node_label, 7

drop_nodes *(*keep_drop_nodes*)*, 9
drop_nodes(), *10*

is_dv_residvar, 8

keep_drop_nodes, 9
keep_nodes *(*keep_drop_nodes*)*, 9
keep_nodes(), *10*

lavaan, *13*, *15*
lavaan::cfa(), *2*
lavaan::lavaan, *8*
lavaan::lavParseModelString(), *30*
lavaan::model.syntax, *6*, *17*, *19*, *20*, *23*, *24*
lavaan::parameterEstimates(), *23*, *24*
lavaan::sem(), *2*
lavaan_indicator_order, 11
layout_matrix, 12

mark_se, 13
mark_sig, *13*, 15

pa_example, 16
pa_example_3covs, 17
parameterEstimates, *13*

qgraph::qgraph, *2*, *3*, *5*, *7*, *8*, *13*, *15*, *18*, *21*, *23–28*

rotate_resid, 18

sem_2nd_order_example, 19
sem_example, 20
semPaths, *13*
semPlot::semPaths, *7*, *8*, *18*, *23–25*
semPlot::semPaths(), *2*, *3*, *5*, *7*, *10*, *12*
semPlot::semPlotModel, *10*
semPlot::semPlotModel(), *10*
semptools, *2*, *5*, *7*, *13*, *15*, *18*, *21*, *23*, *24*, *26*, *30*
set_cfa_layout, 20
set_cfa_layout(), *5*, *11*
set_curve, 23
set_edge_label_position, 24
set_sem_layout, 25
set_sem_layout(), *4*, *5*, *11*

to_list_of_lists, 30