Package 'diy.sem.plot'

November 11, 2026

Type Package

Title Manually Plot Path Diagrams for Structural Equation Models

Version 1.1.0

Description Manually plot fully customisable path diagrams for structural equation models (SEM). Map out node positions using simple coordinates and specify where on the perimeter of each node paths begin and end. Extensive fine-tuning options allow the creation of a path diagram exactly as envisioned, entirely within R.

Encoding UTF-8

Imports ggplot2, ggtext, ggforce, lavaan, patchwork

Suggests knitr, rmarkdown, testthat

Config/testthat/edition 3

URL https://github.com/snagy86/diy.sem.plot

BugReports https://github.com/snagy86/diy.sem.plot/issues

VignetteBuilder knitr

License MIT + file LICENSE

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Sebastian Nagy [aut, cre, cph]

Maintainer Sebastian Nagy <snagy8610@gmail.com>

Repository CRAN

Date/Publication 2026-11-10 23:40:07 UTC

Contents

diyPaths 2
node 7
panel_note 8
panel_title 9
path 9
template 11

Index **13**

`diyPaths` *Manually plot path diagrams for structural equation models*

Description

Plot fully customisable path diagrams for structural equation models fitted with **lavaan**, rendered using **ggplot2**.

To render the `ggplot` object of the diagram, the function only requires users to supply a fitted `lavaan` model, specify node positions using x-y coordinates, and detail where on the perimeter of each node (top, bottom, left, or right) each path should begin and end. The render automatically inserts estimates centred on the midpoint and adjusts each node's shape to match its variable type. A range of optional fine-tuning arguments are included, facilitating the creation of a path diagram exactly as you envision it, entirely within R.

Usage

```
diyPaths(
  fit,
  node_positions,
  path_positions,
  standardised = FALSE,
  digits = 3,
  est_stars = FALSE,
  est_p = FALSE,
  est_ci = FALSE,
  variance_stars = FALSE,
  variance_p = FALSE,
  variance_ci = FALSE,
  sig_linetype = FALSE,
  p_threshold = 0.05,
  show_variances = FALSE,
  show_grid = FALSE,
  grid_axis_scale = 1,
  non_transparent_text = TRUE,
  latent_node_text_size = 4,
  observed_node_text_size = 4,
  path_text_size = 3.5,
  line_thickness = 0.6,
  arrow_size = 0.2,
  node_width = 1.5,
  node_height = 1,
  latent_node_size_adjust = 1,
  observed_node_size_adjust = 1,
  show_group_labels = FALSE,
  panel_titles = NULL,
```

```
  panel_cols = NULL,
  margin_x = 0.5,
  margin_y = 0.5,
  look_up_table = FALSE,
  panel_notes = NULL,
  note_text_size = NULL,
  title_text_size = NULL,
  font_family = "",
  title_note_font_family = NULL
)
```

Arguments

`fit`	A fitted model object of class `lavaan`.
`node_positions`	Specify a list of node position objects. Use the `node()` helper function to assist with this.
`path_positions`	Specify a list of path configuration objects. Use the `path()` helper function to assist with this.
`standardised`	Logical. If `TRUE`, uses standardised parameter estimates (`est.std`). Default is `FALSE`.
`digits`	Number of decimal places to display for estimates. Default is `3`.
`est_stars`	Logical. Whether to display significance stars on path estimates. Default is `FALSE`.
`est_p`	Logical. Whether to display p-values on path estimates. Default is `FALSE`.
`est_ci`	Logical. Whether to display confidence intervals on path estimates. Default is `FALSE`.
`variance_stars`	Logical. Whether to display significance stars on variance paths. Default is `FALSE`.
`variance_p`	Logical. Whether to display p-values on variance paths. Default is `FALSE`.
`variance_ci`	Logical. Whether to display confidence intervals on variance paths. Default is `FALSE`.
`sig_linetype`	Logical. If `TRUE`, renders non-significant paths with dashed lines. Default is `FALSE`.
`p_threshold`	Statistical significance threshold used to determine non-significant paths when `sig_linetype = TRUE`. Default is `0.05`.
`show_variances`	Logical. Whether to display variance and residual paths. Default is `FALSE`.
`show_grid`	Logical. Whether to overlay a coordinate grid. Default is `FALSE`.
`grid_axis_scale`	Sets the spacing of grid-lines when `show_grid = TRUE`. Default is `1`.
`non_transparent_text`	Logical. If `TRUE`, path estimates receive a white background mask, if `FALSE`, they have a transparent background. Default is `TRUE`.
`latent_node_text_size`	Text font size for latent node labels. Default is `4`.

`observed_node_text_size`
: Text font size for observed node labels. Default is `4`.

`path_text_size`
: Text font size for path estimate labels. Default is `3.5`.

`line_thickness`
: Sets thickness of paths. Default is `0.6`.

`arrow_size`
: Sets the size of arrow heads. Default is `0.2`.

`node_width`
: Base width for nodes. Default is `1.5`.

`node_height`
: Base height for node shapes. Default is `1`.

`latent_node_size_adjust`
: Numeric multiplier scaling latent node ellipses. Default is `1`.

`observed_node_size_adjust`
: Numeric multiplier scaling observed node rectangles. Default is `1`.

`show_group_labels`
: Logical. For multi-group models, whether to annotate each panel with its panel number and the raw group value it represents (e.g. "Panel 1: Group = male"). Needed to identify which diagram represents each group and its internal panel number when creating panel titles and notes. Default is `FALSE`.

`panel_titles`
: Specify a list of titles for panels. Use the `panel_title()` helper function to assist with this. Any panel not referenced is left untitled. Default is `NULL`.

`panel_cols`
: Integer for number of columns to use when arranging multi-group panels. Default is `NULL`.

`margin_x`
: Padding for plot limits along the x-axis. Default is `0.5`.

`margin_y`
: Padding for plot limits along the y-axis. Default is `0.5`.

`look_up_table`
: Logical. If `TRUE`, also returns a look-up table detailing the width (x scale) and height (y scale) of latent and observed nodes, along with the text sizes used for latent, observed, and path labels. Default is `FALSE`.

`panel_notes`
: Specify a list of notes, displayed as captions beneath each panel. Use the `panel_note()` helper function to assist with this. Any panel not referenced is left without a note. Default is `NULL`.

`note_text_size`
: Font size for panel notes (captions). Default is NULL (ggplot2's default caption size).

`title_text_size`
: Font size for panel titles. Default is NULL (ggplot2's default title size).

`font_family`
: Font family used for all text in the diagram. Default is `""`, the graphics device's default font.

`title_note_font_family`
: Font family for panel titles and notes. Default is `NULL`, which uses `font_family`.

Details

The easiest way to start is with `template()`, which prints ready-to-run code for your fitted model. Edit the node positions and path sides it generates, render with `diyPaths()`, then fine-tune and re-render until the layout looks right. See `vignette("diy.sem.plot")` for worked examples.

Value

A ggplot (or patchwork, for multi-group models) object representing the SEM path diagram(s). If look_up_table = TRUE, a list containing the diagram(s) ($plot) and a look-up data.frame of node width/height and text sizes for latent, observed, and path labels ($look_up_table).

See Also

template() to generate starting code for node_positions, path_positions, panel_titles and panel_notes; node(), path(), panel_title(), and panel_note() for the helper functions used within them.

Examples

```
#full SEM example, this model may not make theoretical sense.

library(lavaan)
library(ggplot2)

data(HolzingerSwineford1939, package = "lavaan")

#specify the model

sem_model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9

  speed ~ visual + textual
  visual ~~ textual
'

#fit the model

fit <- sem(sem_model, data = HolzingerSwineford1939)

#generate a starting template

temp <- template(fit = fit, diyPaths_arguments = "max", quiet = TRUE)

#### edited template ####

node_list <- list(
  node(name = "visual", x = 1, y = 1, label = "Visual"),
  node(name = "textual", x = 1, y = 2, label = "Textual"),
  node(name = "speed", x = 4, y = 1.5, label = "Speed"),
  node(name = "x1", x = -0.06, y = -0.5, label = "Visual\nPerception"),
  node(name = "x2", x = 1, y = -0.5, label = "Cubes"),
  node(name = "x3", x = 2.06, y = -0.5, label = "Lozenges"),
  node(name = "x4", x = -0.06, y = 3.5, label = "Paragraph\nComprehension"),
  node(name = "x5", x = 1, y = 3.5, label = "Sentence\nCompletion"),
  node(name = "x6", x = 2.06, y = 3.5, label = "Word\nMeaning"),
  node(name = "x7", x = 6, y = 0.5, label = "Speeded\nAddition"),
```

```
  node(name = "x8", x = 6, y = 1.5, label = "Speeded\nCounting"),
  node(name = "x9", x = 6, y = 2.5, label = "Speeded\nDiscrimination")
)

path_list <- list(
  # regression paths
  path(from = "visual", to = "speed", side_from = "right", side_to = "left"),
  path(from = "textual", to = "speed", side_from = "right", side_to = "left"),

  # loading paths
 path(from = "visual", to = "x1", side_from = "bottom", side_to = "top", nudge_text_x = -0.1),
  path(from = "visual", to = "x2", side_from = "bottom", side_to = "top"),
 path(from = "visual", to = "x3", side_from = "bottom", side_to = "top", nudge_text_x = 0.1),
 path(from = "textual", to = "x4", side_from = "top", side_to = "bottom", nudge_text_x = -0.1),
  path(from = "textual", to = "x5", side_from = "top", side_to = "bottom"),
 path(from = "textual", to = "x6", side_from = "top", side_to = "bottom", nudge_text_x = 0.1),
  path(from = "speed", to = "x7", side_from = "right", side_to = "left"),
  path(from = "speed", to = "x8", side_from = "right", side_to = "left"),
  path(from = "speed", to = "x9", side_from = "right", side_to = "left"),

  # covariance/correlation paths
 path(from = "visual", to = "textual", side_from = "left", side_to = "left", cov_curve = -0.6)
)

title_list <- list(
  panel_title(panel_num = 1, title = "My SEM Plot")
)

note_list <- list(
  panel_note(panel_num = 1, note_text = "Standardised estimates shown.")
)

p <- diyPaths(fit = fit, node_positions = node_list, path_positions = path_list,
              standardised = TRUE,
              digits = 3,
              est_stars = TRUE,
              est_p = FALSE,
              est_ci = FALSE,
              show_variances = FALSE,
              variance_stars = FALSE,
              variance_p = FALSE,
              variance_ci = FALSE,
              sig_linetype = FALSE,
              p_threshold = 0.05,
              panel_titles = title_list,
              panel_notes = note_list,
              show_grid = TRUE,
              grid_axis_scale = 0.5,
              non_transparent_text = TRUE,
              latent_node_text_size = 4,
              observed_node_text_size = 3,
              path_text_size = 3.5,
              line_thickness = 0.6,
```

```
                arrow_size = 0.2,
                node_width = 1.5, node_height = 1,
                latent_node_size_adjust = 0.8,
                observed_node_size_adjust = 0.55,
                show_group_labels = FALSE,
                panel_cols = NULL,
                font_family = "sans",
                title_note_font_family = "serif",
                margin_x = 0.5,
                margin_y = 0.5,
                look_up_table = TRUE)

print(p)
```

---

node *Create a node position list*

Description

Helper function that creates a list of arguments which specify a given node's position and its label in the diagram, designed for use within the `node_positions` argument of `diyPaths()`.

Usage

```
node(name, x = 0, y = 0, label = NULL)
```

Arguments

`name`	The name of the variable within the `lavaan` model.
`x`	Numeric value for the x-coordinate of the node's centre. Default is `0`.
`y`	Numeric value for the y-coordinate of the node's centre. Default is `0`.
`label`	Custom label to display instead of `lavaan` variable name. Defaults to the `lavaan` variable name.

Value

A list containing arguments that specify a node's position and label, for use within the `node_positions` argument of `diyPaths()`.

See Also

`path()` to specify the paths between nodes; `diyPaths()` to render the diagram; `template()` to generate a starting `node_positions` list.

Examples

```
#a list that specifies a node positioned on x = 1, y = 2,
#and relabelled from its `lavaan` model name.

node(name = "bpm", x = 1, y = 2,  label = "Beats per Minute")
```

`panel_note` *Create a note for a diyPaths panel*

Description

Helper function that creates a list of arguments which specify a custom note, displayed as a caption beneath its target panel, designed for use within the `panel_notes` argument of `diyPaths()`. Use the `show_group_labels` argument in `diyPaths()` to view each panel's number and which group it refers to.

Usage

```
panel_note(panel_num, note_text = NULL)
```

Arguments

`panel_num`	Integer value for the panel number this note applies to.
`note_text`	Text to display in the note. If `NULL`, no note is drawn.

Value

A list containing arguments that specify a panel's note text, for use within the `panel_notes` argument of `diyPaths()`.

See Also

`diyPaths()`, where the list of notes is passed to `panel_notes`; `panel_title()` for adding titles to panels; `template()`, whose output includes a starting note list.

Examples

```
# a note on the first panel noting sample size was 100.
panel_note(panel_num = 1, note_text = "n = 100")

# a note on the second panel of a multi-group model noting sample size for Group 2 was 50
panel_note(panel_num = 2, note_text = "Group 2: n = 50")
```

panel_title *Create a title for a diyPaths panel*

Description

Helper function that creates a list of arguments which specify a custom title and its target panel, designed for use within the `panel_titles` argument of `diyPaths()`. Use the `show_group_labels` argument in `diyPaths()` to view each panel's number and which group it refers to.

Usage

```
panel_title(panel_num = 1, title = NULL)
```

Arguments

`panel_num`	Integer value for the panel number this title applies to. Default is `1`.
`title`	The title text to display.

Value

A list containing arguments that specify a panel's number and title, for use within the `panel_titles` argument of `diyPaths()`.

See Also

`diyPaths()`, where the list of titles is passed to `panel_titles`; `panel_note()` for adding notes to panels; `template()`, whose output includes a starting title list.

Examples

```
# titling a single, non-grouped model
panel_title(title = "My SEM Model")

# titling the second panel of a multi-group model
panel_title(panel_num = 2, title = "Female Participants")
```

path *Create a path position list*

Description

Helper function that creates a list of arguments which specify a given path's position and fine-tuning adjustments, designed for use within the `path_positions` argument of `diyPaths()`.

Usage

```
path(
  from,
  to,
  side_from = "right",
  side_to = "left",
  cov_curve = NULL,
  nudge_text_x = 0,
  nudge_text_y = 0,
  variance_position = "top"
)
```

Arguments

from	The source variable name within the lavaan model.
to	The target variable name within the lavaan model.
side_from	Side of source node where path starts ("top", "bottom", "left", "right"). Default is "right".
side_to	Side of target node where path ends ("top", "bottom", "left", "right"). Default is "left".
cov_curve	Numeric value for curvature of covariance/correlation paths. Default is NULL.
nudge_text_x	Numeric fine tuning adjustment for path estimate text along the x-axis. Default is 0.
nudge_text_y	Numeric fine tuning adjustment for path estimate text along the y-axis. Default is 0.
variance_position	Placement of variance/residual paths ("top", "bottom", "left", or "right"). Default is "top".

Details

from/to in path() must exactly match the variable name used in the lavaan model syntax. Furthermore, the order must also be correct for regression or loading paths. A misspelled or mismatched path() entry will not raise an error, instead, it will use default attachment points and values, not applying specific customisations. This can be completely avoided using template().

cov_curve controls the curvature of covariance/correlation paths: its size sets how strongly the path bends, and its sign sets which way. If a path bends the wrong way, flip the sign. Best results typically range from -1 to 1.

Value

A list containing arguments that specify a path's position and fine-tuning adjustments, for use within the path_positions argument of diyPaths().

See Also

node() to specify node positions; diyPaths() to render the diagram; template() to generate a starting path_positions list.

Examples

```
#a list that specifies a regression path of node "anxiety" predicting node "depression".

path(from = "anx", to = "dep", side_from = "right", side_to = "left")

#a list that specifies a covariance/correlation path of node "anxiety" and node "depression".
#Order of "from" and "to" does not matter for covariance/correlation.


path(from = "dep", to = "anx", side_from = "left", side_to = "left", cov_curve = -0.6)
```

---

template *Create a template for use in diyPaths*

Description

Takes your fitted model and prints a ready-to-run starting template (node list, path list, title list, note list, and `diyPaths()` call) to the console. Copy the output into your script, then fine-tune the diagram by adjusting values in the lines marked `#EDIT ME!`

The template is also returned invisibly as a character string, so it can be stored as an object then viewed with `writeLines()` (see examples).

Usage

```
template(
  fit,
  include_variance_paths = TRUE,
  diyPaths_arguments = c("max", "min"),
  quiet = FALSE
)
```

Arguments

`fit`	A fitted model object of class `lavaan`.
`include_variance_paths`	Logical. Whether to include variance/residual paths in the path list. Default is `TRUE`.
`diyPaths_arguments`	Use either "max" (the printed call shows every `diyPaths()` argument) or "min" (only the required arguments). Default is "max".
`quiet`	Logical. If `TRUE`, the template is not printed to the console and is only returned invisibly. Default is `FALSE`.

Value

Invisibly returns the template as a single character string. Unless `quiet = TRUE`, it is also printed to the console.

See Also

diyPaths() to render the diagram from the template; node(), path(), panel_title(), and panel_note() for the helper functions the template uses.

Examples

```
library(lavaan)

data(HolzingerSwineford1939, package = "lavaan")


#specify the model

sem_model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9

  speed ~ visual + textual
  visual ~~ textual
'

#fit the model

fit <- sem(sem_model, data = HolzingerSwineford1939)

#print the maximal template to the console

# template(fit = fit, include_variance_paths = TRUE, diyPaths_arguments = "max", quiet = FALSE)

#store the template without printing it, then view it

tmpl <- template(fit = fit, diyPaths_arguments = "min", quiet = TRUE)
writeLines(tmpl)
```

Index

diyPaths, 2
diyPaths(), *7–12*

node, 7
node(), *3*, *5*, *10*, *12*

panel_note, 8
panel_note(), *4, 5*, *9*, *12*
panel_title, 9
panel_title(), *4, 5*, *8*, *12*
path, 9
path(), *3*, *5*, *7*, *12*

template, 11
template(), *4, 5*, *7–10*