Package 'diy.sem.plot'

September 26, 2026

Type Package

Title Manually Plot Path Diagrams for Structural Equation Models

Version 1.0.0

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 2026-09-26 16:20:13 UTC

Contents

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 centered 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
)
```

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 digits 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 estimate labels receive a white background mask. 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. 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 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 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`.

Details

Using the function requires 4 steps and is illustrated by the example below. See `vignette("diy.sem.plot")` for in-depth examples and guidance on using the function's arguments.

1. Specify and fit the SEM using **lavaan**.
2. Specify `node_positions` as a list (`node_positions = list(...)`) and, within it, define each node's position and label using the `node()` helper function.
3. Specify `path_positions` as a list (`path_positions = list(...)`) and, within it, define each path's connection points, curvature, and label adjustments using the `path()` helper function.
4. Call `diyPaths()`, passing in the fitted model, `node_positions`, and `path_positions`, along with any additional display augmentations (i.e. show significance stars on estimates).

Creating a diagram for a specific model is inherently an iterative process. As such, steps 2-4 will likely need to be repeated with additional fine-tuning adjustments to achieve the desired result.

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`).

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)

#specify the node position, this was done iteratively with show_grid to help with layout.

node_list <- list(
  #main latent variable structure
  node("visual", x = 1, y = 1, label = "Visual"),
  node("textual", x = 1, y = 2, label = "Textual"),
  node("speed", x = 4, y = 1.5, label = "Speed"),

  # observed variables that visual perception ability loads onto
  node("x1", x = -0.06, y = -0.5, label = "Visual\nPerception"), #\n creates a line break
  node("x2", x = 1, y = -0.5, label = "Cubes"),
  node("x3", x = 2.06, y = -0.5, label = "Lozenges"),

  #observed variables that textual ability loads onto
  node("x4", x = -0.06, y = 3.5, label = "Paragraph\nComprehension"),
  node("x5", x = 1, y = 3.5, label = "Sentence\nCompletion"),
  node("x6", x = 2.06, y = 3.5, label = "Word\nMeaning"),

  #observed variables that speeded cognitive processing loads onto
  node("x7", x = 6, y = 0.5, label = "Speeded\nAddition"),
  node("x8", x = 6, y = 1.5, label = "Speeded\nCounting"),
  node("x9", x = 6, y = 2.5, label = "Speeded\nDiscrimination")
)

#Specify the paths

path_list <- list(
 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"),

  path(from = "visual",  to = "speed", side_from = "right", side_to = "left"),
  path(from = "textual", to = "speed", side_from = "right", side_to = "left"),

 path(from = "visual",  to = "textual", side_from = "left", side_to = "left", cov_curve = -0.6)
)

#Creating title

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

#creating the diagram

p <- diyPaths(
  fit = fit,
  node_positions = node_list,
  path_positions = path_list,
  panel_titles = title_list,
  standardised = TRUE,
  est_stars = TRUE,
  observed_node_size_adjust = 0.55,
  observed_node_text_size = 3,
  latent_node_size_adjust = 0.8,
  show_grid = TRUE,
  grid_axis_scale = 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 name 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().

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_title *Create a title for a diyPaths panel*

Description

Helper function that creates a list of arguments which specify custom titles 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().

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.

cov_curve's value can be used to adjust direction of curve on covariance/correlation path. For a mostly vertical path (i.e. node1: x = 0, y = 1 -> node2: x = 0, y = 2), positive curvature bends it left and negative curvature bends it right. For a mostly horizontal path (i.e. node1: x = 1, y = 0 -> node2: x = 2, y = 0), positive curvature bends it down and negative curvature bends it up. 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().

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)
```

Index