Package: semEffect (via r-universe)

May 15, 2026

Title Unified Effect Analysis and Visualization for Structural Equation Models

Version 1.3.0

Description Provides a unified interface to calculate, extract, and visualize standardized effect decompositions (direct, indirect, and total effects) for three major structural equation modeling (SEM) frameworks: 'lavaan', 'piecewiseSEM', and 'plspm'. The package simplifies comparative SEM analysis by offering consistent functions across different modeling paradigms. Key features include automated handling of zero-effect variables, generation of publication-ready 'ggplot2' visualizations, and returning both wide-format and long-format effect tables. It supports flexible effect filtering, accepts multi-model object inputs, and allows extensive customization of visualization parameters. The 'sem_modeling()' function serves as a central wrapper, enabling users to fit models using different packages with a unified syntax. For methodological background, see 'Rosseel' (2012) <doi:10.18637/jss.v048.i02> for 'lavaan', 'Lefcheck' (2016) <doi:10.1111/2041-210X.12512> for 'piecewiseSEM', and 'Sanchez' (2013) <doi:10.1007/s11747-012-0306-5> for 'PLS-PM'.

URL https://github.com/PhDMeiwp/semEffect/

BugReports https://github.com/PhDMeiwp/semEffect/issues

Depends R (>= 4.4.0)

License GPL-3

Encoding UTF-8

Imports lavaan, piecewiseSEM, plspm, ggplot2, tidyr, dplyr, checkmate

Suggests testthat, RColorBrewer, blavaan, semEff

RoxygenNote 7.3.2

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev libx11-dev

Repository https://phdmeiwp.r-universe.dev

Date/Publication 2025-09-11 05:15:13 UTC

RemoteUrl https://github.com/phdmeiwp/semeffect

RemoteRef HEAD

RemoteSha 7a6436d8d38a3f5fd339a956d8eec2edb05435ae

Contents

`get_effect` *Extract Effect Values from Structural Equation Models*

Description

Extracts standardized effect decomposition (direct, indirect, and total effects) for three major structural equation modeling frameworks: 'lavaan', 'piecewiseSEM', and 'plspm'. Returns both wide-format and long-format effect tables.

Usage

```
get_effect(object, target, delete_zero_effect = TRUE, zero_threshold = 1e-10)
```

Arguments

`object`	SEM object (lavaan/psem/plspm).
`target`	Character string specifying the target variable name for effect analysis.
`delete_zero_effect`	Logical indicating whether to remove rows where all specified effect columns contain only zeros (default: `TRUE`).
`zero_threshold`	Threshold for negligible variables (default: 1e-10). A variable will be removed if 'delete_zero_effect' is TRUE and both of the following conditions are met: (i) the absolute value each effect type ('Direct_Effect','Indirect_Effect', and 'Total_Effect') is less than the threshold; AND (ii) the absolute value of the sum of the three effect values is also less than the threshold.

Value

A list containing two components:

- `effect_table`: A data frame with variables and their standardized effect values (direct, indirect, total)
- `effect_long`: A long-format version of `effect_table`

Author(s)

Weiping Mei

See Also

plot_effect, sem, psem, plspm

Examples

```
# Example 01: lavaan -------------------------------
if (requireNamespace("lavaan", quietly = TRUE)) {
library(lavaan)

model <- '
  # Measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

  # Structural model
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
'

# Fit the model using sem_modeling
fit <- sem_modeling(model, data = PoliticalDemocracy, type = "lavaan")

# Extract effects for target variable "dem65"
effects <- get_effect(fit, target = "dem65")

print(effects$effect_table)
print(effects$effect_long)
}

# Example 02: piecewiseSEM -------------------------
if (requireNamespace("piecewiseSEM", quietly = TRUE)) {
library(piecewiseSEM)

# Create model using sem_modeling compatible format
model_list <- list(
  lm(rich ~ cover, data = keeley),
  lm(cover ~ firesev, data = keeley),
  lm(firesev ~ age, data = keeley)
)

pmod <- sem_modeling(model_list, data = keeley, type = "piecewiseSEM")

effects <- get_effect(pmod, target = "rich")
print(effects$effect_table)
}

# Example 03: plspm with lavaan-style syntax -------------------------
```

```
if (requireNamespace("plspm", quietly = TRUE)) {
  library(plspm)
  data(satisfaction)

  # Define model using lavaan-style syntax
  model_plspm <- "
    # Measurement model (latent variable definitions)
    IMAG =~ imag1 + imag2 + imag3 + imag4 + imag5
    EXPE =~ expe1 + expe2 + expe3 + expe4 + expe5
    QUAL =~ qual1 + qual2 + qual3 + qual4 + qual5
    VAL =~ val1 + val2 + val3 + val4
    SAT =~ sat1 + sat2 + sat3 + sat4
    LOY =~ loy1 + loy2 + loy3 + loy4

    # Structural model (regression relationships)
    EXPE ~ IMAG
    QUAL ~ EXPE
    VAL ~ EXPE + QUAL
    SAT ~ IMAG + EXPE + QUAL + VAL
    LOY ~ SAT + IMAG
  "

  # Fit the model using sem_modeling (modes will default to all "A")
  pls_fit <- sem_modeling(model_plspm, data = satisfaction, type = "plspm")

  # Extract effects for target variable "LOY"
  effects <- get_effect(pls_fit, target = "LOY")
  print(effects$effect_table)

  # Example with explicit modes specification
  sat_modes <- rep("A", 6)  # All reflective modes
  pls_fit2 <- sem_modeling(model_plspm, data = satisfaction,
                           type = "plspm", modes = sat_modes)
  effects2 <- get_effect(pls_fit2, target = "LOY")
  print(effects2$effect_table)
 }
```

---

plot_effect *Plot Effect Values from Structural Equation Models*

Description

Creates ggplot2 visualizations of effect values extracted from SEM models. Supports plotting total effects only or all effect types (direct, indirect, total).

Usage

```
plot_effect(
  object,
```

```
  target,
  total_only = FALSE,
  delete_zero_effect = TRUE,
  zero_threshold = 1e-10,
  total_color = "skyblue",
  color_palette = c("darkgreen", "skyblue", "orange")
)
```

Arguments

object	SEM object (lavaan/psem/plspm) from which to extract effect values.
target	Character string specifying the target variable name for effect analysis.
total_only	Logical controlling plot mode. If TRUE, shows only total effects with customizable colors; if FALSE, displays all effect types with palette coloring (default: FALSE).
delete_zero_effect	Logical indicating whether to remove rows where all specified effect columns contain only zeros (default: TRUE).
zero_threshold	Threshold for negligible variables (default: 1e-10).
total_color	Single color or vector of colors for total effect bars when total_only=TRUE (default: "skyblue").
color_palette	Character vector of 3 colors for direct/indirect/total effects when total_only=FALSE (default: c("darkgreen", "skyblue", "orange")).

Value

A ggplot2 object displaying the effect values.

Author(s)

Weiping Mei

See Also

get_effect

Examples

```
# Example using lavaan model
if (requireNamespace("lavaan", quietly = TRUE)) {
library(lavaan)

model <- '
  # Measurement model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

  # Structural model
```

```
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
'

# Fit the model using sem_modeling
fit <- sem_modeling(model, data = PoliticalDemocracy, type = "lavaan")

# Plot all effect types for target variable "dem65"
p1 <- plot_effect(fit, target = "dem65", total_only = FALSE)
print(p1)

# Plot only total effects for target variable "dem65"
p2 <- plot_effect(fit, target = "dem65", total_only = TRUE,
                  total_color = "steelblue")
print(p2)
 }
```

`sem_lavaan` *Fitting Structural Equation Models (SEM) with lavaan*

Description

A wrapper for the `sem` function from the `lavaan` package. It performs basic input checks before fitting a Structural Equation Model (SEM).

Usage

```
sem_lavaan(model, data, ...)
```

Arguments

`model`	A character string specifying the structural equation model.
`data`	A data frame containing the observed variables used in the model.
`...`	Additional arguments passed to `sem`.

Value

An object of class `lavaan-class`, representing the fitted SEM.

See Also

`sem`

Examples

```
## Not run:
if (requireNamespace("lavaan", quietly = TRUE)) {
library(lavaan)
data(PoliticalDemocracy)

model_spec <- '
  # Measurement Model
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

  # Structural Model
  dem60 ~ ind60
  dem65 ~ ind60 + dem60

  # Residual Covariances
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

fit <- sem_lavaan(model = model_spec, data = PoliticalDemocracy)
summary(fit, standardized = TRUE, fit.measures = TRUE)
 }

## End(Not run)
```

`sem_modeling` *sem_modeling*

Description

Provides a unified interface to fit Structural Equation Models (SEM) using three different approaches: lavaan-style modeling, piecewise SEM, and PLS-PM. This function serves as a wrapper for custom sem_lavaan, sem_piecewise, and sem_plspm functions.

Usage

```
sem_modeling(model, data, type = 1, modes = NULL, ...)
```

Arguments

`model`	Model specification. For lavaan and plspm, a character string specifying the model syntax. For piecewiseSEM, can be a character string with formulas, a list of regression models, or a list compatible with piecewiseSEM's native format.
`data`	A data.frame containing the observed data.

type	Type of SEM model to fit. Can be numeric (1, 2, 3) or character ("lavaan", "piecewiseSEM", "plspm"). Default is 1 ("lavaan").
modes	A character vector specifying the measurement mode for each block in plspm (only required for plspm). Each element should be either "A" (reflective) or "B" (formative). If NULL (default), all modes are set to "A" (reflective).
...	Additional arguments passed to the underlying SEM fitting functions.

Value

A fitted model object whose class depends on the selected package: - lavaan: lavaan object - piecewiseSEM: psem object - plspm: plspm object The returned object has an additional attribute "model_type" indicating the type of model fitted.

See Also

sem_lavaan, sem_piecewise, sem_plspm

Examples

```
# Example 1: Using lavaan with built-in PoliticalDemocracy data
if (requireNamespace("lavaan", quietly = TRUE)) {
  library(lavaan)
  data(PoliticalDemocracy)

  model_lavaan <- '
    # measurement model
    ind60 =~ x1 + x2 + x3
    dem60 =~ y1 + y2 + y3 + y4
    dem65 =~ y5 + y6 + y7 + y8
    # structural model
    dem60 ~ ind60
    dem65 ~ ind60 + dem60
  '

  # Fit the model
  fit_lavaan <- sem_modeling(model_lavaan, data = PoliticalDemocracy,
                             type = "lavaan")
  summary(fit_lavaan)
}

# Example 2: Using piecewiseSEM with built-in keeley data
if (requireNamespace("piecewiseSEM", quietly = TRUE)) {
  library(piecewiseSEM)
  data(keeley)

  model_psem <- "
    firesev ~ age + abiotic
    cover ~ firesev + hetero
    rich ~ cover + abiotic
  "
```

```
  # Fit the model
  fit_psem <- sem_modeling(model_psem, data = keeley, type = "piecewiseSEM")
  summary(fit_psem)
}

# Example 3: Using plspm with built-in satisfaction dataset
if (requireNamespace("plspm", quietly = TRUE)) {
  library(plspm)
  data(satisfaction)

  model_plspm <- "
    # Measurement model
    IMAG =~ imag1 + imag2 + imag3 + imag4 + imag5
    EXPE =~ expe1 + expe2 + expe3 + expe4 + expe5
    QUAL =~ qual1 + qual2 + qual3 + qual4 + qual5
    VAL =~ val1 + val2 + val3 + val4
    SAT =~ sat1 + sat2 + sat3 + sat4
    LOY =~ loy1 + loy2 + loy3 + loy4

    # Structural model
    EXPE ~ IMAG
    QUAL ~ EXPE
    VAL ~ EXPE + QUAL
    SAT ~ IMAG + EXPE + QUAL + VAL
    LOY ~ SAT + IMAG
  "

  # Example 3a: Fit without specifying modes (will default to all "A")
  fit_plspm1 <- sem_modeling(model_plspm, data = satisfaction,
                             type = "plspm")
  summary(fit_plspm1)

  # Example 3b: Fit with explicit modes specification
  sat_modes <- rep("A", 6)
  fit_plspm2 <- sem_modeling(model_plspm, data = satisfaction,
                             type = "plspm", modes = sat_modes)
  summary(fit_plspm2)
  plot(fit_plspm2, what = "inner")
}
```

`sem_piecewise` *Fit a Piecewise Structural Equation Model (SEM) from Multiple Specification Formats*

Description

Fits a piecewise structural equation model using the `psem` function from the `piecewiseSEM` package. It accepts three types of model specifications: lavaan-style formula text, regression function text (lm/glm), or a list of models compatible with piecewiseSEM's native format.

Usage

```
sem_piecewise(model, data, ...)
```

Arguments

model	A model specification. Can be a character string containing formulas, a list of regression models, or a list compatible with piecewiseSEM's native format. For character inputs, each line should represent a structural equation, with the dependent variable on the left and predictors on the right of a '~' operator. Lines can be separated by newlines or semicolons. Comments starting with '#' are allowed.
data	A data frame containing the variables used in the model.
...	Additional arguments passed to `piecewiseSEM::psem`.

Details

The function first checks the input model type and content. For character specifications, it validates that the model does not contain latent variable ('=~') or covariance ('~~') operators, which are not supported in piecewiseSEM. It then parses the input model string into separate equations and constructs appropriate regression function calls.

Three input formats are supported:

1. Lavaan-style formula text: Multiple formulas as a character string
2. Regression function text: Explicit lm()/glm() calls as a character string
3. Native list format: A list of regression models as expected by piecewiseSEM

Value

An object of class `psem` returned by `piecewiseSEM::psem`, representing the fitted piecewise structural equation model. This contains information about individual regressions, goodness-of-fit statistics, and coefficients.

See Also

`psem` for the underlying fitting function.

Examples

```
if (requireNamespace("piecewiseSEM", quietly = TRUE)) {
  library(piecewiseSEM)
  data(keeley)

  # Example 1: Lavaan-style formula text
  model_text1 <- "
    firesev ~ age + abiotic
    cover ~ firesev + hetero
    rich ~ cover + abiotic
  "
  fit1 <- sem_piecewise(model = model_text1, data = keeley)
```

```
  summary(fit1)

  # Example 2: Regression function text
  model_text2 <- "
    lm(firesev ~ age + abiotic, data = data)
    lm(cover ~ firesev + hetero, data = data)
    lm(rich ~ cover + abiotic, data = data)
  "
  fit2 <- sem_piecewise(model = model_text2, data = keeley)
  summary(fit2)

  # Example 3: Native list format (same as piecewiseSEM)
  model_list3 <- list(
    lm(firesev ~ age + abiotic, data = keeley),
    lm(cover ~ firesev + hetero, data = keeley),
    lm(rich ~ cover + abiotic, data = keeley)
  )
  fit3 <- sem_piecewise(model = model_list3, data = keeley)
  summary(fit3)
}
```

`sem_plspm` *Convert Lavaan Format Model to Plspm Required Input Format and Perform PLS-PM Analysis*

Description

Parses a structural equation model specified in lavaan syntax, extracts the latent variable definitions and structural relationships, converts them into the path matrix and blocks list required by the plspm package, and performs Partial Least Squares Path Modeling analysis.

Usage

```
sem_plspm(model, data, modes = NULL, verbose = FALSE, ...)
```

Arguments

`model`	Character string, lavaan format model definition. Should include latent variable definitions (using =~ operator) and structural relationships (using ~ operator). Covariance relationships (~~) are not directly handled by plspm and will be ignored with a warning.
`data`	Data frame containing all observed variables used in the model.
`modes`	Character vector specifying measurement mode for each latent variable in the model. "A" for reflective mode, "B" for formative mode. If NULL (default), all modes are set to "A" (reflective).
`verbose`	Logical value, whether to display detailed intermediate processes including parsed model components and path matrix construction (default FALSE).
`...`	Other parameters passed to plspm::plspm() function.

Value

Returns the result object from plspm::plspm() function.

Examples

```
# Load the plspm package
if (requireNamespace("plspm", quietly = TRUE)) {
library(plspm)

# Example using the built-in satisfaction dataset
data(satisfaction)

# Define a lavaan-style model for customer satisfaction
model <- "
  # Measurement model (latent variable definitions)
  IMAG =~ imag1 + imag2 + imag3 + imag4 + imag5
  EXPE =~ expe1 + expe2 + expe3 + expe4 + expe5
  QUAL =~ qual1 + qual2 + qual3 + qual4 + qual5
  VAL =~ val1 + val2 + val3 + val4
  SAT =~ sat1 + sat2 + sat3 + sat4
  LOY =~ loy1 + loy2 + loy3 + loy4

  # Structural model (regression relationships)
  EXPE ~ IMAG
  QUAL ~ EXPE
  VAL ~ EXPE + QUAL
  SAT ~ IMAG + EXPE + QUAL + VAL
  LOY ~ SAT + IMAG
"

# Example 1: Run without specifying modes (will default to all "A")
result1 <- sem_plspm(model = model,
                     data = satisfaction,
                     verbose = TRUE)

# Example 2: Run with explicit modes specification
modes <- c("A", "A", "A", "A", "A", "A")
result2 <- sem_plspm(model = model,
                     data = satisfaction,
                     modes = modes,
                     verbose = TRUE)

# View summary of results
summary(result1)

# Plot the inner model path coefficients
plot(result1, what = "inner")
}
```

Index