Package: lavaanExtra (via r-universe)

October 17, 2026

Title Convenience Functions for Package 'lavaan'

Version 0.2.2.5

Date 2025-11-23

Description Affords an alternative, vector-based syntax to 'lavaan', as well as other convenience functions such as naming paths and defining indirect links automatically, in addition to convenience formatting optimized for a publication and script sharing workflow.

License MIT + file LICENSE

URL https://lavaanExtra.remi-theriault.com

BugReports https://github.com/rempsyc/lavaanExtra/issues

Depends R (>= 3.5)

Imports lavaan, insight

Suggests rempsyc (> 0.1.6), flextable, lavaanPlot, DiagrammeR, DiagrammeRsvg, rsvg, ggplot2, png, webshot2, htmlwidgets, tidySEM, tmvnsim, knitr, tibble, sjlabelled, stringdist, psych, testthat (>= 3.0.0), rmarkdown, markdown, covr, spelling, xml2

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Language en-US

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

Date/Publication 2025-12-01 01:04:29 UTC

RemoteUrl https://github.com/rempsyc/lavaanExtra

RemoteRef HEAD

RemoteSha 1019774eed779ae66df1144063d87d6a189083f9

Contents

cfa_fit_plot . . . 2
lavaan_cov . . . 3
lavaan_defined . . . 4
lavaan_extract . . . 6
lavaan_reg . . . 8
lavaan_var . . . 10
nice_fit . . . 11
nice_lavaanPlot . . . 13
nice_modindices . . . 16
nice_tidySEM . . . 17
save_plot . . . 19
write_lavaan . . . 22

Index **25**

`cfa_fit_plot` *Fit and plot CFA simultaneously*

Description

Prints and saves CFA fit, as well as plots CFA factor loadings, simultaneously.

Usage

```
cfa_fit_plot(
  model,
  data,
  covs = FALSE,
  estimator = "MLR",
  remove.items = "",
  print = TRUE,
  save.as.pdf = FALSE,
  file.name,
  ...
)
```

Arguments

`model`	CFA model to fit.
`data`	Data set on which to fit the CFA model.
`covs`	Logical, whether to include covariances on the lavaan plot.
`estimator`	What estimator to use for the CFA.
`remove.items`	Optional, if one wants to remove items from the CFA model without having to redefine it completely again.
`print`	Logical, whether to print model summary to console.

save.as.pdf	Logical, whether to save as PDF for a high-resolution, scalable vector graphic quality plot. Defaults to saving to the "/model" subfolder of the working directory. If it doesn't exist, it creates it. Then automatically open the created PDF in the default browser. Defaults to false.
file.name	Optional (when save.as.pdf is set to TRUE), if one wants something different than the default file name. It saves to pdf per default, so the .pdf extension should not be specified as it will add it automatically.
...	Arguments to be passed to function lavaan::cfa.

Value

The function returns a lavaan fit object. However, it also: prints a summary of the lavaan fit object to the console, and; prints a lavaanPlot of the lavaan fit object.

Illustrations

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

HS.model <- write_lavaan(latent = latent)
cat(HS.model)

library(lavaan)
fit <- cfa_fit_plot(HS.model, HolzingerSwineford1939)
```

---

lavaan_cov *Extract relevant covariance/correlation indices from lavaan model*

Description

Extract relevant covariance/correlation indices from lavaan lavaan::parameterEstimates and lavaan::standardizedsolution.

Usage

```
lavaan_cov(fit, nice_table = FALSE, ...)

lavaan_cor(fit, nice_table = FALSE, ...)
```

Arguments

`fit`	lavaan fit object to extract covariance indices from
`nice_table`	Logical, whether to print the table as a rempsyc::nice_table as well as print the reference values at the bottom of the table.
`...`	Arguments to be passed to rempsyc::nice_table

Value

A dataframe of covariances/correlation, including the covaried variables, the covariance/correlation, and corresponding p-value.

Functions

- `lavaan_cor()`: Shortcut for `lavaan_cov(fit, estimate = "r")`

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

(regression <- list(
  ageyr = c("visual", "textual", "speed"),
  grade = c("visual", "textual", "speed")
))

(covariance <- list(speed = "textual", ageyr = "grade"))

HS.model <- write_lavaan(
  regression = regression, covariance = covariance,
  latent = latent, label = TRUE
)
cat(HS.model)

library(lavaan)
fit <- sem(HS.model, data = HolzingerSwineford1939)
lavaan_cov(fit)
```

`lavaan_defined` *Extract relevant user-defined parameter (e.g., indirect or total effects) indices from lavaan model*

Description

Extract relevant user-defined parameters (e.g., indirect or total effects) indices from lavaan model through lavaan::parameterEstimates and lavaan::standardizedsolution.

Uncertainty for standardized coefficients: When `standardized_se = "delta"`, standard errors (SE) and confidence intervals (CI) for standardized coefficients are computed via the delta method (as in lavaan::standardizedsolution). When `standardized_se = "bootstrap"`, CIs for standardized coefficients are obtained from the bootstrap distribution of the standardized statistic (std.all) returned by lavaan::parameterEstimates with `standardized = TRUE`. In this case, lavaan reports SE for the corresponding unstandardized parameter; a bootstrap SE for standardized coefficients is not provided by lavaan. lavaanExtra preserves this behavior and labels the SE source in the output.

The default `standardized_se = "model"` chooses "bootstrap" if the fitted model used `se = "bootstrap"` (and `bootstrap > 0`), and "delta" otherwise.

Usage

```
lavaan_defined(
  fit,
  underscores_to_symbol = "→",
  lhs_name = "User-Defined Parameter",
  rhs_name = "Paths",
  standardized_se = "model",
  nice_table = FALSE,
  ...
)
```

Arguments

`fit`	lavaan fit object to extract fit indices from
`underscores_to_symbol`	Character to convert underscores to arrows in the first column, like for indirect effects. Default to the right arrow symbol, but can be set to NULL or "_", or to any other desired symbol. It is also possible to provide a vector of replacements if they they are not all the same.
`lhs_name`	Name of first column, referring to the left-hand side expression (lhs).
`rhs_name`	Name of first column, referring to the right-hand side expression (rhs).
`standardized_se`	Character string indicating the method to use for computing standard errors and confidence intervals of standardized estimates. Options are "model" (default, auto-detects based on model fitting method), "delta" (uses delta method via lavaan::standardizedsolution), or "bootstrap" (uses bootstrap method via lavaan::parameterEstimates with `standardized = TRUE`, only available when the model was fitted with bootstrap standard errors). When `standardized_se = "model"`, the function chooses "bootstrap" if the fitted model used `se = "bootstrap"` (and `bootstrap > 0`), and "delta" otherwise.
`nice_table`	Logical, whether to print the table as a rempsyc::nice_table as well as print the reference values at the bottom of the table.
`...`	Arguments to be passed to rempsyc::nice_table

Value

A dataframe, including the indirect effect ("lhs"), corresponding paths ("rhs"), standardized regression coefficient ("std.all"), corresponding p-value, as well as the unstandardized regression coefficient ("est") and its confidence interval ("ci.lower", "ci.upper"). When `standardized_se = "delta"`, standardized SE and CI are computed using the delta method. When `standardized_se = "bootstrap"`, standardized CI are computed using bootstrap and SE represents the unstandardized bootstrap SE (lavaan limitation). The SE computation method is stored as an attribute (`standardized_se_method`) for verification.

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

(mediation <- list(
  speed = "visual",
  textual = "visual",
  visual = c("ageyr", "grade")
))

(indirect <- list(
  IV = c("ageyr", "grade"),
  M = "visual",
  DV = c("speed", "textual")
))

HS.model <- write_lavaan(mediation,
  indirect = indirect,
  latent = latent, label = TRUE
)
cat(HS.model)

library(lavaan)
fit <- sem(HS.model, data = HolzingerSwineford1939)
lavaan_defined(fit, lhs_name = "Indirect Effect")
```

`lavaan_extract` *Extract relevant indices from lavaan model based on specified operator*

Description

Extract relevant indices from lavaan model through lavaan::parameterEstimates and lavaan::standardizedsolution.

Uncertainty for standardized coefficients: When `standardized_se = "delta"`, standard errors (SE) and confidence intervals (CI) for standardized coefficients are computed via the delta method (as in lavaan::standardizedsolution). When `standardized_se = "bootstrap"`, CIs for standardized coefficients are obtained from the bootstrap distribution of the standardized statistic (std.all) returned by lavaan::parameterEstimates with `standardized = TRUE`. In this case, lavaan reports SE for the corresponding unstandardized parameter; a bootstrap SE for standardized coefficients is not provided by lavaan. lavaanExtra preserves this behavior and labels the SE source in the output.

The default `standardized_se = "model"` chooses "bootstrap" if the fitted model used `se = "bootstrap"` (and `bootstrap > 0`), and "delta" otherwise.

Usage

```
lavaan_extract(
  fit,
  operator = NULL,
  lhs_name = "Left-Hand Side",
  rhs_name = "Right-Hand Side",
  underscores_to_symbol = "→",
  diag = NULL,
  standardized_se = "model",
  nice_table = FALSE,
  ...
)
```

Arguments

`fit`	lavaan fit object to extract fit indices from
`operator`	Which operator to subselect with.
`lhs_name`	Name of first column, referring to the left-hand side expression (lhs).
`rhs_name`	Name of first column, referring to the right-hand side expression (rhs).
`underscores_to_symbol`	Character to convert underscores to arrows in the first column, like for indirect effects. Default to the right arrow symbol, but can be set to NULL or "_", or to any other desired symbol. It is also possible to provide a vector of replacements if they they are not all the same.
`diag`	When extracting covariances (~~), whether to include or exclude diagonal values (one of "exclude" or "include").
`standardized_se`	Character string indicating the method to use for computing standard errors and confidence intervals of standardized estimates. Options are "model" (default, auto-detects based on model fitting method), "delta" (uses delta method via lavaan::standardizedsolution), or "bootstrap" (uses bootstrap method via lavaan::parameterEstimates with `standardized = TRUE`, only available when the model was fitted with bootstrap standard errors). When `standardized_se = "model"`, the function chooses "bootstrap" if the fitted model used `se = "bootstrap"` (and `bootstrap > 0`), and "delta" otherwise.

`nice_table`	Logical, whether to print the table as a rempsyc::nice_table as well as print the reference values at the bottom of the table.
`...`	Arguments to be passed to rempsyc::nice_table

Value

A dataframe, including the indirect effect ("lhs"), corresponding paths ("rhs"), standardized regression coefficient ("std.all"), corresponding p-value, as well as the unstandardized regression coefficient ("est") and its confidence interval ("ci.lower", "ci.upper"). When `standardized_se = "delta"`, standardized SE and CI are computed using the delta method. When `standardized_se = "bootstrap"`, standardized CI are computed using bootstrap and SE represents the unstandardized bootstrap SE (lavaan limitation). The SE computation method is stored as an attribute (`standardized_se_method`) for verification.

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

(mediation <- list(
  speed = "visual",
  textual = "visual",
  visual = c("ageyr", "grade")
))

(indirect <- list(
  IV = c("ageyr", "grade"),
  M = "visual",
  DV = c("speed", "textual")
))

HS.model <- write_lavaan(mediation,
  indirect = indirect,
  latent = latent, label = TRUE
)
cat(HS.model)

library(lavaan)
fit <- sem(HS.model, data = HolzingerSwineford1939)
lavaan_extract(fit, lhs_name = "Indirect Effect", operator = ":=")
```

`lavaan_reg` *Extract relevant regression indices from lavaan model*

Description

Extract relevant regression indices from lavaan model through lavaan::parameterEstimates and lavaan::standardizedsolution.

Uncertainty for standardized coefficients: When `standardized_se = "delta"`, standard errors (SE) and confidence intervals (CI) for standardized coefficients are computed via the delta method (as in lavaan::standardizedsolution). When `standardized_se = "bootstrap"`, CIs for standardized coefficients are obtained from the bootstrap distribution of the standardized statistic (std.all) returned by lavaan::parameterEstimates with `standardized = TRUE`. In this case, lavaan reports SE for the corresponding unstandardized parameter; a bootstrap SE for standardized coefficients is not provided by lavaan. lavaanExtra preserves this behavior and labels the SE source in the output.

The default `standardized_se = "model"` chooses "bootstrap" if the fitted model used `se = "bootstrap"` (and `bootstrap > 0`), and "delta" otherwise.

Usage

```
lavaan_reg(fit, standardized_se = "model", nice_table = FALSE, ...)
```

Arguments

`fit`	lavaan fit object to extract fit indices from
`standardized_se`	Character string indicating the method to use for computing standard errors and confidence intervals of standardized estimates. Options are "model" (default, auto-detects based on model fitting method), "delta" (uses delta method via lavaan::standardizedsolution), or "bootstrap" (uses bootstrap method via lavaan::parameterEstimates with `standardized = TRUE`, only available when the model was fitted with bootstrap standard errors). When `standardized_se = "model"`, the function chooses "bootstrap" if the fitted model used `se = "bootstrap"` (and `bootstrap > 0`), and "delta" otherwise.
`nice_table`	Logical, whether to print the table as a rempsyc::nice_table as well as print the reference values at the bottom of the table.
`...`	Arguments to be passed to rempsyc::nice_table

Value

A dataframe, including the outcome ("lhs"), predictor ("rhs"), standardized regression coefficient ("std.all"), corresponding p-value, as well as the unstandardized regression coefficient ("est") and its confidence interval ("ci.lower", "ci.upper"). When `standardized_se = "delta"`, standardized SE and CI are computed using the delta method. When `standardized_se = "bootstrap"`, standardized CI are computed using bootstrap and SE represents the unstandardized bootstrap SE (lavaan limitation). The SE computation method is stored as an attribute (`standardized_se_method`) for verification.

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
```

```
  speed = x[7:9]
))

(regression <- list(
  ageyr = c("visual", "textual", "speed"),
  grade = c("visual", "textual", "speed")
))

HS.model <- write_lavaan(latent = latent, regression = regression)
cat(HS.model)

library(lavaan)
fit <- sem(HS.model, data = HolzingerSwineford1939)
lavaan_reg(fit)
```

`lavaan_var` *Extract relevant variance indices from lavaan model*

Description

Extract relevant variance indices from lavaan model through lavaan::parameterEstimates (when estimate = "sigma", `est` column)) or lavaan::standardizedsolution (when estimate = "r2", `est.std` column). R2 values are then calculated as `1 - est.std`, and the new *p* values for the R2, with the following formula: `stats::pnorm((1 - est) / se)`.

Usage

```
lavaan_var(fit, estimate = "r2", nice_table = FALSE, ...)
```

Arguments

`fit`	lavaan fit object to extract covariance indices from
`estimate`	What estimate to use, either the standardized estimate ("r2", default), or unstandardized estimate ("sigma2").
`nice_table`	Logical, whether to print the table as a rempsyc::nice_table as well as print the reference values at the bottom of the table.
`...`	Arguments to be passed to rempsyc::nice_table

Value

A dataframe of covariances/correlation, including the covaried variables, the covariance/correlation, and corresponding p-value.

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

(regression <- list(
  ageyr = c("visual", "textual", "speed"),
  grade = c("visual", "textual", "speed")
))

(covariance <- list(speed = "textual", ageyr = "grade"))

HS.model <- write_lavaan(
  regression = regression, covariance = covariance,
  latent = latent, label = TRUE
)
cat(HS.model)

library(lavaan)
fit <- sem(HS.model, data = HolzingerSwineford1939)
lavaan_var(fit)
```

---

nice_fit *Extract relevant fit indices from lavaan model*

Description

Compares fit from one or several lavaan models. Also optionally includes references values. The reference fit values are based on Schreiber (2017), Table 3.

Usage

```
nice_fit(
  model,
  model.labels,
  nice_table = FALSE,
  guidelines = TRUE,
  stars = FALSE,
  verbose = TRUE
)
```

Arguments

model	lavaan model object(s) to extract fit indices from

model.labels	Model labels to use. If a named list is provided for model, default to the names of the list. Otherwise, if the list is unnamed, defaults to generic numbering.
nice_table	Logical, whether to print the table as a rempsyc::nice_table.
guidelines	Logical, if nice_table = TRUE, whether to display include reference values based on Schreiber (2017), Table 3, at the bottom of the table.
stars	Logical, if nice_table = TRUE, whether to display significance stars (defaults to FALSE).
verbose	Logical, whether to display messages and warnings.

Details

Note that nice_fit reports the unbiased SRMR through lavaan::lavResiduals() because the standard SRMR is upwardly biased (doi:10.1007/s1133601695527) in a noticeable way for smaller samples (thanks to James Uanhoro for this change).

If using guidelines = TRUE, please carefully consider the following 2023 quote from Terrence D. Jorgensen:

I do not recommend including cutoffs in the table, as doing so would perpetuate their misuse. Fit indices are not test statistics, and their suggested cutoffs are not critical values associated with known Type I error rates. Numerous simulation studies have shown how poorly cutoffs perform in model selection (e.g., , Jorgensen et al. (2018). Instead of test statistics, fit indices were designed to be measures of effect size (practical significance), which complement the chi-squared test of statistical significance. The range of RMSEA interpretations above is more reminiscent of the range of small/medium/large effect sizes proposed by Cohen for use in power analyses, which are as arbitrary as alpha levels, but at least they better respect the idea that (mis)fit is a matter of magnitude, not nearly so simple as "perfect or imperfect."

Value

A dataframe, representing select fit indices (chi2, df, chi2/df, p-value of the chi2 test, CFI, TLI, RMSEA and its 90% CI, unbiased SRMR, AIC, and BIC).

References

Schreiber, J. B. (2017). Update to core reporting practices in structural equation modeling. *Research in social and administrative pharmacy*, *13*(3), 634-643. doi:10.1016/j.sapharm.2016.06.006

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

(regression <- list(
  ageyr = c("visual", "textual", "speed"),
  grade = c("visual", "textual", "speed")
```

```
))

HS.model <- write_lavaan(latent = latent, regression = regression)
cat(HS.model)

library(lavaan)
fit <- sem(HS.model, data = HolzingerSwineford1939)
nice_fit(fit)
```

`nice_lavaanPlot` *Make a quick* `lavaanPlot`

Description

Make a quick and decent-looking `lavaanPlot`.

Usage

```
nice_lavaanPlot(
  model,
  node_options = list(shape = "box", fontname = "Helvetica"),
  edge_options = c(color = "black"),
  coefs = TRUE,
  stand = TRUE,
  covs = FALSE,
  stars = c("regress", "latent", "covs"),
  sig = 0.05,
  graph_options = c(rankdir = "LR"),
  title = NULL,
  note = NULL,
  fit_stats = NULL,
  fit_stats_type = c("regular", "scaled", "robust"),
  title_size = 14,
  note_size = 10,
  fit_stats_size = 9,
  wrap_width = 60,
  ...
)
```

Arguments

`model`	SEM or CFA model to plot.
`node_options`	Shape and font name.
`edge_options`	Colour of edges.
`coefs`	Logical, whether to plot coefficients. Defaults to TRUE.
`stand`	Logical, whether to use standardized coefficients. Defaults to TRUE.

covs	Logical, whether to plot covariances. Defaults to FALSE.
stars	Which links to plot significance stars for. One of c("regress", "latent", "covs").
sig	Which significance threshold to use to plot coefficients (defaults to .05). To plot all coefficients, set sig to 1.
graph_options	Read from left to right, rather than from top to bottom.
title	Optional title for the plot, positioned at the top. Plain text only; special characters like <, >, & are automatically escaped for Graphviz compatibility. Note: This will override any label or labelloc settings in graph_options.
note	Optional note or caption for the plot, positioned at the bottom when used alone, or displayed below the title with smaller font when both are provided. Plain text only; special characters are automatically escaped. Note: This will override any label or labelloc settings in graph_options.
fit_stats	Logical or character vector. If TRUE, displays a default set of fit statistics (CFI, TLI, RMSEA, SRMR) at the bottom of the plot. If a character vector, displays only the specified fit indices (e.g., c("cfi", "rmsea")). Available indices include: "chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr", "aic", "bic". Defaults to NULL (no fit statistics displayed).
fit_stats_type	Character vector specifying which types of fit statistics to display when using robust estimators. Options are "regular" (standard fit indices), "scaled" (scaled fit indices), and "robust" (robust fit indices). Defaults to c("regular", "scaled", "robust") to show all available types. Each type is displayed on a separate line. Only applicable when the model uses a robust estimator (e.g., MLR, MLM) that provides scaled/robust versions.
title_size	Numeric value specifying the font size in points for the title text. Defaults to 14.
note_size	Numeric value specifying the font size in points for the note/caption text. Defaults to 10.
fit_stats_size	Numeric value specifying the font size in points for the fit statistics text. Defaults to 9. Increase this value for larger diagrams where the default size is too small to read.
wrap_width	Numeric value or NULL. Specifies the maximum character width before text is automatically wrapped to the next line. The wrapping is intelligent, breaking at word boundaries. Defaults to 60 which works well for most plots. Set to NULL to disable automatic text wrapping. The actual wrapping is adjusted based on font size - larger fonts will wrap at proportionally fewer characters. A 20% safety margin is applied to prevent text cutoff.
...	Arguments to be passed to function lavaanPlot::lavaanPlot.

Value

A lavaanPlot, of classes c("grViz", "htmlwidget"), representing the specified lavaan model. Use save_plot() to export the plot to PNG, PDF, SVG, or JPG formats.

Illustrations

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

HS.model <- write_lavaan(latent = latent)
cat(HS.model)

library(lavaan)
fit <- cfa(HS.model, HolzingerSwineford1939)
nice_lavaanPlot(fit)

# With title and note
nice_lavaanPlot(fit, title = "Three-Factor CFA Model", note = "Holzinger-Swineford Dataset")

# With fit statistics
nice_lavaanPlot(fit, title = "Three-Factor CFA Model", fit_stats = TRUE)

# With specific fit statistics
nice_lavaanPlot(fit, fit_stats = c("cfi", "tli", "rmsea", "srmr", "chisq", "pvalue"))

# With robust estimator showing multiple fit statistic types
fit_robust <- cfa(HS.model, HolzingerSwineford1939, estimator = "MLR")
nice_lavaanPlot(fit_robust,
  title = "CFA Model", fit_stats = TRUE,
  fit_stats_type = c("regular", "scaled", "robust")
)

# With custom font sizes (useful for large diagrams)
nice_lavaanPlot(fit,
  title = "Three-Factor CFA Model",
  fit_stats = TRUE,
  title_size = 18, fit_stats_size = 12
)

# With automatic text wrapping for long titles
long_title <- "A Very Long Title That Would Otherwise Be Cut Off When Displayed"
nice_lavaanPlot(fit, title = long_title, fit_stats = TRUE, wrap_width = 60)

# Wrapping adapts to font size - larger fonts wrap at fewer characters
nice_lavaanPlot(fit, title = long_title, title_size = 18, wrap_width = 60)

# Save plot to file
## Not run:
plot <- nice_lavaanPlot(fit)
save_plot(plot, "my_plot.png") # PNG format
save_plot(plot, "my_plot.pdf") # PDF format (lossless)
save_plot(plot, "my_plot.svg") # SVG format (lossless)
save_plot(plot, "my_plot.jpg") # JPG format
```

```
## End(Not run)
```

`nice_modindices` *Extract relevant modification indices along item labels*

Description

Extract relevant modification indices along item labels, with a similarity score provided to help guide decision-making for removing redundant items with high covariance.

Usage

```
nice_modindices(fit, labels = NULL, method = "lcs", sort = TRUE, ...)
```

Arguments

`fit`	lavaan fit object to extract modification indices from
`labels`	Dataframe of labels. If the original data frame is provided, and that it contains labelled variables, will automatically attempt to extract the correct labels from the dataframe.
`method`	Method for distance calculation from stringdist::stringsim. Defaults to `"lcs"`.
`sort`	Logical. If TRUE, sort the output using the values of the modification index values. Higher values appear first. Defaults to `TRUE`.
`...`	Arguments to be passed to lavaan::modindices

Value

A dataframe, including the outcome ("lhs"), predictor ("rhs"), standardized regression coefficient ("std.all"), corresponding p-value, as well as the unstandardized regression coefficient ("est") and its confidence interval ("ci.lower", "ci.upper").

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

(regression <- list(
  ageyr = c("visual", "textual", "speed"),
  grade = c("visual", "textual", "speed")
))

HS.model <- write_lavaan(latent = latent, regression = regression)
```

```
cat(HS.model)

library(lavaan)
fit <- sem(HS.model, data = HolzingerSwineford1939)
nice_modindices(fit, maximum.number = 5)
data_labels <- data.frame(
  x1 = "I have good visual perception",
  x2 = "I have good cube perception",
  x3 = "I have good at lozenge perception",
  x4 = "I have paragraph comprehension",
  x5 = "I am good at sentence completion",
  x6 = "I excel at finding the meaning of words",
  x7 = "I am quick at doing mental additions",
  x8 = "I am quick at counting dots",
  x9 = "I am quick at discriminating straight and curved capitals"
)
nice_modindices(fit, maximum.number = 10, labels = data_labels, op = "~~")

x <- HolzingerSwineford1939
x <- sjlabelled::set_label(x, label = c(rep("", 6), data_labels))
fit <- sem(HS.model, data = x)
nice_modindices(fit, maximum.number = 10, op = "~~")
```

`nice_tidySEM` *Make a quick* `tidySEM` *plot*

Description

Make a quick and decent-looking `tidySEM` plot.

Usage

```
nice_tidySEM(
  fit,
  layout = NULL,
  hide_nonsig_edges = FALSE,
  hide_var = TRUE,
  hide_cov = FALSE,
  hide_mean = TRUE,
  est_std = TRUE,
  label,
  label_location = NULL,
  reduce_items = NULL,
  plot = TRUE,
  ...
)
```

Arguments

fit	SEM or CFA model fit to plot.
layout	A matrix (or data.frame) that describes the structure; see tidySEM::get_layout. If a named list is provided, with names "IV" (independent variables), "M" (mediator), and "DV" (dependent variables), nice_tidySEM attempts to write the layout matrix automatically.
hide_nonsig_edges	Logical, hides non-significant edges. Defaults to FALSE.
hide_var	Logical, hides variances. Defaults to TRUE.
hide_cov	Logical, hides co-variances. Defaults to FALSE.
hide_mean	Logical, hides means/node labels. Defaults to TRUE.
est_std	Logical, whether to use the standardized coefficients. Defaults to TRUE.
label	Labels to be used on the plot. As elsewhere in lavaanExtra, it is provided as a named list with format (colname = "label").
label_location	Location of label along the path, as a percentage (defaults to middle, 0.5).
reduce_items	A numeric vector of length 1 (x) or 2 (x & y) defining how much space to trim from the nodes (boxes) of the items defining the latent variables. Can be provided either as reduce_items = 0.4 (will only affect horizontal space, x), or reduce_items = c(x = 0.4, y = 0.2) (will affect both horizontal x and vertical y).
plot	Logical, whether to plot the result (default). If FALSE, returns the tidy_sem object, which can be further edited as needed.
...	Arguments to be passed to tidySEM::prepare_graph.

Value

A tidySEM plot, of class ggplot, representing the specified lavaan model. Use save_plot() to export the plot to PNG, PDF, SVG, or JPG formats.

Illustrations

Examples

```
# Calculate scale averages
library(lavaan)
data <- HolzingerSwineford1939
data$visual <- rowMeans(data[paste0("x", 1:3)])
data$textual <- rowMeans(data[paste0("x", 4:6)])
data$speed <- rowMeans(data[paste0("x", 7:9)])

# Define our variables
IV <- c("sex", "ageyr", "agemo", "school")
M <- c("visual", "grade")
DV <- c("speed", "textual")
```

```
# Define our lavaan lists
mediation <- list(speed = M, textual = M, visual = IV, grade = IV)

# Define indirect object
structure <- list(IV = IV, M = M, DV = DV)

# Write the model, and check it
model <- write_lavaan(mediation, indirect = structure, label = TRUE)
cat(model)

# Fit model
fit <- sem(model, data)

# Plot model

nice_tidySEM(fit, layout = structure)


# Save plot to file
## Not run:
plot <- nice_tidySEM(fit, layout = structure)
save_plot(plot, "my_plot.png") # PNG format
save_plot(plot, "my_plot.pdf") # PDF format

## End(Not run)
```

`save_plot` *Save a lavaanExtra plot to file*

Description

Save plots from `nice_lavaanPlot()` or `nice_tidySEM()` to various file formats including PNG, PDF, SVG, and JPG. Automatically detects the plot type and applies the appropriate saving method.

Usage

```
save_plot(
  plot,
  filename,
  width = NULL,
  height = NULL,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  use_webshot = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

plot	A plot object from nice_lavaanPlot() (class c("grViz", "htmlwidget")) or nice_tidySEM() (class ggplot).
filename	The path where the file should be saved, including the file extension (e.g., "myplot.png", "myplot.pdf", "myplot.svg", "myplot.jpg"). The file format is determined by the extension.
width	The width of the output image. Units are specified by the units parameter. If NULL (default for grViz objects), uses the SVG's natural dimensions which automatically crops to content. For ggplot objects, defaults to 7 (inches).
height	The height of the output image. Units are specified by the units parameter. If NULL (default for grViz objects), uses the SVG's natural dimensions. For ggplot objects, defaults to 5 (inches).
units	Units for width and height. One of "in" (inches, default), "cm" (centimeters), "mm" (millimeters), or "px" (pixels). For ggplot objects, this is passed directly to ggplot2::ggsave(). For grViz objects, units are converted to pixels using the dpi parameter.
dpi	Dots per inch for converting units to pixels and for raster formats (PNG/JPG). Defaults to 300. Used for unit conversion for grViz objects and passed to ggplot2::ggsave() for ggplot objects.
use_webshot	Logical. If TRUE, uses browser-based rendering via webshot2 package (headless Chrome) for pixel-perfect screenshots that match exactly what you see in the RStudio viewer. This avoids font substitution issues that can cause text misalignment with librsvg. If FALSE (default), uses rsvg library to render SVG to PNG/JPG/PDF, which may have font rendering differences compared to browser display.
verbose	Logical. If TRUE (default), prints a message indicating where the file was saved. Set to FALSE to suppress messages.
...	Additional arguments passed to the underlying save functions (ggplot2::ggsave() for ggplot objects or rsvg::rsvg_*() functions for grViz objects).

Details

For plots from nice_lavaanPlot() (grViz/htmlwidget objects):

- The plot is first converted to SVG using DiagrammeRsvg::export_svg()
- Graphviz centering attributes (ALIGN, BALIGN, labeljust, center) ensure proper centering
- By default (no width/height specified), PNG/JPG/PDF use the SVG's intrinsic viewBox dimensions to ensure pixel-perfect centering that matches the SVG output. This prevents rsvg from guessing dimensions inconsistently across formats.
- Dimension units are specified by the units parameter and converted to pixels internally
- For PNG: SVG is rendered directly to PNG using rsvg::rsvg_png()
- For JPG: SVG is rendered to raster array using rsvg::rsvg() and saved with grDevices::jpeg()
- For PDF: SVG is rendered to PDF using rsvg::rsvg_pdf()
- For SVG: The SVG string is saved directly to file

When `use_webshot = TRUE`:

- Uses a headless browser (Chrome via webshot2/chromote) to render the SVG
- Produces pixel-perfect output that matches exactly what you see in the RStudio viewer
- Avoids font substitution issues that can cause text misalignment with rsvg
- Requires the webshot2 package (`install.packages("webshot2")`)
- Supports PNG, JPG, and PDF formats

When `use_webshot = FALSE` (default):

- Uses rsvg library for SVG rendering (faster but may have font differences)
- May experience font substitution (e.g., Helvetica -> DejaVu Sans) causing text misalignment

For plots from `nice_tidySEM()` (ggplot objects):

- The plot is saved using `ggplot2::ggsave()` with the specified format and units

Value

Invisibly returns the path to the saved file.

Examples

```
# Create a simple CFA model
library(lavaan)
x <- paste0("x", 1:9)
latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
)

HS.model <- write_lavaan(latent = latent)
fit <- cfa(HS.model, HolzingerSwineford1939)

# Create and save plot in different formats
plot <- nice_lavaanPlot(fit)

# Save as PNG (default)
tmp_png <- tempfile(fileext = ".png")
save_plot(plot, tmp_png)
unlink(tmp_png)

# Save as PDF (lossless)
tmp_pdf <- tempfile(fileext = ".pdf")
save_plot(plot, tmp_pdf)
unlink(tmp_pdf)

# Save as SVG (lossless)
tmp_svg <- tempfile(fileext = ".svg")
save_plot(plot, tmp_svg)
```

```
unlink(tmp_svg)

# Save as JPG
tmp_jpg <- tempfile(fileext = ".jpg")
save_plot(plot, tmp_jpg)
unlink(tmp_jpg)

# Custom dimensions with different units
tmp_in <- tempfile(fileext = ".png")
save_plot(plot, tmp_in, width = 10, height = 7.5, units = "in")
unlink(tmp_in)

tmp_cm <- tempfile(fileext = ".pdf")
save_plot(plot, tmp_cm, width = 20, height = 15, units = "cm")
unlink(tmp_cm)

tmp_px <- tempfile(fileext = ".jpg")
save_plot(plot, tmp_px, width = 2400, height = 1800, units = "px")
unlink(tmp_px)
```

`write_lavaan` *Vector-based lavaan syntax interpreter*

Description

Vector-based lavaan syntax interpreter.

Usage

```
write_lavaan(
  mediation = NULL,
  regression = NULL,
  covariance = NULL,
  indirect = NULL,
  latent = NULL,
  intercept = NULL,
  threshold = NULL,
  constraint.equal = NULL,
  constraint.smaller = NULL,
  constraint.larger = NULL,
  custom = NULL,
  label = FALSE,
  use.letters = FALSE
)
```

Arguments

mediation	Mediation indicators (~ symbol: "is regressed on"). Differs from argument regression because path names can be optionally specified automatically with argument label.
regression	Regression indicators (~ symbol: "is regressed on").
covariance	(Residual) (co)variance indicators (~~ symbol: "is correlated with").
indirect	Indirect effect indicators (:= symbol: "indirect effect defined as"). If a named list is provided, with names "IV" (independent variables), "M" (mediator), and "DV" (dependent variables), write_lavaan attempts to write indirect effects automatically. In this case, the mediation argument must be specified too.
latent	Latent variable indicators (=~ symbol: "is measured by").
intercept	Intercept indicators (~ 1 symbol: "intercept").
threshold	Threshold indicators (| symbol: "threshold").
constraint.equal	Equality indicators (== symbol).
constraint.smaller	Smaller than indicators (< symbol).
constraint.larger	Greater than indicators (> symbol).
custom	Custom specifications. Takes a *single* string just like regular lavaan syntax would. Always added at the end of the model.
label	Logical, whether to display path names for the mediation argument.
use.letters	Logical, for the labels, whether to use letters instead of the variable names.

Value

A character string, representing the specified lavaan model.

See Also

The corresponding vignette: https://lavaanextra.remi-theriault.com/articles/write_lavaan.html

Examples

```
x <- paste0("x", 1:9)
(latent <- list(
  visual = x[1:3],
  textual = x[4:6],
  speed = x[7:9]
))

HS.model <- write_lavaan(latent = latent)
cat(HS.model)

library(lavaan)
```

```
fit <- lavaan(HS.model,
  data = HolzingerSwineford1939,
  auto.var = TRUE, auto.fix.first = TRUE,
  auto.cov.lv.x = TRUE
)
summary(fit, fit.measures = TRUE)
```

Index

cfa_fit_plot, 2

lavaan::cfa, *3*
lavaan::lavResiduals(), *12*
lavaan::modindices, *16*
lavaan::parameterEstimates, *3*, *5–7*, *9, 10*
lavaan::standardizedsolution, *3*, *5–7*, *9*, *10*
lavaan_cor *(*lavaan_cov*)*, 3
lavaan_cov, 3
lavaan_defined, 4
lavaan_extract, 6
lavaan_ind *(*lavaan_defined*)*, 4
lavaan_reg, 8
lavaan_var, 10
lavaanPlot::lavaanPlot, *14*

nice_fit, 11
nice_lavaanPlot, 13
nice_modindices, 16
nice_tidySEM, 17

rempsyc::nice_table, *4, 5*, *8–10*, *12*

save_plot, 19
save_plot(), *14*, *18*
stringdist::stringsim, *16*

tidySEM::get_layout, *18*
tidySEM::prepare_graph, *18*

write_lavaan, 22