Package 'semlbci'

July 27, 2026

Title Likelihood-Based Confidence Interval in Structural Equation Models

Version 0.12.1

Description Forms likelihood-based confidence intervals (LBCIs) for parameters in structural equation modeling, introduced in Cheung and Pesigan (2023) <doi:10.1080/10705511.2023.2183860>. Currently implements the algorithm illustrated by Pek and Wu (2018) <doi:10.1037/met0000163>, and supports the robust LBCI proposed by Falk (2018) <doi:10.1080/10705511.2017.1367254>.

URL https://sfcheung.github.io/semlbci/

BugReports https://github.com/sfcheung/semlbci/issues

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Depends R (>= 4.0.0)

Imports lavaan (>= 0.6.13), nloptr, stats, utils, MASS, ggplot2, ggrepel, rlang, pbapply, callr, methods

VignetteBuilder knitr

Config/testthat/parallel true

Config/testthat/edition 3

Config/testthat/start-first semlbci_wn_mg_*, ur_ci_bound_ur*

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Shu Fai Cheung [aut, cre] (ORCID: <https://orcid.org/0000-0002-9871-9448>), Ivan Jacob Agaloos Pesigan [ctb] (ORCID: <https://orcid.org/0000-0003-4818-8420>)

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

Repository CRAN

Date/Publication 2026-07-27 15:20:02 UTC

Contents

cfa_evar_near_zero 2
cfa_two_factors 3
cfa_two_factors_mg 4
check_sem_out 5
ci_bound_ur 7
ci_bound_ur_i 10
ci_bound_wn_i 13
ci_i_one 17
ci_order 19
confint.semlbci 21
gen_userp 22
get_cibound 25
loglike_compare 27
mediation_latent 33
mediation_latent_skewed 34
nearby_levels 35
plot.loglike_compare 36
print.cibound 38
print.semlbci 39
reg_cor_near_one 42
semlbci 43
set_constraint 46
simple_med 47
simple_med_mg 48
syntax_to_i 49

Index **51**

`cfa_evar_near_zero` *Dataset (CFA, Two Factors, One Standardized Error Variance Close to Zero)*

Description

Generated from a two-factor model, with one standardized error variance close to zero.

Usage

```
cfa_evar_near_zero
```

Format

A data frame with 120 rows and six variables, x1 to x6

Details

This model is used for examples like this one:

```
# If fitted by the following model, the standardized
# error variance of `x3` is close to zero.
# Consequently, the R-square of `x3` is close to one:

library(lavaan)
mod <- "f1 =~ x1 + x2 + x3
        f2 =~ x4 + x5 + x6"
fit <- cfa(mod, cfa_evar_near_zero)
summary(fit, standardized = TRUE, rsquare = TRUE)
```

Examples

```
print(head(cfa_evar_near_zero), digits = 3)
nrow(cfa_evar_near_zero)
```

---

cfa_two_factors *Dataset (CFA, Two Factors, Six Variables)*

Description

Generated from a two-factor model with six variables, n = 500

Usage

```
cfa_two_factors
```

Format

A data frame with 500 rows and six variables, x1 to x6.

Details

This model is used for examples like this one:

```
library(lavaan)
mod <- "f1 =~ x1 + x2 + x3
        f2 =~ x4 + x5 + x6"
fit <- cfa(mod, cfa_two_factors)
summary(fit)
```

Examples

```
print(head(cfa_two_factors), digits = 3)
nrow(cfa_two_factors)
```

`cfa_two_factors_mg` *Dataset (CFA, Two Factors, Six Variables, Two Groups)*

Description

Generated from a two-factor model with six variables, n = 500, two groups, n = 250 each.

Usage

```
cfa_two_factors_mg
```

Format

A data frame with 500 rows, one grouping variable, gp, six variables, x1 to x6.

Details

This model is used for examples like this one:

```
library(lavaan)
mod <- "f1 =~ x1 + x2 + x3
        f2 =~ x4 + x5 + x6"
fit <- cfa(mod, cfa_two_factors_mg, group = "gp")
summary(fit)
```

Examples

```
print(head(cfa_two_factors_mg), digits = 3)
nrow(cfa_two_factors_mg)
table(cfa_two_factors_mg$gp)
```

check_sem_out *Pre-analysis Check For 'semlbci'*

Description

Check the output passed to `semlbci()`

Usage

```
check_sem_out(
  sem_out,
  robust = c("none", "satorra.2000"),
  multigroup_ok = TRUE
)
```

Arguments

`sem_out`	The output from an SEM analysis. Currently only supports a lavaan::lavaan object.
`robust`	Whether the LBCI based on robust likelihood ratio test is to be found. Only "satorra.2000" in `lavaan::lavTestLRT()` is supported for now. If `"none"`, the default, then likelihood ratio test based on maximum likelihood estimation will be used.
`multigroup_ok`	If `TRUE`, will not check whether the model is a multiple-group model. Default is `TRUE`.

Details

It checks whether the model and the estimation method in the `sem_out` object passed to `semlbci()` are supported by the current version of `semlbci()`. This function is to be used by `semlbci()` but is exported such that the compatibility of an SEM output can be checked directly.

Estimation methods (`estimator` in `lavaan::lavaan()`) currently supported:

- Maximum likelihood (`ML`) and its variants (e.g., `MLM`, `MLR`). For methods with robust test statistics (e.g., `MLR`), only robust LBCIs (`robust = "satorra.2000"` in calling `semlbci()`) can be requested.

Estimation methods not yet supported:

- Generalized least squares (`GLS`).
- Weighted least squares (a.k.a. asymptotically distribution free) (`WLS`) and its variants (e.g., `WLSMV`).
- Unweighted least squares (`ULS`).
- Diagonally weighted least squares (`DWLS`).
- Other methods not listed.

Models supported:

- Single-group models with continuous variables.
- Multiple-group models with continuous variables.

Models not tested:

- Models with categorical variables.

Models not yet supported:

- Models with formative factors.
- Multilevel models.

Value

A numeric vector of one element. If 0, the model and estimation method are officially supported. If larger than zero, then the model and method are not officially supported but users can still try to use `semlbci()` on it at their own risks. If less than zero, then the model and/or the method are officially not supported.

The attributes `info` contains the reason for a value other than zero.

See Also

`semlbci()`, `ci_i_one()`

Examples

```
library(lavaan)
data(cfa_two_factors)
mod <-
"
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
"

fit <- sem(mod, cfa_two_factors)

# Should be 0
check_sem_out(fit)

fit2 <- sem(mod, cfa_two_factors, estimator = "DWLS", ordered = FALSE)

# Should be negative because DWLS is officially not supported
check_sem_out(fit2)

fit3 <- sem(mod, cfa_two_factors, estimator = "MLR")

# Should be negative because MLR is supported only if
# robust is set to "satorra.2000"
check_sem_out(fit3)

# Should be zero because robust is set to "satorra.2000"
check_sem_out(fit3, robust = "satorra.2000")
```

ci_bound_ur *Find a Likelihood-Based Confidence Bound By Root Finding*

Description

Find the lower or upper bound of the likelihood-based confidence interval (LBCI) for one parameter in a structural equation model fitted in `lavaan::lavaan()` using `uniroot()`.

Usage

```
ci_bound_ur(
  sem_out,
  func,
  ...,
  level = 0.95,
  which = c("lbound", "ubound"),
  interval = NULL,
  progress = FALSE,
  method = "uniroot",
  lrt_method = "default",
  tol = 5e-04,
  root_target = c("chisq", "pvalue"),
  d = 5,
  uniroot_extendInt = switch(which, lbound = "downX", ubound = "upX"),
  uniroot_trace = 0,
  uniroot_maxiter = 1000,
  use_callr = TRUE,
  rs = NULL
)

gen_est_i(i, sem_out, standardized = FALSE)
```

Arguments

`sem_out`	The fit object. Currently supports lavaan::lavaan objects only.
`func`	A function that receives a lavaan object and returns a scalar. This function is to be used by `gen_userp()` and so there are special requirements on it. Alternatively, it can be the output of `gen_est_i()`.
`...`	Optional arguments to be passed to `func`. Usually not used but included in case the function has such arguments.
`level`	The level of confidence of the confidence interval. Default is .95, or 95%.
`which`	Whether the lower bound or the upper bound is to be found. Must be `"lbound"` or `"ubound"`.
`interval`	A numeric vector of two values, which is the initial interval to be searched. If `NULL`, the default, it will be determined internally using Wald or delta method confidence interval, if available.

progress — Whether progress will be reported on screen during the search. Default is FALSE.

method — The actual function to be used in the search. which can only be "uniroot", the default, for now. May include other function in the future.

lrt_method — The method used in lavaan::lavTestLRT(). Default is "default". It is automatically set to "satorra.2000" and cannot be overridden if a scaled test statistic is requested in sem_out.

tol — The tolerance used in uniroot(), default is .005.

root_target — Whether the chi-square difference ("chisq"), the default, or its *p*-value ("pvalue") is used as the function value in finding the root. Should have little impact on the results.

d — A value used to determine the width of the interval in the initial search. Larger this value, *narrow* the interval. Default is 5.

uniroot_extendInt — To be passed to the argument extendInt of uniroot(). Whether the interval should be extended if the root is not found. Default value depends on the bound to be searched. Refer to the help page of uniroot() for possible values.

uniroot_trace — To be passed to the argument trace of uniroot(). How much information is printed during the search. Default is 0, and no information is printed during the search. Refer to the help page of uniroot() for possible values.

uniroot_maxiter — The maximum number of iteration in the search. Default is 1000.

use_callr — Whether the callr package will be used to do the search in a separate R process. Default is TRUE. Should not set to FALSE if used in an interactive environment unless this is intentional.

rs — Optional. If set to a persistent R process created by callr, it will be used instead of starting a new one, and it will not be terminated on exit.

i — The position of the target parameter as appeared in the parameter table of an lavaan object, generated by lavaan::parameterTable().

standardized — If TRUE, the standardized estimate is to be retrieved. Default is FALSE. Only support "std.all" for now.

Details

This function is called xby ci_bound_ur_i(). This function is exported because it is a stand-alone function that can be used directly for any function that receives a lavaan object and returns a scalar.

The function ci_bound_ur_i() is a wrapper of this function, with an interface similar to that of ci_bound_wn_i() and returns a cibound-class object. The user-parameter function is generated internally by ci_bound_wn_i().

This function, on the other hand, requires users to supply the function directly through the func argument. This provides the flexibility to find the bound for any function of the model parameter, even one that cannot be easily coded in lavaan model syntax.

Value

The function ci_bound_ur() returns a list with the following elements:

- bound: The bound found.
- optimize_out: THe output of the root finding function, uniroot() for now. (Called optimize_out because an earlier version of this function also uses optimize()).
- sem_out_bound: The lavaan model with the user-defined parameter fixed to the bound.
- lrt: The output of lavaan::lavTestLRT() comparing sem_out and sem_out_bound.
- bound_start: The Wald or delta method confidence bound returned when determining the interval internally.
- user_est: The estimate of the user-defined parameter when determining the interval internally.

The function gen_est_i() returns a special function can inspects the Model slot (and implied slot if necessary) of a modified lavaan object and return the parameter estimate. This function is to be used by ci_bound_ur() or gen_sem_out_userp().

Examples

```
library(lavaan)
data(simple_med)
dat <- simple_med
mod <-
"
m ~ x
y ~ m
"
fit_med <- lavaan::sem(mod, simple_med, fixed.x = FALSE)
parameterTable(fit_med)
# Create a function to get the second parameter
est_i <- gen_est_i(i = 2, sem_out = fit_med)
# Find the lower bound of the likelihood-based confidence interval
# of the second parameter.
# user_callr should be TRUE or omitted in read research.
# Remove interval in read research. It is added to speed up the example.
out1l <- ci_bound_ur(sem_out = fit_med,
                     func = est_i,
                     which = "lbound",
                     use_callr = FALSE,
                     interval = c(.39070, .39075))
out1l
```

ci_bound_ur_i *Likelihood-Based Confidence Bound By Root Finding*

Description

Using root finding to find the lower or upper bound of the likelihood-based confidence interval (LBCI) for one parameter in a structural equation model fitted in `lavaan::lavaan()`.

Usage

```
ci_bound_ur_i(
  i = NULL,
  npar = NULL,
  sem_out = NULL,
  f_constr = NULL,
  which = NULL,
  history = FALSE,
  perturbation_factor = 0.9,
  lb_var = -Inf,
  standardized = FALSE,
  wald_ci_start = !standardized,
  opts = list(),
  ciperc = 0.95,
  ci_limit_ratio_tol = 1.5,
  verbose = FALSE,
  sf = 1,
  sf2 = 0,
  p_tol = 5e-04,
  std_method = "internal",
  bounds = "none",
  xtol_rel_factor = 1,
  ftol_rel_factor = 1,
  lb_prop = 0.05,
  lb_se_k = 3,
  d = 5,
  ...
)
```

Arguments

i	The position of the target parameter as appeared in the parameter table of a lavaan object, generated by `lavaan::parameterTable()`.
npar	Ignored by this function. Included consistency in the interface.
sem_out	The fit object. Currently supports lavaan::lavaan objects only.
f_constr	Ignored by this function. Included consistency in the interface.

`which` Whether the lower bound or the upper bound is to be found. Must be `"lbound"` or `"ubound"`.

`history` Not used. Kept for backward compatibility.

`perturbation_factor` Ignored by this function. Included consistency in the interface.

`lb_var` Ignored by this function. Included consistency in the interface.

`standardized` If `TRUE`, the LBCI is for the requested estimate in the standardized solution. Default is `FALSE`.

`wald_ci_start` Ignored by this function. Included consistency in the interface.

`opts` Options to be passed to `stats::uniroot()`. Default is `list()`.

`ciperc` The intended coverage probability for the confidence interval. Default is .95, and the bound for a 95% confidence interval will be sought.

`ci_limit_ratio_tol` The tolerance for the ratio of `a` to `b`, where `a` is the distance between an LBCI limit and the point estimate, and the `b` is the distance between the original confidence limit (by default the Wald CI in `lavaan::lavaan()`) and the point estimate. If the ratio is larger than this value or smaller than the reciprocal of this value, a warning is set in the status code. Default is 1.5.

`verbose` If `TRUE`, the function will store more diagnostic information in the attribute `diag`. Default is `FALSE`.

`sf` Ignored by this function. Included consistency in the interface.

`sf2` Ignored by this function. Included consistency in the interface.

`p_tol` Tolerance for checking the achieved level of confidence. If the absolute difference between the achieved level and `ciperc` is greater than this amount, a warning is set in the status code and the bound is set to `NA`. Default is 5e-4.

`std_method` The method used to find the standardized solution. If equal to `"lavaan"`, `lavaan::standardizedSoluti` will be used. If equal to `"internal"`, an internal function will be used. The `"lavaan"` method should work in all situations, but the `"internal"` method is usually much faster. Default is `"internal"`.

`bounds` Ignored by this function. Included consistency in the interface.

`xtol_rel_factor` Ignored by this function. Included consistency in the interface.

`ftol_rel_factor` Ignored by this function. Included consistency in the interface.

`lb_prop` Ignored by this function. Included consistency in the interface.

`lb_se_k` Ignored by this function. Included consistency in the interface.

`d` A value used to determine the width of the interval in the initial search. Larger this value, *narrow* the interval. Default is 5. Used by `ci_bound_ur()`.

`...` Optional arguments. Not used.

Details

Important Notice:

This function is not supposed to be used directly by users in typical scenarios. Its interface is user-*unfriendly* because it should be used through `semlbci()`. It is exported such that interested users can examine how a confidence bound is found, or use it for experiments or simulations.

Usage:

This function is the lowest level function used by `semlbci()`. `semlbci()` calls this function once for each bound of each parameter.

For consistency in the interface, most of the arguments in `ci_bound_wn_i()` are also included in this function, even those not used internally.

Algorithm:

This function, unlike `ci_bound_wn_i()`, use a simple root finding algorithm. Basically, it tries fixing the target parameter to different values until the likelihood ratio test p-value, or the corresponding chi-square difference, is equal to the value corresponding to the desired level of confidence. (Internally, the difference between the p-value and the target p-value, that for the chi-square difference, is the function value.)

For finding the bound, this algorithm can be inefficient compared to the one proposed by Wu and Neale (2012). The difference can be less than one second versus 10 seconds. It is included as a backup algorithm for parameters which are difficult for the method by Wu and Neale.

Internally, it uses `uniroot()` to find the root.

Limitation(s):

This function does not handle an estimate close to an attainable bound using the method proposed by Wu and Neale (2012). Use it for such parameters with cautions.

Value

A `cibound`-class object which is a list with three elements:

- `bound`: A single number. The value of the bound located. `NA` is the search failed for various reasons.
- `diag`: A list of diagnostic information.
- `call`: The original call.

A detailed and organized output can be printed by the default print method (`print.cibound()`).

References

Wu, H., & Neale, M. C. (2012). Adjusted confidence intervals for a bounded parameter. *Behavior Genetics, 42*(6), 886-898. doi:10.1007/s105190129560z

See Also

`print.cibound()`, `semlbci()`, `ci_i_one()`; see `ci_bound_wn_i()` on the version for the method by Wu and Neale (2012).

Examples

```
library(lavaan)
data(simple_med)
dat <- simple_med
mod <-
"
m ~ x
y ~ m
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
# Remove `opts` in real cases.
# The options are added just to speed up the example
out1l <- ci_bound_ur_i(i = 1,
                       sem_out = fit_med,
                       which = "lbound",
                       opts = list(use_callr = FALSE,
                                   interval = c(0.8277, 0.8278)))
out1l
```

---

ci_bound_wn_i *Likelihood-based Confidence Bound By Wu-Neale-2012*

Description

User the method proposed by Wu and Neale (2012) to find the lower or upper bound of the likelihood-based confidence interval (LBCI) for one parameter in a structural equation model fitted in `lavaan::lavaan()`.

Usage

```
ci_bound_wn_i(
  i = NULL,
  npar = NULL,
  sem_out = NULL,
  f_constr = NULL,
  which = NULL,
  history = FALSE,
  perturbation_factor = 0.9,
  lb_var = -Inf,
  standardized = FALSE,
  wald_ci_start = !standardized,
  opts = list(),
  ciperc = 0.95,
  ci_limit_ratio_tol = 1.5,
  verbose = FALSE,
  sf = 1,
  sf2 = 0,
  p_tol = 5e-04,
```

```
  std_method = "internal",
  bounds = "none",
  xtol_rel_factor = 1,
  ftol_rel_factor = 1,
  lb_prop = 0.05,
  lb_se_k = 3,
  try_harder = 0,
  fit_lb = -Inf,
  fit_ub = +Inf,
  timeout = 300,
  ...
)
```

Arguments

i	The position of the target parameter as appeared in the parameter table of an lavaan object, generated by lavaan::parameterTable().
npar	The number of free parameters, including those constrained to be equal.
sem_out	The fit object. Currently supports lavaan::lavaan objects only.
f_constr	The constraint function generated by set_constraint().
which	Whether the lower bound or the upper bound is to be found. Must be "lbound" or "ubound".
history	Not used. Kept for backward compatibility.
perturbation_factor	A number multiplied to the parameter estimates in sem_out. Using the parameter estimates as starting values may lead to errors in the first few iterations. Default is .90. This argument is ignored if wald_ci_start is 'TRUE.
lb_var	The lower bound for free parameters that are variances. If equal to -Inf, the default, lb_prop and lb_se_k will be used to set the lower bounds for free variances. If it is a number, it will be used to set the lower bounds for all free variances.
standardized	If TRUE, the LBCI is for the requested estimate in the standardized solution. Default is FALSE.
wald_ci_start	If TRUE, there are no equality constraints in the model, and the target parameter is not a user-defined parameter, the Wald confidence bounds will be used as the starting value.
opts	Options to be passed to nloptr::nloptr(), the current optimizer. Default is list().
ciperc	The intended coverage probability for the confidence interval. Default is .95, and the bound for a 95% confidence interval will be sought.
ci_limit_ratio_tol	The tolerance for the ratio of a to b, where a is the distance between an LBCI limit and the point estimate, and the b is the distance between the original confidence limit (by default the Wald CI in lavaan::lavaan()) and the point estimate. If the ratio is larger than this value or smaller than the reciprocal of this value, a warning is set in the status code. Default is 1.5.

verbose	If TRUE, the function will store more diagnostic information in the attribute diag. Default is FALSE.
sf	A scaling factor. Used for robust confidence bounds. Default is 1. Computed by an internal function called by semlbci() when robust = "satorra.2000".
sf2	A shift factor. Used for robust confidence bounds. Default is 0. Computed by an internal function called by semlbci() when robust = "satorra.2000".
p_tol	Tolerance for checking the achieved level of confidence. If the absolute difference between the achieved level and ciperc is greater than this amount, a warning is set in the status code and the bound is set to NA. Default is 5e-4.
std_method	The method used to find the standardized solution. If equal to "lavaan", lavaan::standardizedSoluti will be used. If equal to "internal", an internal function will be used. The "lavaan" method should work in all situations, but the "internal" method is usually much faster. Default is "internal".
bounds	Default is "" and this function will set the lower bounds to lb_var for variances. Other valid values are those accepted by lavaan::lavaan(). Ignored for now.
xtol_rel_factor	Multiply the default xtol_rel by a number, usually a positive number equal to or less than 1, to change the default termination criterion. Default is 1.
ftol_rel_factor	Multiply the default ftol_rel by a number, usually a positive number equal to or less than 1, to change the default termination criterion. Default is 1.
lb_prop	Used by an internal function to set the lower bound for free variances. Default is .05, setting the lower bound to .05 * estimate. Used only if the lower bound set by lb_se_k is negative.
lb_se_k	Used by an internal function to set the lower bound for free variances. Default is 3, the estimate minus 3 standard error. If negative, the lower bound is set using lb_prop.
try_harder	If error occurred in the optimization, how many more times to try. In each new attempt, the starting values will be randomly jittered. Default is 0.
fit_lb	The vector of lower bounds of parameters. Default is -Inf, setting the lower bounds to -Inf for all parameters except for free variances which are controlled by lb_var.
fit_ub	The vector of upper bounds of parameters. Default is +Inf, setting the lower bounds to +Inf for all parameters.
timeout	The approximate maximum time for the search, in second. Default is 300 seconds (5 minutes).
...	Optional arguments. Not used.

Details

Important Notice:

This function is not supposed to be used directly by users in typical scenarios. Its interface is user-*unfriendly* because it should be used through semlbci(). It is exported such that interested users can examine how a confidence bound is found, or use it for experiments or simulations.

Usage:

This function is the lowest level function used by `semlbci()`. `semlbci()` calls this function once for each bound of each parameter. To use it, `set_constraint()` needs to be called first to create the equality constraint required by the algorithm proposed by Wu and Neale (2012).

Algorithm:

This function implements the algorithm presented in Wu and Neale (2012; see also Pek & Wu, 2015, Equation 12) that estimates all free parameters in the optimization.

Limitation(s):

This function does not yet implement the method by Wu and Neale (2012) for an estimate close to an attainable bound.

Value

A `cibound`-class object which is a list with three elements:

- `bound`: A single number. The value of the bound located. `NA` is the search failed for various reasons.
- `diag`: A list of diagnostic information.
- `call`: The original call.

A detailed and organized output can be printed by the default print method (`print.cibound()`).

References

Pek, J., & Wu, H. (2015). Profile likelihood-based confidence intervals and regions for structural equation models. *Psychometrika, 80*(4), 1123-1145. doi:10.1007/s1133601594611

Wu, H., & Neale, M. C. (2012). Adjusted confidence intervals for a bounded parameter. *Behavior Genetics, 42*(6), 886-898. doi:10.1007/s105190129560z

See Also

`print.cibound()`, `semlbci()`, `ci_i_one()`

Examples

```
data(simple_med)
dat <- simple_med

mod <-
"
m ~ x
y ~ m
"

fit_med <- lavaan::sem(mod, simple_med, fixed.x = FALSE)

fn_constr0 <- set_constraint(fit_med)
```

```
out1l <- ci_bound_wn_i(i = 1,
                       npar = 5,
                       sem_out = fit_med,
                       f_constr = fn_constr0,
                       which = "lbound")
out1l
```

---

ci_i_one *Likelihood-Based Confidence Bound for One Parameter*

Description

Find the likelihood-based confidence bound for one parameter.

Usage

```
ci_i_one(
  i,
  which = NULL,
  sem_out,
  method = c("wn", "ur"),
  standardized = FALSE,
  robust = "none",
  sf_full = NA,
  sf_args = list(),
  sem_out_name = NULL,
  try_k_more_times = 0,
  ...
)
```

Arguments

i	The position (row number) of the target parameters as appeared in the parameter table of the lavaan::lavaan object.
which	Whether the lower bound or the upper bound is to be found. Must be "lbound" or "ubound".
sem_out	The SEM output. Currently supports lavaan::lavaan outputs only.
method	The approach to be used. Default is "wn" (Wu-Neale-2012 Method). Another method is "ur", root finding by stats::uniroot().
standardized	Logical. Whether the bound of the LBCI of the standardized solution is to be searched. Default is FALSE.
robust	Whether the LBCI based on robust likelihood ratio test is to be found. Only "satorra.2000" in lavaan::lavTestLRT() is supported for now. If "none", the default, then likelihood ratio test based on maximum likelihood estimation will be used. For "ur", "satorra.2000" is automatically used if a scaled test statistic is requested in sem_out.

sf_full
: A list with the scaling and shift factors. Ignored if robust is "none". If robust is "satorra.2000" and sf_full is supplied, then its value will be used. If robust is "satorra.2000" but sf_full is NA, then scaling factors will be computed internally.

sf_args
: The list of arguments to be used for computing scaling factors if robust is "satorra.2000". Used only by semlbci(). Ignored if robust is not "satorra.2000".

sem_out_name
: The name of the object supplied to sem_out. NULL by default. Originally used by some internal functions. No longer used in the current version but kept for backward compatibility.

try_k_more_times
: How many more times to try if the status code is not zero. Default is 0.

...
: Arguments to be passed to the function corresponds to the requested method (ci_bound_wn_i() for "wn").

Details

Important Notice:

This function is not supposed to be used directly by users in typical scenarios. Its interface is user-*unfriendly* because it should be used through semlbci(). It is exported such that interested users can examine how a confidence bound is found, or use it for experiments or simulations.

Usage:

ci_i_one() is the link between semlbci() and the lowest level function (currently ci_bound_wn_i()). When called by semlbci() to find the bound of a parameter, ci_i_one() calls a function (ci_bound_wn_i() by default) one or more times to find the bound (limit) for a likelihood-based confidence interval.

Value

A list of the following elements.

- bound: The bound located. NA if the search failed.
- diags: Diagnostic information.
- method: Method used. Currently only "wn" is the only possible value.
- times: Total time used in the search.
- sf_full: The scaling and shift factors used.
- ci_bound_i_out: The original output from ci_bound_wn_i().
- attempt_lb_var: How many attempts used to reduce the lower bounds of free variances.
- attempt_more_times: How many additional attempts used to search for the bounds. Controlled by try_k_more_times.

See Also

semlbci(), ci_bound_wn_i()

Examples

```
data(simple_med)

library(lavaan)
mod <-
"
m ~ x
y ~ m
"
fit_med <- lavaan::sem(mod, simple_med, fixed.x = FALSE)

parameterTable(fit_med)

# Find the LBCI for the first parameter
# The method "wn" needs the constraint function.
# Use set_constraint() to generate this function:
fn_constr0 <- set_constraint(fit_med)

# Call ci_i to find the bound, the lower bound in this example.
# The constraint function, assigned to f_constr, is passed
# to ci_bound_wn_i().
# npar is an argument for ci_bound_wn_i().
out <- ci_i_one(i = 1,
                which = "lbound",
                sem_out = fit_med,
                npar = 5,
                f_constr = fn_constr0)
out$bounds
```

---

ci_order *Check The Order of Bounds in a List of* semlbci *Objects*

Description

Check whether the LBCIs in a list of semlbci-class of objects are consistent with their levels of confidence.

Usage

```
ci_order(semlbci_list)

## S3 method for class 'ci_order'
print(x, digits = 3, ...)
```

Arguments

semlbci_list	An object of class semlbci_list, such as the output of nearby_levels().
x	The output of ci_order().

digits	The number of decimal places in the printout.
...	Additional arguments. Not used.

Value

A ci_order-class object with a print method print.ci_order(). The number of rows is equal to the number of parameters in semlbci_list, and the columns stores the confidence limits from the list, ordered according to the level of confidence.

x is returned invisibly. Called for its side effect.

Methods (by generic)

- print(ci_order): The print method of the output of ci_order().

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

See Also

nearby_levels(), semlbci()

Examples

```
library(lavaan)
mod <-
"
m ~ x
y ~ m
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
lbci_fit <- semlbci(fit_med)
lbci_fit_nb <- nearby_levels(lbci_fit,
                             ciperc_levels = c(-.050, .050))

# Check the order of the confidence bounds.
# A confidence interval with a higher level of confidence
# should enclose a confidence interval with
# a lower level of confidence.
ci_order(lbci_fit_nb)
```

confint.semlbci *Confidence Intervals for a 'smelbci' Object*

Description

Return the confidence intervals of the parameters in the output of `semlbci()`.

Usage

```
## S3 method for class 'semlbci'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	The output of `semlbci()`.
`parm`	The parameters for which the confidence intervals are returned. Not used because parameters are defined by three or more columns (`lhs`, `op`, `rhs`, and `group` for multisample models).
`level`	Ignored. The level of confidence is determined when calling `semlbci()` and cannot be changed.
`...`	Optional arguments. Ignored.

Details

It returns the likelihood-based confidence intervals in the output of `semlbci()`.

Value

A two-column matrix of the confidence intervals.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

See Also

`semlbci()`

Examples

```
library(lavaan)
mod <-
"
m ~ a*x
y ~ b*m
ab := a * b
"
```

```
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
p_table <- parameterTable(fit_med)
p_table
lbci_med <- semlbci(fit_med,
                    pars = "ab :=")
lbci_med

confint(lbci_med)
```

---

gen_userp *Create a Wrapper To Be Used in 'lavaan' Models*

Description

Make a function on lavaan object usable in a lavaan model syntax.

Usage

```
gen_userp(func, sem_out)

gen_sem_out_userp(
  userp,
  sem_out,
  userp_name = "semlbciuserp1234",
  fix = TRUE,
  control_args = list(),
  iter.max = 10000,
  max_attempts = 5
)
```

Arguments

func	A function that receives a lavaan-object and returns a scalar. See Details on the restriction on this function.
sem_out	A lavaan-class object to be modified.
userp	A function that is generated by gen_userp().
userp_name	The name of the function userp to be used in the lavaan model. It does not have to be the name of the function in userp. Should be changed only if it conflicts with another object in the parent environment, which should not happen if the model is always fitted in a clean R session.
fix	If TRUE, the default, the function generated is used to fix the value of userp to a target value using an equality constraint. If FALSE, then the function simply fits the model to the data.
control_args	To be passed to the argument of the same name in lavaan::lavaan(). Default is list(). Can be used to set the default values of this argument in the generated function.

iter.max	The maximum number of iteration when the generated function fit the model. Default is 10000.
max_attempts	If the initial fit with the equality constraint fails, how many more attempts will be made by the generated function. Default is 5.

Details

gen_userp:

There are cases in which we want to create a user parameter which is a function of other free parameters, computed by a function. However such a function may work only on a lavaan object.

If the target function works by extracting parameter estimates stored in the Model slot and/or the implied slot, then gen_userp() can be used to convert it to a function that retrieves the parameter estimates when being called by lavaan::lavaan() or its wrappers, modifies the stored lavaan object using lavaan::lav_model_set_parameters() and lavaan::lav_model_implied() to change the estimates, and call the target function.

Note that this is an unconventional way to define a user parameter and the generated function should always be checked to see whether it works as expected.

As shown in the examples, the parameter computed this may not have standard error nor p-value.

The main purpose is for the point estimate, for searching the likelihood-based confidence bound using ci_bound_ur() and ci_bound_ur_i().

Note that the target function specified in func should work directly on the parameter estimates stored in the Model slot and then get the estimates using lavaan::lav_model_get_parameters(). Functions that work on the unmodified output generated by lavaan::lavaan() usually do not work.

Users are not recommended to use gen_userp() and gen_sem_out_userp() directly because they require unconventional way to extract parameter estimates from a lavaan model. However, developers may use them to include functions they wrote in a lavaan model. This is the technique used by ci_bound_ur_i() to constrain any parameter in a model to an arbitrary value.

gen_sem_out_userp:

The function gen_sem_out_userp() is to be used internally for generating a function for searching a likelihood-based confidence bound. It is exported because it needs to be run in an fresh external R process, usually created by callr in other internal functions.

Value

gen_userp:

It returns a function that accepts a numeric vector of length equals to the number of free parameters in sem_out, and returns a scalar which is the output of func. If this vector is not supplied, it will try to find it in the parent.frame(). This is how it works inside a lavaan model.

gen_sem_out_userp:

If fix is TRUE, it returns a function with these arguments:

- target: The value to which the user-defined parameter will be fixed to.
- verbose: If TRUE, additional information will be printed when fitting the model.
- control: The values to be passed as a list to the argument of the same name in lavaan::lavaan().

- seed: Numeric. If supplied, it will be used in set.seed() to initialize the random number generator. Necessary to reproduce some results because random numbers are used in some steps in lavaan. If NULL, the default, set.seed() will not be called.

If fix is 'FALSE, then it returns a function with optional arguments that will be ignored, Calling it will simply fit the modified model to the data. Useful for getting the value of the user-defined parameter.

Examples

```
library(lavaan)

data(simple_med)
dat <- simple_med
mod <-
"
m ~ a*x
y ~ b*m
ab := a*b
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
parameterEstimates(fit_med)

# A trivial example for verifying the results
my_ab <- function(object) {
    # Need to use lav_model_get_parameters()
    # because the object is only a modified
    # lavaan-object, not one directly
    # generated by lavaan function
    est <- lavaan::lav_model_get_parameters(object@Model, type = "user")
    unname(est[1] * est[2])
  }

# Check the function
my_ab(fit_med)
coef(fit_med, type = "user")["ab"]

# Create the function
my_userp <- gen_userp(func = my_ab,
                      sem_out = fit_med)
# Try it on the vector of free parameters
my_userp(coef(fit_med))

# Generate a modified lavaan model
fit_userp <- gen_sem_out_userp(userp = my_userp,
                               userp_name = "my_userp",
                               sem_out = fit_med)

# This function can then be used in the model syntax.

# Note that the following example only work when called inside the
# workspace or inside other functions such as ci_bound_ur()`
# and `ci_bound_ur_i()` because `lavaan::sem()` will
```

```
# search `my_userp()` in the global environment.

# Therefore, the following lines are commented out.
# They should be run only in a "TRUE" interactive
# session.

# mod2 <-
# "
# m ~ x
# y ~ m
# ab := my_userp()
# "
# fit_med2 <- sem(mod2, simple_med, fixed.x = FALSE)
# parameterEstimates(fit_med2)
#
# # Fit the model with the output of the function, a*b
# # fixed to .50
#
# fit_new <- fit_userp(.50)
#
# # Check if the parameter ab is fixed to .50
# parameterEstimates(fit_new)
```

`get_cibound` *A 'cibound' Output From a 'semlbci' Object*

Description

Get the `cibound` output of a bound from a `semlbci` object, the output of `semlbci()`.

Usage

```
get_cibound(x, row_id, which = c("lbound", "ubound"))

get_cibound_status_not_0(x)
```

Arguments

`x`	The output of `semlbci()`.
`row_id`	The row number in `x`. Should be the number on the left, not the actual row number, because some rows may be omitted in the printout of `x`.
`which`	The bound for which the `ci_bound_wn_i()` is to be extracted. Either `"lbound"` or `"ubound"`.

Details

The function get_cibound() returns the original output of ci_bound_wn_i() for a bound. Usually for diagnosis.

The function get_cibound_status_not_0() checks the status code of each bound, and returns the cibound outputs of bounds with status code not equal to zero (i.e., something wrong in the search). Printing it can print the diagnostic information for all bounds that failed in the search.

Value

get_cibound() returns a cibound-class object. See ci_bound_wn_i() for details. get_cibound_status_not_0() returns a list of cibound-class objects with status not equal to zero. If all bounds have status equal to zero, it returns an empty list.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

See Also

semlbci()

Examples

```
library(lavaan)
mod <-
"
m ~ a*x
y ~ b*m
ab := a * b
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
p_table <- parameterTable(fit_med)
p_table
lbci_med <- semlbci(fit_med,
                    pars = c("ab :="))
lbci_med

# Get the output of ci_bound_wn_i() of the lower
# bound of the LBCI for the indirect effect:
get_cibound(lbci_med, row_id = 6, which = "lbound")

# Get the output of ci_bound_wn_i() of the upper
# bound of the LBCI for the indirect effect:
get_cibound(lbci_med, row_id = 6, which = "ubound")
```

loglike_compare *Log Profile likelihood of a Parameter*

Description

These functions compute the log profile likelihood of a parameter when it is fixed to a value or a range of values

Usage

```
loglike_compare(
  sem_out,
  semlbci_out = NULL,
  par_i,
  confidence = 0.95,
  n_points = 21,
  start = "default",
  try_k_more = 5,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  use_pbapply = TRUE
)

loglike_range(
  sem_out,
  par_i,
  confidence = 0.95,
  n_points = 21,
  interval = NULL,
  verbose = FALSE,
  start = "default",
  try_k_more = 5,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  use_pbapply = TRUE
)

loglike_point(
  theta0,
  sem_out,
  par_i,
  verbose = FALSE,
  start = "default",
  try_k_more = 5
)

loglike_quad_range(
```

```
  sem_out,
  par_i,
  confidence = 0.95,
  n_points = 21,
  interval = NULL,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  use_pbapply = TRUE,
  try_k_more = 5,
  start = "default"
)

loglike_quad_point(theta0, sem_out, par_i)

loglike_compare_ur(
  sem_out,
  semlbci_out = NULL,
  par_i,
  confidence = 0.95,
  n_points = 21,
  standardized = FALSE,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  use_pbapply = TRUE,
  loadbalancing = TRUE
)

loglike_range_ur(
  sem_out,
  par_i,
  standardized = FALSE,
  confidence = 0.95,
  n_points = 21,
  interval = NULL,
  verbose = FALSE,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  use_pbapply = TRUE,
  loadbalancing = TRUE
)

loglike_point_ur(theta0, sem_out, par_i, standardized = FALSE, verbose = FALSE)

loglike_quad_range_ur(
  sem_out,
  par_i,
  confidence = 0.95,
  standardized = FALSE,
```

```
  n_points = 21,
  interval = NULL,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  use_pbapply = TRUE,
  loadbalancing = TRUE
)

loglike_quad_point_ur(theta0, sem_out, par_i, standardized = FALSE)
```

Arguments

sem_out	The SEM output. Currently the outputs of lavaan::lavaan() or its wrappers, such as lavaan::sem() and lavaan::cfa() are supported.
semlbci_out	The output of semlbci(). If supplied, it will extract the likelihood-based confidence interval from the output. If not, it will call semlbci().
par_i	The row number of the parameter in the output of lavaan::parameterTable(). Can also be a lavaan::model.syntax specification for a parameter, e.g., "y ~ x" or ab := . It will be converted to the row number by syntax_to_i(). Refer to syntax_to_i() for details.
confidence	The level of confidence of the Wald-type confidence interval. If interval is NULL, this confidence is used to form the interval.
n_points	The number of points to be evaluated in the interval. Default is 21.
start	How the start values are set in lavaan::lavaan(). See lavaan::lavOptions() on this argument. Default is "default". If the plot is too irregular, try setting it to "simple".
try_k_more	How many more times to try finding the p-values, by randomizing the starting values. Default is 5. Try increasing this number if the plot is too irregular.
parallel	If TRUE, parallel processing will be used. A cluster will be created by parallel::makeCluster(), with the number of workers equal to ncpus. Parallel processing, though not enabled by default, is recommended because it can speed up the computation a lot.
ncpus	The number of workers if parallel is TRUE. Default is parallel::detectCores(logical = FALSE) - 1, the number of physical cores minus 1.
use_pbapply	If TRUE and pbapply::pbapply is installed, pbapply::pbapply will be used to display the progress in computing the log profile likelihood. Default is TRUE.
interval	A vector of numbers. If provided and has two elements, this will be used as the end points of the interval. If it has more than two elements, the elements will be used directly to form the values in the interval. Default is NULL.
verbose	Whether some diagnostic information will be printed. Default is FALSE.
theta0	The value at which the parameter is fixed to.
standardized	Logical. Whether the parameter requested is in the standardized solution. Default is FALSE.
loadbalancing	Logical. When using parallel processing, whether load balancing is used. Default is TRUE.

Details

It uses the methods presented in Pawitan (2013) to compute and visualize the log profile likelihood of a parameter in a structural equation model when this parameter is fixed to a value or a range of values. `loglike_range()` and `loglike_point()` compute the so-called "true" log profile likelihood, while `loglike_quad_range()` and `loglike_quad_point()` approximate the log profile likelihood by a quadratic function.

These functions are for creating illustrative examples and learning only, not for research use. Therefore, they are not as versatile as `semlbci()` in the types of models and parameters supported. They can be used for free parameters and user-defined parameters not involved in any constraints. Only a model fitted by maximum likelihood is supported.

They will not check whether the computation is appropriate for a model. It is the responsibility of the users to ensure that the computation is appropriate for the model and parameter.

In version 0.11.2.1, added variants of the function, with suffix `_ur`, which use root finding (`"ur"` in `semlbci()`). These variants are slower to run but can be used for parameter in the standardized solution. Therefore, they can used to generate plots for parameters such as standardized regression paths and correlations.

Value

`loglike_compare()` calls `loglike_range()` and `loglike_quad_range()` and returns their results in a `loglike_compare`-class object, a list with these elements:

- `quadratic`: The output of `loglike_quad_range()`.
- `loglikelihood`: The output of `loglike_range()`.
- `pvalue_quadratic`: The likelihood ratio test p-values at the quadratic approximation confidence bounds.
- `pvalue_loglikelihood`: The likelihood ratio test p-values at the likelihood-based confidence bounds.
- `est`: The point estimate of the parameter in `sem_out`.

`loglike_compare`-class object has a `plot` method (`plot.loglike_compare()`) that can be used to plot the log profile likelihood.

`loglike_point()` returns a list with these elements:

- `loglike`: The log profile likelihood of the parameter when it is fixed to `theta0`.
- `pvalue`: The p-values based on the likelihood ratio difference test between the original model and the model with the parameter fixed to `theta0`.
- `fit`: A lavaan::lavaan object. The original model with the parameter fixed to `theta0`.
- `lrt`: The output of `lavaan::lavTestLRT()`, comparing the original model to the model with the parameter fixed to `theta0`.

`loglike_quad_range()` returns a data frame with these columns:

- `theta`: The values to which the parameter is fixed to.
- `loglike`: The log profile likelihood values of the parameter using quadratic approximation.

- pvalue: The *p*-values based on the likelihood ratio difference test between the original model and the model with the parameter fixed to theta.

loglike_quad_point() returns a single number of the class lavaan.vector (because it is the output of lavaan::fitMeasures()). This number is the quadratic approximation of the log profile likelihood when the parameter is fixed to theta0.

loglike_range() returns a data frame with these columns:

- theta: The values to which the parameter is fixed to.
- loglike: The log profile likelihood at theta.
- pvalue: The *p*-values based on the likelihood ratio difference test between the original model and model with the parameter fixed to theta.

Functions

- loglike_compare(): Generates points for log profile likelihood and quadratic approximation, by calling the helper functions loglike_range() and loglike_quad_range().
- loglike_range(): Find the log profile likelihood for a range of values.
- loglike_point(): Find the log likelihood at a value.
- loglike_quad_range(): Find the approximated log likelihood for a range of values.
- loglike_quad_point(): Find the approximated log likelihood at a value.
- loglike_compare_ur(): Generates points for log profile likelihood and quadratic approximation using root finding, by calling the helper functions loglike_range_ur() and loglike_quad_range_ur().
- loglike_range_ur(): Find the log profile likelihood for a range of values using root finding.
- loglike_point_ur(): Find the log likelihood at a value.
- loglike_quad_range_ur(): Find the approximated log likelihood for a range of values using root finding.
- loglike_quad_point_ur(): Find the approximated log likelihood at a value. Support a parameter in the standardized solution.

References

Pawitan, Y. (2013). *In all likelihood: Statistical modelling and inference using likelihood*. Oxford University Press.

See Also

plot.loglike_compare()

Examples

```
## loglike_compare

library(lavaan)
data(simple_med)
dat <- simple_med
mod <-
```

```
"
m ~ a * x
y ~ b * m
ab := a * b
"
fit <- lavaan::sem(mod, simple_med, fixed.x = FALSE)

# 4 points are used just for illustration
# At least 21 points should be used for a smooth plot
# Remove try_k_more in real applications. It is set
# to zero such that this example does not take too long to run.
# use_pbapply can be removed or set to TRUE to show the progress.
ll_a <- loglike_compare(fit, par_i = "m ~ x", n_points = 4,
                        try_k_more = 0,
                        use_pbapply = FALSE)
plot(ll_a)

# See the vignette "loglike" for an example for the
# indirect effect.


## loglike_range

# Usually not to be used directly.
# Used by loglike_compare().
# 3 points are used just for illustration
ll_1 <- loglike_range(fit, par_i = "y ~ m", n_points = 2)
head(ll_1)


## loglike_point

# Usually not to be used directly.
# Used by loglike_compare().
llp_1 <- loglike_point(theta0 = 0.3, sem_out = fit, par_i = "y ~ m")
llp_1$loglike
llp_1$pvalue
llp_1$lrt



## loglike_quad_range

# Usually not to be used directly.
# Used by loglike_compare().
# 2 points are used just for illustration
lq_1 <- loglike_quad_range(fit, par_i = "y ~ m", n_points = 2)
head(lq_1)



## loglike_quad_point
```

```
# Usually not to be used directly.
# Used by loglike_compare().
lqp_1 <- loglike_quad_point(theta0 = 0.3, sem_out = fit, par_i = "y ~ m")
lqp_1
```

`mediation_latent` *Dataset (SEM, Three Factors, Nine Variables, Mediation)*

Description

Generated from a three-factor model with nine variables, n = 150

Usage

```
mediation_latent
```

Format

A data frame with 150 rows and nine variables:

x1 x1

x2 x2

x3 x3

x4 x4

x5 x5

x6 x6

x7 x7

x8 x8

x9 x9

Details

This model is used for examples like this one:

```
mod <-
"
fx =~ x1 + x2 + x3
fm =~ x4 + x5 + x6
fy =~ x7 + x8 + x9
fm ~ a*fx
fy ~ b*fm + cp*fx
ab := a*b
"
fit <- lavaan::sem(mod, mediation_latent)
```

Examples

```
print(head(mediation_latent), digits = 3)
nrow(mediation_latent)
```

`mediation_latent_skewed` *Dataset (SEM, Three Factors, Nine Variables, Mediation, Skewed)*

Description

Generated from a three-factor model with nine variables, n = 150, with some observed variables positively skewed.

Usage

```
mediation_latent_skewed
```

Format

A data frame with 150 rows and nine variables:

x1 x1

x2 x2

x3 x3

x4 x4

x5 x5

x6 x6

x7 x7

x8 x8

x9 x9

Details

This model is used for examples like this one:

```
mod <-
"
fx =~ x1 + x2 + x3
fm =~ x4 + x5 + x6
fy =~ x7 + x8 + x9
fm ~ a*fx
fy ~ b*fm + cp*fx
ab := a*b
"
fit <- lavaan::sem(mod, mediation_latent)
```

Examples

```
print(head(mediation_latent_skewed), digits = 3)
nrow(mediation_latent_skewed)
```

---

nearby_levels *LBCI Bounds of Nearby Levels of Confidence*

Description

Find LBCIs with levels of confidence different from those stored in a `semlbci`- class object.

Usage

```
nearby_levels(x, ciperc_levels = c(-0.025, 0.025), ciperc_range = c(0.6, 0.99))
```

Arguments

`x`	The output of `semlbci()`.
`ciperc_levels`	A numeric vector of deviations from the original level of confidence. The default is `c(-.025, .025)`. Therefore, if the original level is .95, the levels to be used is `c(-.025, .025) + .95` or `c(.925, .975)`.
`ciperc_range`	A numeric vector of two numbers, which are the minimum and maximum levels of confidence to be used, respectively. Default is `c(.60, .99)`.

Details

It receives a `semlbci`-class object, gets the original level of confidence, generates one or more levels of confidence different from this level by certain amounts, and repeats the original call to `semlbci()` with these levels of confidence. The results are returned as a list of class `semlbci_list`, with the original`semlbci`-class included.

Value

A `semlbci_list`-class object, which is simply a named list of `semlbci`-class object, names being the levels of confidence.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

See Also

`semlbci()`, `ci_order()`

Examples

```
library(lavaan)
mod <-
"
m ~ x
y ~ m
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
lbci_fit <- semlbci(fit_med)
lbci_fit_nb <- nearby_levels(lbci_fit,
                             ciperc_levels = c(-.050, .050))
names(lbci_fit_nb)
# Check the order of the confidence bounds.
# A confidence interval with a higher level of confidence
# should enclose a confidence interval with
# a lower level of confidence.
ci_order(lbci_fit_nb)
```

---

plot.loglike_compare *Plot the Output of 'loglike_compare()'*

Description

Visualize the log profile likelihood of a parameter fixed to values in a range.

Usage

```
## S3 method for class 'loglike_compare'
plot(
  x,
  y,
  type = c("ggplot2", "default"),
  size_label = 4,
  size_point = 4,
  nd_theta = 3,
  nd_pvalue = 3,
  size_theta = 4,
  size_pvalue = 4,
  add_pvalues = FALSE,
  ...
)
```

Arguments

x	The output of loglike_compare().
y	Not used.

type	Character. If "ggplot2", will use ggplot2::ggplot() to plot the graph. If "default", will use R base graphics, The ggplot2 version plots more information. Default is "ggplot2".
size_label	The relative size of the labels for thetas (and *p*-values, if requested) in the plot, determined by ggplot2::rel(). Default is 4.
size_point	The relative size of the points to be added if *p*-values are requested in the plot, determined by ggplot2::rel(). Default is 4.
nd_theta	The number of decimal places for the labels of theta. Default is 3.
nd_pvalue	The number of decimal places for the labels of *p*-values. Default is 3.
size_theta	Deprecated. No longer used.
size_pvalue	Deprecated. No longer used.
add_pvalues	If TRUE, likelihood ratio test *p*-values will be included for the confidence limits. Only available if type = "ggplot2".
...	Optional arguments. Ignored.

Details

Given the output of loglike_compare(), it plots the log profile likelihood based on quadratic approximation and that based on the original log-likelihood. The log profile likelihood is scaled to have a maximum of zero (at the point estimate) as suggested by Pawitan (2013).

Value

Nothing if type = "default", the generated ggplot2::ggplot() graph if type = "ggplot2".

References

Pawitan, Y. (2013). *In all likelihood: Statistical modelling and inference using likelihood*. Oxford University Press.

Examples

```
## loglike_compare

library(lavaan)
data(simple_med)
dat <- simple_med
mod <-
"
m ~ a * x
y ~ b * m
ab := a * b
"
fit <- lavaan::sem(mod, simple_med, fixed.x = FALSE)

# Four points are used just for illustration
# At least 21 points should be used for a smooth plot
# Remove try_k_more in real applications. It is set
```

```
# to run such that this example is not too slow.
# use_pbapply can be removed or set to TRUE to show the progress.
ll_a <- loglike_compare(fit, par_i = "m ~ x", n_points = 4,
                        try_k_more = 0,
                        use_pbapply = FALSE)

plot(ll_a)
plot(ll_a, add_pvalues = TRUE)

# See the vignette "loglike" for an example for the
# indirect effect.
```

`print.cibound` *Print Method of a 'cibound'-class Object*

Description

Print the diagnostic information of a `cibound`-class object.

Usage

```
## S3 method for class 'cibound'
print(x, digits = 5, ...)
```

Arguments

`x`	The output of a `ci_bound_xx_i` function. Currently the only such function is `ci_bound_wn_i()`.
`digits`	The number of digits after the decimal point. To be passed to `round()`. Default is 5.
`...`	Other arguments. They will be ignored.

Details

This is the print method for the output of `ci_bound_wn_i()`, a `cibound`-class object. It prints the diagnostic information on the bound being found and the search process.

Value

`x` is returned invisibly. Called for its side effect.

Examples

```
data(simple_med)
dat <- simple_med

mod <-
"
m ~ x
y ~ m
"

fit_med <- lavaan::sem(mod, simple_med, fixed.x = FALSE)

fn_constr0 <- set_constraint(fit_med)

out1l <- ci_bound_wn_i(i = 1,
                       npar = 5,
                       sem_out = fit_med,
                       f_constr = fn_constr0,
                       which = "lbound")

# Print the output
out1l
```

---

print.semlbci *Print Method of a 'semlbci' Object*

Description

Prints the results of a semlbci object, the output of semlbci().

Usage

```
## S3 method for class 'semlbci'
print(
  x,
  digits = 3,
  annotation = TRUE,
  time = FALSE,
  verbose = FALSE,
  verbose_if_needed = TRUE,
  drop_no_lbci = TRUE,
  output = c("table", "text", "lavaan"),
  sem_out = NULL,
  lbci_only = drop_no_lbci,
  ratio_digits = 1,
  se = TRUE,
  zstat = TRUE,
```

```
  pvalue = TRUE,
  boot.ci.type = "perc",
  ...
)
```

Arguments

x	The output of semlbci().
digits	The number of digits after the decimal point. To be passed to formatC(). Default is 3.
annotation	If TRUE, print table notes. Default is TRUE.
time	If TRUE, print the time spent on each bound. Default is FALSE.
verbose	If TRUE, additional diagnostic information will always be printed. This argument overrides verbose_if_needed. Default is FALSE.
verbose_if_needed	If TRUE, additional diagnostic information will be printed only if necessary. If FALSE, additional diagnostic information will always be printed. Default is TRUE.
drop_no_lbci	If TRUE, parameters without LBCIs will be removed. Default is TRUE.
output	The type of printout. If "table", the default, the results will be printed in a table. If "text" or "lavaan", then the results will be printed in the lavaan style, as in the summary() method for the output of lavaan.
sem_out	If output is "text" or "lavaan", the original output of lavaan used in calling semlbci() needs to be supplied to this argument.
lbci_only	Used only if output is "text" or "lavaan". If TRUE, only the likelihood-based confidence intervals (LBCIs) will be printed. If FALSE, and LBCIs will be printed alongside the confidence intervals by lavaan. Its default value depend on the argument drop_no_lbci. If drop_no_lbci is TRUE, then lbci_only is TRUE by default. If drop_no_lbci is FALSE, then lbci_only is FALSE by default.
ratio_digits	The number of digits after the decimal points for the ratios of distance from the confidence limits to the point estimates. Default is 1.
se	Logical. To be passed to lavaan::parameterEstimates(). Whether standard error (S.E.) will be printed. Only applicable if output is "text" or "lavaan".
zstat	Logical. To be passed to lavaan::parameterEstimates(). Whether z-values will be printed. Only applicable if output is "text" or "lavaan".
pvalue	Logical. To be passed to lavaan::parameterEstimates(). Whether p-values will be printed. Only applicable if output is "text" or "lavaan".
boot.ci.type	Logical. To be passed to lavaan::parameterEstimates(). The type of bootstrap confidence intervals to be printed if bootstrapping confidence intervals available. Possible values are "norm", "basic", "perc", or "bca.simple". The default value is "perc". Refer to the help of lavaan::parameterEstimates() for further information. Only applicable if output is "text" or "lavaan".
...	Other arguments. They will be ignored.

Details

Prints the results of `semlbci()` as a table.

Value

`x` is returned invisibly. Called for its side effect.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

See Also

`semlbci()`

Examples

```
library(lavaan)
mod <-
"
m ~ a*x
y ~ b*m
ab := a * b
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
p_table <- parameterTable(fit_med)
p_table
lbci_med <- semlbci(fit_med,
                    pars = c("ab :="))
lbci_med

print(lbci_med, verbose_if_needed = FALSE)

print(lbci_med, verbose = TRUE)

print(lbci_med, time = TRUE)

print(lbci_med, annotation = FALSE)

print(lbci_med, digits = 4)

# Text output

print(lbci_med, output = "lavaan", sem_out = fit_med)

print(lbci_med, output = "lavaan", sem_out = fit_med, lbci_only = FALSE)

print(lbci_med, output = "lavaan", sem_out = fit_med, lbci_only = FALSE,
      se = FALSE, zstat = FALSE, pvalue = FALSE)
```

reg_cor_near_one *Dataset (Six Variables, One Correlation Close to One)*

Description

Generated from a regression model six variables, x4~~x5 correlation close to one.

Usage

```
reg_cor_near_one
```

Format

A data frame with 100 rows and six variables:

x1 x1

x2 x2

x3 x3

x4 x4, with correlation with x5 nearly equal to 1

x5 x5, with correlation with x4 nearly equal to 1

y y, the dependent variable

Details

This model is used for examples like this one:

```
out <- lm(y ~ x1 + x2 + x3 + x4 + x5, reg_cor_near_one)
summary(out)
cor(reg_cor_near_one[, c("x4", "x5")])
```

Examples

```
print(head(reg_cor_near_one), digits = 3)
nrow(reg_cor_near_one)
```

semlbci *Likelihood-Based Confidence Interval*

Description

Find the likelihood-based confidence intervals (LBCIs) for selected free parameters in an SEM output.

Usage

```
semlbci(
  sem_out,
  pars = NULL,
  include_user_pars = TRUE,
  remove_variances = TRUE,
  remove_intercepts = TRUE,
  ciperc = 0.95,
  standardized = FALSE,
  method = c("wn", "ur"),
  robust = c("none", "satorra.2000"),
  try_k_more_times = 2,
  semlbci_out = NULL,
  check_fit = TRUE,
  ...,
  parallel = FALSE,
  ncpus = 2,
  use_pbapply = TRUE,
  loadbalancing = TRUE
)
```

Arguments

sem_out	The SEM output. Currently supports lavaan::lavaan outputs only.
pars	The positions of the parameters for which the LBCIs are to be searched. Use the position as appeared on the parameter tables of the sem_out. If NULL, the default, then LBCIs for all free parameters will be searched. Can also be a vector of strings to indicate the parameters on the parameter table. The parameters should be specified in lavaan::lavaan() syntax. The vector of strings will be converted by syntax_to_i() to parameter positions. See syntax_to_i() on how to specify the parameters.
include_user_pars	Logical. Whether all user-defined parameters are automatically included when pars is not set. Default is TRUE. If pars is explicitly set, this argument will be ignored.
remove_variances	Logical. Whether variances and error variances will be removed. Default is TRUE, removing all variances and error variances even if specified in pars.

remove_intercepts
: Logical. Whether intercepts will be removed. Default is TRUE, removing all intercepts (parameters with operator ~1). Intercepts are not yet supported in standardized solution and so will always be removed if standardized = TRUE.

ciperc
: The proportion of coverage for the confidence interval. Default is .95, requesting a 95 percent confidence interval.

standardized
: If TRUE, the LBCI is for the standardized estimates.

method
: The method to be used to search for the confidence bounds. Supported methods are"wn" (Wu-Neale-2012), the default, and "ur" (root finding by stats::uniroot()).

robust
: Whether the LBCI based on robust likelihood ratio test is to be found. Only "satorra.2000" in lavaan::lavTestLRT() is supported for now, implemented by the method proposed by Falk (2018). If "none", the default, then likelihood ratio test based on maximum likelihood estimation will be used.

try_k_more_times
: How many more times to try if failed. Default is 2.

semlbci_out
: An semlbci-class object. If provided, parameters already with LBCIs formed will be excluded from pars.

check_fit
: If TRUE (default), the input (sem_out) will be checked by check_sem_out(). If not supported, an error will be raised. If FALSE, the check will be skipped and the LBCIs will be searched even for a model or parameter not supported. Set to TRUE only for testing.

...
: Arguments to be passed to ci_bound_wn_i().

parallel
: If TRUE, will use parallel processing to do the search.

ncpus
: The number of workers, if parallel is TRUE. Default is 2. This number should not be larger than the number CPU cores.

use_pbapply
: If TRUE and pbapply is installed, pbapply::pbapply() will be used to display a progress bar when finding the intervals. Default is TRUE. Ignored if pbapply is not installed.

loadbalancing
: Whether load balancing is used when parallel is TRUE and use_pbapply is TRUE.

Details

semlbci() finds the positions of the selected parameters in the parameter table and then calls ci_i_one() once for each of them. For the technical details, please see ci_i_one() and the functions it calls to find a confidence bound, currently ci_bound_wn_i(). ci_bound_wn_i() uses the approach proposed by Wu and Neale (2012) and illustrated by Pek and Wu (2015).

It supports updating an output of semlbci() by setting semlbci_out. This allows forming LBCIs for some parameters after those for some others have been formed.

If possible, parallel processing should be used (see parallel and ncpus), especially for a model with many parameters.

If the search for some of the confidence bounds failed, with NA for the bounds, try increasing try_k_more_times.

The SEM output will first be checked by `check_sem_out()` to see whether the model and the estimation method are supported. To skip this test (e.g., for testing or experimenting with some models and estimators), set `check_fit` to `FALSE`.

Examples and technical details can be found at Cheung and Pesigan (2023), the website of the `semlbci` package (https://sfcheung.github.io/semlbci/), and the technical appendices at (https://sfcheung.github.io/semlbci/art

It currently supports lavaan::lavaan outputs only.

Value

A `semlbci`-class object similar to the parameter table generated by `lavaan::parameterEstimates()`, with the LBCIs for selected parameters added. Diagnostic information, if requested, will be included in the attributes. See `print.semlbci()` for options available.

Author(s)

Shu Fai Cheung https://orcid.org/0000-0002-9871-9448

References

Cheung, S. F., & Pesigan, I. J. A. (2023). *semlbci*: An R package for forming likelihood-based confidence intervals for parameter estimates, correlations, indirect effects, and other derived parameters. *Structural Equation Modeling: A Multidisciplinary Journal*, *30*(6), 985–999. doi:10.1080/10705511.2023.2183860

Falk, C. F. (2018). Are robust standard errors the best approach for interval estimation with non-normal data in structural equation modeling? *Structural Equation Modeling: A Multidisciplinary Journal, 25*(2), 244-266. doi:10.1080/10705511.2017.1367254

Pek, J., & Wu, H. (2015). Profile likelihood-based confidence intervals and regions for structural equation models. *Psychometrika, 80*(4), 1123-1145. doi:10.1007/s1133601594611

Wu, H., & Neale, M. C. (2012). Adjusted confidence intervals for a bounded parameter. *Behavior Genetics, 42*(6), 886-898. doi:10.1007/s105190129560z

Pritikin, J. N., Rappaport, L. M., & Neale, M. C. (2017). Likelihood-based confidence intervals for a parameter with an upper or lower bound. *Structural Equation Modeling: A Multidisciplinary Journal, 24*(3), 395-401. doi:10.1080/10705511.2016.1275969

See Also

`print.semlbci()`, `confint.semlbci()`, `ci_i_one()`, `ci_bound_wn_i()`

Examples

```
library(lavaan)
mod <-
"
m ~ a*x
y ~ b*m
ab := a * b
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
```

```
p_table <- parameterTable(fit_med)
p_table
lbci_med <- semlbci(fit_med,
                    pars = c("m ~ x",
                             "y ~ m",
                             "ab :="))
lbci_med
```

`set_constraint` *Equality Constraint for Finding the LBCI by Wu-Neale-2012*

Description

Create the equality constraint for finding the likelihood-based confidence interval (LBCI) by the Wu-Neale-2012 method.

Usage

```
set_constraint(sem_out, ciperc = 0.95)
```

Arguments

`sem_out`	The SEM output. Currently supports lavaan::lavaan outputs only.
`ciperc`	The intendeted coverage probability of the confidence interval. Default is .95.

Details

Important Notice:

This function is not supposed to be used directly by users in typical scenarios. Its interface is user-*unfriendly* because it should be used through `semlbci()`. It is exported such that interested users can examine how a confidence bound is found, or use it for experiments or simulations.

Usage:

The Wu-Neale-2012 method uses a simple objective function that is optimized with an equality constraint. `set_constraint()` generates the equality constraint function to be used by `ci_bound_wn_i()`.

It currently supports lavaan::lavaan outputs only.

Value

An equality constraint function to be used by `ci_bound_wn_i()`.

Examples

```
library(lavaan)
data(simple_med)
dat <- simple_med
mod <-
"
m ~ x
y ~ m
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)

fn_constr0 <- set_constraint(fit_med)
out <- fn_constr0(coef(fit_med), sem_out = fit_med)
out
lavTech(fit_med, "optim")$fx
```

---

simple_med *Dataset (Simple Mediation Model)*

Description

Generated from a simple mediation model, n = 200

Usage

```
simple_med
```

Format

A data frame with 200 rows and three variables:

x x, the independent variable

m m, the mediator

y y, the dependent variable

Details

This model is used for examples like this one:

```
library(lavaan)
mod <- "m ~ x
        y ~ m"
fit <- cfa(mod, simple_med)
summary(fit)
```

Examples

```
print(head(simple_med), digits = 3)
nrow(simple_med)
```

`simple_med_mg` *Dataset (Simple Mediation Model, Two Groups)*

Description

Generated from a simple mediation model, n = 200, two groups, n = 100 each.

Usage

```
simple_med_mg
```

Format

A data frame with 500 rows and four variables:

gp gp, the grouping variable

x x, the independent variable

m m, the mediator

y y, the dependent variable

Details

This model is used for examples like this one:

```
library(lavaan)
mod <- "m ~ x
        y ~ m"
fit <- sem(mod, simple_med_mg, gp = "group")
summary(fit)
```

Examples

```
print(head(simple_med_mg), digits = 3)
nrow(simple_med_mg)
table(simple_med_mg$gp)
```

`syntax_to_i` *Parameter Positions From lavaan Syntax*

Description

Converts lavaan syntax to positions in the model parameter table.

Usage

```
syntax_to_i(syntax, sem_out)
```

Arguments

`syntax`	A vector of parameters, defined as in lavaan.
`sem_out`	The SEM output. Currently `lavaan` output only.

Details

`syntax_to_i()` converts a vector of strings, in lavaan syntax, to the positions in the parameter table of a lavaan::lavaan fit object.

Each element in the vector should have left hand side (`lhs`), operator (`op`), and/or right hand side (`rhs`). For example:all.x

- `"m ~ x"` denotes the coefficient of the path from `x` to `m`.
- `"y ~~ x"` denotes the covariance between `y` and `x`.

For user-defined parameters, only `lhs` and `op` will be interpreted. For example:

- To specify the user parameter `ab`, both `"ab := ..."` and `"ab :="` will do, `...` the definition of `ab` in the model. The right-hand side will be ignored.

To denote a labelled parameters, such as `"y ~ a*x"`, treat it as a user-defined parameters and use `:=`, e.g., `"a :="` in this example.

For multiple-group models, if a parameter is specified as in a single-group models, then this parameter in all groups will be selected. For example:all.x

- If a model has three groups, `"y ~ x"` denotes this path parameter in all three groups, and it will be converted to three row numbers.

To select the parameter in a specific group, "multiply" the right-hand-side variable by the group number. For example:

- `"y ~ 2*x"` denotes the path coefficient from `x` to `y` in Group 2.

To denote the parameters in more than one group, multiply the right-hand side variable by a vector of number. For example:all.x

- `"f1 =~ c(2,3)*x2"` denotes the factor loading of `x2` on `f1` in Group 2 and Group 3.

Elements that cannot be converted to a parameter in the parameter table will be ignored.

Currently supports lavaan::lavaan outputs only.

Value

A numeric vector of positions (row numbers) in the parameter table.

Examples

```
library(lavaan)
data(simple_med)
mod <-
"
m ~ a*x
y ~ b*m
ab:= a*b
asq:= a^2
"
fit_med <- sem(mod, simple_med, fixed.x = FALSE)
p_table <- parameterTable(fit_med)

pars <- c("m ~ x",
          "y ~ m",
          "asq := 1",
          "ab  := 2")
out <- syntax_to_i(pars, fit_med)
out
p_table[out, ]
```

Index

∗ **datasets**
cfa_evar_near_zero, 2
cfa_two_factors, 3
cfa_two_factors_mg, 4
mediation_latent, 33
mediation_latent_skewed, 34
reg_cor_near_one, 42
simple_med, 47
simple_med_mg, 48

cfa_evar_near_zero, 2
cfa_two_factors, 3
cfa_two_factors_mg, 4
check_sem_out, 5
check_sem_out(), *44, 45*
ci_bound_ur, 7
ci_bound_ur(), *9*, *11*, *23*
ci_bound_ur_i, 10
ci_bound_ur_i(), *8*, *23*
ci_bound_wn_i, 13
ci_bound_wn_i(), *8*, *12*, *18*, *25, 26*, *38*, *44–46*
ci_i_one, 17
ci_i_one(), *6*, *12*, *16*, *18*, *44, 45*
ci_order, 19
ci_order(), *19, 20*, *35*
confint.semlbci, 21
confint.semlbci(), *45*

formatC(), *40*

gen_est_i *(*ci_bound_ur*)*, 7
gen_est_i(), *7*, *9*
gen_sem_out_userp *(*gen_userp*)*, 22
gen_sem_out_userp(), *9*, *23*
gen_userp, 22
gen_userp(), *7*, *22, 23*
get_cibound, 25
get_cibound(), *26*
get_cibound_status_not_0 *(*get_cibound*)*, 25
get_cibound_status_not_0(), *26*
ggplot2::ggplot(), *37*
ggplot2::rel(), *37*

lavaan::cfa(), *29*
lavaan::fitMeasures(), *31*
lavaan::lav_model_get_parameters(), *23*
lavaan::lav_model_implied(), *23*
lavaan::lav_model_set_parameters(), *23*
lavaan::lavaan, *5*, *7*, *10*, *14*, *17*, *30*, *43*, *45, 46*, *49*
lavaan::lavaan(), *5*, *7*, *10, 11*, *13–15*, *22, 23*, *29*, *43*
lavaan::lavOptions(), *29*
lavaan::lavTestLRT(), *5*, *8, 9*, *17*, *30*, *44*
lavaan::model.syntax, *29*
lavaan::parameterEstimates(), *40*, *45*
lavaan::parameterTable(), *8*, *10*, *14*, *29*
lavaan::sem(), *29*
lavaan::standardizedSolution(), *11*, *15*
loglike_compare, 27
loglike_compare(), *30*, *36, 37*
loglike_compare_ur *(*loglike_compare*)*, 27
loglike_point *(*loglike_compare*)*, 27
loglike_point(), *30*
loglike_point_ur *(*loglike_compare*)*, 27
loglike_quad_point *(*loglike_compare*)*, 27
loglike_quad_point(), *30, 31*
loglike_quad_point_ur *(*loglike_compare*)*, 27
loglike_quad_range *(*loglike_compare*)*, 27
loglike_quad_range(), *30*
loglike_quad_range_ur *(*loglike_compare*)*, 27
loglike_quad_range_ur(), *31*
loglike_range *(*loglike_compare*)*, 27
loglike_range(), *30, 31*
loglike_range_ur *(*loglike_compare*)*, 27
loglike_range_ur(), *31*
loglikelihood *(*loglike_compare*)*, 27

mediation_latent, 33
mediation_latent_skewed, 34

nearby_levels, 35
nearby_levels(), *19, 20*
nloptr::nloptr(), *14*

optimize(), *9*

parallel::makeCluster(), *29*
pbapply::pbapply, *29*
pbapply::pbapply(), *44*
plot.loglike_compare, 36
plot.loglike_compare(), *30, 31*
print.ci_order *(*ci_order*)*, 19
print.ci_order(), *20*
print.cibound, 38
print.cibound(), *12*, *16*
print.semlbci, 39
print.semlbci(), *45*

reg_cor_near_one, 42
round(), *38*

semlbci, 43
semlbci(), *5, 6*, *12*, *15, 16*, *18*, *20, 21*, *25, 26*, *29, 30*, *35*, *39–41*, *44*, *46*
set.seed(), *24*
set_constraint, 46
set_constraint(), *14*, *16*, *46*
simple_med, 47
simple_med_mg, 48
stats::uniroot(), *11*, *17*, *44*
summary(), *40*
syntax_to_i, 49
syntax_to_i(), *29*, *43*, *49*

uniroot(), *7–9*, *12*