Package 'semlbci'

May 7, 2023

Title Likelihood-Based Confidence Interval in Structural Equation Models

Version 0.10.3

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

RoxygenNote 7.2.3

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

VignetteBuilder knitr

Config/testthat/parallel true

Config/testthat/edition 3

Config/testthat/start-first semlbci_wn_mg_*

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-05-07 10:20:02 UTC

R topics documented:

`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")

# 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_wn_i *Likelihood-based Confidence Bound For One parameter*

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()`.

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,
  ...
)
```

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::standardizedSol
ternal", an internal function will be used. The "lavaan"method should work in all situation

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.
...	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 = "wn",
  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), the only supported method.
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.
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)
```

---

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"))
```

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

It returns the original output of `ci_bound_wn_i()` for a bound. Usually for diagnosis.

Value

A cibound-class object. See `ci_bound_wn_i()` for details.

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)
```

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.

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.

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.

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 originalsemlbci-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,
  ...
)
```

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.
...	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)
```

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 = "wn",
  robust = c("none", "satorra.2000"),
  try_k_more_times = 2,
  semlbci_out = NULL,
  check_fit = TRUE,
  ...,
  parallel = FALSE,
  ncpus = 2,
  use_pbapply = 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. Currently only "wn" (Wu-Neale-2012), the default, is supported.

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.

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/ar

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*. Advance online publication. 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",
          "not in table")
out <- syntax_to_i(pars, fit_med)
out
p_table[out, ]
```

Index