Package 'betaselectr'

July 11, 2026

Title Betas-Select in Structural Equation Models and Linear Models

Version 0.2.2

Description It computes betas-select, coefficients after standardization in structural equation models and regression models, standardizing only selected variables. Supports models with moderation, with product terms formed after standardization. It also offers confidence intervals that account for standardization, including bootstrap confidence intervals as proposed by Cheung et al. (2022) <doi:10.1037/hea0001188>. An introduction to the package can be found in Sun et al. (2026) <doi:10.1080/00273171.2026.2672692>.

License GPL (>= 3)

Encoding UTF-8

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Imports boot, stats, lavaan, methods, numDeriv, manymome, pbapply, lavaan.printer

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first lav_betaselect_mg*

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

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

Depends R (>= 3.5.0)

LazyData true

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Shu Fai Cheung [aut, cre] (ORCID: <https://orcid.org/0000-0002-9871-9448>), Rong Wei Sun [aut] (ORCID: <https://orcid.org/0000-0003-0034-1422>), Florbela Chang [aut] (ORCID: <https://orcid.org/0009-0003-9931-501X>), Wendie Yang [aut] (ORCID: <https://orcid.org/0009-0000-8388-6481>), Sing-Hang Cheung [aut] (ORCID: <https://orcid.org/0000-0001-5182-0752>)

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

Repository CRAN

Date/Publication 2026-07-11 08:40:08 UTC

Contents

anova.lm_betaselect 2
coef.lav_betaselect 4
coef.lm_betaselect 5
confint.lav_betaselect 7
confint.lm_betaselect 8
data_test_medmod 10
data_test_mod_cat 11
data_test_mod_cat2 12
data_test_mod_cat_binary 13
getCall.lm_betaselect 13
lav_betaselect 15
lm_betaselect 20
predict.glm_betaselect 25
predict.lm_betaselect 27
print.lav_betaselect 28
std_data 30
summary.glm_betaselect 31
summary.lm_betaselect 34
vcov.lm_betaselect 37

Index **40**

`anova.lm_betaselect` *ANOVA Tables For 'lm_betaselect' and 'glm_betaselect' Objects*

Description

Return the analysis of variance tables for the outputs of `lm_betaselect()` and `glm_betaselect()`.

Usage

```
## S3 method for class 'lm_betaselect'
anova(object, ..., type = c("beta", "standardized", "raw", "unstandardized"))

## S3 method for class 'glm_betaselect'
anova(
  object,
  ...,
  type = c("beta", "standardized", "raw", "unstandardized"),
  dispersion = NULL,
  test = NULL
)
```

Arguments

`object`	The output of `lm_betaselect()` or `glm_betaselect()`.
`...`	Additional outputs of `lm_betaselect()` or `glm_betaselect()`.
`type`	String. If `"unstandardized"` or `"raw"`, the output *before* standardization are used If `"beta"` or `"standardized"`, then the output *after* selected variables standardized are returned. Default is `"beta"`.
`dispersion`	To be passed to `stats::anova.glm()`. The dispersion parameter. Default ia `NULL` and it is extracted from the model.
`test`	String. The test to be conducted. Please refer to `stats::anova.glm()` for details.

Details

By default, it calls `stats::anova()` on the results with selected variables standardized. By setting `type` to `"raw"` or `"unstandardized"`, it calls `stats::anova()` on the results *before* standardization.

Value

It returns an object of class `anova`, which is identical to the output of `stats::anova()` in structure.

Author(s)

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

See Also

`lm_betaselect()`

Examples

```
data(data_test_mod_cat)

lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                           data = data_test_mod_cat,
                           to_standardize = "iv",
                           do_boot = FALSE)
anova(lm_beta_x)
anova(lm_beta_x, type = "raw")


data_test_mod_cat$p <- scale(data_test_mod_cat$dv)[, 1]
data_test_mod_cat$p <- ifelse(data_test_mod_cat$p > 0,
                              yes = 1,
                              no = 0)
logistic_beta_x <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                  data = data_test_mod_cat,
                                  family = binomial,
                                  to_standardize = "iv")
```

```
anova(logistic_beta_x)
anova(logistic_beta_x, type = "raw")
```

`coef.lav_betaselect` *Coefficients of a 'lav_betaselect'-Class Object*

Description

Return the betas-select in a 'lav_betaselect'-class object.

Usage

```
## S3 method for class 'lav_betaselect'
coef(object, drop_na = FALSE, ...)
```

Arguments

`object`	The output of `lav_betaselect()`.
`drop_na`	Logical. Whether betas-select with `NA` are dropped. Default is `FALSE`.
`...`	Optional arguments. Not used.

Details

It just extracts and returns the column `est` from the object: the betas-select, with selected variables standardized.

Value

A numeric vector: The betas-select in the object. The names of parameters follow the convention in `lavaan`.

Author(s)

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

See Also

`lav_betaselect()`

Examples

```
library(lavaan)
# Need to mean-center iv and mod
data_test_medmod$iv <- data_test_medmod$iv - mean(data_test_medmod$iv)
data_test_medmod$mod <- data_test_medmod$mod - mean(data_test_medmod$mod)
mod <-
"
med ~ iv + mod + iv:mod
dv ~ med + iv
"
fit <- sem(mod,
           data_test_medmod,
           fixed.x = TRUE)
summary(fit)
fit_beta <- lav_betaselect(fit,
                           to_standardize = c("iv", "dv"))
coef(fit_beta)
```

---

coef.lm_betaselect *Coefficients of Beta-Select in Linear Models*

Description

Return the estimates of coefficients in an `lm_betaselect`-class or `glm_betaselect`-class object.

Usage

```
## S3 method for class 'lm_betaselect'
coef(
  object,
  complete = FALSE,
  type = c("beta", "standardized", "raw", "unstandardized"),
  ...
)

## S3 method for class 'glm_betaselect'
coef(
  object,
  complete = FALSE,
  type = c("beta", "standardized", "raw", "unstandardized"),
  ...
)
```

Arguments

`object`	The output of `lm_betaselect()` or `glm_betaselect()`, or an `lm_betaselect`-class or `glm_betaselect`-class object.

complete	If TRUE, it returns the full vector of coefficients, including those of terms dropped in an over-determined system. See stats::coef() for further information. Default is FALSE.
type	String. If "unstandardized" or "raw", the coefficients *before* standardization are returned. If "beta" or "standardized", then the coefficients *after* selected variables standardized are returned. Default is "beta".
...	Other arguments. Ignored.

Details

By default, it extracts the regression coefficients *after* the selected variables have been standardized. If requested, it can also return the regression coefficients *before* standardization.

Value

A numeric vector: The estimate of regression coefficients.

Author(s)

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

See Also

lm_betaselect() and glm_betaselect()

Examples

```
data(data_test_mod_cat)

lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                           data = data_test_mod_cat,
                           to_standardize = "iv")
coef(lm_beta_x)
coef(lm_beta_x, type = "raw")


data_test_mod_cat$p <- scale(data_test_mod_cat$dv)[, 1]
data_test_mod_cat$p <- ifelse(data_test_mod_cat$p > 0,
                              yes = 1,
                              no = 0)
logistic_beta_x <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                  data = data_test_mod_cat,
                                  family = binomial,
                                  to_standardize = "iv")
coef(logistic_beta_x)
coef(logistic_beta_x, type = "raw")
```

confint.lav_betaselect *Confidence Intervals for a 'lav_betaselect'-Class Object*

Description

Return the confidence intervals of betas-select in the output of `lav_betaselect()`.

Usage

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

Arguments

`object`	The output of `lav_betaselect()`.
`parm`	Ignored due to the complexity in the naming. The confidence intervals of all parameters are always returned.
`level`	The level of confidence. Ignored because the intervals should be formed when calling `lav_betaselect()`.
`...`	Optional arguments. Ignored.

Details

The type of confidence intervals depends on the call to `lav_betaselect()`. This function does not recompute the confidence interval.

Value

A two-column matrix of the confidence intervals.

Author(s)

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

See Also

`lav_betaselect()`

Examples

```
library(lavaan)
# Need to mean-center iv and mod
data_test_medmod$iv <- data_test_medmod$iv - mean(data_test_medmod$iv)
data_test_medmod$mod <- data_test_medmod$mod - mean(data_test_medmod$mod)
mod <-
"
med ~ iv + mod + iv:mod
```

```
dv ~ med + iv
"
fit <- sem(mod,
           data_test_medmod,
           fixed.x = TRUE)
summary(fit)
fit_beta <- lav_betaselect(fit,
                           to_standardize = c("iv", "dv"))
confint(fit_beta)
```

`confint.lm_betaselect` *Confidence Interval for 'lm_betaselect' or 'glm_betaselect' Objects*

Description

Return the confidence interval of the regression coefficients in the output of `lm_betaselect()` or `glm_betaselect()`.

Usage

```
## S3 method for class 'lm_betaselect'
confint(
  object,
  parm,
  level = 0.95,
  method = c("boot", "bootstrap", "ls"),
  type = c("beta", "standardized", "raw", "unstandardized"),
  warn = TRUE,
  boot_type = c("perc", "bc"),
  ...
)

## S3 method for class 'glm_betaselect'
confint(
  object,
  parm,
  level = 0.95,
  trace = FALSE,
  test = c("LRT", "Rao"),
  method = c("boot", "bootstrap", "default", "ls"),
  type = c("beta", "standardized", "raw", "unstandardized"),
  warn = TRUE,
  boot_type = c("perc", "bc"),
  transform_b = NULL,
  ...
)
```

Arguments

object	The output of lm_betaselect() or glm_betaselect().
parm	The terms for which the confidence intervals are returned. If missing, the confidence intervals of all terms will be returned.
level	The level of confidence, default is .95, returning the 95% confidence interval.
method	The method used to compute the confidence intervals/ If bootstrapping was requested when calling lm_betaselect() and this argument is set to "bootstrap" or "boot", the bootstrap confidence intervals are returned. If bootstrapping was not requested or if this argument is set to "ls", then the usual lm confidence intervals are returned, with a warning raised unless type is "raw" or "unstandardized". Default is "boot".
type	String. If "unstandardized" or "raw", the confidence intervals of the coefficients *before* standardization are returned. If "beta" or "standardized", then the confidence intervals of the coefficients *after* selected variables standardized are returned. Default is "beta".
warn	Logical. Whether a warning will be raised is OLS (or WLS) confidence intervals are requested for the model with some variables standardized (i.e., type is "beta" or "standardized"). Default is TRUE.
boot_type	The type of bootstrap confidence intervals. Currently, it supports "perc", percentile bootstrap confidence intervals, and "bc", bias-corrected bootstrap confidence interval.
...	Optional arguments. Ignored.
trace	Logical. Whether profiling will be traced. See stats::confint.glm() for details. ignored if method is "boot" or "bootstrap".
test	The test used for profiling. See stats::confint.glm for details. ignored if method is "boot" or "bootstrap".
transform_b	The function to be used to transform the confidence limits. For example, if set to exp, the confidence limits will be exponentiated. Users need to decide whether the transformed limits are meaningful. Default is NULL.

Details

The type of confidence intervals depends on the object. If bootstrapping was requested, by default it returns the percentile bootstrap confidence intervals. Otherwise, it returns the default confidence intervals.

Support for other type of confidence intervals may be added in the future.

Value

A p by 2 matrix of the confidence intervals, p being the number of coefficients.

Author(s)

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

See Also

lm_betaselect()

Examples

```
data(data_test_mod_cat)

# bootstrap should be set to 2000 or 5000 in real studies
lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                           data = data_test_mod_cat,
                           to_standardize = "iv",
                           do_boot = TRUE,
                           bootstrap = 100,
                           iseed = 1234)
confint(lm_beta_x)
confint(lm_beta_x, method = "ls")
confint(lm_beta_x, type = "raw")


data_test_mod_cat$p <- scale(data_test_mod_cat$dv)[, 1]
data_test_mod_cat$p <- ifelse(data_test_mod_cat$p > 0,
                              yes = 1,
                              no = 0)
# bootstrap should be set to 2000 or 5000 in real studies
logistic_beta_x <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                  data = data_test_mod_cat,
                                  family = binomial,
                                  to_standardize = "iv",
                                  do_boot = TRUE,
                                  bootstrap = 100,
                                  iseed = 1234)

confint(logistic_beta_x, method = "default")
confint(logistic_beta_x, type = "raw")
```

---

data_test_medmod *Test Dataset with Moderator and Mediator*

Description

This dataset has one mediator, one moderator, one independent variable, one dependent variable, and two control variables.

Usage

```
data_test_medmod
```

Format

A data frame with 200 rows and five variables:

dv Dependent variable, continuous

iv Independent variable, continuous

mod Moderator, continuous

med Mediator, continuous

cov1 Control variable, continuous

cov2 Control variable, continuous

Examples

```
library(lavaan)
mod <-
"
med ~ iv + mod + iv:mod + cov1 + cov2
dv ~ med + iv + cov1 + cov2
"
fit <- sem(mod,
           data_test_medmod)
summary(fit)
```

---

data_test_mod_cat *Test Dataset with Moderator and Categorical Variables*

Description

This dataset has one predictor, one moderator, one control variable, one dependent variable, and a categorical variable.

Usage

```
data_test_mod_cat
```

Format

A data frame with 500 rows and five variables:

dv Dependent variable, continuous

iv Independent variable, continuous

mod Moderator, continuous

cov1 Control variable, continuous

cat1 String variable with these values: "gp1", "gp2", and "gp3"

Examples

```
lm_out <- lm(dv ~ iv * mod + cov1 + cat1, data_test_mod_cat)
summary(lm_out)
```

`data_test_mod_cat2` *Test Dataset with Moderator and Categorical Variables (Version 2)*

Description

This dataset has one predictor, one moderator, one control variable, one dependent variable, and a categorical variable.

Similar to `data_test_mod_cat` but generated from another population.

Usage

```
data_test_mod_cat2
```

Format

A data frame with 300 rows and five variables:

dv Dependent variable, continuous

iv Independent variable, continuous

mod Moderator, continuous

cov1 Control variable, continuous

cat1 String variable with these values: "gp1", "gp2", and "gp3"

Examples

```
lm_out <- lm(dv ~ iv * mod + cov1 + cat1, data_test_mod_cat)
summary(lm_out)
```

data_test_mod_cat_binary *Test Dataset with a Binary Outcome Variable*

Description

This dataset has one predictor, one moderator, one control variable, one binary dependent variable, and a categorical variable.

Usage

```
data_test_mod_cat_binary
```

Format

A data frame with 300 rows and five variables:

dv Dependent variable, binary: 0, 1

iv Independent variable, continuous

mod Moderator, continuous

cov1 Control variable, continuous

cat1 String variable with these values: "gp1", "gp2", and "gp3"

Examples

```
glm_out <- glm(dv ~ iv * mod + cov1 + cat1, data_test_mod_cat_binary, family = binomial())
summary(glm_out)
```

getCall.lm_betaselect *Call in an 'lm_betaselect' or 'glm_betaselect' Object*

Description

The getCall-method for an lm_betaselect-class or glm_betaselectd-class objects.

Usage

```
## S3 method for class 'lm_betaselect'
getCall(
  x,
  what = c("lm_betaselect", "beta", "standardized", "raw", "unstandardized"),
  ...
)

## S3 method for class 'glm_betaselect'
getCall(
  x,
  what = c("glm_betaselect", "beta", "standardized", "raw", "unstandardized"),
  ...
)
```

Arguments

x	An lm_betaselect-class or glm_betaselect-class object from which the call is to be extracted.
what	Which call to extract. For "lm_betaselect" or "glm_betaselect" the call to lm_betaselect() or glm_betaselect() is extracted. For "beta" or "standardized", the call used to fit the model *after* selected variables standardized is extracted. For "raw" or "unstandardized", the call used to fit hte model *before* standardization is extracted.
...	Additional arguments. Ignored.

Details

This works in the same way the default getCall-method does for the outputs of stats::lm() and stats::glm().

Value

It returns the call requested.

Author(s)

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

See Also

lm_betaselect(), glm_betaselect(), and stats::getCall()

Examples

```
data(data_test_mod_cat)

lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1,
                           data = data_test_mod_cat,
```

```
                            to_standardize = "iv")
getCall(lm_beta_x)
getCall(lm_beta_x, what = "beta")
getCall(lm_beta_x, what = "raw")
```

---

lav_betaselect *Betas-Select in a 'lavaan'-Model*

Description

Can standardize selected variables in a lavaan model without refitting the models, can handle product term correctly and skip categorical predictors in standardization.

Usage

```
lav_betaselect(
  object,
  to_standardize = ".all.",
  not_to_standardize = NULL,
  skip_categorical_x = TRUE,
  output = c("data.frame", "text"),
  std_se = c("none", "delta", "bootstrap"),
  std_z = TRUE,
  std_pvalue = TRUE,
  std_ci = TRUE,
  level = 0.95,
  progress = TRUE,
  boot_out = NULL,
  bootstrap = 100L,
  store_boot_est = TRUE,
  parallel = c("no", "snow", "multicore"),
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  cl = NULL,
  iseed = NULL,
  find_product_terms = TRUE,
  check_mean_centering = FALSE,
  std_intercept = FALSE,
  ...,
  delta_method = c("lavaan", "numDeriv"),
  vector_form = TRUE
)
```

Arguments

object	The output of lavaan model fit functions, such as lavaan::sem() and lavaan::cfa().

`to_standardize` A string vector, which should be the names of the variables to be standardized. Default is `".all."`, indicating all variables are to be standardized (but see `skip_categorical_x`).

`not_to_standardize`
A string vector, which should be the names of the variables that should not be standardized. This argument is useful when most variables, except for a few, are to be standardized. This argument cannot be ued with `to_standardize` at the same time. Default is `NULL`, and only `to_standardize` is used.

`skip_categorical_x`
Logical. If `TRUE`, the default, all categorical predictors, defined as variables with only two possible values in the data analyzed, will be skipped in standardization. This overrides the argument `to_standardize`. That is, a categorical predictor will not be standardized even if listed in `to_standardize`, unless users set this argument to `FALSE`.

`output` The format of the output. Not used because the format of the printout is now controlled by the `print`-method of the output of this function. Kept for backward compatibility.

`std_se` String. If set to `"none"`, the default, standard errors will not be computed for the standardized solution. If set to `"delta"`, delta method will be used to compute the standard errors. If set to `"bootstrap"`, then what it does depends whether `boot_out` is set. If `boot_out` is to an output of `manymome::do_boot()`, its content will be used. If `boot_out` is `NULL` *and* bootstrap estimates are available in `object` (e.g., bootstrapping is requested when fitting the model in `lavaan`), then the stored bootstrap estimates will be sued. If not available, the bootstrapping will be conducted using `lavaan::bootstrapLavaan()`, using arguments `bootstrap`, `parallel`, `ncpus`, `cl`, and `iseed`.‘

`std_z` Logical. If `TRUE` and `std_se` is not set to `"none"`, standard error will be computed using the method specified in `std_se`. Default is `TRUE`.

`std_pvalue` Logical. If `TRUE`, `std_se` is not set to `"none"`, and `std_z` is `TRUE`, *p*-values will be computed using the method specified in `std_se`. For bootstrapping, the method proposed by Asparouhov and Muthén (2021) is used. Default is `TRUE`.

`std_ci` Logical. If `TRUE` and `std_se` is not set to `"none"`, confidence intervals will be computed using the method specified in `std_se`. Default is `FALSE`.

`level` The level of confidence of the confidence intervals. Default is .95. It will be used in the confidence intervals of both the unstandardized and standardized solution.

`progress` Logical. If `TRUE`, progress bars will be displayed for long process.

`boot_out` If `std_se` is `"bootstrap"` and this argument is set to an output of `manymome::do_boot()`, its output will be used in computing statistics such as standard errors and confidence intervals. This allows users to use methods other than bootstrapping when fitting the model, while they can still request bootstrapping for the standardized solution.

`bootstrap` If `std_se` is `"bootstrap"` but bootstrapping is not requested when fitting the model and `boot_out` is not set, `lavaan::bootstrapLavaan()` will be called to do bootstrapping. This argument is the number of bootstrap samples to draw. Default is 100. Should be set to 5000 or even 10000 for stable results.

`store_boot_est` Logical. If `std_se` is `"bootstrap"` and this argument is `TRUE`, the default, the bootstrap estimates of the standardized solution will be stored in the attribute `"boot_est"`. These estimates can be used for diagnosis of the bootstrapping. If `FALSE`, then the bootstrap estimates will not be stored.

`parallel` If `std_se` is `"bootstrap"` but bootstrapping is not requested when fitting the model and `boot_out` is not set, `lavaan::bootstrapLavaan()` will be called to do bootstrapping. This argument is to be passed to `lavaan::bootstrapLavaan()`. Default is `"no"`.

`ncpus` If `std_se` is `"bootstrap"` but bootstrapping is not requested when fitting the model and `boot_out` is not set, `lavaan::bootstrapLavaan()` will be called to do bootstrapping. This argument is to be passed to `lavaan::bootstrapLavaan()`. Default is `parallel::detectCores(logical = FALSE) - 1`. Ignored if `parallel` is `"no"`.

`cl` If `std_se` is `"bootstrap"` but bootstrapping is not requested when fitting the model and `boot_out` is not set, `lavaan::bootstrapLavaan()` will be called to do bootstrapping. This argument is to be passed to `lavaan::bootstrapLavaan()`. Default is `NULL`. Ignored if `parallel` is `"no"`.

`iseed` If `std_se` is `"bootstrap"` but bootstrapping is not requested when fitting the model and `boot_out` is not set, `lavaan::bootstrapLavaan()` will be called to do bootstrapping. This argument is to be passed to `lavaan::bootstrapLavaan()` to set the seed for the random resampling. Default is `NULL`. Should be set to an integer for reproducible results. Ignored if `parallel` is `"no"`.

`find_product_terms` String. If it is certain that a model does not have product terms, setting this to `FALSE` will skip the search, which is time consuming for a models with many paths and/or many variables. Default is `TRUE`, and the function will automatically identify product terms, if any.

`check_mean_centering` Logical. If `TRUE`, it will check whether variables involved in a product term has been mean-centered. If not, an error will be raised.

`std_intercept` Logical. If `TRUE`, intercepts of `y` variables will also be computed based on the variables standardized.

`...` Optional arguments to be passed to the `lavaan::parameterEstimates()`, which will be use to generate the output.

`delta_method` The method used to compute delta-method standard errors. For internal use and should not be changed.

`vector_form` The internal method used to compute standardized solution. For internal use and should not be changed.

Details

This function lets users select which variables to be standardized when computing the standardized solution. It has the following features:

- It automatically skips predictors which has only two unique values, assuming that they are dummy variables.

- It does not standardize product term, which is incorrect. Instead, it computes the product term with its component variables standardized first.
- It can be used to generate bootstrap confidence intervals for the standardized solution (Falk, 2018). Bootstrap confidence interval is better than doing standardization *before* fitting a model because it correctly takes into account the sampling variance of the standard deviations. It is also better than delta-method confidence interval because it takes into account the usually asymmetric distribution of parameters after standardization, such as standardized loadings and correlations.
- For comparison, it can also report delta-method standard errors and confidence intervals if requested.

Problems With Common Approaches:

In most SEM programs, users have limited control on which variables to standardize when requesting the standardized solution. The solution may be uninterpretable or misleading in these conditions:

- Dummy variables are standardized and their coefficients cannot be interpreted as the difference between two groups on the outcome variables.
- Product terms (interaction terms) are standardized and they cannot be interpreted as the changes in the effects of focal variables when the moderators change (Cheung, Cheung, Lau, Hui, & Vong, 2022).
- Variables with meaningful units can be more difficult to interpret when they are standardized (e.g., age).

Moreover, the delta method is usually used in standardization, which is suboptimal for standardization unless the sample size is large (Falk, 2018). For example, the covariance with variables standardized is a correlation, and its sampling distribution is skewed unless its population value is zero. However, delta-method confidence interval for the correlation is necessarily symmetric around the point estimate.

Limitations:

- It only supports observed variable interaction terms, and only support two-way interactions.
- It does not support multilevel models.
- It only supports models fitted to raw data.

Value

A `lav_betaselect`-class object, which is a data frame storing the parameter estimates, similar in form to the output of `lavaan::parameterEstimates()`.

Author(s)

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

References

Asparouhov, A., & Muthén, B. (2021). Bootstrap p-value computation. Retrieved from https://www.statmodel.com/download
Bootstrap%20-%20Pvalue.pdf

Cheung, S. F., Cheung, S.-H., Lau, E. Y. Y., Hui, C. H., & Vong, W. N. (2022) Improving an old way to measure moderation effect in standardized units. *Health Psychology*, *41*(7), 502-505. doi:10.1037/hea0001188

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

Sun, R. wei, Chang, F., Yang, W., Cheung, S. F., & Cheung, S.-H. (2026). `betaselectr`: Selective (and proper) standardization in structural equation models. *Multivariate Behavioral Research*. Advance online publication. doi:10.1080/00273171.2026.2672692

See Also

`print.lav_betaselect()` for its print method.

Examples

```
library(lavaan)
# Need to mean-center iv and mod
data_test_medmod$iv <- data_test_medmod$iv - mean(data_test_medmod$iv)
data_test_medmod$mod <- data_test_medmod$mod - mean(data_test_medmod$mod)
mod <-
"med ~ iv + mod + iv:mod
 dv ~ med + iv"
fit <- sem(mod,
           data_test_medmod,
           fixed.x = TRUE)
summary(fit)
fit_beta <- lav_betaselect(fit,
                           to_standardize = c("iv", "dv"))
fit_beta
print(fit_beta, standardized_only = FALSE)

# In real studies:
# - should set bootstrap to at least 5000
# - should set parallel to "snow" or "multicore"
fit_beta_boot <- lav_betaselect(fit,
                                to_standardize = c("iv", "dv"),
                                std_se = "bootstrap",
                                std_ci = TRUE,
                                bootstrap = 100,
                                iseed = 1234)
fit_beta_boot
print(fit_beta_boot, standardized_only = FALSE)

# Print full results
print(fit_beta_boot,
      standardized_only = FALSE)
```

`lm_betaselect` *Betas-Select in a Regression Model*

Description

Can fit a linear regression models with selected variables standardized; handle product terms correctly and skip categorical predictors in standardization.

Usage

```
lm_betaselect(
  ...,
  to_standardize = NULL,
  not_to_standardize = NULL,
  skip_response = FALSE,
  do_boot = TRUE,
  bootstrap = 100L,
  iseed = NULL,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  progress = TRUE,
  load_balancing = FALSE,
  model_call = c("lm", "glm")
)

glm_betaselect(
  ...,
  to_standardize = NULL,
  not_to_standardize = NULL,
  skip_response = FALSE,
  do_boot = TRUE,
  bootstrap = 100L,
  iseed = NULL,
  parallel = FALSE,
  ncpus = parallel::detectCores(logical = FALSE) - 1,
  progress = TRUE,
  load_balancing = FALSE
)

## S3 method for class 'lm_betaselect'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  type = c("beta", "standardized", "raw", "unstandardized"),
  ...
)
```

```
## S3 method for class 'glm_betaselect'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  type = c("beta", "standardized", "raw", "unstandardized"),
  ...
)

raw_output(x)
```

Arguments

...	For lm_betaselect(). these arguments will be passed directly to lm(). For glm_betaselect(), these arguments will be passed to glm(). For the print-method of lm_betaselect or glm_betaselect objects, this will be passed to other methods.
to_standardize	A string vector, which should be the names of the variables to be standardized. Default is NULL, indicating all variables are to be standardized.
not_to_standardize	A string vector, which should be the names of the variables that should *not* be standardized. This argument is useful when most variables, except for a few, are to be standardized. This argument cannot be ued with to_standardize at the same time. Default is NULL, and only to_standardize is used.
skip_response	Logical. If TRUE, will not standardize the response (outcome) variable even if it appears in to_standardize or to_standardize is not specified. Used for models such as logistic regression models in which there are some restrictions on the response variables (e.g., only 0 or 1 for logistic regression).
do_boot	Whether bootstrapping will be conducted. Default is TRUE.
bootstrap	If do_boot is TRUE, this argument is the number of bootstrap samples to draw. Default is 100. Should be set to 5000 or even 10000 for stable results.
iseed	If do_boot is TRUE and this argument is not NULL, it will be used by set.seed() to set the seed for the random number generator. Default is NULL.
parallel	If do_boot is TRUE and this argument is TRUE, parallel processing will be used to do bootstrapping. Default is FALSE because bootstrapping for models fitted by stats::lm() or stats::glm() is rarely slow. Actually, if both parallel and progress are set to TRUE, the speed may even be slower than serial processing.
ncpus	If do_boot is TRUE and parallel is also TRUE, this argument is the number of processes to be used in parallel processing. Default is parallel::detectCores(logical = FALSE) - 1
progress	Logical. If TRUE, progress bars will be displayed for long process. Default is TRUE.
load_balancing	Logical. If parallel is TRUE, this determines whether load balancing will be used. Default is FALSE because the gain in speed is usually minor.
model_call	The model function to be called. If "lm", the default, the model will be fitted by stats::lm(). If "glm", the model will be fitted by stats::glm(). Users should call the corresponding function directly rather than setting this argument manually.

x	An lm_betaselect or glm_betaselect object.
digits	The number of significant digits to be printed for the coefficients.
type	The coefficients to be printed. For "beta" or "standardized", the coefficients after selected variables standardized will be printed. For "raw" or "unstandardized", the coefficients before standardization was done will be printed.

Details

The functions lm_betaselect() and glm_betaselect() let users select which variables to be standardized when computing the standardized solution. They have the following features:

- They automatically skip categorical predictors (i.e., factor or string variables).
- They do not standardize a product term, which is incorrect. Instead, they compute the product term with its component variables standardized, if requested.
- They standardize the selected variables *before* fitting a model. Therefore, If a model has the term log(x) and x is one of the selected variables, the model used the logarithm of the *standardized* x in the model, instead of standardized log(x) which is difficult to interpret.
- They can be used to generate nonparametric bootstrap confidence intervals for the standardized solution. Bootstrap confidence interval is better than the default confidence interval ignoring the standardization because it takes into account the sampling variance of the standard deviations. Preliminary support for bootstrap confidence has been found for forming confidence intervals for coefficients involving standardized variables in linear regression (Jones & Waller, 2013).

Problems With Common Approaches:

In some regression programs, users have limited control on which variables to standardize when requesting the so-called "betas". The solution may be uninterpretable or misleading in these conditions:

- Dummy variables are standardized and their coefficients cannot be interpreted as the difference between two groups on the outcome variables.
- Product terms (interaction terms) are standardized and they cannot be interpreted as the changes in the effects of focal variables when the moderators change (Cheung, Cheung, Lau, Hui, & Vong, 2022).
- Variables with meaningful units can be more difficult to interpret when they are standardized (e.g., age).

How The Function Work:

They standardize the original variables *before* they are used in the model. Therefore, strictly speaking, they do not standardize the predictors in model, but standardize the *input variable* (Gelman et al., 2021).

The requested model is then fitted to the dataset with selected variables standardized. For the ease of follow-up analysis, both the results with selected variables standardized and the results without standardization are stored. If required, the results without standardization can be retrieved by raw_output().

Methods:

The output of lm_betaselect() is an lm_betaselect-class object, and the output of glm_betaselect() is a glm_betaselect-class object. They have the following methods:

- A coef-method for extracting the coefficients of the model. (See coef.lm_betaselect() and coef.glm_betaselect() for details.)
- A vcov-method for extracting the variance-covariance matrix of the estimates of the coefficients. If bootstrapping is requested, it can return the matrix based on the bootstrapping estimates. (See vcov.lm_betaselect() and vcov.glm_betaselect() for details.)
- A confint-method for forming the confidence intervals of the estimates of the coefficients. If bootstrapping is requested, it can return the bootstrap confidence intervals. (See confint.lm_betaselect() and confint.glm_betaselect() for details.)
- A summary-method for printing the summary of the results, with additional information such as the number of bootstrap samples and which variables have been standardized. (See summary.lm_betaselect() and summary.glm_betaselect() for details.)
- An anova-method for printing the ANOVA table. Can also be used to compare two or more outputs of lm_betaselect() or glm_betaselect() (See anova.glm_betaselect() and anova.glm_betaselect() for details.)
- A predict-method for computing predicted values. It can be used to compute the predicted values given a set of new unstandardized data. The data will be standardized before computing the predicted values in the models with standardization. (See predict.lm_betaselect() and predict.glm_betaselect() for details.)
- The default update-method for updating a call also works for an lm_betaselect object or a glm_betaselect() object. It can update the model in the same way it updates a model fitted by stats::lm() or stats::glm(), and also update the arguments of lm_betaselect() or glm_betaselect() such as the variables to be standardized. (See stats::update() for details.)

Most other methods for the output of stats::lm() and stats::glm() should also work on an lm_betaselect-class object or a glm_betaselect-class object, respectively. Some of them will give the same results regardless of the variables standardized. Examples are rstandard() and cooks.distance(). For some others, they should be used with cautions if they make use of the variance-covariance matrix of the estimates.

To use the methods for lm objects or glm objects on the results without standardization, simply use raw_output(). For example, to get the fitted values without standardization, call fitted(raw_output(x)), where x is the output of lm_betaselect() or glm_betaselect().

The function raw_output() simply extracts the regression output by stats::lm() or stats::glm() on the variables without standardization.

Value

The function lm_betaselect() returns an object of the class lm_betaselect, The function glm_betaselect() returns an object of the class glm_betaselect. They are similar in structure to the output of stats::lm() and stats::glm(), with additional information stored.

The function raw_output() returns an object of the class lm or glm, which are the results of fitting the model to the data by stats::lm() or stats::glm() without standardization.

Author(s)

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

References

Cheung, S. F., Cheung, S.-H., Lau, E. Y. Y., Hui, C. H., & Vong, W. N. (2022) Improving an old way to measure moderation effect in standardized units. *Health Psychology*, *41*(7), 502-505. doi:10.1037/hea0001188

Craig, C. C. (1936). On the frequency function of xy. *The Annals of Mathematical Statistics, 7*(1), 1–15. doi:10.1214/aoms/1177732541

Gelman, A., Hill, J., & Vehtari, A. (2021). *Regression and other stories*. Cambridge University Press. doi:10.1017/9781139161879

Jones, J. A., & Waller, N. G. (2013). Computing confidence intervals for standardized regression coefficients. *Psychological Methods, 18*(4), 435–453. doi:10.1037/a0033269

Sun, R. wei, Chang, F., Yang, W., Cheung, S. F., & Cheung, S.-H. (2026). `betaselectr`: Selective (and proper) standardization in structural equation models. *Multivariate Behavioral Research*. Advance online publication. doi:10.1080/00273171.2026.2672692

See Also

`print.lm_betaselect()` and `print.glm_betaselect()` for the `print`-methods.

Examples

```
data(data_test_mod_cat)

# Standardize only iv

lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                           data = data_test_mod_cat,
                           to_standardize = "iv")
lm_beta_x
summary(lm_beta_x)

# Manually standardize iv and call lm()

data_test_mod_cat$iv_z <- scale(data_test_mod_cat[, "iv"])[, 1]

lm_beta_x_manual <- lm(dv ~ iv_z*mod + cov1 + cat1,
                       data = data_test_mod_cat)

coef(lm_beta_x)
coef(lm_beta_x_manual)

# Standardize all numeric variables

lm_beta_all <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                             data = data_test_mod_cat)
# Note that cat1 is not standardized
summary(lm_beta_all)


data(data_test_mod_cat)
```

```
data_test_mod_cat$p <- scale(data_test_mod_cat$dv)[, 1]
data_test_mod_cat$p <- ifelse(data_test_mod_cat$p > 0,
                              yes = 1,
                              no = 0)
# Standardize only iv
logistic_beta_x <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                  family = binomial,
                                  data = data_test_mod_cat,
                                  to_standardize = "iv")
summary(logistic_beta_x)

logistic_beta_x
summary(logistic_beta_x)

# Manually standardize iv and call glm()

data_test_mod_cat$iv_z <- scale(data_test_mod_cat[, "iv"])[, 1]

logistic_beta_x_manual <- glm(p ~ iv_z*mod + cov1 + cat1,
                              family = binomial,
                              data = data_test_mod_cat)

coef(logistic_beta_x)
coef(logistic_beta_x_manual)

# Standardize all numeric predictors

logistic_beta_allx <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                     family = binomial,
                                     data = data_test_mod_cat,
                                     to_standardize = c("iv", "mod", "cov1"))
# Note that cat1 is not standardized
summary(logistic_beta_allx)


summary(raw_output(lm_beta_x))
```

`predict.glm_betaselect` *Predict Method for a 'glm_betaselect' Object*

Description

Compute the predicted values in a model fitted by `glm_betaselect()`.

Usage

```
## S3 method for class 'glm_betaselect'
predict(
```

```
  object,
  model_type = c("beta", "standardized", "raw", "unstandardized"),
  newdata,
  ...
)
```

Arguments

object	A glm_betaselect-class object.
model_type	The model from which the the predicted values are computed. For "beta" or "standardized", the model is the one after selected variables standardized. For "raw" or "unstandardized", the model is the one before standardization was done.
newdata	If set to a data frame, the predicted values are computed using this data frame. The data must be unstandardized. That is, the variables are of the same units as in the data frame used in glm_betaselect(). If model_type is "beta" or "standardized", it will be standardized using the setting of to_standardize when object is created in glm_betaselect().
...	Arguments to be passed to stats::predict.glm(). Please refer to the help page of stats::predict.glm().

Details

It simply passes the model *before* or *after* selected variables are standardized to the predict-method of a glm object.

IMPORTANT:

Some statistics, such as prediction or confidence interval, which make use of the sampling variances and covariances of coefficient estimates *may* not be applicable to the models with one or more variables standardized. Therefore, they should only be used for exploratory purpose.

Value

It returns the output of stats::predict.glm().

Author(s)

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

See Also

glm_betaselect() and stats::predict.glm()

Examples

```
data_test_mod_cat$p <- scale(data_test_mod_cat$dv)[, 1]
data_test_mod_cat$p <- ifelse(data_test_mod_cat$p > 0,
                              yes = 1,
                              no = 0)
```

```
logistic_beta_x <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                  data = data_test_mod_cat,
                                  family = binomial,
                                  to_standardize = "iv")

predict(logistic_beta_x)
predict(logistic_beta_x, model_type = "raw")
```

`predict.lm_betaselect` *Predict Method for an 'lm_betaselect' Object*

Description

Compute the predicted values in a model fitted by `lm_betaselect()`.

Usage

```
## S3 method for class 'lm_betaselect'
predict(
  object,
  model_type = c("beta", "standardized", "raw", "unstandardized"),
  newdata,
  ...
)
```

Arguments

`object`	An `lm_betaselect`-class object.
`model_type`	The model from which the the predicted values are computed. For `"beta"` or `"standardized"`, the model is the one after selected variables standardized. For `"raw"` or `"unstandardized"`, the model is the one before standardization was done.
`newdata`	If set to a data frame, the predicted values are computed using this data frame. The data must be unstandardized. That is, the variables are of the same units as in the data frame used in `lm_betaselect()`. If `model_type` is `"beta"` or `"standardized"`, it will be standardized using the setting of `to_standardize` when `object` is created in `lm_betaselect()`.
`...`	Arguments to be passed to `stats::predict.lm()`. Please refer to the help page of `stats::predict.lm()`.

Details

It simply passes the model *before* or *after* selected variables are standardized to the `predict`-method of an `lm` object.

IMPORTANT:
Some statistics, such as prediction or confidence interval, which make use of the sampling variances and covariances of coefficient estimates *may* not be applicable to the models with one or more variables standardized. Therefore, they should only be used for exploratory purpose.

Value

It returns the output of `stats::predict.lm()`.

Author(s)

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

See Also

`lm_betaselect()` and `stats::predict.lm()`

Examples

```
data(data_test_mod_cat)

lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                           data = data_test_mod_cat,
                           to_standardize = "iv")
predict(lm_beta_x)
predict(lm_beta_x, model_type = "raw")
```

---

print.lav_betaselect *Print a 'lav_betaselect' Object*

Description

Print method for a 'lav_betaselect' object, which is the output of `lav_betaselect()`.

Usage

```
## S3 method for class 'lav_betaselect'
print(
  x,
  ...,
  nd = 3,
  output = c("lavaan.printer", "table"),
  standardized_only = TRUE,
  show_Bs.by = FALSE,
  show_ustd = !show_Bs.by,
  by_group = TRUE,
  na_str = " ",
```

```
  sig_stars = TRUE,
  ci_sig = TRUE
)
```

Arguments

x	A lav_betaselect-class object, such as the output of lav_betaselect().
...	Optional arguments to be passed to print() methods.
nd	The number of digits after the decimal place. Default is 3.
output	String. How the results are printed. Default is "lavaan.printer", and the results will be printed in a format similar to the printout of the output of the summary-method of a 'lavaan'-class object. If set to "table", the results are printed in a table format similar to that of lavaan::parameterEstimates() with output set to "data.frame".
standardized_only	Logical. If TRUE, the default, only the results for the standardized solution will be printed. If FALSE, then the standardized solution is printed alongside the unstandardized solution, as in the printout of the output of summary() of a 'lavaan'-class object.
show_Bs.by	Logical. If TRUE and output is "lavaan.printer", then the column "Bs.by" is shown, indicating, for each parameter, the variables standardized. This column is not shown if output is not "lavaan.printer".
show_ustd	Logical. If TRUE, then a column will be added to indicate whether a parameter is *unstandardized*. That is, its estimate is equal to that in the original unstandardized solution.
by_group	If TRUE, the default, and the model has more than one group, sections will be grouped by groups first, as in the print out of summary() in lavaan. If FALSE, then the sections will be grouped by sections first.
na_str	The string to be used for cells with NA. Default is " ", a whitespace.
sig_stars	If TRUE, the default, symbols such as asterisks (*, **, ***) will be used to denote whether a beta-select is significant.
ci_sig	If TRUE, the default, a beta-select will be denoted as significant or not significant based on its confidence interval.

Details

The default format of the printout, "lavaan.printer", is similar to that of the summary() of a lavaan object. Users can also select whether only the standardized solution is printed or whether the standardized solution is appended to the right of the printout.

If output is set to "table"' the format is that of [lavaan::parameterEstimates()] with output = "data.frame"', which is compact but not easy to read.

Value

x is returned invisibly. Called for its side effect.

Author(s)

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

See Also

lav_betaselect(). This function is adapted from semhelpinghands::print.std_solution_boot().

Examples

```
library(lavaan)
# Need to mean-center iv and mod
data_test_medmod$iv <- data_test_medmod$iv - mean(data_test_medmod$iv)
data_test_medmod$mod <- data_test_medmod$mod - mean(data_test_medmod$mod)
mod <-
"
med ~ iv + mod + iv:mod
dv ~ med + iv
"
fit <- sem(mod,
           data_test_medmod,
           fixed.x = TRUE)
summary(fit)
fit_beta <- lav_betaselect(fit,
                           to_standardize = c("iv", "dv"))
fit_beta
print(fit_beta)
print(fit_beta, show_Bs.by = TRUE)
print(fit_beta, output = "table")
```

---

std_data *Standardize Selected Variables*

Description

Standardize selected variables in a data frame or similar object.

Usage

```
std_data(data, to_standardize)
```

Arguments

data	A data frame or similar object.
to_standardize	A character vector of the column names of variables to be standardized.

Details

This is a helper functions to be used by lm_betaselect() and glm_betaselect(). It assumes that the variables selected has been checked whether they are numeric.

Value

A data frame similar to data, with selected variables standardized.

Author(s)

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

Examples

```
data(data_test_mod_cat)
dat <- data_test_mod_cat
dat <- std_data(dat, to_standardize = c("iv", "dv"))
colMeans(dat[, c("dv", "iv")])
apply(dat[, c("dv", "iv")], 2, sd)
```

---

summary.glm_betaselect *Summary of an 'glm_betaselect'-Class Object*

Description

The summary method for glm_betaselect-class objects.

Usage

```
## S3 method for class 'glm_betaselect'
summary(
  object,
  dispersion = NULL,
  correlation = FALSE,
  symbolic.cor = FALSE,
  trace = FALSE,
  test = c("LRT", "Rao"),
  se_method = c("boot", "bootstrap", "z", "glm", "default"),
  ci = TRUE,
  level = 0.95,
  boot_type = c("perc", "bc"),
  boot_pvalue_type = c("asymmetric", "norm"),
  type = c("beta", "standardized", "raw", "unstandardized"),
  print_raw = c("none", "before_ci", "after_ci"),
  transform_b = NULL,
  transform_b_name = NULL,
  ...
)

## S3 method for class 'summary.glm_betaselect'
```

```
print(
  x,
  est_digits = 3,
  symbolic.cor = x$symbolic.cor,
  signif.stars = getOption("show.signif.stars"),
  show.residuals = FALSE,
  z_digits = 3,
  pvalue_less_than = 0.001,
  ...
)
```

Arguments

object	The output of glm_betaselect().
dispersion	The dispersion parameter. If NULL, then it is extracted from the object. If a scalar, it will be used as the dispersion parameter. See stats::summary.glm() for details.
correlation	If TRUE, the correlation matrix of the estimates will be returned. The same argument in stats::summary.glm(). Default is FALSE.
symbolic.cor	If TRUE, correlations are printed in symbolic form as in stats::summary.glm(). Default is FALSE.
trace	Logical. Whether profiling will be traced when forming the confidence interval if se_method is "default", "z", or "glm". Ignored if ci is FALSE. See stats::confint.glm() for details.
test	The test used for se_method is "default", "z", or "glm". Ignored if ci is FALSE. See stats::confint.glm() for details.
se_method	The method used to compute the standard errors and confidence intervals (if requested). If bootstrapping was requested when calling glm_betaselect() and this argument is set to "bootstrap" or "boot", the bootstrap standard errors are returned. If bootstrapping was not requested or if this argument is set to "z", "glm", or "default", then the usual glm standard errors are returned. Default is "boot".
ci	Logical. Whether confidence intervals are computed. Default is FALSE.
level	The level of confidence, default is .95, returning the 95% confidence interval.
boot_type	The type of bootstrap confidence intervals, if requested. Currently, it supports "perc", percentile bootstrap confidence intervals, and "bc", bias-corrected bootstrap confidence interval.
boot_pvalue_type	The type of *p*-values if se_method is "boot" or "bootstrap". If "norm", then the *z* score is used to compute the *p*-value using a standard normal distribution. If "asymmetric", the default, then the method presented in Asparouhov and Muthén (2021) is used to compute the *p*-value based on the bootstrap distribution.
type	String. If "unstandardized" or "raw", the output *before* standardization are used If "beta" or "standardized", then the output *after* selected variables standardized are returned. Default is "beta".

print_raw — Control whether the estimates before selected standardization are printed when type is "beta" or "standardized". If "none", the default, then it will not be printed. If set to "before_ci" and ci is TRUE, then will be inserted to the left of the confidence intervals. If set to "after_ci"andciisTRUE, then will be printed to the right of the standardized estimates.

transform_b — The function to be used to transform the confidence limits. For example, if set to exp, the confidence limits will be exponentiated. Users need to decide whether the transformed limits are meaningful. Default is NULL.

transform_b_name — If transform_b is a function, then this is the name of the transformed coefficients. Default is "Estimate(Transformed)"

... — Additional arguments passed to other methods.

x — The output of summary.glm_betaselect().

est_digits — The number of digits after the decimal to be displayed for the coefficient estimates, their standard errors, and confidence intervals (if present). Note that the values will be rounded to this number of digits before printing. If all digits at this position are zero for all values, the values may be displayed with fewer digits. Note that the coefficient table is printed by stats::printCoefmat(). If some numbers are vary large, the number of digits after the decimal may be smaller than est_digits due to a limit on the column width.

signif.stars — Whether "stars" (asterisks) are printed to denote the level of significance achieved for each coefficient. Default is TRUE.

show.residuals — If TRUE, a summary of the deviance residuals will be printed. Default is FALSE.

z_digits — The number of digits after the decimal to be displayed for the z or similar statistic (in the column "z value").

pvalue_less_than — If a p-value is less than this value, it will be displayed with "<(this value)". For example, if pvalue_less_than is .001, the default, p-values less than .001 will be displayed as <.001. This value also determines the printout of the p-value of the F statistic. (This argument does what eps.Pvalue does in stats::printCoefmat().)

Details

By default, it returns a summary.glm_betaselect-class object for the results with selected variables standardized. By setting type to "raw" or "unstandardized", it returns the summary for the results *before* standardization.

The print method of summary.glm_betaselect-class objects is adapted from stdmod::print.summary.std_selected().

Value

It returns an object of class summary.glm_betaselect, which is similar to the output of stats::summary.glm(), with additional information on the standardization and bootstrapping, if requested.

The print-method of summary.glm_betaselect is called for its side effect. The object x is returned invisibly.

Author(s)

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

References

Asparouhov, A., & Muthén, B. (2021). Bootstrap p-value computation. Retrieved from https://www.statmodel.com/download Bootstrap%20-%20Pvalue.pdf

See Also

glm_betaselect()

Examples

```
data_test_mod_cat$p <- scale(data_test_mod_cat$dv)[, 1]
data_test_mod_cat$p <- ifelse(data_test_mod_cat$p > 0,
                              yes = 1,
                              no = 0)
# bootstrap should be set to 2000 or 5000 in real studies
logistic_beta_x <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                  data = data_test_mod_cat,
                                  family = binomial,
                                  to_standardize = "iv",
                                  do_boot = TRUE,
                                  bootstrap = 100,
                                  iseed = 1234)
summary(logistic_beta_x)
```

---

summary.lm_betaselect *Summary of an 'lm_betaselect'-Class Object*

Description

The summary method for lm_betaselect-class objects.

Usage

```
## S3 method for class 'lm_betaselect'
summary(
  object,
  correlation = FALSE,
  symbolic.cor = FALSE,
  se_method = c("boot", "bootstrap", "t", "lm", "ls"),
  ci = TRUE,
  level = 0.95,
  boot_type = c("perc", "bc"),
  boot_pvalue_type = c("asymmetric", "norm"),
```

```
  type = c("beta", "standardized", "raw", "unstandardized"),
  print_raw = c("none", "before_ci", "after_ci"),
  ...
)

## S3 method for class 'summary.lm_betaselect'
print(
  x,
  est_digits = 3,
  symbolic.cor = x$symbolic.cor,
  signif.stars = getOption("show.signif.stars"),
  tz_digits = 3,
  pvalue_less_than = 0.001,
  ...
)
```

Arguments

object	The output of lm_betaselect().
correlation	If TRUE, the correlation matrix of the estimates will be returned. The same argument in stats::summary.lm(). Default is FALSE.
symbolic.cor	If TRUE, correlations are printed in symbolic form as in stats::summary.lm(). Default is FALSE.
se_method	The method used to compute the standard errors and confidence intervals (if requested). If bootstrapping was requested when calling lm_betaselect() and this argument is set to "bootstrap" or "boot", the bootstrap standard errors are returned. If bootstrapping was not requested or if this argument is set to "t", "lm", or "ls", then the usual lm standard errors are returned. Default is "boot".
ci	Logical. Whether confidence intervals are computed. Default is TRUE.
level	The level of confidence, default is .95, returning the 95% confidence interval.
boot_type	The type of bootstrap confidence intervals, if requested. Currently, it supports "perc", percentile bootstrap confidence intervals, and "bc", bias-corrected bootstrap confidence interval.
boot_pvalue_type	The type of *p*-values if se_method is "boot" or "bootstrap". If "norm", then the *z* score is used to compute the *p*-value using a standard normal distribution. If "asymmetric", the default, then the method presented in Asparouhov and Muthén (2021) is used to compute the *p*-value based on the bootstrap distribution.
type	String. If "unstandardized" or "raw", the output *before* standardization are used If "beta" or "standardized", then the output *after* selected variables standardized are returned. Default is "beta".
print_raw	Control whether the estimates before selected standardization are printed when type is "beta" or "standardized". If "none", the default, then it will not be printed. If set to "before_ci" and ci is TRUE, then will be inserted to the left of the confidence intervals. If set to "after_ci" and ci is TRUE, then will be printed to the right of the standardized estimates.

`...`	Additional arguments passed to other methods.
`x`	The output of `summary.lm_betaselect()`.
`est_digits`	The number of digits after the decimal to be displayed for the coefficient estimates, their standard errors, and confidence intervals (if present). Note that the values will be rounded to this number of digits before printing. If all digits at this position are zero for all values, the values may be displayed with fewer digits. Note that the coefficient table is printed by `stats::printCoefmat()`. If some numbers are vary large, the number of digits after the decimal may be smaller than `est_digits` due to a limit on the column width. This value also determines the number of digits for displayed R-squared.
`signif.stars`	Whether "stars" (asterisks) are printed to denote the level of significance achieved for each coefficient. Default is `TRUE`.
`tz_digits`	The number of digits after the decimal to be displayed for the *t* or similar statistic (in the column `"t value"` or `"z value"`). This value also determines the number of digits for the *F* statistic for the R-squared.
`pvalue_less_than`	If a *p*-value is less than this value, it will be displayed with `"<(this value)"`. For example, if `pvalue_less_than` is .001, the default, *p*-values less than .001 will be displayed as `<.001`. This value also determines the printout of the *p*-value of the *F* statistic. (This argument does what `eps.Pvalue` does in `stats::printCoefmat()`.)

Details

By default, it returns a `summary.lm_betaselect`-class object for the results with selected variables standardized. By setting `type` to `"raw"` or `"unstandardized"`, it return the summary for the results *before* standardization.

The `print` method of `summary.lm_betaselect`-class objects is adapted from `stdmod::print.summary.std_selected()`.

Value

It returns an object of class `summary.lm_betaselect`, which is similar to the output of `stats::summary.lm()`, with additional information on the standardization and bootstrapping, if requested.

The `print`-method of `summary.lm_betaselect` is called for its side effect. The object `x` is returned invisibly.

Author(s)

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

References

Asparouhov, A., & Muthén, B. (2021). Bootstrap p-value computation. Retrieved from https://www.statmodel.com/download Bootstrap%20-%20Pvalue.pdf

See Also

`lm_betaselect()`

Examples

```
data(data_test_mod_cat)

# bootstrap should be set to 2000 or 5000 in real studies
lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                           data = data_test_mod_cat,
                           to_standardize = "iv",
                           do_boot = TRUE,
                           bootstrap = 100,
                           iseed = 1234)

summary(lm_beta_x)
summary(lm_beta_x, ci = TRUE)
summary(lm_beta_x, boot_pvalue_type = "norm")
summary(lm_beta_x, type = "raw")
```

---

vcov.lm_betaselect *The 'vcov' Method for 'lm_betaselect' and* glm_betaselect *Objects*

Description

Compute the variance-covariance matrix of estimates in the output of lm_betaselect() or glm_betaselect().

Usage

```
## S3 method for class 'lm_betaselect'
vcov(
  object,
  method = c("boot", "bootstrap", "ls", "default"),
  type = c("beta", "standardized", "raw", "unstandardized"),
  warn = TRUE,
  ...
)

## S3 method for class 'glm_betaselect'
vcov(
  object,
  method = c("boot", "bootstrap", "ls", "default"),
  type = c("beta", "standardized", "raw", "unstandardized"),
  warn = TRUE,
  ...
)
```

Arguments

object	The output of lm_betaselect() or an lm_betaselect-class object, or the output of glm_betaselect() or a glm_betaselect-class object.

method	The method used to compute the variance-covariance matrix. If bootstrapping was requested when calling lm_betaselect() or glm_betaselect() and this argument is set to "bootstrap" or "boot", the bootstrap variance-covariance matrix is returned. If bootstrapping was not requested or if this argument is set to "ls" or "default", then the usual lm or glm variance-covariance matrix is returned, with a warning raised unless type is "raw" or "unstandardized". Default is "boot".
type	String. If "unstandardized" or "raw", the variance-covariance matrix of the coefficients *before* standardization are returned. If "beta" or "standardized", then the variance-covariance matrix of the coefficients *after* selected variables standardized are returned. Default is "beta".
warn	Logical. WHether a warning will be raised is OLS (or WLS) variance-covariance matrix is requested for the model with some variables standardized (i.e., type is "beta" or "standardized"). Default is TRUE.
...	Other arguments to be passed to stats::vcov().

Details

The type of variance-covariance matrix depends on the object. If bootstrapping was requested, by default it returns the bootstrap variance-covariance matrix. Otherwise, it returns the default variance-covariance matrix and raises a warning.

Support for other type of variance-covariance matrix will be added.

Value

A matrix of the variances and covariances of the parameter estimates.

Author(s)

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

See Also

lm_betaselect() and glm_betaselect()

Examples

```
data(data_test_mod_cat)

# bootstrap should be set to 2000 or 5000 in real studies
lm_beta_x <- lm_betaselect(dv ~ iv*mod + cov1 + cat1,
                           data = data_test_mod_cat,
                           to_standardize = "iv",
                           do_boot = TRUE,
                           bootstrap = 100,
                           iseed = 1234)
vcov(lm_beta_x)
# A warning is expected for the following call
vcov(lm_beta_x, method = "ls")
```

```
vcov(lm_beta_x, type = "raw")



data_test_mod_cat$p <- scale(data_test_mod_cat$dv)[, 1]
data_test_mod_cat$p <- ifelse(data_test_mod_cat$p > 0,
                              yes = 1,
                              no = 0)
# bootstrap should be set to 2000 or 5000 in real studies
logistic_beta_x <- glm_betaselect(p ~ iv*mod + cov1 + cat1,
                                  data = data_test_mod_cat,
                                  family = binomial,
                                  to_standardize = "iv",
                                  do_boot = TRUE,
                                  bootstrap = 100,
                                  iseed = 1234)
vcov(logistic_beta_x)
# A warning is expected for the following call
vcov(logistic_beta_x, method = "default")
vcov(logistic_beta_x, type = "raw")
```

Index

∗ **datasets**
    data_test_medmod, 10
    data_test_mod_cat, 11
    data_test_mod_cat2, 12
    data_test_mod_cat_binary, 13

anova.glm_betaselect
        *(*anova.lm_betaselect*)*, 2
anova.glm_betaselect(), *23*
anova.lm_betaselect, 2

coef.glm_betaselect
        *(*coef.lm_betaselect*)*, 5
coef.glm_betaselect(), *23*
coef.lav_betaselect, 4
coef.lm_betaselect, 5
coef.lm_betaselect(), *23*
confint.glm_betaselect
        *(*confint.lm_betaselect*)*, 8
confint.glm_betaselect(), *23*
confint.lav_betaselect, 7
confint.lm_betaselect, 8
confint.lm_betaselect(), *23*
cooks.distance(), *23*

data_test_medmod, 10
data_test_mod_cat, 11
data_test_mod_cat2, 12
data_test_mod_cat_binary, 13

getCall.glm_betaselect
        *(*getCall.lm_betaselect*)*, 13
getCall.lm_betaselect, 13
glm(), *21*
glm_betaselect *(*lm_betaselect*)*, 20
glm_betaselect(), *2, 3*, *5, 6*, *8, 9*, *14*, *21–23*, *25, 26*, *30*, *32*, *34*, *37, 38*

lav_betaselect, 15
lav_betaselect(), *4*, *7*, *28–30*
lavaan::bootstrapLavaan(), *16, 17*
lavaan::cfa(), *15*
lavaan::parameterEstimates(), *17, 18*, *29*
lavaan::sem(), *15*
lm(), *21*
lm_betaselect, 20
lm_betaselect(), *2, 3*, *5, 6*, *8–10*, *14*, *21–23*, *27, 28*, *30*, *35–38*

manymome::do_boot(), *16*

predict.glm_betaselect, 25
predict.glm_betaselect(), *23*
predict.lm_betaselect, 27
predict.lm_betaselect(), *23*
print(), *29*
print.glm_betaselect *(*lm_betaselect*)*, 20
print.glm_betaselect(), *24*
print.lav_betaselect, 28
print.lav_betaselect(), *19*
print.lm_betaselect *(*lm_betaselect*)*, 20
print.lm_betaselect(), *24*
print.summary.glm_betaselect
        *(*summary.glm_betaselect*)*, 31
print.summary.lm_betaselect
        *(*summary.lm_betaselect*)*, 34

raw_output *(*lm_betaselect*)*, 20
raw_output(), *22, 23*
rstandard(), *23*

semhelpinghands::print.std_solution_boot(), *30*
set.seed(), *21*
stats::anova(), *3*
stats::anova.glm(), *3*
stats::coef(), *6*
stats::confint.glm, *9*
stats::confint.glm(), *9*, *32*
stats::getCall(), *14*
stats::glm(), *14*, *21*, *23*

stats::lm(), *14*, *21*, *23*
stats::predict.glm(), *26*
stats::predict.lm(), *27, 28*
stats::printCoefmat(), *33*, *36*
stats::summary.glm(), *32, 33*
stats::summary.lm(), *35, 36*
stats::update(), *23*
stats::vcov(), *38*
std_data, 30
stdmod::print.summary.std_selected(), *33*, *36*
summary(), *29*
summary.glm_betaselect, 31
summary.glm_betaselect(), *23*, *33*
summary.lm_betaselect, 34
summary.lm_betaselect(), *23*, *36*

vcov.glm_betaselect *(*vcov.lm_betaselect*)*, 37
vcov.glm_betaselect(), *23*
vcov.lm_betaselect, 37
vcov.lm_betaselect(), *23*