Package 'stdmod'

September 22, 2024

Title Standardized Moderation Effect and Its Confidence Interval

Version 0.2.11

Description Functions for computing a standardized moderation effect in moderated regression and forming its confidence interval by nonparametric bootstrapping as proposed in Cheung, Cheung, Lau, Hui, and Vong (2022) <doi:10.1037/hea0001188>. Also includes simple-to-use functions for computing conditional effects (unstandardized or standardized) and plotting moderation effects.

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

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

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests testthat, knitr, rmarkdown, visreg, lm.beta

Config/testthat/edition 3

Config/testthat/parallel true

Depends R (>= 4.0.0)

Imports boot, ggplot2, stats, utils, lavaan, manymome, rlang

VignetteBuilder knitr

NeedsCompilation no

Author Shu Fai Cheung [aut, cre] (<https://orcid.org/0000-0002-9871-9448>), David Weng Ngai Vong [ctb]

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

Repository CRAN

Date/Publication 2024-09-22 12:30:02 UTC

Contents

`add1.std_selected` *The 'add1' Method for a 'std_selected' Class Object*

Description

Intercept the `add1()` method and raise an error.

Usage

```
## S3 method for class 'std_selected'
add1(object, ...)
```

Arguments

`object`	The output of `std_selected()` or `std_selected_boot()`.
`...`	Additional arguments. They will be ignored.

Details

add1() should not be used after the output of lm() is processed by std_selected() or std_selected_boot().

Value

It returns nothing. It is called for its side effect.

Author(s)

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

---

coef.cond_effect *Conditional Effect in a 'cond_effect'-Class Object*

Description

Return the estimates of the conditional effects in the output of cond_effect() or cond_effect_boot().

Usage

```
## S3 method for class 'cond_effect'
coef(object, ...)
```

Arguments

object	The output of cond_effect() or cond_effect_boot().
...	Optional arguments. Ignored by the function.

Details

It just extracts and returns the column of conditional effects in a cond_effect-class object.

Value

A numeric vector: The estimates of the conditional effects in a cond_effect-class object.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)
out <- cond_effect(lm_raw, x = iv, w = mod)
out
coef(out)

lm_std <- std_selected(lm_raw, to_standardize = ~ iv + mod)
out <- cond_effect(lm_std, x = iv, w = mod)
out
coef(out)

# Categorical moderator
lm_cat <- lm(dv ~ iv*cat1 + v1, dat)
summary(lm_cat)
out <- cond_effect(lm_cat, x = iv, w = cat1)
out
coef(out)
```

---

coef.stdmod_lavaan *Standardized Moderation Effect in a 'stdmod_lavaan' Class Object*

Description

Return the estimate of the standardized moderation effect in the output of `stdmod_lavaan()`.

Usage

```
## S3 method for class 'stdmod_lavaan'
coef(object, ...)
```

Arguments

`object`	The output of `stdmod_lavaan()`.
`...`	Optional arguments. Ignored by the function.

Details

It just extracts and returns the element `stdmod`.

Value

A scalar: The estimate of the standardized moderation effect.

Author(s)

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

Examples

```
# Load a test data of 500 cases
dat <- test_mod1
library(lavaan)

mod <-
"
med ~ iv + mod + iv:mod + cov1
dv ~ med + cov2
"
fit <- sem(mod, dat)
coef(fit)

# Compute the standardized moderation effect
out_noboot <- stdmod_lavaan(fit = fit,
                            x = "iv",
                            y = "med",
                            w = "mod",
                            x_w = "iv:mod")
coef(out_noboot)

# Compute the standardized moderation effect and
# its confidence interval based on nonparametric bootstrapping
# Fit the model with bootstrap confidence intervals
# At least 2000 bootstrap samples should be used
# in real research. 50 is used here only for
# illustration.
fit <- sem(mod, dat, se = "boot", bootstrap = 50,
           iseed = 89574)
out_boot <- stdmod_lavaan(fit = fit,
                          x = "iv",
                          y = "med",
                          w = "mod",
                          x_w = "iv:mod",
                          boot_ci = TRUE)
coef(out_boot)
```

---

cond_effect *Conditional Effects*

Description

Compute the conditional effects in a moderated regression model.

Usage

```
cond_effect(
  output,
  x = NULL,
  w = NULL,
  w_method = c("sd", "percentile"),
  w_percentiles = c(0.16, 0.5, 0.84),
  w_sd_to_percentiles = NA,
  w_from_mean_in_sd = 1,
  w_values = NULL
)

cond_effect_boot(
  output,
  x = NULL,
  w = NULL,
  ...,
  conf = 0.95,
  nboot = 100,
  boot_args = NULL,
  save_boot_est = TRUE,
  full_output = FALSE,
  do_boot = TRUE
)
```

Arguments

output	The output from stats::lm(). It can also accept the output from std_selected() or std_selected_boot().
x	The focal variable (independent variable), that is, the variable with its effect on the outcome variable (dependent) being moderated. It must be a numeric variable.
w	The moderator. Both numeric variables and categorical variables (character or factor) are supported.
w_method	How to define "low", "medium", and "high" for the moderator levels. Default is in terms of mean and standard deviation (SD) of the moderator, "sd": "low", "medium", and "high" are one SD below mean, mean, and one SD above mean, respectively. If equal to "percentile", then percentiles of the moderator in the dataset are used: "low", "medium", and "high" are 16th, 50th (median), and 84th percentiles, respectively. Ignored if w is categorical.
w_percentiles	If w_method is "percentile", then this argument specifies the three percentiles to be used, divided by 100. It must be a vector of two numbers. The default is c(.16, .50, .84), the 16th, 50th, and 84th percentiles, which corresponds approximately to one SD below and above mean in a normal distribution, respectively. Ignored if w is categorical.
w_sd_to_percentiles	If w_method is "percentile" and this argument is set to a number, this number

will be used to to determine the percentiles to be used. The lower percentile is the percentile in a normal distribution that is w_sd_to_percentiles SD below the mean. The upper percentile is the percentile in a normal distribution that is w_sd_to_percentiles SD above the mean. Therefore, if w_sd_to_percentiles is set to 1, then the lower and upper percentiles are 16th and 84th, respectively. Default is NA.

w_from_mean_in_sd
: How many SD from mean is used to define "low" and "high" for the moderator. Default is 1. Ignored if w is categorical.

w_values
: The values of w to be used. Default is NULL. If a numeric vector is supplied, these values will be used to compute the conditional effects. Other arguments on generating levels are ignored. Note that, if w has been standardized or centered, these values are for the standardized or centered w. The values will always be sorted. This argument is ignored if w is categorical.

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

conf
: The level of confidence for the confidence interval. Default is .95, to get 95% confidence intervals.

nboot
: The number of bootstrap samples. Default is 100.

boot_args
: A named list of arguments to be passed to boot::boot(). Default is NULL.

save_boot_est
: If TRUE, the default, the bootstrap estimates will be saved in the element boot_est of the output.

full_output
: Whether the full output from boot::boot() will be returned. Default is FALSE. If TRUE, the full output from boot::boot() will be saved in the element boot_out of the output.

do_boot
: Whether bootstrapping confidence intervals will be formed. Default is TRUE. If FALSE, all arguments related to bootstrapping will be ignored.

Details

cond_effect() uses the centering approach to find the conditional effect of the focal variable. For each level of the moderator, the value for this level is subtracted from the moderator scores, and the model is fitted to the modified data. The coefficient of the focal variable is then the conditional effect of the focal variable when the moderator's score is equal this value.

cond_effect_boot() function is a wrapper of cond_effect(). It calls cond_effect() once for each bootstrap sample, and then computes the nonparametric bootstrap percentile confidence intervals (Cheung, Cheung, Lau, Hui, & Vong, 2022). If the output object is the output of std_selected() or std_selected_boot(), in which mean-centering and/or standardization have been conducted, they will be repeated in each bootstrap sample. Therefore, like std_selected_boot(), it can be used for form nonparametric bootstrap confidence intervals for standardized effects, though cond_effect_boot() does this for the standardized conditional effects.

This function ignores bootstrapping done by std_selected_boot(). It will do its own bootstrapping.

If do_boot is FALSE, then the object it returns is identical to that by cond_effect().

This function intentionally does not have an argument for setting the seed for random number. Users are recommended to set the seed, e.g., using set.seed() before calling it, to ensure reproducibility.

Value

`cond_effect()` returns a data-frame-like object of the conditional effects. The class is `cond_effect` and the print method will print additional information of the conditional effects. Additional information is stored in the following attributes:

- `call`: The original call.
- `output`: The `output` object, such as the output from `lm()`.
- `x`, `y`, and `w`: The three variables used to compute the conditional effects: focal variable (`x`), outcome variable (`y`), and the moderator (`w`).
- `w_method`: The method used to determine the values of the moderator at the selected levels.
- `w_percentiles` The percentiles to use if `w_method = "percentile"`.
- `w_sd_to_percentiles`: If not equal to `NA`, this is a scalar, the number of standard deviation from the mean used to determine the percentiles for the "low" and "high" levels of the moderator.
- `w_from_mean_in_sd`: The number of SD above or below the mean, for determining the "low" and "high" levels of the moderator if `w_method` is `"sd"`.
- `w_empirical_percentiles`: The actual percentile levels in the dataset for the selected levels of the moderator. A numeric vector.
- `w_empirical_z`: The actual distance from the mean, in SD, of each selected level of the moderator. A numeric vector.
- `y_standardized`, `x_standardized`, and `w_standardized`: Each of them is a logical scalar, indicating whether the outcome variable, focal variable, and moderator are standardized.

`cond_effect_boot()` also returns a data-frame-like object of the conditional effects of the class `cond_effect`, with additional information from the bootstrapping stored in these attributes:

- `boot_ci`: A data frame of the bootstrap confidence intervals of the conditional effects.
- `nboot`: The number of bootstrap samples requested.
- `conf`: The level of confidence, in proportion.
- `boot_est`: A matrix of the bootstrap estimates of the conditional effects. The number of rows equal to `nboot`, and the number of columns equal to the number of levels of the moderator.
- `cond_effect_boot_call`: The call to `cond_effect_boot()`.
- `boot_out`: If available, the original output from `boot::boot()`.

Functions

- `cond_effect_boot()`: A wrapper of `cond_effect()` that forms nonparametric bootstrap confidence intervals.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)
cond_effect(lm_raw, x = iv, w = mod)

lm_std <- std_selected(lm_raw, to_standardize = ~ iv + mod)
cond_effect(lm_std, x = iv, w = mod)

# Categorical moderator
lm_cat <- lm(dv ~ iv*cat1 + v1, dat)
summary(lm_cat)
cond_effect(lm_cat, x = iv, w = cat1)


# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)

lm_std <- std_selected(lm_raw, to_standardize = ~ iv + mod)
cond_effect(lm_std, x = iv, w = mod)

# Form nonparametric bootstrap confidence intervals
# Use 2000 or even 5000 for nboot in real research
out <- cond_effect_boot(lm_std, x = iv, w = mod, nboot = 50)
out
```

---

confint.cond_effect *Confidence Intervals for a 'cond_effect' Class Object*

Description

Return the confidence intervals of estimates conditional effect in the output of `cond_effect()` or `cond_effect_boot()`.

Usage

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

Arguments

object	The output of cond_effect() or cond_effect_boot().
parm	Ignored by this function. The confidence intervals for all available levels will be returned.
level	The level of confidence. For the confidence intervals returned by lm(), default is .95, i.e., 95%. For the bootstrap percentile confidence intervals, default is the level used in calling cond_effect_boot().
type	The type of the confidence intervals. If est to "lm", returns the confidence interval given by the confint() method of lm(). If set to "boot", the bootstrap percentile confidence intervals are returned. Default is "boot" if bootstrap estimates are stored in object, and "lm" if bootstrap estimates are not stored.
...	Additional arguments. Ignored.

Details

If bootstrapping is used to form the confidence interval by cond_effect_boot(), users can request the percentile confidence intervals of the bootstrap estimates. This method does not do the bootstrapping itself.

Value

A matrix of the confidence intervals.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)
out <- cond_effect(lm_raw, x = iv, w = mod)
print(out, t_ci = TRUE)
confint(out)

lm_std <- std_selected(lm_raw, to_center = ~ iv + mod, to_scale = ~ iv + mod)
# Alternative: use to_standardize as a shortcut
# lm_std <- std_selected(lm_raw, to_standardize = ~ iv + mod)
out <- cond_effect(lm_std, x = iv, w = mod)
print(out, t_ci = TRUE)
confint(out)

# Categorical moderator
lm_cat <- lm(dv ~ iv*cat1 + v1, dat)
```

```
summary(lm_cat)
out <- cond_effect(lm_cat, x = iv, w = cat1)
print(out, t_ci = TRUE)
confint(out)
```

`confint.stdmod_lavaan` *Confidence Intervals for a 'stdmod_lavaan' Class Object*

Description

Return the confidence interval of the standardized moderation effect in the output of `stdmod_lavaan()`.

Usage

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

Arguments

`object`	The output of `stdmod_lavaan()`.
`parm`	Ignored. Always return the bootstrap confidence interval of the standardized moderation effect.
`level`	The level of confidence, default is .95, returning the 95% confidence interval.
`...`	Additional arguments. Ignored by the function.

Details

If bootstrapping is used to form the confidence interval by `stdmod_lavaan()`, users can request the percentile confidence interval of using the stored bootstrap estimate.

Value

A one-row matrix of the confidence intervals.

Author(s)

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

Examples

```
# Load a test data of 500 cases
dat <- test_mod1
library(lavaan)

mod <-
"
med ~ iv + mod + iv:mod + cov1
```

```
dv ~ med + cov2
"
fit <- sem(mod, dat)
coef(fit)

# Compute the standardized moderation effect and
# its confidence interval based on nonparametric bootstrapping
# Fit the model with bootstrap confidence intervals
# At least 2000 bootstrap samples should be used
# in real research. 50 is used here only for
# illustration.
fit <- sem(mod, dat, se = "boot", bootstrap = 50,
           iseed = 89574)
out_boot <- stdmod_lavaan(fit = fit,
                          x = "iv",
                          y = "med",
                          w = "mod",
                          x_w = "iv:mod",
                          boot_ci = TRUE)
confint(out_boot)
```

`confint.std_selected` *Confidence Intervals for a 'std_selected' Class Object*

Description

Return the confidence intervals of estimates in the output of `std_selected()` or `std_selected_boot()`.

Usage

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

Arguments

`object`	The output of `std_selected()` or `std_selected_boot()`.
`parm`	The parameters (coefficients) for which confidence intervals should be returned. If missing, the confidence intervals of all parameters will be returned.
`level`	The level of confidence. For the confidence intervals returned by `lm()`, default is .95, i.e., 95%. For the bootstrap percentile confidence intervals, default is the level used in calling `std_selected_boot()`. If a level different from that in the original call is specified, `full_output` needs to be set in the call to `std_selected_boot()` such that the original bootstrapping output is stored.
`type`	The type of the confidence intervals. If est to `"lm"`, returns the confidence interval given by the `confint()` method of `lm()`. If set to `"boot"`, the bootstrap percentile confidence intervals are returned. Default is `"boot"` if bootstrap estimates are stored in `object`, and `"lm"` if bootstrap estimates are not stored.
`...`	Arguments to be passed to `summary.lm()`.

Details

If bootstrapping is used to form the confidence interval by `std_selected_boot()`, users can request the percentile confidence intervals of the bootstrap estimates. This method does not do the bootstrapping itself.

Value

A matrix of the confidence intervals.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)

# Standardize all variables except for categorical variables.
# Interaction terms are formed after standardization.
lm_std <- std_selected(lm_raw, to_center = ~ .,
                               to_scale = ~ .)
# Alternative: use to_standardize as a shortcut
# lm_std <- std_selected(lm_raw, to_standardize = ~ .)
summary(lm_std)

confint(lm_std)

# Use to_standardize as a shortcut
lm_std2 <- std_selected(lm_raw, to_standardize = ~ .)
# The results are the same
confint(lm_std)
confint(lm_std2)
all.equal(confint(lm_std), confint(lm_std2))

# With bootstrapping
# nboot = 100 just for illustration. nboot >= 2000 should be used in read
# research.
set.seed(89572)
lm_std_boot <- std_selected_boot(lm_raw, to_scale = ~ .,
                                         to_center = ~ .,
                                         nboot = 100)
summary(lm_std_boot)

# Bootstrap percentile intervals, default when bootstrap was conduced

confint(lm_std_boot)
```

```
# Force OLS confidence intervals

confint(lm_std_boot, type = "lm")

# Use to_standardize as a shortcut
set.seed(89572)
lm_std_boot2 <- std_selected_boot(lm_raw, to_standardize = ~ .,
                                          nboot = 100)
# The results are the same
confint(lm_std_boot)
confint(lm_std_boot2)
all.equal(confint(lm_std_boot), confint(lm_std_boot2))
```

---

plotmod *Moderation Effect Plot*

Description

Plot the moderation effect in a regression model

Usage

```
plotmod(
  output,
  x,
  w,
  x_label,
  w_label,
  y_label,
  title,
  digits = 3,
  x_from_mean_in_sd = 1,
  w_from_mean_in_sd = 1,
  w_method = c("sd", "percentile"),
  w_percentiles = c(0.16, 0.84),
  x_method = c("sd", "percentile"),
  x_percentiles = c(0.16, 0.84),
  w_sd_to_percentiles = NA,
  x_sd_to_percentiles = NA,
  w_values = NULL,
  note_standardized = TRUE,
  no_title = FALSE,
  line_width = 1,
  point_size = 5,
  graph_type = c("default", "tumble")
)
```

Arguments

output	The output of stats::lm(), std_selected(), or std_selected_boot().
x	The name of the focal variable (x-axis) in 'output". It can be the name of the variable, with or without quotes. Currently only numeric variables are supported.
w	The name of the moderator in output. It can be the name of the variable, with or without quotes.
x_label	The label for the X-axis. Default is the value of x.
w_label	The label for the legend for the lines. Default is the value ofw.
y_label	The label for the Y-axis. Default is the name of the response variable in the model.
title	The title of the graph. If not supplied, it will be generated from the variable names or labels (in x_label, y_label, and w_label). If "", no title will be printed. This can be used when the plot is for manuscript submission and figures are required to have no titles.
digits	Number of decimal places to print. Default is 3.
x_from_mean_in_sd	How many SD from mean is used to define "low" and "high" for the focal variable. Default is 1.
w_from_mean_in_sd	How many SD from mean is used to define "low" and "high" for the moderator. Default is 1. Ignored if w is categorical.
w_method	How to define "high" and "low" for the moderator levels. Default is in terms of the standard deviation of the moderator, "sd". If equal to "percentile", then the percentiles of the moderator in the dataset are used. Ignored if w is categorical.
w_percentiles	If w_method is "percentile", then this argument specifies the two percentiles to be used, divided by 100. It must be a vector of two numbers. The default is c(.16, .84), the 16th and 84th percentiles, which corresponds approximately to one SD below and above mean for a normal distribution, respectively. Ignored if w is categorical.
x_method	How to define "high" and "low" for the focal variable levels. Default is in terms of the standard deviation of the focal variable, "sd". If equal to "percentile", then the percentiles of the focal variable in the dataset is used.
x_percentiles	If x_method is "percentile", then this argument specifies the two percentiles to be used, divided by 100. It must be a vector of two numbers. The default is c(.16, .84), the 16th and 84th percentiles, which corresponds approximately to one SD below and above mean for a normal distribution, respectively.
w_sd_to_percentiles	If w_method is "percentile" and this argument is set to a number, this number will be used to determine the percentiles to be used. The lower percentile is the percentile in a normal distribution that is w_sd_to_percentiles SD below the mean. The upper percentile is the percentile in a normal distribution that is w_sd_to_percentiles SD above the mean. Therefore, if w_sd_to_percentiles

is set to 1, then the lower and upper percentiles are 16th and 84th, respectively. Default is NA.

`x_sd_to_percentiles`
: If `x_method` is `"percentile"` and this argument is set to a number, this number will be used to determine the percentiles to be used. The lower percentile is the percentile in a normal distribution that is `x_sd_to_percentiles` SD below the mean. The upper percentile is the percentile in a normal distribution that is `x_sd_to_percentiles` SD above the mean. Therefore, if `x_sd_to_percentiles` is set to 1, then the lower and upper percentiles are 16th and 84th, respectively. Default is NA.

`w_values`
: The values of `w` to be used. Default is `NULL`. If a numeric vector is supplied, these values will be used to compute the conditional effects. Other arguments on generating levels are ignored. Note that, if `w` has been standardized or centered, these values are for the standardized or centered `w`. The values will always be sorted. This argument is ignored if `w` is categorical.

`note_standardized`
: If `TRUE`, will check whether a variable has SD nearly equal to one. If yes, will report this in the plot. Default is `TRUE`.

`no_title`
: If `TRUE`, title will be suppressed. Default is `FALSE`.

`line_width`
: The width of the lines as used in `ggplot2::geom_segment()`. Default is 1.

`point_size`
: The size of the points as used in `ggplot2::geom_point()`. Default is 5.

`graph_type`
: If `"default"`, the typical line-graph with equal end-points will be plotted. If `"tubmle"`, then the tumble graph proposed by Bodner (2016) will be plotted. Default is `"default"`.

Details

This function generate a basic `ggplot2` graph typically found in psychology manuscripts. It tries to check whether one or more variables are standardized, and report this in the plot if required.

This function only has features for typical plots of moderation effects. It is not intended to be a flexible tool for a fine control on the plots.

Value

A `ggplot2` graph. Plotted if not assigned to a name. It can be further modified like a usual `ggplot2` graph.

Author(s)

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

References

Bodner, T. E. (2016). Tumble graphs: Avoiding misleading end point extrapolation when graphing interactions from a moderated multiple regression analysis. *Journal of Educational and Behavioral Statistics, 41*(6), 593-604. doi:10.3102/1076998616657080

Examples

```
# Do a moderated regression by lm
lm_out <- lm(sleep_duration ~ age + gender + emotional_stability*conscientiousness, sleep_emo_con)
plotmod(lm_out,
        x = emotional_stability,
        w = conscientiousness,
        x_label = "Emotional Stability",
        w_label = "Conscientiousness",
        y_label = "Sleep Duration")

# Standardize all variables except for categorical variables
# Alternative: use to_standardize as a shortcut
# lm_std <- std_selected(lm_out,
#                        to_standardize = ~ .)
lm_std <- std_selected(lm_out,
                       to_scale = ~ .,
                       to_center = ~ .)
plotmod(lm_std,
        x = emotional_stability,
        w = conscientiousness,
        x_label = "Emotional Stability",
        w_label = "Conscientiousness",
        y_label = "Sleep Duration")

# Tumble Graph
plotmod(lm_std,
        x = emotional_stability,
        w = conscientiousness,
        x_label = "Emotional Stability",
        w_label = "Conscientiousness",
        y_label = "Sleep Duration",
        graph_type = "tumble")
```

---

print.cond_effect *Print a 'cond_effect' Class Object*

Description

Print the output of cond_effect() or cond_effect_boot().

Usage

```
## S3 method for class 'cond_effect'
print(
  x,
  nd = 3,
  nd_stat = 3,
  nd_p = 3,
```

```
  title = TRUE,
  model = TRUE,
  level_info = TRUE,
  standardized = TRUE,
  boot_info = TRUE,
  table_only = FALSE,
  t_ci = FALSE,
  t_ci_level = 0.95,
  ...
)
```

Arguments

x	The output of cond_effect() or cond_effect_boot().
nd	The number of digits for the variables.
nd_stat	The number of digits for test statistics (e.g., *t*).
nd_p	The number of digits for *p*-values.
title	If TRUE, print a title. Default is TRUE.
model	If TRUE, print the regression model. Default is TRUE.
level_info	If TRUE, print information for interpreting the levels of the moderator, such as the values of the levels and distance from the mean. Default is TRUE.
standardized	If TRUE and one or more variables are standardized, report it. Default is TRUE.‘
boot_info	If TRUE and bootstrap estimates are in x, print information about the bootstrapping, such as the number of bootstrap samples. Default is TRUE.
table_only	If TRUE, will suppress of other elements except for the table of conditional effects. Override arguments such as title, model, and level_info.
t_ci	If TRUE, will print the confidence intervals based on t statistics. These confidence intervals should not be used if some variables are standardized.
t_ci_level	The level of confidence of the confidence intervals based on t statistics. Default is .95.
...	Additional arguments. Ignored by this function.

Value

x is returned invisibility.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500
```

```
# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)

cond_effect(lm_raw, x = iv, w = mod)

lm_std <- std_selected(lm_raw, to_scale = ~ iv + mod,
                               to_center = ~ iv + mod)

cond_effect(lm_std, x = iv, w = mod)
```

`print.stdmod_lavaan` *Print a 'stdmod_lavaan' Class Object*

Description

Print the output of `stdmod_lavaan()`.

Usage

```
## S3 method for class 'stdmod_lavaan'
print(x, conf = 0.95, nd = 3, ...)
```

Arguments

`x`	The output of `stdmod_lavaan()`.
`conf`	If nonparametric bootstrapping has been conducted by `stdmod_lavaan()`, this is the level of confidence in proportion (.95 denotes 95%), of the confidence interval. Default is .95.
`nd`	The number of digits to be printed.
`...`	Optional arguments. Ignored.

Value

`x` is returned invisibly.

Author(s)

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

Examples

```
# Load a test data of 500 cases

dat <- test_mod1
library(lavaan)

mod <-
```

```
"
med ~ iv + mod + iv:mod + cov1
dv ~ med + cov2
"
fit <- sem(mod, dat)
coef(fit)

# Compute the standardized moderation effect
out_noboot <- stdmod_lavaan(fit = fit,
                            x = "iv",
                            y = "med",
                            w = "mod",
                            x_w = "iv:mod")
out_noboot

# Compute the standardized moderation effect and
# its percentile confidence interval based on nonparametric bootstrapping
# Fit the model with bootstrap confidence intervals
# At least 2000 bootstrap samples should be used
# in real research. 50 is used here only for
# illustration.
fit <- sem(mod, dat, se = "boot", bootstrap = 50,
           iseed = 89574)
out_boot <- stdmod_lavaan(fit = fit,
                          x = "iv",
                          y = "med",
                          w = "mod",
                          x_w = "iv:mod",
                          boot_ci = TRUE)

out_boot
```

`print.std_selected` *Print Basic Information of a 'std_selected' Class Object*

Description

Provide information of centering and scaling, along with basic model information printed by the `print()` method of `lm()`.

Usage

```
## S3 method for class 'std_selected'
print(x, ...)
```

Arguments

`x`	The output of `std_selected()` or `std_selected_boot()`.
`...`	Arguments to be passed to `print()` method of `lm()`.

Value

x is returned invisibly.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)

# Standardize all variables except for categorical variables.
# Interaction terms are formed after standardization.
lm_std <- std_selected(lm_raw, to_scale = ~ .,
                               to_center = ~ .)
lm_std

# With bootstrapping
# nboot = 100 just for illustration. nboot >= 2000 should be used in read
# research.
lm_std_boot <- std_selected_boot(lm_raw, to_scale = ~ .,
                                         to_center = ~ .,
                                         nboot = 100)
lm_std_boot
```

---

print.summary.std_selected

Print the Summary of a 'std_selected' Class Object

Description

Print the summary generated by summary() on the output of std_selected() or std_selected_boot().

Usage

```
## S3 method for class 'summary.std_selected'
print(
  x,
  ...,
  est_digits = 4,
  t_digits = 4,
  pvalue_less_than = 0.001,
```

```
  default_style = FALSE
)
```

Arguments

x	The output of `summary()`.
...	Arguments to be passed to `summary()`.
est_digits	The number of digits after the decimal to be displayed for the coefficient estimates, their standard errors, and bootstrap 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 if `default_style` is `FALSE`. Default if 4.
t_digits	The number of digits after the decimal to be displayed for the *t* statistic (in the column `"t value"`). This value also determines the number of digits for the *F* statistic for the R-squared if `default_style` is `FALSE`. Default is 4.
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 if `default_style` is `FALSE`. (This argument does what `eps.Pvalue` does in `stats::printCoefmat()`.)
default_style	Logical. If `FALSE`, the default, R-squared and *F* statistic will be displayed in a more readable style. If `TRUE`, then the default style in the printout of the `summary` of `lm()` output will be used.

Value

`x` is returned invisibly.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)

# Standardize all variables except for categorical variables.
# Interaction terms are formed after standardization.
lm_std <- std_selected(lm_raw, to_scale = ~ .,
```

```
                                to_center = ~ .)
summary(lm_std)

# With bootstrapping
# nboot = 100 just for illustration. nboot >= 2000 should be used in read
# research.
lm_std_boot <- std_selected_boot(lm_raw, to_scale = ~ .,
                                         to_center = ~ .,
                                         nboot = 100)
summary(lm_std_boot)
```

`sleep_emo_con` *Sample Dataset: Predicting Sleep Duration*

Description

A random subset from a real dataset. For illustration.

Usage

```
sleep_emo_con
```

Format

A data frame with 500 rows and six variables:

case_id Case ID, integer

sleep_duration Sleep duration in hours

conscientiousness Conscientiousness score, continuous

emotional_stability Emotional stability score, continuous

age Age in years

gender Gender, string, "female" or "male"

`stdmod` *Standardized Moderation Effect Given an 'lm' Output*

Description

Compute the standardized moderation effect in a moderated regression model.

Usage

```
stdmod(
  lm_out,
  x = NULL,
  w = NULL,
  y = NULL,
  x_rescale = TRUE,
  w_rescale = TRUE,
  y_rescale = TRUE
)

stdmod_boot(
  lm_out,
  ...,
  nboot = 100,
  conf = 0.95,
  boot_args = NULL,
  full_output = FALSE
)
```

Arguments

lm_out	The output from lm().
x	The focal variable, that is, the variable with its effect being moderated. If supplied, its standard deviation will be used for rescaling. Also called the independent variable in some models. Default is NULL.
w	The moderator. If supplied, its standard deviation will be used for rescaling. Default is NULL.
y	The outcome variable (dependent variable) . If supplied, its standard deviation will be used for rescaling. Default is NULL.
x_rescale	If TRUE, will rescale x by its standard deviation. Default is TRUE.
w_rescale	If TRUE, will rescale w by its standard deviation. Default is TRUE.
y_rescale	If TRUE, will rescale y by its standard deviation. Default is TRUE.
...	Parameters to be passed to stdmod().
nboot	The number of bootstrap samples. Default is 100.
conf	The level of confidence for the confidence interval. Default is .95.
boot_args	A named list of arguments to be passed to boot::boot(). Default is NULL.
full_output	Whether the full output from boot::boot() is returned. Default is FALSE.

Details

Two more general functions, std_selected() and std_selected_boot(), have been developed and can do what these functions do and more. Users are recommended to use them instead of stdmod() and stdmod_boot(). These two functions will not be updated in the near future.

Nevertheless, if computing the standardized moderation effect and forming its nonparametric bootstrap interval are all required, then these functions can still be used.

`stdmod()` computes the standardized moderation effect given an `lm()` output using the formula from Cheung, Cheung, Lau, Hui, and Vong (2022). Users specify the moderator, the focal variable (the variable with its effect on the outcome variable moderated), the outcome variable (dependent variable) , and the corresponding standardized moderation effect. Users can also select which variable(s) will be standardized.

`stdmod_boot()` is a wrapper of `stdmod()`. It computes the nonparametric bootstrap confidence interval of the standardized moderation effect, as suggested by Cheung, Cheung, Lau, Hui, and Vong (2022), given the output of `lm()`

Percentile interval from `boot::boot.ci()` is returned by this function. If other types of confidence intervals are desired, set `full_output = TRUE` and use `boot::boot.ci()` on the element `boot_out` in the output of this function.

Value

`stdmod()` returns a scalar: The standardized moderation effect.

`stdmod_boot()` returns a list with two elements. The element `ci` is a numeric vector of the bootstrap confidence interval. The element `boot_out`, if not `NA`, is the output of `boot::boot()`, which is used to do the bootstrapping.

Functions

- `stdmod()`: The base function for computing standardized moderation effect
- `stdmod_boot()`: A wrapper of `stdmod()` that computes the nonparametric bootstrap confidence interval of the standardized moderation effect.

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

Examples

```
# Load a test data of 500 cases

dat <- test_x_1_w_1_v_2_n_500

# Do regression as usual:
lm_raw <- lm(dv ~ iv*mod + v1 + v2, dat)
summary(lm_raw)

# The standard deviations of iv, dv, and mod:
sds <- apply(dat, 2, sd)
```

```
sds

# Compute the standardized moderation effect:
stdmod_xyw <- stdmod(lm_raw, x = iv, y = dv, w = mod)
stdmod_xyw
# By default, all three variables will be standardized.

# Check against self-computed standardized moderation effect:
coef(lm_raw)["iv:mod"] * sds["iv"] * sds["mod"] / sds["dv"]

# Standardize only the iv, i.e., do not standardized dv and the moderator:
stdmod_x <- stdmod(lm_raw, x = iv, y = dv, w = mod,
                   x_rescale = TRUE,  y_rescale = FALSE, w_rescale = FALSE)
stdmod_x
# Check against self-computed moderation effect with only iv standardized:
coef(lm_raw)["iv:mod"] * sds["iv"]


dat <- test_x_1_w_1_v_2_n_500
# Do regression as usual:
lm_raw <- lm(dv ~ iv*mod + v1 + v2, dat)

# Compute the standardized moderation effect.
# Form its confidence interval by nonparametric bootstrapping.
set.seed(85740917)
stdmod_xyw_boot <- stdmod_boot(lm_raw, x = iv, w = mod, y = dv, nboot = 100)
# In real analysis, nboot should be at least 2000.

# Print the ci
stdmod_xyw_boot$ci

# Repeat the analysis but keep the results from boot:
set.seed(85740917)
stdmod_xyw_boot <- stdmod_boot(lm_raw, x = iv, w = mod, y = dv,
                               nboot = 200, full_output = TRUE)
# In real analysis, nboot should be at least 2000.

# Print the 95% percentile confidence interval
stdmod_xyw_boot$ci
```

---

stdmod_lavaan *Standardized Moderation Effect and Its Bootstrap CI in 'lavaan'*

Description

Compute the standardized moderation effect in a structural equation model fitted by `lavaan::lavaan()` or its wrappers and form the nonparametric bootstrap confidence interval.

Usage

```
stdmod_lavaan(
  fit,
  x,
  y,
  w,
  x_w,
  standardized_x = TRUE,
  standardized_y = TRUE,
  standardized_w = TRUE,
  boot_ci = FALSE,
  boot_out = NULL,
  R = 100,
  conf = 0.95,
  use_old_version = FALSE,
  ...
)
```

Arguments

fit	The SEM output by lavaan::lavaan() or its wrappers.
x	The name of the focal variable in the model, the variable with its effect on the outcome variable being moderated.
y	The name of the outcome variable (dependent variable) in the model.
w	The name of the moderator in the model.
x_w	The name of the product term (x * w) in the model. It can be the variable generated by the colon operator, e.g., "x:w", which is only in the model and not in the original data set.
standardized_x	If TRUE, the default, x is standardized when computing the standardized moderation effect.
standardized_y	If TRUE, the default, y is standardized when computing the standardized moderation effect.
standardized_w	If TRUE, the default, w is standardized when computing the standardized moderation effect.
boot_ci	Boolean. Whether nonparametric bootstrapping will be conducted. Default is FALSE.
boot_out	If set to the output of manymome::do_boot(), the stored bootstrap estimates will be retrieved to form the bootstrap confidence interval. If set, bootstrap estimates stored in fit, if any, will not be used. Default is NULL.
R	(Not used in the current version. Used when use_old_version is set to TRUE.) The number of nonparametric bootstrapping samples. Default is 100. Set this to at least 2000 in actual use.
conf	The level of confidence. Default is .95, i.e., 95%.
use_old_version	If set to TRUE, it will use the bootstrapping method used in 0.2.7.4 or before. Included only for reproducing previous results if necessary. Default is FALSE.

... (Not used in the current version. Used when use_old_version is set to TRUE.) Optional arguments to be passed to boot::boot(). Parallel processing can be used by adding the appropriate arguments in boot::boot().

Details

Important Notes:

Starting from Version 0.2.7.5, of stdmod_lavaan() adopts an approach to bootstrapping different from that in the previous versions (0.2.7.4 and before), yielding bootstrapping results different from those in previous versions (for reasons explained later).

To reproduce results from the older version of this function, set use_old_version to TRUE.

How it works:

stdmod_lavaan() accepts a lavaan::lavaan object, the structural equation model output returned by lavaan::lavaan() and its wrappers (e.g, lavaan::sem()) and computes the standardized moderation effect using the formula in the appendix of Cheung, Cheung, Lau, Hui, and Vong (2022).

The standard deviations of the focal variable (the variable with its effect on the outcome variable being moderated), moderator, and outcome variable (dependent variable) are computed from the implied covariance matrix returned by lavaan::lavInspect(). Therefore, models fitted to data sets with missing data (e.g., with missing = "fiml") are also supported.

Partial standardization can also be requested. For example, standardization can be requested for only the focal variable and the outcome variable.

There are two ways to request nonparametric bootstrap confidence interval. First, the model is fitted with se = "boot" or se = "bootstrap" in lavaan. The stored bootstrap estimates will then be retrieved automatically to compute the standardized moderation effect. This is the most efficient approach if the bootstrap confidence intervals are also needed for other parameters in the model. Bootstrapping needs to be done only once.

Second, bootstrap estimates can be generated by manymome::do_boot(). The output is then supplied through the argument boot_out. Bootstrapping also only needs to be done once. This approach is appropriate when bootstrapping confidence intervals are not needed for other model parameters, or another type of confidence interval is needed when fitting the model. Please refer to the help page of manymome::do_boot() on how to use this function.

In both approaches, the standard deviations are also computed in each bootstrap samples. This ensures that the sampling variability of the standard deviations is also taken into account in computing the bootstrap confidence interval of the standardized moderation effect.

Note on the differences between the current version (Version 0.2.7.5 or later) and previous versions (0.2.7.4 and before):

In older versions, stdmod_lavaan() does not allow for partial standardization. Moreover, it uses boot::boot() to do the bootstrapping. Even with the same seed, the results from boot::boot() are not identical to those of lavaan with se = "boot" because they differ in the way the indices of resampling are generated. Both approaches are correct, They just use the generated random numbers differently. To have results consistent with those from lavaan, the current version of stdmod_lavaan() adopts a resampling algorithm identical to that of lavaan. Last, in older versions, stdmod_lavaan() does bootstrapping every time it is called. This is inefficient.

The bootstrapping results in the current version are not identical to those in older versions due to the use of different resampling algorithms, To reproduce previous results, set use_old_version to TRUE

Value

A list of class `stdmod_lavaan` with these elements:

- `stdmod`: The standardized moderation effect.
- `ci`: The nonparametric bootstrap confidence interval. `NA` if confidence interval not requested.
- `boot_out`: The raw output from `boot::boot()`. `NA` if confidence interval not requested.
- `fit`: The original fit object.

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

Examples

```
#Load a test data of 500 cases

dat <- test_mod1
library(lavaan)
mod <-
"
med ~ iv + mod + iv:mod + cov1
dv ~ med + cov2
"

fit <- sem(mod, dat)

# Compute the standardized moderation effect
out_noboot <- stdmod_lavaan(fit = fit,
                            x = "iv",
                            y = "med",
                            w = "mod",
                            x_w = "iv:mod")
out_noboot

# Compute the standardized moderation effect and
# its percentile confidence interval using
# nonparametric bootstrapping
# Fit the model with bootstrap confidence intervals
# At least 2000 bootstrap samples should be used
# in real research. 50 is used here only for
# illustration.
fit <- sem(mod, dat, se = "boot", bootstrap = 50,
           iseed = 89574)
out_boot <- stdmod_lavaan(fit = fit,
                          x = "iv",
```

```
                        y = "med",
                        w = "mod",
                        x_w = "iv:mod",
                        boot_ci = TRUE)
out_boot
```

`std_selected` *Standardize Variables in a Regression Model*

Description

Standardize, mean center, or scale by standard deviation selected variables in a regression model and refit the model

Usage

```
std_selected(lm_out, to_scale = NULL, to_center = NULL, to_standardize = NULL)

std_selected_boot(
  lm_out,
  to_scale = NULL,
  to_center = NULL,
  to_standardize = NULL,
  conf = 0.95,
  nboot = 100,
  boot_args = NULL,
  save_boot_est = TRUE,
  full_output = FALSE,
  do_boot = TRUE
)
```

Arguments

`lm_out`	The output from `lm()`.
`to_scale`	The terms to be rescaled by standard deviation, specified by a formula as in `lm()`. For example, if the terms to be scaled are `x1` and `x3`, use `~ x1 + x3`. No need to specify the interaction term. To scale the outcome variable, list it on the *right hand side* as a predictor. Specify only the original variables. If `NULL`, then no terms will be rescaled by their standard deviations. Variables that are not numeric will be ignored. Default is `NULL`.
`to_center`	The terms to be mean centered, specified by a formula as in `lm()`. For example, if the terms to be centered is `x1` and `x3`, use `~ x1 + x3`. No need to specify the interaction term. To center the outcome variable, list it on the *right hand side* as a predictor. Specify only the original variables. If `NULL`, then no term will be centered. Default is `NULL`.

to_standardize	The terms to be standardized, specified by a formula as in lm(). For example, if the terms to be standardized is x1 and x3, use ~ x1 + x3. No need to specify the interaction term. To standardize the outcome variable, list it on the *right hand side* as a predictor. Specify only the original variables. This is a shortcut to to_center and to_scale. Listing a variable in to_standardize is equivalent to listing this variable in both to_center and to_scale. Default is NULL.
conf	The level of confidence for the confidence interval. Default is .95.
nboot	The number of bootstrap samples. Default is 100.
boot_args	A named list of arguments to be passed to boot::boot(). Default is NULL.
save_boot_est	If TRUE, the default, the bootstrap estimates will be saved in the element boot_est of the output.
full_output	Whether the full output from boot::boot() is returned. Default is FALSE. If TRUE, the full output from boot::boot() will be saved in the element boot_out of the output.
do_boot	Whether bootstrapping confidence intervals will be formed. Default is TRUE. If FALSE, all arguments related to bootstrapping will be ignored.

Details

std_selected() was originally developed to compute the standardized moderation effect and the standardized coefficients for other predictors given an lm() output (Cheung, Cheung, Lau, Hui, & Vong, 2022). It has been extended such that users can specify which variables in a regression model are to be mean-centered and/or rescaled by their standard deviations. If the model has one or more interaction terms, they will be formed after the transformation, yielding the correct standardized solution for a moderated regression model. Moreover, categorical predictors will be automatically skipped in mean-centering and rescaling.

Standardization is conducted when a variable is mean-centered and then rescaled by its standard deviation. Therefore, if the goal is to get the standardized solution of a moderated regression, users just instruct the function to standardize all non-categorical variables in the regression model.

std_selected_boot() is a wrapper of std_selected(). It calls std_selected() once for each bootstrap sample, and then computes the nonparametric bootstrap percentile confidence intervals (Cheung, Cheung, Lau, Hui, & Vong, 2022).

If do_boot is FALSE, then the object it returns is identical to that by std_selected().

This function intentionally does not have an argument for setting the seed for random number. Users are recommended to set the seed, e.g., using set.seed() before calling it, to ensure reproducibility.

Value

The updated lm() output, with the class std_selected added. It will be treated as a usual lm() object by most functions. These are the major additional element in the list:

- scaled_terms: If not NULL, a character vector of the variables scaled.
- centered_terms: If not NULL, a character vector of the variables mean-centered.
- scaled_by: A numeric vector of the scaling factors for all the variables in the model. The value is 1 for terms not scaled.

- centered_by: A numeric vector of the numbers used for centering for all the variables in the model. The value is 0 for terms not centered.
- std_selected_call: The original call.
- lm_out_call: The call in lm_out.

Like std_selected(), std_selected_boot() returns the updated lm() output, with the class std_selected added. The output of std_selected_boot() contain these additional elements in the list:

- boot_ci: A data frame of the bootstrap confidence intervals of the regression coefficient.
- nboot: The number of bootstrap samples requested.
- conf: The level of confidence, in proportion.
- boot_est: A matrix of the bootstrap estimates of the regression coefficients. The number of rows equal to nboot, and the number of columns equal to the number of terms in the regression model.
- std_selected_boot_call: The call to std_selected_boot().
- boot_out: If available, the original output from boot::boot().

Functions

- std_selected(): The base function to center or scale selected variables in a regression model
- std_selected_boot(): A wrapper of std_selected() that forms nonparametric bootstrap confidence intervals.

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500
head(dat)

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)

# Mean center mod only
lm_cw <- std_selected(lm_raw, to_center = ~ mod)
summary(lm_cw)
```

```
# Mean center mod and iv
lm_cwx <- std_selected(lm_raw, to_center = ~ mod + iv)
summary(lm_cwx)

# Standardize both mod and iv
lm_stdwx <- std_selected(lm_raw, to_scale = ~ mod + iv,
                                 to_center = ~ mod + iv)
summary(lm_stdwx)

# Standardize all variables except for categorical variables.
# Interaction terms are formed after standardization.
lm_std <- std_selected(lm_raw, to_scale = ~ .,
                               to_center = ~ .)
summary(lm_std)

# Use to_standardize as a shortcut
lm_stdwx2 <- std_selected(lm_raw, to_standardize = ~ mod + iv)
# The results are the same
coef(lm_stdwx)
coef(lm_stdwx2)
all.equal(coef(lm_stdwx), coef(lm_stdwx2))



dat <- test_x_1_w_1_v_1_cat1_n_500
head(dat)

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)
# Standardize all variables as in std_selected above, and compute the
# nonparametric bootstrapping percentile confidence intervals.
set.seed(87053)
lm_std_boot <- std_selected_boot(lm_raw,
                                 to_scale = ~ .,
                                 to_center = ~ .,
                                 conf = .95,
                                 nboot = 100)
# In real analysis, nboot should be at least 2000.
summary(lm_std_boot)

# Use to_standardize as a shortcut
set.seed(87053)
lm_std_boot2 <- std_selected_boot(lm_raw,
                                  to_standardize = ~ .,
                                  conf = .95,
                                  nboot = 100)
# The results are the same
confint(lm_std_boot)
confint(lm_std_boot2)
all.equal(confint(lm_std_boot), confint(lm_std_boot2))
```

summary.std_selected *Summary Method for a 'std_selected' Class Object*

Description

Summarize the results of std_selected() or std_selected_boot().

Usage

```
## S3 method for class 'std_selected'
summary(object, ...)
```

Arguments

object	The output of std_selected() or std_selected_boot().
...	Additional arguments. Ignored by this function.

Value

An object of class summary.std_selected, with bootstrap confidence intervals added if present in the object. The object is a list. Its main element coefficients is similar to the coefficient table in the summary() printout of lm(). This object is for printing summary information of the results from std_selected() or std_selected_boot().

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)

# Standardize all variables except for categorical variables.
# Interaction terms are formed after standardization.
lm_std <- std_selected(lm_raw, to_scale = ~ .,
                               to_center = ~ .)
summary(lm_std)

# With bootstrapping
# nboot = 100 just for illustration. nboot >= 2000 should be used in read
# research.
lm_std_boot <- std_selected_boot(lm_raw, to_scale = ~ .,
                                         to_center = ~ .,
```

```
                                        nboot = 100)
summary(lm_std_boot)
```

`test_mod1` *Sample Dataset: A Path Model With A Moderator*

Description

For testing. Generated from the following model.

```
mod <-
"
med ~ iv + mod + iv:mod + cov1
dv ~ med + cov2
"
```

Usage

```
test_mod1
```

Format

A data frame with 300 rows and 6 variables:

dv Dependent variable, continuous

iv Independent variable, continuous

med Mediator, continuous

mod Moderator, continuous

cov1 Covariate, continuous

cov2 Covariate, continuous

`test_mod2` *Sample Dataset: A Path Model With A Moderator*

Description

For testing. Generated from the following model.

```
mod <-
"
med ~ iv + cov1
dv ~ med + mod + med:mod + cov2
"
```

Usage

```
test_mod2
```

Format

A data frame with 300 rows and 6 variables:

dv Dependent variable, continuous

iv Independent variable, continuous

med Mediator, continuous

mod Moderator, continuous

cov1 Covariate, continuous

cov2 Covariate, continuous

---

test_mod3_miss *Sample Dataset: A Path Model With A Moderator*

Description

For testing the handling of warnings in `stdmod_lavaan()`. Generated from the following model. dv has about 88% missing. A warning on missing data will be raised in some bootstrap samples.

```
mod <-
"
med ~ iv + mod + iv:mod + cov1
dv ~ med + cov2
"
```

Usage

```
test_mod3_miss
```

Format

A data frame with 500 rows and 6 variables:

dv Dependent variable, continuous

iv Independent variable, continuous

med Mediator, continuous

mod Moderator, continuous

cov1 Covariate, continuous

cov2 Covariate, continuous

test_x_1_w_1_v_1_cat1_n_500

Sample Dataset: One IV, One Moderator, Two Covariates

Description

A covariate (cat1) is categorical. For testing.

Usage

```
test_x_1_w_1_v_1_cat1_n_500
```

Format

A data frame with 500 rows and five variables:

dv Dependent variable, continuous

iv Independent variable, continuous

mod Moderator variable, continuous

v1 Covariate, continuous

cat1 Covariate, categorical (string) with three values: "gp1", "gp2", and "gp3"

test_x_1_w_1_v_1_cat1_xw_cov_n_500

Sample Dataset: One IV, One Moderator, Two Covariates

Description

The independent variable and the moderator are associated. For demonstrating the use of tumble graph.

Usage

```
test_x_1_w_1_v_1_cat1_xw_cov_n_500
```

Format

A data frame with 500 rows and 5 variables:

dv Dependent variable, continuous

iv Independent variable, continuous

mod Moderator variable, continuous

v1 Covariate, continuous

cat1 Covariate, categorical (string) with three values, "gp1", "gp2", and "gp3"

test_x_1_w_1_v_1_cat1_xw_cov_wcat3_n_500

Sample Dataset: One IV, One 3-Category Moderator, Two Covariates

Description

The independent variable and the categorical moderator are associated. For demonstrating the use of tumble graph.

Usage

```
test_x_1_w_1_v_1_cat1_xw_cov_wcat3_n_500
```

Format

A data frame with 500 rows and 5 variables:

dv Dependent variable, continuous

iv Independent variable, continuous

mod Moderator variable, categorical (string) with three categories, "City Alpha", "City Gamma", and "City Beta"

v1 Covariate, continuous

cat1 Covariate, categorical (string) with three values, "gp1", "gp2", and "gp3"

test_x_1_w_1_v_2_n_500

Sample Dataset: One IV, One Moderator, Two Covariates

Description

All variables are continuous. For testing.

Usage

```
test_x_1_w_1_v_2_n_500
```

Format

A data frame with 500 rows and five variables:

dv Dependent variable, continuous

iv Independent variable, continuous

mod Moderator variable, continuous

v1 Covariate, continuous

v2 Covariate, continuous

update.std_selected *The 'update' Method for a 'std_selected' Class Object*

Description

This should be used only to update the call to `lm()`, not to the call to `std_selected()` or `std_selected_boot()`.

Usage

```
## S3 method for class 'std_selected'
update(object, formula., ..., evaluate = TRUE)
```

Arguments

`object`	The output of the class `std_selected()`.
`formula.`	Changes to the formula.
`...`	Optional arguments to be changed.
`evaluate`	Whether the call will be evaluated.

Details

Although supported, it is not recommended to update an analysis processed by `std_selected()` or `std_selected_boot()`. It is recommended to call `lm()` again and pass the output to `std_selected()` or `std_selected_boot()`.

Value

If `evaluate = TRUE`, it returns the updated fitted object, otherwise, the updated call.

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500
head(dat)

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)
summary(lm_raw)

# Standardize all variables except for categorical variables.
# Interaction terms are formed after standardization.
lm_std <- std_selected(lm_raw, to_scale = ~ .,
                               to_center = ~ .)
```

```
summary(lm_std)

# Update the model
lm_std2 <- update(lm_std, . ~ . - v1)
summary(lm_std2)
```

`vcov.std_selected` *The 'vcov' Method for a 'std_selected' Class Object*

Description

Compute the variance-covariance matrix of estimates in the output of `std_selected()` or `std_selected_boot()`.

Usage

```
## S3 method for class 'std_selected'
vcov(object, type, ...)
```

Arguments

`object`	The output of `std_selected()` or `std_selected_boot()`.
`type`	The type of variance-covariance matrix. If set to `"lm"`, returns the results of the `stats::vcov()` method for the output of `lm()`. If set to `"boot"`, the variance-covariance matrix of the bootstrap estimates is returned. Default depends on `object`. If bootstrap estimates were stored, then the default is `"boot"`. Otherwise, the default is `"lm"`.
`...`	Arguments to be passed to `stats::vcov()`.

Details

If bootstrapping was used to form the confidence intervals, users can request the variance-covariance matrix of the bootstrap estimates.

Value

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

Author(s)

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

Examples

```
# Load a sample data set

dat <- test_x_1_w_1_v_1_cat1_n_500
head(dat)

# Do a moderated regression by lm
lm_raw <- lm(dv ~ iv*mod + v1 + cat1, dat)

# Standardize all variables except for categorical variables.
# Interaction terms are formed after standardization.
lm_std <- std_selected(lm_raw, to_scale = ~ .,
                               to_center = ~ .)

# VCOV of lm output
vcov(lm_std)

# Standardize all variables as in std_selected above, and compute the
# nonparametric bootstrapping percentile confidence intervals.
lm_std_boot <- std_selected_boot(lm_raw,
                                 to_scale = ~ .,
                                 to_center = ~ .,
                                 conf = .95,
                                 nboot = 100)
# In real analysis, nboot should be at least 2000.

# VCOV of bootstrap estimates, default when bootstrap was conducted
vcov(lm_std_boot)

# For OLS VCOV
vcov(lm_std_boot, type = "lm")
```

Index