Package 'interactions'

May 8, 2026

Type Package

Title Comprehensive, User-Friendly Toolkit for Probing Interactions

Version 1.2.0

Description A suite of functions for conducting and interpreting analysis of statistical interaction in regression models that was formerly part of the 'jtools' package. Functionality includes visualization of two- and three-way interactions among continuous and/or categorical variables as well as calculation of ``simple slopes'' and Johnson-Neyman intervals (see e.g., Bauer & Curran, 2005 <doi:10.1207/s15327906mbr4003_5>). These capabilities are implemented for generalized linear models in addition to the standard linear regression context.

URL https://interactions.jacob-long.com

BugReports https://github.com/jacob-long/interactions/issues

License MIT + file LICENSE

Encoding UTF-8

Imports ggplot2 (>= 3.4.0), cli, generics, broom, jtools (>= 2.0.3), rlang (>= 0.3.0), tibble

Suggests cowplot, broom.mixed, glue, huxtable (>= 3.0.0), lme4, margins, sandwich, survey, knitr, rmarkdown, testthat, vdiffr

Enhances brms, rstanarm

VignetteBuilder knitr

RoxygenNote 7.3.2.9000

NeedsCompilation no

Author Jacob A. Long [aut, cre] (ORCID: <https://orcid.org/0000-0002-1582-6214>)

Maintainer Jacob A. Long <jacob.long@sc.edu>

Repository CRAN

Date/Publication 2024-07-29 21:40:06 UTC

Contents

`as_huxtable.sim_margins` *Create tabular output for simple margins analysis*

Description

This function converts a `sim_margins` object into a `huxtable` object, making it suitable for use in external documents.

Usage

```
as_huxtable.sim_margins(
  x,
  format = "{estimate} ({std.error})",
  sig.levels = c(`***` = 0.001, `**` = 0.01, `*` = 0.05, `#` = 0.1),
  digits = getOption("jtools-digits", 2),
  conf.level = 0.95,
  intercept = attr(x, "cond.int"),
  int.format = format,
  ...
)
```

Arguments

`x`	A `sim_margins` object.
`format`	The method for sharing the slope and associated uncertainty. Default is `"{estimate} ({std.error})"`. See the instructions for the `error_format` argument of `jtools::export_summs()` for more on your options.
`sig.levels`	A named vector in which the values are potential p value thresholds and the names are significance markers (e.g., "*") for when p values are below the threshold. Default is `c(`***` = .001, `**` = .01, `*` = .05, `#` = .1)`.

digits	How many digits should the outputted table round to? Default is 2.
conf.level	How wide the confidence interval should be, if it is used. .95 (95% interval) is the default.
intercept	Should conditional intercepts be included? Default is whatever the cond.int argument to x was.
int.format	If conditional intercepts were requested, how should they be formatted? Default is the same as format.
...	Ignored.

Details

For more on what you can do with a huxtable, see **huxtable**.

`as_huxtable.sim_slopes` *Create tabular output for simple slopes analysis*

Description

This function converts a sim_slopes object into a huxtable object, making it suitable for use in external documents.

Usage

```
as_huxtable.sim_slopes(
  x,
  format = "{estimate} ({std.error})",
  sig.levels = c(`***` = 0.001, `**` = 0.01, `*` = 0.05, `#` = 0.1),
  digits = getOption("jtools-digits", 2),
  conf.level = 0.95,
  intercept = attr(x, "cond.int"),
  int.format = format,
  ...
)
```

Arguments

x	The sim_slopes() object.
format	The method for sharing the slope and associated uncertainty. Default is "{estimate} ({std.error})". See the instructions for the error_format argument of jtools::export_summs() for more on your options.
sig.levels	A named vector in which the values are potential p value thresholds and the names are significance markers (e.g., "*") for when p values are below the threshold. Default is c(`***` = .001, `**` = .01, `*` = .05, `#` = .1).
digits	How many digits should the outputted table round to? Default is 2.

conf.level	How wide the confidence interval should be, if it is used. .95 (95% interval) is the default.
intercept	Should conditional intercepts be included? Default is whatever the cond.int argument to x was.
int.format	If conditional intercepts were requested, how should they be formatted? Default is the same as format.
...	Ignored.

Details

For more on what you can do with a huxtable, see **huxtable**.

---

cat_plot *Plot interaction effects between categorical predictors.*

Description

cat_plot is a complementary function to interact_plot() that is designed for plotting interactions when both predictor and moderator(s) are categorical (or, in R terms, factors).

Usage

```
cat_plot(
  model,
  pred,
  modx = NULL,
  mod2 = NULL,
  data = NULL,
  geom = c("point", "line", "bar"),
  pred.values = NULL,
  modx.values = NULL,
  mod2.values = NULL,
  interval = TRUE,
  plot.points = FALSE,
  point.shape = FALSE,
  vary.lty = FALSE,
  centered = "all",
  int.type = c("confidence", "prediction"),
  int.width = 0.95,
  line.thickness = 1.1,
  point.size = 1.5,
  pred.point.size = 3.5,
  jitter = 0.1,
  geom.alpha = NULL,
  dodge.width = NULL,
  errorbar.width = NULL,
```

```
  interval.geom = c("errorbar", "linerange"),
  outcome.scale = "response",
  robust = FALSE,
  cluster = NULL,
  vcov = NULL,
  pred.labels = NULL,
  modx.labels = NULL,
  mod2.labels = NULL,
  set.offset = 1,
  x.label = NULL,
  y.label = NULL,
  main.title = NULL,
  legend.main = NULL,
  colors = NULL,
  partial.residuals = FALSE,
  point.alpha = 0.6,
  color.class = NULL,
  at = NULL,
  ...
)
```

Arguments

model	A regression model. The function is tested with lm, glm, svyglm, merMod, rq, brmsfit, stanreg models. Models from other classes may work as well but are not officially supported. The model should include the interaction of interest.
pred	A categorical predictor variable that will appear on the x-axis. Note that it is evaluated using rlang, so programmers can use the !! syntax to pass variables instead of the verbatim names.
modx	A categorical moderator variable.
mod2	For three-way interactions, the second categorical moderator.
data	Optional, default is NULL. You may provide the data used to fit the model. This can be a better way to get mean values for centering and can be crucial for models with variable transformations in the formula (e.g., log(x)) or polynomial terms (e.g., poly(x, 2)). You will see a warning if the function detects problems that would likely be solved by providing the data with this argument and the function will attempt to retrieve the original data from the global environment.
geom	What type of plot should this be? There are several options here since the best way to visualize categorical interactions varies by context. Here are the options: • "point": The default. Simply plot the point estimates. You may want to use point.shape = TRUE with this and you should also consider interval = TRUE to visualize uncertainty. • "line": This connects observations across levels of the pred variable with a line. This is a good option when the pred variable is ordinal (ordered). You may still consider point.shape = TRUE and interval = TRUE is still a good idea.

- "bar": A bar chart. Some call this a "dynamite plot." Many applied researchers advise against this type of plot because it does not represent the distribution of the observed data or the uncertainty of the predictions very well. It is best to at least use the interval = TRUE argument with this geom.

pred.values — Which values of the predictor should be included in the plot? By default, all levels are included.

modx.values — For which values of the moderator should lines be plotted? There are two basic options:

- A vector of values (e.g., c(1, 2, 3))
- A single argument asking to calculate a set of values. See details below.

Default is NULL. If NULL (or mean-plus-minus), then the customary +/- 1 standard deviation from the mean as well as the mean itself are used for continuous moderators. If "plus-minus", plots lines when the moderator is at +/- 1 standard deviation without the mean. You may also choose "terciles" to split the data into equally-sized groups and choose the point at the mean of each of those groups.

If the moderator is a factor variable and modx.values is NULL, each level of the factor is included. You may specify any subset of the factor levels (e.g., c("Level 1", "Level 3")) as long as there is more than 1. The levels will be plotted in the order you provide them, so this can be used to reorder levels as well.

mod2.values — For which values of the second moderator should the plot be facetted by? That is, there will be a separate plot for each level of this moderator. Defaults are the same as modx.values.

interval — Logical. If TRUE, plots confidence/prediction intervals.

plot.points — Logical. If TRUE, plots the actual data points as a scatterplot on top of the interaction lines. Note that if geom = "bar", this will cause the bars to become transparent so you can see the points.

point.shape — For plotted points—either of observed data or predicted values with the "point" or "line" geoms—should the shape of the points vary by the values of the factor? This is especially useful if you aim to be black and white printing- or colorblind-friendly.

vary.lty — Should the resulting plot have different shapes for each line in addition to colors? Defaults to TRUE.

centered — A vector of quoted variable names that are to be mean-centered. If "all", all non-focal predictors are centered. You may instead pass a character vector of variables to center. User can also use "none" to base all predictions on variables set at 0. The response variable, pred, modx, and mod2 variables are never centered.

int.type — Type of interval to plot. Options are "confidence" or "prediction". Default is confidence interval.

int.width — How large should the interval be, relative to the standard error? The default, .95, corresponds to roughly 1.96 standard errors and a .05 alpha level for values outside the range. In other words, for a confidence interval, .95 is analogous to a 95% confidence interval.

`line.thickness` How thick should the plotted lines be? Default is 1.

`point.size` What size should be used for observed data when `plot.points` is TRUE? Default is 1.5.

`pred.point.size`
If TRUE and `geom` is `"point"` or `"line"`, sets the size of the predicted points. Default is 3.5. Note the distinction from `point.size`, which refers to the observed data points.

`jitter` How much should `plot.points` observed values be "jittered" via `ggplot2::position_jitter()`? When there are many points near each other, jittering moves them a small amount to keep them from totally overlapping. In some cases, though, it can add confusion since it may make points appear to be outside the boundaries of observed values or cause other visual issues. Default is 0.1, but increase as needed if your points are overlapping too much or set to 0 for no jitter. If the argument is a vector with two values, then the first is assumed to be the jitter for width and the second for the height.

`geom.alpha` What should the alpha aesthetic be for the plotted lines/bars? Default is NULL, which means it is set depending on the value of `geom` and `plot.points`.

`dodge.width` What should the `width` argument to `ggplot2::position_dodge()` be? Default is NULL, which means it is set depending on the value of `geom`.

`errorbar.width` How wide should the error bars be? Default is NULL, meaning it is set depending on the value `geom`. Ignored if `interval` is FALSE.

`interval.geom` For categorical by categorical interactions. One of "errorbar" or "linerange". If the former, `ggplot2::geom_errorbar()` is used. If the latter, `ggplot2::geom_linerange()` is used.

`outcome.scale` For nonlinear models (i.e., GLMs), should the outcome variable be plotted on the link scale (e.g., log odds for logit models) or the original scale (e.g., predicted probabilities for logit models)? The default is `"response"`, which is the original scale. For the link scale, which will show straight lines rather than curves, use `"link"`.

`robust` Should robust standard errors be used to find confidence intervals for supported models? Default is FALSE, but you should specify the type of sandwich standard errors if you'd like to use them (i.e., `"HC0"`, `"HC1"`, and so on). If `TRUE`, defaults to `"HC3"` standard errors.

`cluster` For clustered standard errors, provide the column name of the cluster variable in the input data frame (as a string). Alternately, provide a vector of clusters.

`vcov` Optional. You may supply the variance-covariance matrix of the coefficients yourself. This is useful if you are using some method for robust standard error calculation not supported by the **sandwich** package.

`pred.labels` A character vector of equal length to the number of factor levels of the predictor (or number specified in `predvals`). If `NULL`, the default, the factor labels are used.

`modx.labels` A character vector of labels for each level of the moderator values, provided in the same order as the `modx.values` argument. If `NULL`, the values themselves are used as labels unless `modx,values` is also `NULL`. In that case, "+1 SD" and "-1 SD" are used.

mod2.labels	A character vector of labels for each level of the 2nd moderator values, provided in the same order as the mod2.values argument. If NULL, the values themselves are used as labels unless mod2.values is also NULL. In that case, "+1 SD" and "-1 SD" are used.
set.offset	For models with an offset (e.g., Poisson models), sets an offset for the predicted values. All predicted values will have the same offset. By default, this is set to 1, which makes the predicted values a proportion. See details for more about offset support.
x.label	A character object specifying the desired x-axis label. If NULL, the variable name is used.
y.label	A character object specifying the desired x-axis label. If NULL, the variable name is used.
main.title	A character object that will be used as an overall title for the plot. If NULL, no main title is used.
legend.main	A character object that will be used as the title that appears above the legend. If NULL, the name of the moderating variable is used.
colors	Any palette argument accepted by jtools::get_colors(). Default is "CUD Bright" for factor moderators and "blue" for continuous moderators. You may also simply supply a vector of colors accepted by ggplot2 and of equal length to the number of moderator levels.
partial.residuals	Instead of plotting the observed data, you may plot the partial residuals (controlling for the effects of variables besides pred).
point.alpha	What should the alpha aesthetic for plotted points of observed data be? Default is 0.6, and it can range from 0 (transparent) to 1 (opaque).
color.class	Deprecated. Now known as colors.
at	If you want to manually set the values of other variables in the model, do so by providing a named list where the names are the variables and the list values are vectors of the values. This can be useful especially when you are exploring interactions or other conditional predictions.
...	extra arguments passed to make_predictions

Details

This function provides a means for plotting conditional effects for the purpose of exploring interactions in the context of regression. You must have the package ggplot2 installed to benefit from these plotting functions.

The function is designed for two and three-way interactions. For additional terms, the effects package may be better suited to the task.

This function supports nonlinear and generalized linear models and by default will plot them on their original scale (outcome.scale = "response").

While mixed effects models from lme4 are supported, only the fixed effects are plotted. lme4 does not provide confidence intervals, so they are not supported with this function either.

Note: to use transformed predictors, e.g., log(variable), provide only the variable name to pred, modx, or mod2 and supply the original data separately to the data argument.

Info about offsets:

Offsets are partially supported by this function with important limitations. First of all, only a single offset per model is supported. Second, it is best in general to specify offsets with the offset argument of the model fitting function rather than in the formula. You are much more likely to have success if you provide the data used to fit the model with the `data` argument.

Value

The functions returns a `ggplot` object, which can be treated like a user-created plot and expanded upon as such.

Examples

```
library(ggplot2)
fit <- lm(price ~ cut * color, data = diamonds)
cat_plot(fit, pred = color, modx = cut, interval = TRUE)

# 3-way interaction

## Will first create a couple dichotomous factors to ensure full rank
mpg2 <- mpg
mpg2$auto <- "auto"
mpg2$auto[mpg2$trans %in% c("manual(m5)", "manual(m6)")] <- "manual"
mpg2$auto <- factor(mpg2$auto)
mpg2$fwd <- "2wd"
mpg2$fwd[mpg2$drv == "4"] <- "4wd"
mpg2$fwd <- factor(mpg2$fwd)
## Drop the two cars with 5 cylinders (rest are 4, 6, or 8)
mpg2 <- mpg2[mpg2$cyl != "5",]
mpg2$cyl <- factor(mpg2$cyl)
## Fit the model
fit3 <- lm(cty ~ cyl * fwd * auto, data = mpg2)

# The line geom looks good for an ordered factor predictor
cat_plot(fit3, pred = cyl, modx = fwd, mod2 = auto, geom = "line",
 interval = TRUE)
```

---

interact_plot *Plot interaction effects in regression models*

Description

`interact_plot` plots regression lines at user-specified levels of a moderator variable to explore interactions. The plotting is done with `ggplot2` rather than base graphics, which some similar functions use.

Usage

```
interact_plot(
  model,
  pred,
  modx,
  modx.values = NULL,
  mod2 = NULL,
  mod2.values = NULL,
  centered = "all",
  data = NULL,
  at = NULL,
  plot.points = FALSE,
  interval = FALSE,
  int.type = c("confidence", "prediction"),
  int.width = 0.95,
  outcome.scale = "response",
  linearity.check = FALSE,
  facet.modx = FALSE,
  robust = FALSE,
  cluster = NULL,
  vcov = NULL,
  set.offset = 1,
  x.label = NULL,
  y.label = NULL,
  pred.labels = NULL,
  modx.labels = NULL,
  mod2.labels = NULL,
  main.title = NULL,
  legend.main = NULL,
  colors = NULL,
  line.thickness = 1,
  vary.lty = TRUE,
  point.size = 1.5,
  point.shape = FALSE,
  jitter = 0,
  rug = FALSE,
  rug.sides = "b",
  partial.residuals = FALSE,
  point.alpha = 0.6,
  color.class = NULL,
  ...
)
```

Arguments

model	A regression model. The function is tested with `lm`, `glm`, `svyglm`, `merMod`, `rq`, `brmsfit`, `stanreg` models. Models from other classes may work as well but are not officially supported. The model should include the interaction of interest.

`pred` The name of the predictor variable involved in the interaction. This can be a bare name or string. Note that it is evaluated using `rlang`, so programmers can use the `!!` syntax to pass variables instead of the verbatim names.

`modx` The name of the moderator variable involved in the interaction. This can be a bare name or string. The same `rlang` proviso applies as with `pred`.

`modx.values` For which values of the moderator should lines be plotted? There are two basic options:

- A vector of values (e.g., `c(1, 2, 3)`)
- A single argument asking to calculate a set of values. See details below.

Default is `NULL`. If `NULL` (or `mean-plus-minus`), then the customary +/- 1 standard deviation from the mean as well as the mean itself are used for continuous moderators. If `"plus-minus"`, plots lines when the moderator is at +/- 1 standard deviation without the mean. You may also choose `"terciles"` to split the data into equally-sized groups and choose the point at the mean of each of those groups.

If the moderator is a factor variable and `modx.values` is `NULL`, each level of the factor is included. You may specify any subset of the factor levels (e.g., `c("Level 1", "Level 3")`) as long as there is more than 1. The levels will be plotted in the order you provide them, so this can be used to reorder levels as well.

`mod2` Optional. The name of the second moderator variable involved in the interaction. This can be a bare name or string. The same `rlang` proviso applies as with `pred`.

`mod2.values` For which values of the second moderator should the plot be facetted by? That is, there will be a separate plot for each level of this moderator. Defaults are the same as `modx.values`.

`centered` A vector of quoted variable names that are to be mean-centered. If `"all"`, all non-focal predictors are centered. You may instead pass a character vector of variables to center. User can also use "none" to base all predictions on variables set at 0. The response variable, `pred`, `modx`, and `mod2` variables are never centered.

`data` Optional, default is NULL. You may provide the data used to fit the model. This can be a better way to get mean values for centering and can be crucial for models with variable transformations in the formula (e.g., `log(x)`) or polynomial terms (e.g., `poly(x, 2)`). You will see a warning if the function detects problems that would likely be solved by providing the data with this argument and the function will attempt to retrieve the original data from the global environment.

`at` If you want to manually set the values of other variables in the model, do so by providing a named list where the names are the variables and the list values are vectors of the values. This can be useful especially when you are exploring interactions or other conditional predictions.

`plot.points` Logical. If `TRUE`, plots the actual data points as a scatterplot on top of the interaction lines. The color of the dots will be based on their moderator value.

`interval` Logical. If `TRUE`, plots confidence/prediction intervals around the line using `geom_ribbon`.

`int.type` Type of interval to plot. Options are "confidence" or "prediction". Default is confidence interval.

`int.width` How large should the interval be, relative to the standard error? The default, .95, corresponds to roughly 1.96 standard errors and a .05 alpha level for values outside the range. In other words, for a confidence interval, .95 is analogous to a 95% confidence interval.

`outcome.scale` For nonlinear models (i.e., GLMs), should the outcome variable be plotted on the link scale (e.g., log odds for logit models) or the original scale (e.g., predicted probabilities for logit models)? The default is `"response"`, which is the original scale. For the link scale, which will show straight lines rather than curves, use `"link"`.

`linearity.check` For two-way interactions only. If `TRUE`, plots a pane for each level of the moderator and superimposes a loess smoothed line (in gray) over the plot. This enables you to see if the effect is linear through the span of the moderator. See Hainmueller et al. (2016) in the references for more details on the intuition behind this. It is recommended that you also set `plot.points = TRUE` and use `modx.values = "terciles"` with this option.

`facet.modx` Create separate panels for each level of the moderator? Default is FALSE, except when `linearity.check` is TRUE.

`robust` Should robust standard errors be used to find confidence intervals for supported models? Default is FALSE, but you should specify the type of sandwich standard errors if you'd like to use them (i.e., `"HC0"`, `"HC1"`, and so on). If `TRUE`, defaults to `"HC3"` standard errors.

`cluster` For clustered standard errors, provide the column name of the cluster variable in the input data frame (as a string). Alternately, provide a vector of clusters.

`vcov` Optional. You may supply the variance-covariance matrix of the coefficients yourself. This is useful if you are using some method for robust standard error calculation not supported by the **sandwich** package.

`set.offset` For models with an offset (e.g., Poisson models), sets an offset for the predicted values. All predicted values will have the same offset. By default, this is set to 1, which makes the predicted values a proportion. See details for more about offset support.

`x.label` A character object specifying the desired x-axis label. If `NULL`, the variable name is used.

`y.label` A character object specifying the desired x-axis label. If `NULL`, the variable name is used.

`pred.labels` A character vector of 2 labels for the predictor if it is a 2-level factor or a continuous variable with only 2 values. If `NULL`, the default, the factor labels are used.

`modx.labels` A character vector of labels for each level of the moderator values, provided in the same order as the `modx.values` argument. If `NULL`, the values themselves are used as labels unless `modx,values` is also `NULL`. In that case, "+1 SD" and "-1 SD" are used.

`mod2.labels` A character vector of labels for each level of the 2nd moderator values, provided in the same order as the `mod2.values` argument. If `NULL`, the values themselves are used as labels unless `mod2.values` is also `NULL`. In that case, "+1 SD" and "-1 SD" are used.

`main.title` A character object that will be used as an overall title for the plot. If `NULL`, no main title is used.

`legend.main` A character object that will be used as the title that appears above the legend. If `NULL`, the name of the moderating variable is used.

`colors` See `jtools_colors` for details on the types of arguments accepted. Default is "CUD Bright" for factor moderators, "Blues" for +/- SD and user-specified `modx.values` values.

`line.thickness` How thick should the plotted lines be? Default is 1.

`vary.lty` Should the resulting plot have different shapes for each line in addition to colors? Defaults to `TRUE`.

`point.size` What size should be used for observed data when `plot.points` is TRUE? Default is 1.5.

`point.shape` For plotted points—either of observed data or predicted values with the "point" or "line" geoms—should the shape of the points vary by the values of the factor? This is especially useful if you aim to be black and white printing- or colorblind-friendly.

`jitter` How much should `plot.points` observed values be "jittered" via `ggplot2::position_jitter()`? When there are many points near each other, jittering moves them a small amount to keep them from totally overlapping. In some cases, though, it can add confusion since it may make points appear to be outside the boundaries of observed values or cause other visual issues. Default is 0, but try various small values (e.g., 0.1) and increase as needed if your points are overlapping too much. If the argument is a vector with two values, then the first is assumed to be the jitter for width and the second for the height.

`rug` Show a rug plot in the margins? This uses `ggplot2::geom_rug()` to show the distribution of the predictor (top/bottom) and/or response variable (left/right) in the original data. Default is FALSE.

`rug.sides` On which sides should rug plots appear? Default is "b", meaning bottom. "t" and/or "b" show the distribution of the predictor while "l" and/or "r" show the distribution of the response. "bl" is a good option to show both the predictor and response.

`partial.residuals` Instead of plotting the observed data, you may plot the partial residuals (controlling for the effects of variables besides `pred`).

`point.alpha` What should the `alpha` aesthetic for plotted points of observed data be? Default is 0.6, and it can range from 0 (transparent) to 1 (opaque).

`color.class` Deprecated. Now known as `colors`.

`...` extra arguments passed to `make_predictions`

Details

This function provides a means for plotting conditional effects for the purpose of exploring interactions in regression models.

The function is designed for two and three-way interactions. For additional terms, the **effects** package may be better suited to the task.

This function supports nonlinear and generalized linear models and by default will plot them on their original scale (`outcome.scale = "response"`). To plot them on the linear scale, use "link" for `outcome.scale`.

While mixed effects models from `lme4` are supported, only the fixed effects are plotted. `lme4` does not provide confidence intervals, so they are not supported with this function either.

Note: to use transformed predictors, e.g., `log(variable)`, put its name in quotes or backticks in the argument.

Details on how observed data are split in multi-pane plots:

If you set `plot.points = TRUE` and request a multi-pane (facetted) plot either with a second moderator, `linearity.check = TRUE`, or `facet.modx = TRUE`, the observed data are split into as many groups as there are panes and plotted separately. If the moderator is a factor, then the way this happens will be very intuitive since it's obvious which values go in which pane. The rest of this section will address the case of continuous moderators.

My recommendation is that you use `modx.values = "terciles"` or `mod2.values = "terciles"` when you want to plot observed data on multi-pane plots. When you do, the data are split into three approximately equal-sized groups with the lowest third, middle third, and highest third of the data split accordingly. You can replicate this procedure using `Hmisc::cut2()` with `g = 3` from the `Hmisc` package. Sometimes, the groups will not be equal in size because the number of observations is not divisible by 3 and/or there are multiple observations with the same value at one of the cut points.

Otherwise, a more ad hoc procedure is used to split the data. Quantiles are found for each `mod2.values` or `modx.values` value. These are not the quantiles used to split the data, however, since we want the plotted lines to represent the slope at a typical value in the group. The next step, then, is to take the mean of each pair of neighboring quantiles and use these as the cut points.

For example, if the `mod2.values` are at the 25th, 50th, and 75th percentiles of the distribution of the moderator, the data will be split at the 37.5th and and 62.5th percentiles. When the variable is normally distributed, this will correspond fairly closely to using terciles.

Info about offsets:

Offsets are partially supported by this function with important limitations. First of all, only a single offset per model is supported. Second, it is best in general to specify offsets with the offset argument of the model fitting function rather than in the formula. You are much more likely to have success if you provide the data used to fit the model with the `data` argument.

Value

The functions returns a `ggplot` object, which can be treated like a user-created plot and expanded upon as such.

Author(s)

Jacob Long <jacob.long@sc.edu>

References

Bauer, D. J., & Curran, P. J. (2005). Probing interactions in fixed and multilevel regression: Inferential and graphical techniques. *Multivariate Behavioral Research*, *40*(3), 373-400. doi:10.1207/s15327906mbr4003_5

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analyses for the behavioral sciences* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Hainmueller, J., Mummolo, J., & Xu, Y. (2016). How much should we trust estimates from multiplicative interaction models? Simple tools to improve empirical practice. SSRN Electronic Journal. doi:10.2139/ssrn.2739221

See Also

`plotSlopes` from `rockchalk` performs a similar function, but with R's base graphics—this function is meant, in part, to emulate its features.

Functions from the `margins` and `sjPlot` packages may also be useful if this one isn't working for you.

`sim_slopes` performs a simple slopes analysis with a similar argument syntax to this function.

Examples

```
# Using a fitted lm model
states <- as.data.frame(state.x77)
states$HSGrad <- states$`HS Grad`
fit <- lm(Income ~ HSGrad + Murder * Illiteracy, data = states)
interact_plot(model = fit, pred = Murder, modx = Illiteracy)

# Using interval feature
fit <- lm(accel ~ mag * dist, data = attenu)
interact_plot(fit, pred = mag, modx = dist, interval = TRUE,
  int.type = "confidence", int.width = .8)

# Using second moderator
fit <- lm(Income ~ HSGrad * Murder * Illiteracy, data = states)
interact_plot(model = fit, pred = Murder, modx = Illiteracy, mod2 = HSGrad)

# With svyglm
if (requireNamespace("survey")) {
library(survey)
data(api)
dstrat <- svydesign(id = ~1, strata = ~stype, weights = ~pw,
                    data = apistrat, fpc = ~fpc)
regmodel <- svyglm(api00 ~ ell * meals, design = dstrat)
interact_plot(regmodel, pred = ell, modx = meals)
}

# With lme4
## Not run:
library(lme4)
data(VerbAgg)
mv <- glmer(r2 ~ Anger * mode + (1 | item), data = VerbAgg,
```

```
            family = binomial,
            control = glmerControl("bobyqa"))
interact_plot(mv, pred = Anger, modx = mode)

## End(Not run)
```

---

johnson_neyman *Calculate Johnson-Neyman intervals for 2-way interactions*

Description

johnson_neyman finds so-called "Johnson-Neyman" intervals for understanding where simple slopes are significant in the context of interactions in multiple linear regression.

Usage

```
johnson_neyman(
  model,
  pred,
  modx,
  vmat = NULL,
  alpha = 0.05,
  plot = TRUE,
  control.fdr = FALSE,
  line.thickness = 0.5,
  df = "residual",
  digits = getOption("jtools-digits", 2),
  critical.t = NULL,
  sig.color = "#00BFC4",
  insig.color = "#F8766D",
  mod.range = NULL,
  title = "Johnson-Neyman plot",
  y.label = NULL,
  modx.label = NULL
)
```

Arguments

model	A regression model. It is tested with lm, glm, and svyglm objects, but others may work as well. It should contain the interaction of interest. Be aware that just because the computations work, this does not necessarily mean the procedure is appropriate for the type of model you have.
pred	The predictor variable involved in the interaction.
modx	The moderator variable involved in the interaction.

vmat	Optional. You may supply the variance-covariance matrix of the coefficients yourself. This is useful if you are using robust standard errors, as you could if using the **sandwich** package.
alpha	The alpha level. By default, the standard 0.05.
plot	Should a plot of the results be printed? Default is TRUE. The ggplot2 object is returned either way.
control.fdr	Logical. Use the procedure described in Esarey & Sumner (2017) to limit the false discovery rate? Default is FALSE. See details for more on this method.
line.thickness	How thick should the predicted line be? This is passed to geom_path as the size argument, but because of the way the line is created, you cannot use geom_path to modify the output plot yourself.
df	How should the degrees of freedom be calculated for the critical test statistic? Previous versions used the large sample approximation; if alpha was .05, the critical test statistic was 1.96 regardless of sample size and model complexity. The default is now "residual", meaning the same degrees of freedom used to calculate p values for regression coefficients. You may instead choose any number or "normal", which reverts to the previous behavior. The argument is not used if control.fdr = TRUE.
digits	An integer specifying the number of digits past the decimal to report in the output. Default is 2. You can change the default number of digits for all jtools functions with options("jtools-digits" = digits) where digits is the desired number.
critical.t	If you want to provide the critical test statistic instead relying on a normal or *t* approximation, or the control.fdr calculation, you can give that value here. This allows you to use other methods for calculating it.
sig.color	Sets the color for areas of the Johnson-Neyman plot where the slope of the moderator is significant at the specified level. "black" can be a good choice for greyscale publishing.
insig.color	Sets the color for areas of the Johnson-Neyman plot where the slope of the moderator is insignificant at the specified level. "grey" can be a good choice for greyscale publishing.
mod.range	The range of values of the moderator (the x-axis) to plot. By default, this goes from one standard deviation below the observed range to one standard deviation above the observed range and the observed range is highlighted on the plot. You could instead choose to provide the actual observed minimum and maximum, in which case the range of the observed data is not highlighted in the plot. Provide the range as a vector, e.g., c(0, 10).
title	The plot title. "Johnson-Neyman plot" by default.
y.label	If you prefer to override the automatic labelling of the y axis, you can specify your own label here. The y axis represents a *slope* so it is recommended that you do not simply give the name of the predictor variable but instead make clear that it is a slope. By default, "Slope of [pred]" is used (with whatever pred is).
modx.label	If you prefer to override the automatic labelling of the x axis, you can specify your own label here. By default, the name modx is used.

Details

The interpretation of the values given by this function is important and not always immediately intuitive. For an interaction between a predictor variable and moderator variable, it is often the case that the slope of the predictor is statistically significant at only some values of the moderator. For example, perhaps the effect of your predictor is only significant when the moderator is set at some high value.

The Johnson-Neyman interval provides the two values of the moderator at which the slope of the predictor goes from non-significant to significant. Usually, the predictor's slope is only significant *outside* of the range given by the function. The output of this function will make it clear either way.

One weakness of this method of probing interactions is that it is analogous to making multiple comparisons without any adjustment to the alpha level. Esarey & Sumner (2017) proposed a method for addressing this, which is implemented in the `interactionTest` package. This function implements that procedure with modifications to the `interactionTest` code (that package is not required to use this function). If you set `control.fdr = TRUE`, an alternative *t* statistic will be calculated based on your specified alpha level and the data. This will always be a more conservative test than when `control.fdr = FALSE`. The printed output will report the calculated critical *t* statistic.

This technique is not easily ported to 3-way interaction contexts. You could, however, look at the J-N interval at two different levels of a second moderator. This does forgo a benefit of the J-N technique, which is not having to pick arbitrary points. If you want to do this, just use the `sim_slopes` function's ability to handle 3-way interactions and request Johnson-Neyman intervals for each.

Value

`bounds`	The two numbers that make up the interval.
`cbands`	A dataframe with predicted values of the predictor's slope and lower/upper bounds of confidence bands if you would like to make your own plots
`plot`	The `ggplot` object used for plotting. You can tweak the plot like you could any other from `ggplot`.

Author(s)

Jacob Long <jacob.long@sc.edu>

References

Bauer, D. J., & Curran, P. J. (2005). Probing interactions in fixed and multilevel regression: Inferential and graphical techniques. *Multivariate Behavioral Research*, *40*(3), 373-400. doi:10.1207/s15327906mbr4003_5

Esarey, J., & Sumner, J. L. (2017). Marginal effects in interaction models: Determining and controlling the false positive rate. Comparative Political Studies, 1–33. Advance online publication. doi:10.1177/0010414017730080

Johnson, P.O. & Fay, L.C. (1950). The Johnson-Neyman technique, its theory and application. *Psychometrika*, *15*, 349-367. doi:10.1007/BF02288864

See Also

Other interaction tools: probe_interaction(), sim_margins(), sim_slopes()

Examples

```
# Using a fitted lm model
states <- as.data.frame(state.x77)
states$HSGrad <- states$`HS Grad`
fit <- lm(Income ~ HSGrad + Murder*Illiteracy,
  data = states)
johnson_neyman(model = fit, pred = Murder,
  modx = Illiteracy)
```

---

plot.sim_margins *Plot coefficients from simple slopes analysis*

Description

This creates a coefficient plot to visually summarize the results of simple slopes analysis.

Usage

```
## S3 method for class 'sim_margins'
plot(x, ...)
```

Arguments

x	A sim_margins() object.
...	arguments passed to jtools::plot_coefs()

---

plot.sim_slopes *Plot coefficients from simple slopes analysis*

Description

This creates a coefficient plot to visually summarize the results of simple slopes analysis.

Usage

```
## S3 method for class 'sim_slopes'
plot(x, ...)
```

Arguments

x	A sim_slopes() object.
...	arguments passed to jtools::plot_coefs()

probe_interaction *Probe interaction effects via simple slopes and plotting*

Description

probe_interaction is a convenience function that allows users to call both sim_slopes and interact_plot with a single call.

Usage

```
probe_interaction(model, pred, modx, mod2 = NULL, ...)
```

Arguments

model	A regression model. The function is tested with lm, glm, svyglm, merMod, rq, brmsfit, stanreg models. Models from other classes may work as well but are not officially supported. The model should include the interaction of interest.
pred	The name of the predictor variable involved in the interaction. This can be a bare name or string. Note that it is evaluated using rlang, so programmers can use the !! syntax to pass variables instead of the verbatim names.
modx	The name of the moderator variable involved in the interaction. This can be a bare name or string. The same rlang proviso applies as with pred.
mod2	Optional. The name of the second moderator variable involved in the interaction. This can be a bare name or string. The same rlang proviso applies as with pred.
...	Other arguments accepted by sim_slopes and interact_plot

Details

This function simply merges the nearly-equivalent arguments needed to call both sim_slopes and interact_plot without the need for re-typing their common arguments. Note that each function is called separately and they re-fit a separate model for each level of each moderator; therefore, the runtime may be considerably longer than the original model fit. For larger models, this is worth keeping in mind.

Sometimes, you may want different parameters when doing simple slopes analysis compared to when plotting interaction effects. For instance, it is often easier to interpret the regression output when variables are standardized; but plots are often easier to understand when the variables are in their original units of measure.

probe_interaction does not support providing different arguments to each function. If that is needed, use sim_slopes and interact_plot directly.

Value

simslopes	The sim_slopes object created.
interactplot	The ggplot object created by interact_plot.

Author(s)

Jacob Long <jacob.long@sc.edu>

See Also

Other interaction tools: johnson_neyman(), sim_margins(), sim_slopes()

Examples

```
# Using a fitted model as formula input
fiti <- lm(Income ~ Frost + Murder * Illiteracy,
  data=as.data.frame(state.x77))
probe_interaction(model = fiti, pred = Murder, modx = Illiteracy,
                  modx.values = "plus-minus")
# 3-way interaction
fiti3 <- lm(Income ~ Frost * Murder * Illiteracy,
  data=as.data.frame(state.x77))
probe_interaction(model = fiti3, pred = Murder, modx = Illiteracy,
                  mod2 = Frost, mod2.values = "plus-minus")

# With svyglm
if (requireNamespace("survey")) {
library(survey)
data(api)
dstrat <- svydesign(id = ~1, strata = ~stype, weights = ~pw,
                    data = apistrat, fpc = ~fpc)
regmodel <- svyglm(api00 ~ ell * meals + sch.wide, design = dstrat)
probe_interaction(model = regmodel, pred = ell, modx = meals,
                  modx.values = "plus-minus", cond.int = TRUE)

# 3-way with survey and factor input
regmodel3 <- svyglm(api00 ~ ell * meals * sch.wide, design = dstrat)
probe_interaction(model = regmodel3, pred = ell, modx = meals,
                  mod2 = sch.wide)
# Can try different configurations of 1st vs 2nd mod
probe_interaction(model = regmodel3, pred = ell, modx = sch.wide,
                  mod2 = meals)
}
```

---

sim_margins *Perform a simple margins analysis.*

Description

sim_margins conducts a simple margins analysis for the purposes of understanding two- and three-way interaction effects in linear regression.

Usage

```
sim_margins(
  model,
  pred,
  modx,
  mod2 = NULL,
  modx.values = NULL,
  mod2.values = NULL,
  data = NULL,
  cond.int = FALSE,
  vce = c("delta", "simulation", "bootstrap", "none"),
  iterations = 1000,
  digits = getOption("jtools-digits", default = 2),
  pvals = TRUE,
  confint = FALSE,
  ci.width = 0.95,
  cluster = NULL,
  modx.labels = NULL,
  mod2.labels = NULL,
  ...
)
```

Arguments

`model`	A regression model. The function is tested with `lm`, `glm`, `svyglm`, `merMod`, `rq`, `brmsfit`, `stanreg` models. Models from other classes may work as well but are not officially supported. The model should include the interaction of interest.
`pred`	The name of the predictor variable involved in the interaction. This can be a bare name or string. Note that it is evaluated using `rlang`, so programmers can use the `!!` syntax to pass variables instead of the verbatim names.
`modx`	The name of the moderator variable involved in the interaction. This can be a bare name or string. The same `rlang` proviso applies as with `pred`.
`mod2`	Optional. The name of the second moderator variable involved in the interaction. This can be a bare name or string. The same `rlang` proviso applies as with `pred`.
`modx.values`	For which values of the moderator should lines be plotted? There are two basic options: • A vector of values (e.g., `c(1, 2, 3)`) • A single argument asking to calculate a set of values. See details below. Default is `NULL`. If `NULL` (or `mean-plus-minus`), then the customary +/- 1 standard deviation from the mean as well as the mean itself are used for continuous moderators. If `"plus-minus"`, plots lines when the moderator is at +/- 1 standard deviation without the mean. You may also choose `"terciles"` to split the data into equally-sized groups and choose the point at the mean of each of those groups. If the moderator is a factor variable and `modx.values` is `NULL`, each level of the factor is included. You may specify any subset of the factor levels (e.g.,

c("Level 1", "Level 3")) as long as there is more than 1. The levels will be plotted in the order you provide them, so this can be used to reorder levels as well.

mod2.values — For which values of the second moderator should the plot be facetted by? That is, there will be a separate plot for each level of this moderator. Defaults are the same as modx.values.

data — Optional, default is NULL. You may provide the data used to fit the model. This can be a better way to get mean values for centering and can be crucial for models with variable transformations in the formula (e.g., log(x)) or polynomial terms (e.g., poly(x, 2)). You will see a warning if the function detects problems that would likely be solved by providing the data with this argument and the function will attempt to retrieve the original data from the global environment.

cond.int — Should conditional intercepts be printed in addition to the slopes? Default is FALSE.

vce — A character string indicating the type of estimation procedure to use for estimating variances. The default ("delta") uses the delta method. Alternatives are "bootstrap", which uses bootstrap estimation, or "simulation", which averages across simulations drawn from the joint sampling distribution of model coefficients. The latter two are extremely time intensive.

iterations — If vce = "bootstrap", the number of bootstrap iterations. If vce = "simulation", the number of simulated effects to draw. Ignored otherwise.

digits — An integer specifying the number of digits past the decimal to report in the output. Default is 2. You can change the default number of digits for all jtools functions with options("jtools-digits" = digits) where digits is the desired number.

pvals — Show p values? If FALSE, these are not printed. Default is TRUE.

confint — Show confidence intervals instead of standard errors? Default is FALSE.

ci.width — A number between 0 and 1 that signifies the width of the desired confidence interval. Default is .95, which corresponds to a 95% confidence interval. Ignored if confint = FALSE.

cluster — For clustered standard errors, provide the column name of the cluster variable in the input data frame (as a string). Alternately, provide a vector of clusters.

modx.labels — A character vector of labels for each level of the moderator values, provided in the same order as the modx.values argument. If NULL, the values themselves are used as labels unless modx,values is also NULL. In that case, "+1 SD" and "-1 SD" are used.

mod2.labels — A character vector of labels for each level of the 2nd moderator values, provided in the same order as the mod2.values argument. If NULL, the values themselves are used as labels unless mod2.values is also NULL. In that case, "+1 SD" and "-1 SD" are used.

... — ignored.

Details

This allows the user to perform a simple margins analysis for the purpose of probing interaction effects in a linear regression. Two- and three-way interactions are supported, though one should be warned that three-way interactions are not easy to interpret in this way.

The function is tested with `lm`, `glm`, `svyglm`, and `merMod` inputs. Others may work as well, but are not tested. In all but the linear model case, be aware that not all the assumptions applied to simple slopes analysis apply.

Value

A list object with the following components:

`slopes`	A table of coefficients for the focal predictor at each value of the moderator
`ints`	A table of coefficients for the intercept at each value of the moderator
`modx.values`	The values of the moderator used in the analysis

Author(s)

Jacob Long <jacob.long@sc.edu>

References

Bauer, D. J., & Curran, P. J. (2005). Probing interactions in fixed and multilevel regression: Inferential and graphical techniques. *Multivariate Behavioral Research*, *40*(3), 373-400. doi:10.1207/s15327906mbr4003_5

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analyses for the behavioral sciences* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Hanmer, M. J., & Kalkan, K. O. (2013). Behind the curve: Clarifying the best approach to calculating predicted probabilities and marginal effects from limited dependent variable models. *American Journal of Political Science*, *57*, 263–277. doi:10.1111/j.15405907.2012.00602.x

See Also

`margins::margins()`

Other interaction tools: `johnson_neyman()`, `probe_interaction()`, `sim_slopes()`

`sim_slopes` *Perform a simple slopes analysis.*

Description

`sim_slopes` conducts a simple slopes analysis for the purposes of understanding two- and three-way interaction effects in linear regression.

Usage

```
sim_slopes(
  model,
  pred,
  modx,
  mod2 = NULL,
  modx.values = NULL,
  mod2.values = NULL,
  centered = "all",
  at = NULL,
  data = NULL,
  cond.int = FALSE,
  johnson_neyman = TRUE,
  jnplot = FALSE,
  jnalpha = 0.05,
  robust = FALSE,
  digits = getOption("jtools-digits", default = 2),
  pvals = TRUE,
  confint = FALSE,
  ci.width = 0.95,
  cluster = NULL,
  modx.labels = NULL,
  mod2.labels = NULL,
  v.cov = NULL,
  v.cov.args = NULL,
  ...
)
```

Arguments

model	A regression model. The function is tested with lm, glm, svyglm, merMod, rq, brmsfit, stanreg models. Models from other classes may work as well but are not officially supported. The model should include the interaction of interest.
pred	The name of the predictor variable involved in the interaction. This can be a bare name or string. Note that it is evaluated using rlang, so programmers can use the !! syntax to pass variables instead of the verbatim names.
modx	The name of the moderator variable involved in the interaction. This can be a bare name or string. The same rlang proviso applies as with pred.
mod2	Optional. The name of the second moderator variable involved in the interaction. This can be a bare name or string. The same rlang proviso applies as with pred.
modx.values	For which values of the moderator should lines be plotted? There are two basic options: • A vector of values (e.g., c(1, 2, 3)) • A single argument asking to calculate a set of values. See details below. Default is NULL. If NULL (or mean-plus-minus), then the customary +/- 1 standard deviation from the mean as well as the mean itself are used for continuous

moderators. If "plus-minus", plots lines when the moderator is at +/- 1 standard deviation without the mean. You may also choose "terciles" to split the data into equally-sized groups and choose the point at the mean of each of those groups.

If the moderator is a factor variable and modx.values is NULL, each level of the factor is included. You may specify any subset of the factor levels (e.g., c("Level 1", "Level 3")) as long as there is more than 1. The levels will be plotted in the order you provide them, so this can be used to reorder levels as well.

mod2.values — For which values of the second moderator should the plot be facetted by? That is, there will be a separate plot for each level of this moderator. Defaults are the same as modx.values.

centered — A vector of quoted variable names that are to be mean-centered. If "all", all non-focal predictors as well as the pred variable are centered. You may instead pass a character vector of variables to center. User can also use "none" to base all predictions on variables set at 0. The response variable, modx, and mod2 variables are never centered.

at — If you want to manually set the values of other variables in the model, do so by providing a named list where the names are the variables and the list values are vectors of the values. Note that you cannot alter the values of the pred, modx, or mod2 variables and this will take precedence over the centered argument (but any variables unmentioned by at will be centered as specified by centered). For linear models, this will only change the output of the conditional intercepts.

data — Optional, default is NULL. You may provide the data used to fit the model. This can be a better way to get mean values for centering and can be crucial for models with variable transformations in the formula (e.g., log(x)) or polynomial terms (e.g., poly(x, 2)). You will see a warning if the function detects problems that would likely be solved by providing the data with this argument and the function will attempt to retrieve the original data from the global environment.

cond.int — Should conditional intercepts be printed in addition to the slopes? Default is FALSE.

johnson_neyman — Should the Johnson-Neyman interval be calculated? Default is TRUE. This can be performed separately with johnson_neyman.

jnplot — Should the Johnson-Neyman interval be plotted as well? Default is FALSE.

jnalpha — What should the alpha level be for the Johnson-Neyman interval? Default is .05, which corresponds to a 95% confidence interval.

robust — Should robust standard errors be used to find confidence intervals for supported models? Default is FALSE, but you should specify the type of sandwich standard errors if you'd like to use them (i.e., "HC0", "HC1", and so on). If TRUE, defaults to "HC3" standard errors.

digits — An integer specifying the number of digits past the decimal to report in the output. Default is 2. You can change the default number of digits for all jtools functions with options("jtools-digits" = digits) where digits is the desired number.

pvals	Show p values? If FALSE, these are not printed. Default is TRUE.
confint	Show confidence intervals instead of standard errors? Default is FALSE.
ci.width	A number between 0 and 1 that signifies the width of the desired confidence interval. Default is .95, which corresponds to a 95% confidence interval. Ignored if confint = FALSE.
cluster	For clustered standard errors, provide the column name of the cluster variable in the input data frame (as a string). Alternately, provide a vector of clusters.
modx.labels	A character vector of labels for each level of the moderator values, provided in the same order as the modx.values argument. If NULL, the values themselves are used as labels unless modx,values is also NULL. In that case, "+1 SD" and "-1 SD" are used.
mod2.labels	A character vector of labels for each level of the 2nd moderator values, provided in the same order as the mod2.values argument. If NULL, the values themselves are used as labels unless mod2.values is also NULL. In that case, "+1 SD" and "-1 SD" are used.
v.cov	A function to calculate variances for the model. Examples could be sandwich::vcovPC().
v.cov.args	A list of arguments for the v.cov function. For whichever argument should be the fitted model, put "model".
...	Arguments passed to johnson_neyman and summ.

Details

This allows the user to perform a simple slopes analysis for the purpose of probing interaction effects in a linear regression. Two- and three-way interactions are supported, though one should be warned that three-way interactions are not easy to interpret in this way.

For more about Johnson-Neyman intervals, see johnson_neyman.

The function is tested with lm, glm, svyglm, and merMod inputs. Others may work as well, but are not tested. In all but the linear model case, be aware that not all the assumptions applied to simple slopes analysis apply.

Value

A list object with the following components:

slopes	A table of coefficients for the focal predictor at each value of the moderator
ints	A table of coefficients for the intercept at each value of the moderator
modx.values	The values of the moderator used in the analysis
mods	A list containing each regression model created to estimate the conditional coefficients.
jn	If johnson_neyman = TRUE, a list of johnson_neyman objects from johnson_neyman. These contain the values of the interval and the plots. If a 2-way interaction, the list will be of length 1. Otherwise, there will be 1 johnson_neyman object for each value of the 2nd moderator for 3-way interactions.

Author(s)

Jacob Long <jacob.long@sc.edu>

References

Bauer, D. J., & Curran, P. J. (2005). Probing interactions in fixed and multilevel regression: Inferential and graphical techniques. *Multivariate Behavioral Research*, *40*(3), 373-400. doi:10.1207/s15327906mbr4003_5

Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analyses for the behavioral sciences* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

See Also

`interact_plot` accepts similar syntax and will plot the results with `ggplot`.

`testSlopes()` from `rockchalk` performs a hypothesis test of differences and provides Johnson-Neyman intervals.

`simpleSlope()` from `pequod` performs a similar analysis.

Other interaction tools: `johnson_neyman()`, `probe_interaction()`, `sim_margins()`

Examples

```
# Using a fitted model as formula input
fiti <- lm(Income ~ Frost + Murder * Illiteracy,
  data = as.data.frame(state.x77))
sim_slopes(model = fiti, pred = Murder, modx = Illiteracy)

# With svyglm
if (requireNamespace("survey")) {
library(survey)
data(api)
dstrat <- svydesign(id = ~1, strata = ~stype, weights = ~pw,
                    data = apistrat, fpc = ~fpc)
regmodel <- svyglm(api00 ~ ell * meals, design = dstrat)
sim_slopes(regmodel, pred = ell, modx = meals)

# 3-way with survey and factor input
regmodel <- svyglm(api00 ~ ell * meals * sch.wide, design = dstrat)
sim_slopes(regmodel, pred = ell, modx = meals, mod2 = sch.wide)
}
```

`tidy.sim_margins` *Tidiers for* `sim_margins()` *objects.*

Description

You can use `broom::tidy()` and `broom::glance()` for "tidy" methods of storing `sim_margins` output.

Usage

```
## S3 method for class 'sim_margins'
tidy(x, conf.level = 0.95, ...)

## S3 method for class 'sim_margins'
glance(x, ...)
```

Arguments

x	The sim_margins object
conf.level	The width of confidence intervals. Default is .95 (95%).
...	Ignored.

---

tidy.sim_slopes *Tidiers for* sim_slopes() *objects.*

Description

You can use broom::tidy() and broom::glance() for "tidy" methods of storing sim_slopes output.

Usage

```
## S3 method for class 'sim_slopes'
tidy(x, conf.level = 0.95, ...)

## S3 method for class 'sim_slopes'
glance(x, ...)
```

Arguments

x	The sim_slopes object
conf.level	The width of confidence intervals. Default is .95 (95%).
...	Ignored.

Index