Package: reghelper (via r-universe)

August 28, 2024

Type Package

Title Helper Functions for Regression Analysis

Version 1.1.2

Date 2023-09-02

Description A set of functions used to automate commonly used methods in regression analysis. This includes plotting interactions, and calculating simple slopes, standardized coefficients, regions of significance (Johnson & Neyman, 1936; cf. Spiller et al., 2012), etc. See the reghelper documentation for more information, documentation, and examples.

License GPL-3

URL https://github.com/jeff-hughes/reghelper

BugReports https://github.com/jeff-hughes/reghelper/issues

Depends R (>= 3.1.0)

Imports ggplot2 (>= 1.0.0), rlang, stats, nlme, lme4, MASS, utils

Suggests testthat (>= 0.8.1)

Encoding UTF-8

RoxygenNote 7.2.3

Repository https://jeff-hughes.r-universe.dev

RemoteUrl https://github.com/jeff-hughes/reghelper

RemoteRef HEAD

RemoteSha 64ea7a2dea763fb31cb4b83fab0c20e9e507881e

Contents

`beta` *Standardized coefficients of a model.*

Description

`beta` returns the summary of a linear model where all variables have been standardized. It takes a regression model and standardizes the variables, in order to produce standardized (i.e., beta) coefficients rather than unstandardized (i.e., B) coefficients.

Usage

```
beta(model, ...)

## S3 method for class 'lm'
beta(model, x = TRUE, y = TRUE, skip = NULL, ...)

## S3 method for class 'aov'
beta(model, x = TRUE, y = TRUE, skip = NULL, ...)

## S3 method for class 'glm'
beta(model, x = TRUE, y = FALSE, skip = NULL, ...)
```

Arguments

`model`	A fitted linear model of type 'lm', 'glm', or 'aov'.
`...`	Not currently implemented; used to ensure consistency with S3 generic.
`x`	Logical. Whether or not to standardize predictor variables.
`y`	Logical. Whether or not to standardize criterion variables.
`skip`	A string vector indicating any variables you do *not* wish to be standardized.

Details

Unlike similar functions, this function properly calculates standardized estimates for interaction terms (by first standardizing each of the predictor variables separately, rather than using the standard deviation of the interaction term itself).

Value

Returns the summary of a regression model, with the output showing the standardized coefficients, standard error, t-values, and p-values for each predictor. The exact form of the values returned depends on the class of regression model used.

Methods (by class)

- `beta(lm)`: Standardized coefficients for a linear model.
- `beta(aov)`: Standardized coefficients for ANOVA.
- `beta(glm)`: Standardized coefficients for a generalized linear model.

Examples

```
# iris data, showing use with lm()
model1 <- lm(Sepal.Length ~ Petal.Length + Petal.Width, iris)
beta(model1)  # all three variables standardized

model2 <- lm(Sepal.Width ~ Petal.Width + Species, iris)
beta(model2, skip='Species')  # all variables except Species standardized

# mtcars data, showing use with glm()
model1 <- glm(vs ~ wt + hp, data=mtcars, family='binomial')
beta(model1)  # wt and hp standardized, vs is not by default
```

`build_model` *Incremental block modelling.*

Description

`build_model` allows you to incrementally add terms to a linear regression model. Given a list of names of variables at each step, this function will run a series of models, adding the terms for each block incrementally to "build up" to a final model including all the terms.

Usage

```
build_model(dv, ..., data = NULL, opts = NULL, model = "lm")
```

Arguments

`dv`	The variable name to be used as the dependent variable.
`...`	Pass through variable names (or interaction terms) to add for each block. To add one term to a block, just pass it through directly; to add multiple terms, pass it through in a vector or list. Blocks will be added in the order they are passed to the function, and variables from previous blocks will be included with each subsequent block, so they do not need to be repeated.

data	An optional data frame containing the variables in the model. If not found in data, the variables are taken from the environment from which the function is called.
opts	List of arguments to be passed to the model function.
model	The type of model to use; supports 'lm', 'aov', and 'glm'.

Details

Note: Cases with missing data are dropped based on the *final* model that includes all the relevant terms. This ensures that all the models are tested on the same number of cases.

Value

A named list with the following elements:

formulas	A list of the regression formulas used for each block.
models	A list of all regression models.

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
coef(model1)

# 2 blocks: Species; Species + Petal.Length + Petal.Width + Petal.Length:Petal.Width
model2 <- build_model(Sepal.Length, Species, c(Petal.Length * Petal.Width), data=iris, model='lm')
summary(model2)
coef(model2)
```

---

build_model_q *Incremental block modelling.*

Description

build_model_q allows you to incrementally add terms to a linear regression model. Given a list of names of variables at each step, this function will run a series of models, adding the terms for each block incrementally to "build up" to a final model including all the terms.

Usage

```
build_model_q(dv, blocks = NULL, data = NULL, opts = NULL, model = "lm")
```

Arguments

dv	String of the variable name to be used as the dependent variable.
blocks	List of variable names (or interaction terms) to add for each block. Each list element should be a single string with terms for that block. Variables from previous blocks will be included with each subsequent block, so they do not need to be repeated.
data	An optional data frame containing the variables in the model. If not found in data, the variables are taken from the environment from which the function is called.
opts	List of arguments to be passed to the model function.
model	The type of model to use; supports 'lm', 'aov', and 'glm'.

Details

Note that in most cases it is easier to use build_model and pass variable names in directly instead of strings of variable names. build_model_q uses standard evaluation in cases where such evaluation is easier.

Note: Cases with missing data are dropped based on the *final* model that includes all the relevant terms. This ensures that all the models are tested on the same number of cases.

Value

A named list with the following elements:

formulas	A list of the regression formulas used for each block.
models	A list of all regression models.

See Also

build_model

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model_q('Sepal.Length', list('Petal.Length + Petal.Width'),
    data=iris, model='lm')
summary(model1)
coef(model1)

# 2 blocks: Species; Species + Petal.Length + Petal.Width + Petal.Length:Petal.Width
model2 <- build_model_q('Sepal.Length', list('Species', 'Species + Petal.Length * Petal.Width'),
    data=iris, model='lm')
summary(model2)
coef(model2)
```

`cell_means` *Estimated values of a linear model.*

Description

`cell_means` calculates the predicted values at specific points, given a fitted regression model (linear, generalized, or ANOVA).

Usage

```
cell_means(model, ...)

## S3 method for class 'lm'
cell_means(model, ..., levels = NULL)

## S3 method for class 'aov'
cell_means(model, ..., levels = NULL)

## S3 method for class 'glm'
cell_means(model, ..., levels = NULL, type = c("link", "response"))
```

Arguments

`model`	A fitted linear model of type 'lm', 'aov', or 'glm'.
`...`	Pass through variable names to add them to the table.
`levels`	A list with element names corresponding to some or all of the variables in the model. Each list element should be a vector with the names of factor levels (for categorical variables) or numeric points (for continuous variables) at which to test that variable.
`type`	The type of prediction required. The default 'link' is on the scale of the linear predictors; the alternative 'response' is on the scale of the response variable. For more information, see `predict.glm`.

Details

By default, this function will provide means at -1 SD, the mean, and +1 SD for continuous variables, and at each level of categorical variables. This can be overridden with the `levels` parameter.

If there are additional covariates in the model other than what are selected in the function call, these variables will be set to their respective means. In the case of a categorical covariate, the results will be averaged across all its levels.

Value

A data frame with a row for each predicted value. The first few columns identify the level at which each variable in your model was set. After columns for each variable, the data frame has columns for the predicted value, the standard error of the predicted mean, and the 95% confidence interval.

Methods (by class)

- `cell_means(lm)`: Estimated values for a linear model.
- `cell_means(aov)`: Estimated means for ANOVA.
- `cell_means(glm)`: Estimated values for a generalized linear model.

Examples

```
# iris data
model <- lm(Sepal.Length ~ Petal.Length + Petal.Width, iris)
summary(model)
cell_means(model, Petal.Length)
```

`cell_means_q` *Estimated values of a linear model.*

Description

`cell_means_q` calculates the predicted values at specific points, given a fitted regression model (linear, generalized, or ANOVA).

Usage

```
cell_means_q(model, ...)

## S3 method for class 'lm'
cell_means_q(model, vars = NULL, levels = NULL, ...)

## S3 method for class 'aov'
cell_means_q(model, vars = NULL, levels = NULL, ...)

## S3 method for class 'glm'
cell_means_q(
  model,
  vars = NULL,
  levels = NULL,
  type = c("link", "response"),
  ...
)
```

Arguments

`model`	A fitted linear model of type 'lm', 'aov', or 'glm'.
`...`	Not currently implemented; used to ensure consistency with S3 generic.
`vars`	A vector or list with variable names to be added to the table.

`levels` A list with element names corresponding to some or all of the variables in the model. Each list element should be a vector with the names of factor levels (for categorical variables) or numeric points (for continuous variables) at which to test that variable.

`type` The type of prediction required. The default 'link' is on the scale of the linear predictors; the alternative 'response' is on the scale of the response variable. For more information, see `predict.glm`.

Details

By default, this function will provide means at -1 SD, the mean, and +1 SD for continuous variables, and at each level of categorical variables. This can be overridden with the `levels` parameter.

If there are additional covariates in the model other than what are selected in the function call, these variables will be set to their respective means. In the case of a categorical covariate, the results will be averaged across all its levels.

Note that in most cases it is easier to use `cell_means` and pass variable names in directly instead of strings of variable names. `cell_means_q` uses standard evaluation in cases where such evaluation is easier.

Value

A data frame with a row for each predicted value. The first few columns identify the level at which each variable in your model was set. After columns for each variable, the data frame has columns for the predicted value, the standard error of the predicted mean, and the 95% confidence interval.

Methods (by class)

- `cell_means_q(lm)`: Estimated values for a linear model.
- `cell_means_q(aov)`: Estimated means for ANOVA.
- `cell_means_q(glm)`: Estimated values for a generalized linear model.

See Also

`cell_means`

Examples

```
# iris data
model <- lm(Sepal.Length ~ Petal.Length + Petal.Width, iris)
summary(model)
cell_means_q(model, 'Petal.Length')
```

graph_model *Graph interactions for fitted models.*

Description

graph_model provides an easy way to graph interactions in fitted models (linear, generalized linear, hierarchical linear, or ANOVA). Selected variables will be graphed at +/- 1 SD (if continuous) or at each level of the factor (if categorical).

Usage

```
graph_model(model, ...)

## S3 method for class 'lm'
graph_model(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)

## S3 method for class 'aov'
graph_model(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
```

```
)

## S3 method for class 'glm'
graph_model(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  type = c("link", "response"),
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)

## S3 method for class 'lme'
graph_model(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)

## S3 method for class 'merMod'
graph_model(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
```

```
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)
```

Arguments

model	A fitted linear model of type 'lm', 'aov', 'glm', 'lme', or 'merMod'.
...	Not currently implemented; used to ensure consistency with S3 generic.
y	The variable to be plotted on the y-axis. This variable is required for the graph.
x	The variable to be plotted on the x-axis. This variable is required for the graph.
lines	The variable to be plotted using separate lines (optional).
split	The variable to be split among separate graphs (optional).
errorbars	A string indicating what kind of error bars to show. Acceptable values are "CI" (95% confidence intervals), "SE" (+/-1 standard error of the predicted means), or "none".
ymin	Number indicating the minimum value for the y-axis scale. Default NULL value will adjust position to the lowest y value.
ymax	Number indicating the maximum value for the y-axis scale. Default NULL value will adjust position to the highest y value.
labels	A named list with strings for the various plot labels: 'title' will set the graph title, 'y' sets the y-axis label, 'x' sets the x-axis label, 'lines' sets the legend label, and 'split' sets the label for the facet. If any label is not set, the names of the variables will be used. Setting a label explicitly to NA will set a label with an empty string.
bargraph	Logical. TRUE will draw a bar graph of the results; FALSE will draw a line graph of the results.
draw.legend	Logical. Whether or not to draw legend on the graph.
dodge	A numeric value indicating the amount each point on the graph should be shifted left or right, which can help for readability when points are close together. Default value is 0, with .1 or .2 probably sufficient in most cases.
exp	Logical. If TRUE, the exponential function `exp()` will be used to transform the y-axis (i.e., e to the power of y). Useful for logistic regressions or for converting log-transformed y-values to their original units.
type	The type of prediction required. The default 'link' is on the scale of the linear predictors; the alternative 'response' is on the scale of the response variable. For more information, see `predict.glm`.

Details

If there are additional covariates in the model other than what is indicated to be graphed by the function, these variables will be plotted at their respective means. In the case of a categorical covariate, the results will be averaged across all its levels.

Value

A ggplot2 graph of the plotted variables in the model.

Methods (by class)

- `graph_model(lm)`: Graphing linear models.
- `graph_model(aov)`: Graphing ANOVA.
- `graph_model(glm)`: Graphing generalized linear models.
- `graph_model(lme)`: Graphing hierarchical linear models (nlme).
- `graph_model(merMod)`: Graphing hierarchical linear models (lme4).

Examples

```
# iris data
model <- lm(Sepal.Width ~ Sepal.Length * Species, data=iris)
graph_model(model, y=Sepal.Width, x=Sepal.Length, lines=Species)

# Orthodont data
if (require(nlme, quietly=TRUE)) {
    model <- lme(distance ~ age * Sex, data=Orthodont, random=~1|Subject)
    graph_model(model, y=distance, x=age, lines=Sex)
}

# Arabidopsis data
if (require(lme4, quietly=TRUE)) {
    model <- lmer(total.fruits ~ nutrient * amd + rack + (1|gen), data=Arabidopsis)
    graph_model(model, y=total.fruits, x=nutrient, lines=amd)
}
```

---

graph_model_q *Graph interactions for fitted models.*

Description

`graph_model_q` provides an easy way to graph interactions in fitted models (linear, generalized linear, hierarchical linear, or ANOVA). Selected variables will be graphed at +/- 1 SD (if continuous) or at each level of the factor (if categorical).

Usage

```
graph_model_q(model, ...)

## S3 method for class 'lm'
graph_model_q(
  model,
  y,
  x,
```

```
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)

## S3 method for class 'aov'
graph_model_q(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)

## S3 method for class 'glm'
graph_model_q(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  type = c("link", "response"),
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
```

```
)

## S3 method for class 'lme'
graph_model_q(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)

## S3 method for class 'merMod'
graph_model_q(
  model,
  y,
  x,
  lines = NULL,
  split = NULL,
  errorbars = c("CI", "SE", "none"),
  ymin = NULL,
  ymax = NULL,
  labels = NULL,
  bargraph = FALSE,
  draw.legend = TRUE,
  dodge = 0,
  exp = FALSE,
  ...
)
```

Arguments

model	A fitted linear model of type 'lm', 'aov', 'glm', 'lme', or 'merMod'.
...	Not currently implemented; used to ensure consistency with S3 generic.
y	The variable to be plotted on the y-axis. This variable is required for the graph.
x	The variable to be plotted on the x-axis. This variable is required for the graph.
lines	The variable to be plotted using separate lines (optional).
split	The variable to be split among separate graphs (optional).

errorbars	A string indicating what kind of error bars to show. Acceptable values are "CI" (95% confidence intervals), "SE" (+/-1 standard error of the predicted means), or "none".
ymin	Number indicating the minimum value for the y-axis scale. Default NULL value will adjust position to the lowest y value.
ymax	Number indicating the maximum value for the y-axis scale. Default NULL value will adjust position to the highest y value.
labels	A named list with strings for the various plot labels: 'title' will set the graph title, 'y' sets the y-axis label, 'x' sets the x-axis label, 'lines' sets the legend label, and 'split' sets the label for the facet. If any label is not set, the names of the variables will be used. Setting a label explicitly to NA will set a label with an empty string.
bargraph	Logical. TRUE will draw a bar graph of the results; FALSE will draw a line graph of the results.
draw.legend	Logical. Whether or not to draw legend on the graph.
dodge	A numeric value indicating the amount each point on the graph should be shifted left or right, which can help for readability when points are close together. Default value is 0, with .1 or .2 probably sufficient in most cases.
exp	Logical. If TRUE, the exponential function `exp()` will be used to transform the y-axis (i.e., e to the power of y). Useful for logistic regressions or for converting log-transformed y-values to their original units.
type	The type of prediction required. The default 'link' is on the scale of the linear predictors; the alternative 'response' is on the scale of the response variable. For more information, see `predict.glm`.

Details

If there are additional covariates in the model other than what is indicated to be graphed by the function, these variables will be plotted at their respective means. In the case of a categorical covariate, the results will be averaged across all its levels.

Note that in most cases it is easier to use `graph_model` and pass variable names in directly instead of strings of variable names. `graph_model_q` uses standard evaluation in cases where such evaluation is easier.

Value

A ggplot2 graph of the plotted variables in the model.

Methods (by class)

- `graph_model_q(lm)`: Graphing linear models.
- `graph_model_q(aov)`: Graphing ANOVA.
- `graph_model_q(glm)`: Graphing generalized linear models.
- `graph_model_q(lme)`: Graphing hierarchical linear models (nlme).
- `graph_model_q(merMod)`: Graphing hierarchical linear models (lme4).

See Also

graph_model

Examples

```
# iris data
model <- lm(Sepal.Width ~ Sepal.Length * Species, data=iris)
graph_model_q(model, y='Sepal.Width', x='Sepal.Length', lines='Species')

# Orthodont data
if (require(nlme, quietly=TRUE)) {
    model <- lme(distance ~ age * Sex, data=Orthodont, random=~1|Subject)
    graph_model_q(model, y='distance', x='age', lines='Sex')
}

# Arabidopsis data
if (require(lme4, quietly=TRUE)) {
    model <- lmer(total.fruits ~ nutrient * amd + rack + (1|gen), data=Arabidopsis)
    graph_model_q(model, y='total.fruits', x='nutrient', lines='amd')
}
```

ICC *Intra-class correlation.*

Description

ICC calculates the intra-class correlation (ICC) from a fitted hierarchical linear model using the 'nlme' or 'lme4' packages.

Usage

```
ICC(model, ...)

## S3 method for class 'lme'
ICC(model, ...)

## S3 method for class 'merMod'
ICC(model, ...)
```

Arguments

model	A fitted linear model of type 'lme' (nlme) or 'merMod' (lme4; linear, generalized, or nonlinear).
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

The ICC is the proportion of variance that is between-person variance. For more information, see Hoyt & Kenny (2013).

Value

The intra-class correlation of the model.

Methods (by class)

- `ICC(lme)`: Intra-class correlation for lme (nlme).
- `ICC(merMod)`: Intra-class correlation for lmer (lme4).

Examples

```
# iris data, showing use with lme()
if (require(nlme, quietly=TRUE)) {
    model <- lme(Sepal.Width ~ 1, random=~1|Species, data=iris)
    ICC(model)  # .49 of variance is between-subjects
}

# iris data, showing use with lmer()
if (require(lme4, quietly=TRUE)) {
    model <- lmer(Sepal.Width ~ 1 + (1|Species), data=iris)
    ICC(model)  # .49 of variance is between-subjects
}
```

`print.simple_slopes` *Print simple slopes.*

Description

`print` method for class "`simple_slopes`".

Usage

```
## S3 method for class 'simple_slopes'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

`x`	An object of class "`simple_slopes`", usually, a result of a call to `simple_slopes`.
`digits`	The number of significant digits to use when printing.
`signif.stars`	Logical. If `TRUE`, 'significance stars' are printed for each coefficient.
`...`	Further arguments passed to or from other methods.

See Also

simple_slopes

---

sig_regions *Regions of significance for an interaction.*

Description

sig_regions calculates the Johnson-Neyman (J-N) regions of significance for an interaction – the points at which the simple effect of the categorical predictor changes from non-significant to significant.

Usage

```
sig_regions(model, ...)

## S3 method for class 'lm'
sig_regions(model, alpha = 0.05, precision = 4, ...)

## S3 method for class 'glm'
sig_regions(model, alpha = 0.05, precision = 4, ...)
```

Arguments

model	A fitted linear model of type 'lm' or 'glm' with one two-way interaction including one categorical predictor and one continuous variable.
...	Not currently implemented; used to ensure consistency with S3 generic.
alpha	The level at which to test for significance. Default value is .05.
precision	The number of decimal places to which to round the alpha level (e.g., precision=5 would look for regions of significance at .05000).

Details

This function takes a linear or generalized linear model with one two-way interaction, where one of the predictors in the interaction is categorical (factor) and the other is continuous. For other types of interaction terms, use the simple_slopes function instead.

For more information about regions of significance, see Spiller, Fitzsimons, Lynch, & McClelland (2012).

Value

A named vector with a 'lower' and an 'upper' J-N point. If one or more of the J-N points fall outside the range of your predictor, the function will return NA for that point. If your interaction is not significant, both J-N points will be NA.

Methods (by class)

- `sig_regions(lm)`: Johnson-Neyman points for linear models.
- `sig_regions(glm)`: Johnson-Neyman points for generalized linear models.

See Also

simple_slopes

Examples

```
# mtcars data
mtcars$am <- factor(mtcars$am)  # make 'am' categorical
model <- lm(mpg ~ wt * am, data=mtcars)
summary(model)  # significant interaction
sig_regions(model)
```

---

simple_slopes *Simple slopes of an interaction.*

Description

`simple_slopes` calculates all the simple effects of an interaction in a fitted model (linear, generalized linear, hierarchical linear, or ANOVA).

Usage

```
simple_slopes(model, ...)

## S3 method for class 'lm'
simple_slopes(model, levels = NULL, confint = FALSE, ci.width = 0.95, ...)

## S3 method for class 'glm'
simple_slopes(model, levels = NULL, confint = FALSE, ci.width = 0.95, ...)

## S3 method for class 'lme'
simple_slopes(model, levels = NULL, confint = FALSE, ci.width = 0.95, ...)

## S3 method for class 'merMod'
simple_slopes(
  model,
  levels = NULL,
  confint = FALSE,
  ci.width = 0.95,
  confint.method = c("Wald", "profile", "boot"),
  ...
)
```

Arguments

`model`	A fitted linear model of type 'lm', 'glm', 'aov', 'lme' (nlme), or 'merMod' (lme4), with at least one interaction term.
`...`	Additional parameters to be passed on to the 'confint' method, if 'confint' is TRUE.
`levels`	A list with element names corresponding to some or all of the variables in the model. Each list element should be a vector with the names of factor levels (for categorical variables) or numeric values (for continuous variables) at which to test that variable. **Note:** If you do not include 'sstest' as one of these levels, the function will not test the simple effects for that variable.
`confint`	Whether or not to include confidence intervals for each estimate.
`ci.width`	If 'confint' is TRUE, this represents the width of the confidence intervals to calculate, as a proportion from 0 to 1.
`confint.method`	For 'merMod' models only, specifies what method to use for computing the confidence intervals.

Details

If the model includes interactions at different levels (e.g., three two-way interactions and one three-way interaction), the function will test the simple effects of the highest-order interaction. If there are multiple interactions in the highest order, it will test the first one in the model. If you wish to test simple effects for a different interaction, simply switch the order in the formula.

By default, this function will provide slopes at -1 SD, the mean, and +1 SD for continuous variables, and at each level of categorical variables. This can be overridden with the `levels` parameter.

If a categorical variable with more than two levels is being tested, you may see multiple rows for that test. One row will be shown for each contrast for that variable; the name of the contrast is identified in parentheses after the 'sstest' label.

Value

A data frame with a row for each simple effect. The first few columns identify the level at which each variable in your model was set for that test. A 'sstest' value in a particular column indicates that the simple slope for this variable was being tested. After columns for each variable, the data frame has columns for the slope of the test variable, the standard error, t-value, p-value, and degrees of freedom for the model. For 'merMod' models, the degrees of freedom and p-values will not appear, as these are not calculated by the lme4 package.

Methods (by class)

- `simple_slopes(lm)`: Simple slopes for linear models.
- `simple_slopes(glm)`: Simple slopes for generalized linear models.
- `simple_slopes(lme)`: Simple slopes for hierarchical linear models (nlme).
- `simple_slopes(merMod)`: Simple slopes for hierarchical linear models (lme4).

Examples

```
# linear model
mtcars$am <- factor(mtcars$am)  # make 'am' categorical
model <- lm(mpg ~ wt * am, data=mtcars)
summary(model)  # significant interaction
simple_slopes(model)
simple_slopes(model,
   levels=list(wt=c(2, 3, 4, 'sstest'), am=c(0, 1, 'sstest')))  # test at specific levels

# generalized linear model
model <- glm(vs ~ gear * wt, data=mtcars, family='binomial')
summary(model)  # marginal interaction
simple_slopes(model)
simple_slopes(model,
   levels=list(gear=c(2, 3, 4, 'sstest'), wt=c(2, 3, 'sstest')))  # test at specific levels

# hierarchical linear model (nlme)
if (require(nlme, quietly=TRUE)) {
    model <- lme(Sepal.Width ~ Sepal.Length * Petal.Length, random=~1|Species, data=iris)
    summary(model)  # significant interaction
    simple_slopes(model)
    simple_slopes(model,
        levels=list(Sepal.Length=c(4, 5, 6, 'sstest'),
        Petal.Length=c(2, 3, 'sstest')))  # test at specific levels
}

# hierarchical linear model (lme4)
if (require(lme4, quietly=TRUE)) {
    model <- lmer(Sepal.Width ~ Sepal.Length * Petal.Length + (1|Species), data=iris)
    summary(model)
    simple_slopes(model)
    simple_slopes(model,
        levels=list(Sepal.Length=c(4, 5, 6, 'sstest'),
        Petal.Length=c(2, 3, 'sstest')))  # test at specific levels
}
```

`summary.block_lm` *Summary functions for build_model block regression models.*

Description

These functions offer useful methods for objects created by the `build_model` function: `block_lm`, `block_aov`, and `block_glm`.

Usage

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

```
## S3 method for class 'block_lm_summary'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'block_lm'
coef(object, num = NULL, ...)

## S3 method for class 'block_lm'
residuals(object, num = NULL, ...)

## S3 method for class 'block_lm'
fitted(object, num = NULL, ...)

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

## S3 method for class 'block_aov_summary'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'block_aov'
coef(object, num = NULL, ...)

## S3 method for class 'block_aov'
residuals(object, num = NULL, ...)

## S3 method for class 'block_aov'
fitted(object, num = NULL, ...)

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

## S3 method for class 'block_glm_summary'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

```
## S3 method for class 'block_glm'
coef(object, num = NULL, ...)

## S3 method for class 'block_glm'
residuals(object, num = NULL, ...)

## S3 method for class 'block_glm'
fitted(object, num = NULL, ...)
```

Arguments

object	An object of class block_lm, block_aov, or block_glm, usually, a result of a call to build_model.
...	Further arguments passed to or from other methods.
x	An object of class block_lm_summary, block_aov_summary, or block_glm_summary, usually, a result of a call to the corresponding summary function (e.g., summary.block_lm).
digits	The number of significant digits to use when printing.
signif.stars	Logical. If TRUE, 'significance stars' are printed for each coefficient.
num	Numeric vector with the index of model(s) from which to return the requested output. If NULL, will return output from all blocks.

Value

The summary function computes and returns a named list of summary statistics of the fitted linear models given in object. The list has the following elements:

formulas	A list of the regression formulas used for each block.
residuals	A matrix with quantiles of the residuals for each model.
coefficients	A list with a matrix of coefficients for each model, as well as the standard error, t-statistic, and p-value.
overall	A data frame with information about the overall models, including the multiple R-squared value; adjusted R-

The other functions listed here provide convenient access to the individual components of this summary.

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
coef(model1)  # returns coefficients from both blocks 1 and 2

# 2 blocks: Species; Species + Petal.Length + Petal.Width + Petal.Length:Petal.Width
model2 <- build_model(Sepal.Length, Species, c(Petal.Length * Petal.Width), data=iris, model='lm')
summary(model2)
coef(model2, num=2)  # returns coefficients from second block only
```

Index