Package 'reghelper'

April 8, 2017

Type Package

Title Helper Functions for Regression Analysis

Version 0.3.3

Date 2017-04-07

Description A set of functions used to automate commonly used methods in regression analysis. This includes plotting interactions, calculating simple slopes, calculating standardized coefficients, etc. See the reghelper documentation for more information, documentation, and examples.

License GPL-3

URL

Depends R (>= 3.1.0)

LazyData true

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

Suggests testthat (>= 0.8.1)

RoxygenNote 5.0.1

NeedsCompilation no

Author Jeffrey Hughes [aut, cre]

Maintainer Jeffrey Hughes <jeff.hughes@gmail.com>

Repository CRAN

Date/Publication 2017-04-07 22:53:04 UTC

R topics documented:

beta *Standardized coeffients of a model.*

Description

beta is a generic function for producing standardized coefficients from regression models.

Usage

```
beta(model, ...)
```

Arguments

model	A fitted linear model of type 'lm', 'glm' or 'aov'.
...	Additional arguments to be passed to the particular method for the given model.

Value

The form of the value returned by beta depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

beta.lm, beta.glm, beta.aov

Examples

```
# iris data
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
```

beta.glm *Standardized coeffients of a model.*

Description

beta.glm returns the summary of a linear model where all variables have been standardized.

Usage

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

Arguments

model	A fitted generalized linear model of type 'glm'.
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 standarized.
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function takes a generalized linear regression model and standardizes the variables, in order to produce standardized (i.e., beta) coefficients rather than unstandardized (i.e., B) coefficients.

Note: Unlike beta.lm, the y parameter is set to FALSE by default, to avoid issues with some family functions (e.g., binomial).

Value

Returns the summary of a generalized linear model, with the output showing the beta coefficients, standard error, t-values, and p-values for each predictor.

See Also

beta.glm

Examples

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

---

beta.lm *Standardized coeffients of a model.*

Description

beta.lm returns the summary of a linear model where all variables have been standardized.

Usage

```
## 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, ...)
```

Arguments

`model`	A fitted linear model of type 'lm'.
`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 standarized.
`...`	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function takes a linear regression model and standardizes the variables, in order to produce standardized (i.e., beta) coefficients rather than unstandardized (i.e., B) coefficients.

Unlike similar functions, this function properly calculates standardized estimates for interaction terms (by first standardizing each of the individual predictor variables).

Value

Returns the summary of a linear model, with the output showing the beta coefficients, standard error, t-values, and p-values for each predictor.

See Also

`beta.glm`

Examples

```
# iris data
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
```

`beta.lme` *Standardized coeffients of a model.*

Description

`beta.lme` returns the summary of a linear model where all variables have been standardized.

Usage

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

Arguments

model	A fitted linear model of type 'lme'.
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 standarized.
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function takes a multi-level model and standardizes the variables, in order to produce standardized (i.e., beta) coefficients rather than unstandardized (i.e., B) coefficients.

Unlike similar functions, this function properly calculates standardized estimates for interaction terms (by first standardizing each of the individual predictor variables).

Value

Returns the summary of a multi-level linear model, with the output showing the beta coefficients, standard error, t-values, and p-values for each predictor.

See Also

beta.lm, beta.glm, beta.merMod

Examples

```
# iris data
if (require(nlme, quietly=TRUE)) {
    model <- lme(Sepal.Width ~ Sepal.Length + Petal.Length, random=~1|Species, data=iris)
    beta(model)  # all three variables standardized

    beta(model, skip='Petal.Length')  # all variables except Petal.Length standardized
}
```

---

beta.merMod *Standardized coeffients of a model.*

Description

beta.merMod returns the summary of a linear model where all variables have been standardized.

Usage

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

Arguments

model	A fitted linear model of type 'merMod' (from the 'lme4' package).
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 standarized.
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function takes a multi-level model and standardizes the variables, in order to produce standardized (i.e., beta) coefficients rather than unstandardized (i.e., B) coefficients.

Unlike similar functions, this function properly calculates standardized estimates for interaction terms (by first standardizing each of the individual predictor variables).

Value

Returns the summary of a multi-level linear model, with the output showing the beta coefficients, standard error, t-values, and p-values for each predictor.

See Also

beta.lm, beta.glm, beta.lme

Examples

```
# iris data
if (require(lme4, quietly=TRUE)) {
    model <- lmer(Sepal.Width ~ Sepal.Length + Petal.Length + (1|Species), data=iris)
    beta(model)  # all variables standardized

    beta(model, skip='Petal.Length')  # all variables except Petal.Length standardized
}
```

---

build_model *Incremental block modelling.*

Description

build_model allows you to incrementally add terms to a linear regression model.

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.
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; only supports 'lm' at this time.

Details

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.

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.

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.
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; only supports 'lm' at this time.

Details

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. 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 means.*

Description

`cell_means` is a generic function for calculating the estimated means of a linear model.

Usage

```
cell_means(model, ...)
```

Arguments

`model`	A fitted linear model of type 'lm' or 'aov'.
`...`	Additional arguments to be passed to the particular method for the given model.

Value

The form of the value returned by `cell_means` depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

`cell_means.lm`, `cell_means.aov`

Examples

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

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

Description

`cell_means.glm` calculates the predicted values at specific points, given a fitted general linear model.

Usage

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

Arguments

model	A fitted linear model of type '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 -1SD, the mean, and +1SD for continuous variables, and at each levele 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.

See Also

cell_means.lm

Examples

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

---

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

Description

cell_means.lm calculates the predicted values at specific points, given a fitted regression model.

Usage

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

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

Arguments

`model`	A fitted linear model of type 'lm'.
`...`	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.

Details

By default, this function will provide means at -1SD, the mean, and +1SD for continuous variables, and at each levele 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.

See Also

`cell_means`

Examples

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

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

Description

`cell_means_q.glm` calculates the predicted values at specific points, given a fitted general linear model.

Usage

```
cell_means_q.glm(model, vars = NULL, levels = NULL, type = c("link",
  "response"))
```

Arguments

`model`	A fitted linear model of type 'glm'.
`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 -1SD, the mean, and +1SD for continuous variables, and at each levele 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.glm` and pass variable names in directly instead of strings of variable names. `cell_means_q.glm` 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.

See Also

`cell_means.glm`, `cell_means_q.lm`

Examples

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

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

Description

`cell_means_q.lm` calculates the predicted values at specific points, given a fitted regression model.

Usage

```
cell_means_q.lm(model, vars = NULL, levels = NULL)
```

Arguments

`model`	A fitted linear model of type 'lm'.
`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.

Details

By default, this function will provide means at -1SD, the mean, and +1SD for continuous variables, and at each levele 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.lm` and pass variable names in directly instead of strings of variable names. `cell_means_q.lm` 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.

See Also

`cell_means.lm`

Examples

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

`coef.block_aov` *Extract model coefficients.*

Description

`coef` method for class "`block_aov`".

Usage

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

Arguments

`object`	An object of class "`block_aov`", usually, a result of a call to `build_model`.
`num`	Numeric vector with the index of model(s) from which to return the coefficients.
`...`	Further arguments passed to or from other methods.

Value

The coefficients of block(s) ‘num‘, or if ‘num‘ is NULL, a list of coefficients from all blocks.

See Also

`build_model`, `fitted.block_aov`, `residuals.block_aov`

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 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 second block
```

coef.block_glm *Extract model coefficients.*

Description

coef method for class "block_glm".

Usage

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

Arguments

object	An object of class "block_glm", usually, a result of a call to build_model.
num	Numeric vector with the index of model(s) from which to return the coefficients.
...	Further arguments passed to or from other methods.

Value

The coefficients of block(s) ‘num‘, or if ‘num‘ is NULL, a list of coefficients from all blocks.

See Also

build_model, coef.block_glm_summary, fitted.block_glm, residuals.block_glm

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 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 second block
```

coef.block_glm_summary *Extract model coefficients.*

Description

coef method for class "block_glm_summary".

Usage

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

Arguments

object	An object of class "block_glm_summary", usually, a result of a call to summary.block_glm.
num	Numeric vector with the index of model(s) from which to return the coefficients.
...	Further arguments passed to or from other methods.

Value

The coefficients of block(s) ‘num‘, or if ‘num‘ is NULL, a list of coefficients from all blocks.

See Also

summary.block_glm, coef.block_glm, residuals.block_glm_summary

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
coef(summary(model1))  # returns 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')
coef(summary(model2), num=2)  # returns second block
```

coef.block_lm *Extract model coefficients.*

Description

coef method for class "block_lm".

Usage

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

Arguments

object	An object of class "block_lm", usually, a result of a call to build_model.
num	Numeric vector with the index of model(s) from which to return the coefficients.
...	Further arguments passed to or from other methods.

Value

The coefficients of block(s) ‘num‘, or if ‘num‘ is NULL, a list of coefficients from all blocks.

See Also

build_model, coef.block_lm_summary, fitted.block_lm, residuals.block_lm

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 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 second block
```

coef.block_lm_summary *Extract model coefficients.*

Description

coef method for class "block_lm_summary".

Usage

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

Arguments

object	An object of class "block_lm_summary", usually, a result of a call to summary.block_lm.
num	Numeric vector with the index of model(s) from which to return the coefficients.
...	Further arguments passed to or from other methods.

Value

The coefficients of block(s) ‘num‘, or if ‘num‘ is NULL, a list of coefficients from all blocks.

See Also

summary.block_lm, coef.block_lm, residuals.block_lm_summary

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
coef(summary(model1))  # returns 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')
coef(summary(model2), num=2)  # returns second block
```

fitted.block_aov *Extract model fitted values.*

Description

fitted method for class "block_aov".

Usage

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

Arguments

object	An object of class "block_aov", usually, a result of a call to build_model.
num	Numeric vector with the index of model(s) from which to return the fitted values.
...	Further arguments passed to or from other methods.

Value

The fitted values of block(s) ‘num‘, or if ‘num‘ is NULL, a list of fitted values from all blocks.

See Also

build_model, coef.block_aov, residuals.block_aov

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
fitted(model1)  # returns 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)
fitted(model2, num=2)  # returns second block
```

---

fitted.block_glm *Extract model fitted values.*

Description

fitted method for class "block_glm".

Usage

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

Arguments

object	An object of class "block_glm", usually, a result of a call to build_model.
num	Numeric vector with the index of model(s) from which to return the fitted values.
...	Further arguments passed to or from other methods.

Value

The fitted values of block(s) ‘num‘, or if ‘num‘ is NULL, a list of fitted values from all blocks.

See Also

build_model, coef.block_glm, residuals.block_glm

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
fitted(model1)  # returns 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)
fitted(model2, num=2)  # returns second block
```

---

fitted.block_lm *Extract model fitted values.*

Description

fitted method for class "block_lm".

Usage

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

Arguments

object	An object of class "block_lm", usually, a result of a call to build_model.
num	Numeric vector with the index of model(s) from which to return the fitted values.
...	Further arguments passed to or from other methods.

Value

The fitted values of block(s) ‘num‘, or if ‘num‘ is NULL, a list of fitted values from all blocks.

See Also

build_model, coef.block_lm, residuals.block_lm

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
fitted(model1)  # returns 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)
fitted(model2, num=2)  # returns second block
```

---

graph_model *Graph fitted model interactions.*

Description

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

Usage

```
graph_model(model, ...)
```

Arguments

model	A fitted linear model of type 'lm', 'glm', 'lme', or 'merMod'.
...	Additional arguments to be passed to the particular method for the given model.

Value

A ggplot2 graph of the plotted variables in the model.

See Also

graph_model.lm, graph_model.glm

Examples

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

graph_model.glm *Graph general linear interactions.*

Description

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

Usage

```
## 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, ...)
```

Arguments

model	A fitted linear model of type 'glm'.
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).
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.
errorbars	A string indicating what kind of error bars to show. Acceptable values are "CI" (95 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 an empty label.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

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.

See Also

graph_model.lm

Examples

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

---

graph_model.lm *Graph linear interactions.*

Description

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

Usage

```
## 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, ...)
```

Arguments

model	A fitted linear model of type 'lm'.
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 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 an empty label.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

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.

Examples

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

graph_model.lme *Graph multi-level model interactions.*

Description

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

Usage

```
## 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, ...)
```

Arguments

model	A fitted linear model of type 'lme'.
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 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 an empty label.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function only deals with fixed effects of the model, after the random effects have been accounted for. To look at the lower-level effects, consider using predict.lme to generate predicted values at the desired level.

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.

See Also

graph_model.merMod, graph_model.lm

Examples

```
# 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)
}
```

---

graph_model.merMod *Graph multi-level model interactions.*

Description

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

Usage

```
## 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 'merMod' (from the 'lme4' package).
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 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 an empty label.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function only deals with fixed effects of the model, after the random effects have been accounted for. To look at the lower-level effects, consider using `predict.merMod` to generate predicted values at the desired level.

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.

See Also

`graph_model.lme`, `graph_model.lm`

Examples

```
# 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 fitted model interactions.*

Description

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

Usage

```
graph_model_q(model, ...)
```

Arguments

model	A fitted linear model of type 'lm', 'glm', 'lme', or 'merMod'.
...	Additional arguments to be passed to the particular method for the given model.

Details

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.

See Also

graph_model.lm, graph_model.glm

Examples

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

graph_model_q.glm *Graph general linear interactions.*

Description

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

Usage

```
## 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, ...)
```

Arguments

model	A fitted linear model of type 'glm'.
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).
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.
errorbars	A string indicating what kind of error bars to show. Acceptable values are "CI" (95 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 an empty label.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

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.glm and pass variable names in directly instead of strings of variable names. graph_model_q.glm uses standard evaluation in cases where such evaluation is easier.

Value

A ggplot object of the plotted variables in the model.

See Also

graph_model.glm, graph_model_q.lm

Examples

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

---

graph_model_q.lm *Graph linear interactions.*

Description

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

Usage

```
## 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, ...)
```

Arguments

model	A fitted linear model of type 'lm'.
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 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 an empty label.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

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.lm` and pass variable names in directly instead of strings of variable names. `graph_model_q.lm` uses standard evaluation in cases where such evaluation is easier.

Value

A ggplot object of the plotted variables in the model.

See Also

`graph_model.lm`

Examples

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

`graph_model_q.lme` *Graph multi-level model interactions.*

Description

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

Usage

```
## 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, ...)
```

Arguments

`model`	A fitted linear model of type 'lme'.
`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 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 an empty label.
`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.
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function only deals with fixed effects of the model, after the random effects have been accounted for. To look at the lower-level effects, consider using `predict.lme` to generate predicted values at the desired level.

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.lme` and pass variable names in directly instead of strings of variable names. `graph_model_q.lme` uses standard evaluation in cases where such evaluation is easier.

Value

A ggplot2 graph of the plotted variables in the model.

See Also

`graph_model.lme`, `graph_model_q.merMod`, `graph_model_q.lm`

Examples

```
# 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')
}
```

---

graph_model_q.merMod *Graph multi-level model interactions.*

Description

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

Usage

```
## 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 'merMod' (from the 'lme4' package).
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 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 an empty label.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function only deals with fixed effects of the model, after the random effects have been accounted for. To look at the lower-level effects, consider using `predict.merMod` to generate predicted values at the desired level.

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.merMod and pass variable names in directly instead of strings of variable names. graph_model_q.merMod uses standard evaluation in cases where such evaluation is easier.

Value

A ggplot2 graph of the plotted variables in the model.

See Also

graph_model.merMod, graph_model_q.lme, graph_model_q.lm

Examples

```
# 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 is a generic function for calculating the intra-class correlation (ICC) for a fitted model.

Usage

```
ICC(model, ...)
```

Arguments

model	A fitted linear model of type 'lme' (nlme) or 'merMod' (lme4).
...	Additional arguments to be passed to the particular method for the given model.

Value

The form of the value returned by ICC depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

ICC.lme, ICC.merMod

Examples

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

`ICC.lme` *Intra-class correlation.*

Description

`ICC.lme` calculates the intra-class correlation (ICC) from a fitted multi-level model using the 'nlme' package.

Usage

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

Arguments

`model`	A fitted model of type 'lme'.
`...`	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.

See Also

`ICC.merMod`

Examples

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

ICC.merMod *Intra-class correlation.*

Description

ICC.merMod calculates the intra-class correlation (ICC) from a fitted multi-level model using the 'lme4' package.

Usage

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

Arguments

model	A fitted model of type 'merMod' (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.

See Also

ICC.lme

Examples

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

print.block_aov_summary *Summarizing block ANOVA.*

Description

print method for class "block_aov_summary".

Usage

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

Arguments

x	An object of class "block_aov_summary", usually, a result of a call to summary.block_aov.
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

build_model, summary.block_aov

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)
```

print.block_glm_summary *Summarizing block general linear models.*

Description

print method for class "block_glm_summary".

Usage

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

Arguments

x	An object of class "block_glm_summary", usually, a result of a call to summary.block_glm.
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

build_model, summary.block_glm

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)
```

---

print.block_lm_summary

Summarizing block regression.

Description

print method for class "block_lm_summary".

Usage

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

Arguments

x	An object of class "block_lm_summary", usually, a result of a call to 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.
...	Further arguments passed to or from other methods.

See Also

build_model, summary.block_lm

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)
```

---

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

print.simple_slopes_lme4 *Print simple slopes.*

Description

print method for class "simple_slopes_lme4".

Usage

```
## S3 method for class 'simple_slopes_lme4'
print(x, digits = max(3L, getOption("digits") -
  3L), ...)
```

Arguments

x	An object of class "simple_slopes_lme4", usually, a result of a call to simple_slopes.
digits	The number of significant digits to use when printing.
...	Further arguments passed to or from other methods.

See Also

simple_slopes

reghelper *reghelper: A package to help with running regression analyses.*

Description

The reghelper package offers numerous functions to make some aspects of regression models (and similar types of modelling) simpler.

Details

The following methods are currently implemented:

beta	Calculates standardized beta coefficients.
build_model	Allows variables to be added to a series of regression models sequentially.
ICC	Calculates the intra-class correlation for a multi-level model.
cell_means	Calculate the estimated means for a fitted model.
graph_model	Easily graph interactions at +/- 1 SD (uses ggplot2 package).
simple_slopes	Easily calculate the simple effects of an interaction.
sig_regions	Calculate the Johnson-Neyman regions of significance for an interaction.

`residuals.block_aov` *Extract model residuals.*

Description

`residuals` method for class "`block_aov`".

Usage

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

Arguments

`object`	An object of class "`block_aov`", usually, a result of a call to `build_model`.
`num`	Numeric vector with the index of model(s) from which to return the residuals.
`...`	Further arguments passed to or from other methods.

Value

The residuals of block(s) ‘num‘, or if ‘num‘ is NULL, a list of residuals from all blocks.

See Also

`build_model`, `fitted.block_aov`, `coef.block_aov`

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
residuals(model1)  # returns 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)
residuals(model2, num=2)  # returns second block
```

`residuals.block_glm` *Extract model residuals.*

Description

`residuals` method for class "`block_glm`".

Usage

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

Arguments

`object`	An object of class "`block_glm`", usually, a result of a call to `build_model`.
`num`	Numeric vector with the index of model(s) from which to return the residuals.
`...`	Further arguments passed to or from other methods.

Value

The residuals of block(s) ‘num‘, or if ‘num‘ is NULL, a list of residuals from all blocks.

See Also

`build_model`, `residuals.block_glm_summary`, `fitted.block_glm`, `coef.block_glm`

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
residuals(model1)  # returns 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)
residuals(model2, num=2)  # returns second block
```

`residuals.block_glm_summary` *Extract model residuals.*

Description

`residuals` method for class "`block_glm_summary`".

Usage

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

Arguments

`object`	An object of class "`block_glm_summary`", usually, a result of a call to `summary.block_glm`.
`num`	Numeric vector with the index of model(s) from which to return the residuals.
`...`	Further arguments passed to or from other methods.

Value

The residuals of block(s) ‘num‘, or if ‘num‘ is NULL, a list of residuals from all blocks.

See Also

`summary.block_lm`, `residuals.block_lm`, `coef.block_lm_summary`

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
residuals(summary(model1))  # returns 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')
residuals(summary(model2), num=2)  # returns second block
```

`residuals.block_lm` *Extract model residuals.*

Description

`residuals` method for class "`block_lm`".

Usage

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

Arguments

`object`	An object of class "`block_lm`", usually, a result of a call to `build_model`.
`num`	Numeric vector with the index of model(s) from which to return the residuals.
`...`	Further arguments passed to or from other methods.

Value

The residuals of block(s) ‘num‘, or if ‘num‘ is NULL, a list of residuals from all blocks.

See Also

`build_model`, `residuals.block_lm_summary`, `fitted.block_lm`, `coef.block_lm`

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
summary(model1)
residuals(model1)  # returns 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)
residuals(model2, num=2)  # returns second block
```

`residuals.block_lm_summary` *Extract model residuals.*

Description

`residuals` method for class "`block_lm_summary`".

Usage

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

Arguments

`object`	An object of class "`block_lm_summary`", usually, a result of a call to `summary.block_lm`.
`num`	Numeric vector with the index of model(s) from which to return the residuals.
`...`	Further arguments passed to or from other methods.

Value

The residuals of block(s) ‘num‘, or if ‘num‘ is NULL, a list of residuals from all blocks.

See Also

`summary.block_lm`, `residuals.block_lm`, `coef.block_lm_summary`

Examples

```
# 2 blocks: Petal.Length; Petal.Length + Petal.Width
model1 <- build_model(Sepal.Length, Petal.Length, Petal.Width, data=iris, model='lm')
residuals(summary(model1))  # returns 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')
residuals(summary(model2), num=2)  # returns second block
```

sig_regions *Regions of significance for an interaction.*

Description

sig_regions is a generic function for calculating the Johnson-Neyman 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, ...)
```

Arguments

model	A fitted linear model of type 'lm'.
...	Additional arguments to be passed to the particular method for the given model.

Value

The form of the value returned by sig_regions depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

sig_regions.lm, 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)
```

sig_regions.lm *Regions of significance for an interaction.*

Description

sig_regions.lm 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

```
## 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' with one two-way interaction including one categorical predictor and one continuous variable.
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).
...	Not currently implemented; used to ensure consistency with S3 generic.

Details

This function takes a regression 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.

See Also

simple_slopes.lm

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 interaction.*

Description

simple_slopes is a generic function for calculating the simple effects of an interaction in a regression model.

Usage

```
simple_slopes(model, ...)
```

Arguments

model	A fitted linear model of type 'lm', 'glm', 'aov', 'lme' (nlme), or 'merMod' (lme4).
...	Additional arguments to be passed to the particular method for the given model.

Value

The form of the value returned by simple_slopes depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

simple_slopes.lm, simple_slopes.glm, simple_slopes.aov, simple_slopes.lme, simple_slopes.merMod

Examples

```
# mtcars data
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
```

simple_slopes.glm *Simple slopes of interaction.*

Description

simple_slopes.glm calculates all the simple effects of an interaction in a regression model.

Usage

```
## S3 method for class 'glm'
simple_slopes(model, levels = NULL, ...)
```

Arguments

model	A fitted linear model of type 'glm' with at least one interaction term.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

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 -1SD, the mean, and +1SD 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 order is in the same order shown in `contrasts()`.

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 this was the variable 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.

See Also

`simple_slopes.lm`, `simple_slopes.lme`, `simple_slopes.merMod`

Examples

```
# mtcars data
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
```

`simple_slopes.lm` *Simple slopes of interaction.*

Description

`simple_slopes.lm` calculates all the simple effects of an interaction in a regression model.

Usage

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

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

Arguments

`model`	A fitted linear model of type 'lm' with at least one interaction term.
`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.
`...`	Not currently implemented; used to ensure consistency with S3 generic.

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 -1SD, the mean, and +1SD 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 order is in the same order shown in `contrasts()`.

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 this was the variable 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.

See Also

simple_slopes.glm,simple_slopes.lme, simple_slopes.merMod

Examples

```
# mtcars data
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
```

---

simple_slopes.lme *Simple slopes of interaction.*

Description

simple_slopes.lme calculates all the simple effects of an interaction in a regression model.

Usage

```
## S3 method for class 'lme'
simple_slopes(model, levels = NULL, ...)
```

Arguments

model	A fitted linear model of type 'lme' with at least one interaction term.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

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 -1SD, the mean, and +1SD 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 order is in the same order shown in contrasts().

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 this was the variable 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.

See Also

simple_slopes.lm, simple_slopes.glm, simple_slopes.merMod

Examples

```
# iris data
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
}
```

---

simple_slopes.merMod *Simple slopes of interaction.*

Description

simple_slopes.merMod calculates all the simple effects of an interaction in a regression model.

Usage

```
## S3 method for class 'merMod'
simple_slopes(model, levels = NULL, ...)
```

Arguments

model	A fitted linear model of type 'merMod' with at least one interaction term.
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.
...	Not currently implemented; used to ensure consistency with S3 generic.

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 -1SD, the mean, and +1SD 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 order is in the same order shown in `contrasts()`.

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 this was the variable being tested. After columns for each variable, the data frame has columns for the slope of the test variable, the standard error, and t-value for the model.

See Also

`simple_slopes.lm`, `simple_slopes.glm`, `simple_slopes.lme`

Examples

```
# iris data
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_aov` *Summarizing block ANOVA.*

Description

`summary` method for class "`block_aov`".

Usage

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

Arguments

object	An object of class "block_aov", usually, a result of a call to build_model.
...	Further arguments passed to or from other methods.

Value

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

formulas	A list of the aov formulas used for each block.
residuals	A matrix with quantiles of the residuals for each model.
summaries	A list with an ANOVA table for each model, including the sums of squares, mean squares, F values, and p-value

See Also

build_model, print.block_aov_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)

# 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)
```

---

summary.block_glm *Summarizing block general linear models.*

Description

summary method for class "block_glm".

Usage

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

Arguments

object	An object of class "block_glm", usually, a result of a call to build_model.
...	Further arguments passed to or from other methods.

Value

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

`formulas`	A list of the regression formulas used for each block.
`family`	The link function used in the model fit.
`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-

See Also

`build_model`, `print.block_glm_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)

# 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)
```

---

summary.block_lm *Summarizing block regression.*

Description

`summary` method for class "`block_lm`".

Usage

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

Arguments

`object`	An object of class "`block_lm`", usually, a result of a call to `build_model`.
`...`	Further arguments passed to or from other methods.

Value

The function computes and returns a named list of summary statistics of the fitted linear models given in `model`. 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-

See Also

build_model, print.block_lm_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)

# 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)
```

Index