Package ‘insight’

February 2, 2020

**Type** Package

**Title** Easy Access to Model Information for Various Model Objects

**Description** A tool to provide an easy, intuitive and consistent access to information contained in various R models, like model formulas, model terms, information about random effects, data that was used to fit the model or data from response variables. 'insight' mainly revolves around two types of functions: Functions that find (the names of) information, starting with 'find_'. and functions that get the underlying data, starting with 'get_'. The package has a consistent syntax and works with many different model objects, where otherwise functions to access these information are missing.

**Version** 0.8.1

**Maintainer** Daniel Lüdecke <d.luedecke@uke.de>

**License** GPL-3

**URL** [https://easystats.github.io/insight/](https://easystats.github.io/insight/)

**BugReports** [https://github.com/easystats/insight/issues](https://github.com/easystats/insight/issues)

**Depends** R (>= 3.2)

**Imports** methods, stats, utils

**Suggests** AER, afex, aod, BayesFactor, bayestestR, betareg, biglm, blavaan, blme, brms, censReg, egam, coxme, cplm, crch, estimatr, Feisr, fixest, gam, gamm4, gamlss, gbm, gee, geepack, GLMMadaptive, glmmTMB, gmlnl, HRQoL, httr, lavaan, lfe, logistf, MASS, Matrix, MCMCglmm, mlogit, multgee, lme4, mgcv, nnet, nlmme, ordinal, panelr, plm, pscl, quantreg, rms, robustbase, robustlmm, rstanarm, rstudioapi, speedglm, splines, statmod, survey, survival, tripack, truncreg, testthat, VGAM, knitr, markdown, spelling

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.0.2

**VignetteBuilder** knitr

**Language** en-US
NeedCompilation  no

Author  Daniel Lüdecke [aut, cre] (<https://orcid.org/0000-0002-8895-3206>),
        Dominique Makowski [aut, ctb] (<https://orcid.org/0000-0001-5375-9967>),
        Indrajeet Patil [aut, ctb] (<https://orcid.org/0000-0003-1995-6531>),
        Philip Waggoner [aut, ctb] (<https://orcid.org/0000-0002-7825-7573>)

Repository  CRAN

Date/Publication  2020-02-02 17:10:06 UTC

R topics documented:

all_models_equal ................................. 3
clean_names ....................................... 4
clean_parameters ................................. 5
color_if .......................................... 6
download_model .................................. 7
find_algorithm .................................... 8
find_formula ...................................... 9
find_interactions ................................. 10
find_parameters .................................. 11
find_predictors .................................. 14
find_random ...................................... 15
find_random_slopes ............................... 16
find_response ..................................... 17
find_statistic .................................... 17
find_terms ........................................ 18
find_variables .................................... 19
find_weights ...................................... 21
format_ci ......................................... 21
format_table ...................................... 22
format_value ...................................... 23
get_data .......................................... 24
get_parameters .................................... 27
get_predictors .................................... 30
get_priors ........................................ 31
get_random ........................................ 32
get_response ...................................... 32
get_statistic ...................................... 33
get_varcov ........................................ 34
get_variance ....................................... 36
get_weights ........................................ 39
has_intercept ..................................... 39
is_model .......................................... 40
is_model_supported ............................... 41
is_multivariate ................................... 41
is_nullmodel ...................................... 42
link_function ..................................... 43
link_inverse ....................................... 44
all_models_equal

Checks if all objects are models of same class

Description

Small helper that checks if all objects are supported (regression) model objects and of same class.

Usage

all_models_equal(..., verbose = FALSE)

all_models_same_class(..., verbose = FALSE)

Arguments

... A list of objects.
verbose Toggle off warnings.

Value

A logical, TRUE if x are all supported model objects of same class.

Examples

library(lme4)
data(mtcars)
data(sleepstudy)
m1 <- lm(mpg ~ wt + cyl + vs, data = mtcars)
m2 <- lm(mpg ~ wt + cyl, data = mtcars)
m3 <- lmer(Reaction ~ Days + (1 | Subject), data = sleepstudy)
m4 <- glm(formula = vs ~ wt, family = binomial(), data = mtcars)

all_models_same_class(m1, m2)
all_models_same_class(m1, m2, m3)
all_models_same_class(m1, m4, m2, m3, verbose = TRUE)
all_models_same_class(m1, m4, mtcars, m2, m3, verbose = TRUE)
clean_names

Get clean names of model terms

Description

This function "cleans" names of model terms (or a character vector with such names) by removing patterns like log() or as.factor() etc.

Usage

clean_names(x)

Arguments

x

A fitted model, or a character vector.

Value

The "cleaned" variable names as character vector, i.e. pattern like s() for splines or log() are removed from the model terms.

Note

Typically, this method is intended to work on character vectors, in order to remove patterns that obscure the variable names. For convenience reasons it is also possible to call clean_names() also on a model object. If x is a regression model, this function is (almost) equal to calling find_variables(). The main difference is that clean_names() always returns a character vector, while find_variables() returns a list of character vectors, unless flatten = TRUE. See 'Examples'.

Examples

# example from ?stats: glm
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- c(gl(3, 1, 9))
treatment <- gl(3, 3)
m <- glm(counts ~ log(outcome) + as.factor(treatment), family = poisson())
clean_names(m)

# difference "clean_names()" and "find_variables()"
library(lme4)
m <- glmer(
  cbind(incidence, size - incidence) ~ period + (1 | herd),
  data = cbpp,
  family = binomial
)
clean_names(m)
find_variables(m)
find_variables(m, flatten = TRUE)
clean_parameters

Get clean names of model parameters

Description
This function "cleans" names of model parameters by removing patterns like "r_" or "b[" (mostly applicable to Stan models) and adding columns with information to which group or component parameters belong (i.e. fixed or random, count or zero-inflated...)

The main purpose of this function is to easily filter and select model parameters, in particular of - but not limited to - posterior samples from Stan models, depending on certain characteristics. This might be useful when only selective results should be reported or results from all parameters should be filtered to return only certain results (see print_parameters).

Usage

clean_parameters(x, ...)

Arguments

x A fitted model.
...
Currently not used.

Details

The Effects column indicate if a parameter is a fixed or random effect. The Component can either be conditional or zero_inflated. For models with random effects, the Group column indicates the grouping factor of the random effects. For multivariate response models from brms or rstanarm, an additional Response column is included, to indicate which parameters belong to which response formula. Furthermore, Cleaned_Parameter column is returned that contains "human readable" parameter names (which are mostly identical to Parameter, except for for models from brms or rstanarm, or for specific terms like smooth- or spline-terms).

Value

A data frame with "cleaned" parameter names and information on effects, component and group where parameters belong to. To be consistent across different models, the returned data frame always has at least four columns Parameter, Effects, Component and Cleaned_Parameter. See 'Details'.

Examples

```r
## Not run:
library(brms)
model <- download_model("brms_zi_2")
clean_parameters(model)
## End(Not run)
```
color_if  

Color-formatting for data columns based on condition

Description

Convenient function that formats columns in data frames with color codes, where the color is chosen based on certain conditions. Columns are then printed in color in the console.

Usage

```r
color_if(
  x,
  columns,
  predicate = `>`,
  value = 0,
  color_if = "green",
  color_else = "red",
  digits = 2
)
```

```r
colour_if(
  x,
  columns,
  predicate = `>`,
  value = 0,
  colour_if = "green",
  colour_else = "red",
  digits = 2
)
```

Arguments

- **x**  
  A data frame
- **columns**  
  Character vector with column names of x that should be formatted.
- **predicate**  
  A function that takes columns and value as input and which should return TRUE or FALSE, based on if the condition (in comparison with value) is met.
- **value**  
  The comparator. May be used in conjunction with predicate to quickly set up a function which compares elements in columns to value. May be ignored when predicate is a function that internally computes other comparisons. See 'Examples'.
- **color_if, colour_if**  
  Character vector, indicating the color code used to format values in x that meet the condition of predicate and value. May be one of "red", "yellow", "green", "blue", "violet", "cyan" or "grey". Formatting is also possible with "bold" or "italic".
**Download circus models**

**Description**

Downloads pre-compiled models from the circus-repository. The circus-repository contains a variety of fitted models to help the systematic testing of other packages.

**Usage**

```r
download_model(name, url = NULL)
```

**Arguments**

- `name`  Model name.
- `url`   String with the URL from where to download the model data. Optional, and should only be used in case the repository-URL is changing. By default, models are downloaded from https://raw.githubusercontent.com/easystats/circus/master/data/.
Details

The code that generated the model is available at the https://easystats.github.io/circus/reference/index.html.

Value

A model from the circus-repository.

References

https://easystats.github.io/circus/

find_algorithm | Find sampling algorithm and optimizers

Description

Returns information on the sampling or estimation algorithm as well as optimization functions, or for Bayesian model information on chains, iterations and warmup-samples.

Usage

find_algorithm(x, ...)

Arguments

x A fitted model.

... Currently not used.

Value

A list with elements depending on the model.

For frequentist models:

• algorithm, for instance "OLS" or "ML"
• optimizer, name of optimizing function, only applies to specific models (like gam)

For frequentist mixed models:

• algorithm, for instance "REML" or "ML"
• optimizer, name of optimizing function

For Bayesian models:

• algorithm, the algorithm
• chains, number of chains
• iterations, number of iterations per chain
• warmup, number of warmups per chain
**Examples**

```r
library(lme4)
data(sleepstudy)
m <- lmer(Reaction ~ Days + (1 | Subject), data = sleepstudy)
find_algorithm(m)
## Not run:
library(rstanarm)
m <- stan_lmer(Reaction ~ Days + (1 | Subject), data = sleepstudy)
find_algorithm(m)
## End(Not run)
```

---

**find_formula**

**Find model formula**

**Description**

Returns the formula(s) for the different parts of a model (like fixed or random effects, zero-inflated component, ...).

**Usage**

```r
find_formula(x, ...)
```

**Arguments**

- `x` A fitted model.
- `...` Currently not used.

**Value**

A list of formulas that describe the model. For simple models, only one list-element, `conditional`, is returned. For more complex models, the returned list may have following elements:

- `conditional`, the "fixed effects" part from the model. One exception are `DirichletRegModel` models from `DirichletReg`, which has two or three components, depending on model.
- `random`, the "random effects" part from the model (or the id for gee-models and similar)
- `zero_inflated`, the "fixed effects" part from the zero-inflation component of the model
- `zero_inflated_random`, the "random effects" part from the zero-inflation component of the model
- `dispersion`, the dispersion formula
- `instruments`, for fixed-effects regressions like `ivreg`, `felm` or `plm`, the instrumental variables
- `cluster`, for fixed-effects regressions like `felm`, the cluster specification
- `correlation`, for models with correlation-component like `gls`, the formula that describes the correlation structure
• slopes, for fixed-effects individual-slope models like feis, the formula for the slope parameters
• precision, for DirichletRegModel models from DirichletReg, when parametrization (i.e. model) is "alternative".

Note
For models of class lme or gls the correlation-component is only returned, when it is explicitly defined as named argument (form), e.g. corAR(form = ~1 | Mare)

Examples

data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
find_formula(m)

find_interactions

Find interaction terms from models

Description
Returns all lowest to highest order interaction terms from a model.

Usage

find_interactions(
  x,
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion", "instruments"),
  flatten = FALSE
)

Arguments

x A fitted model.

component Should all predictor variables, predictor variables for the conditional model, the zero-inflated part of the model, the dispersion term or the instrumental variables be returned? Applies to models with zero-inflated and/or dispersion formula, or to models with instrumental variable (so called fixed-effects regressions). May be abbreviated. Note that the conditional component is also called count or mean component, depending on the model.

flatten Logical, if TRUE, the values are returned as character vector, not as list. Duplicated values are removed.
find_parameters

Value
A list of character vectors that represent the interaction terms. Depending on component, the returned list has following elements (or NULL, if model has no interaction term):

- conditional, interaction terms that belong to the "fixed effects" terms from the model
- zero_inflated, interaction terms that belong to the "fixed effects" terms from the zero-inflation component of the model
- instruments, for fixed-effects regressions like ivreg, felm or plm, interaction terms that belong to the instrumental variables

Examples

```r
data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
find_interactions(m)
m <- lm(mpg ~ wt * cyl + vs * hp * gear + carb, data = mtcars)
find_interactions(m)
```

find_parameters

Find names of model parameters

Description
Returns the names of model parameters, like they typically appear in the summary() output. For Bayesian models, the parameter names equal the column names of the posterior samples after coercion from as.data.frame().

Usage

```r
find_parameters(x, ...)  
## S3 method for class 'gam'
find_parameters(
x,
  component = c("all", "conditional", "smooth_terms"),
  flatten = FALSE,
  ...
)

## S3 method for class 'merMod'
find_parameters(x, effects = c("all", "fixed", "random"), flatten = FALSE, ...)

## S3 method for class 'zeroinfl'
find_parameters(
x,
```
component = c("all", "conditional", "zi", "zero_inflated"),
flatten = FALSE,
...)

## S3 method for class 'hurdle'
find_parameters(
x,
  component = c("all", "conditional", "zi", "zero_inflated"),
  flatten = FALSE,
  ...
)

## S3 method for class 'BFBayesFactor'
find_parameters(
x,
  effects = c("all", "fixed", "random"),
  component = c("all", "extra"),
  flatten = FALSE,
  ...
)

## S3 method for class 'brmsfit'
find_parameters(
x,
  effects = c("all", "fixed", "random"),
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion", "simplex", "sigma", "smooth_terms"),
  flatten = FALSE,
  parameters = NULL,
  ...
)

## S3 method for class 'bayesx'
find_parameters(
x,
  component = c("all", "conditional", "smooth_terms"),
  flatten = FALSE,
  parameters = NULL,
  ...
)

## S3 method for class 'stanreg'
find_parameters(
x,
  effects = c("all", "fixed", "random"),
  component = c("all", "conditional", "smooth_terms"),
  flatten = FALSE,
### S3 method for class 'sim.merMod'

```r
find_parameters(
  x,
  effects = c("all", "fixed", "random"),
  flatten = FALSE,
  parameters = NULL,
  ...
)
```

### S3 method for class 'bayesx'

```r
get_parameters(
  x,
  component = c("all", "conditional", "smooth_terms"),
  flatten = FALSE,
  parameters = NULL,
  ...
)
```

#### Arguments

- **x**  
  A fitted model.
- **...**  
  Currently not used.
- **component**  
  Should all parameters, parameters for the conditional model, the zero-inflated part of the model, the dispersion term or the instrumental variables be returned? Applies to models with zero-inflated and/or dispersion formula, or to models with instrumental variable (so called fixed-effects regressions). May be abbreviated. Note that the **conditional** component is also called **count** or **mean** component, depending on the model.
- **flatten**  
  Logical, if TRUE, the values are returned as character vector, not as list. Duplicated values are removed.
- **effects**  
  Should parameters for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
- **parameters**  
  Regular expression pattern that describes the parameters that should be returned.

#### Details

In most cases when models either return different "effects" (fixed, random) or "components" (conditional, zero-inflated, ...), the arguments **effects** and **component** can be used. Not all model classes that support these arguments are listed here in the 'Usage' section.

#### Value

A list of parameter names. For simple models, only one list-element, **conditional**, is returned. For more complex models, the returned list may have following elements:
• conditional, the "fixed effects" part from the model
• random, the "random effects" part from the model
• zero_inflated, the "fixed effects" part from the zero-inflation component of the model
• zero_inflated_random, the "random effects" part from the zero-inflation component of the model
• dispersion, the dispersion parameters
• simplex, simplex parameters of monotonic effects (brms only)
• smooth_terms, the smooth parameters

Examples

data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
find_parameters(m)

Description

Returns the names of the predictor variables for the different parts of a model (like fixed or random effects, zero-inflated component, ...). Unlike find_parameters, the names from find_predictors() match the original variable names from the data that was used to fit the model.

Usage

find_predictors(
x,  
effects = c("fixed", "random", "all"),  
component = c("all", "conditional", "zi", "zero_inflated", "dispersion",  
             "instruments", "correlation", "smooth_terms"),  
flatten = FALSE
)

Arguments

x A fitted model.
effects Should variables for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
component Should all predictor variables, predictor variables for the conditional model, the zero-inflated part of the model, the dispersion term or the instrumental variables be returned? Applies to models with zero-inflated and/or dispersion formula, or to models with instrumental variable (so called fixed-effects regressions). May be abbreviated. Note that the conditional component is also called count or mean component, depending on the model.
flatten Logical, if TRUE, the values are returned as character vector, not as list. Duplicated values are removed.
find_random

Value
A list of character vectors that represent the name(s) of the predictor variables. Depending on the combination of the arguments effects and component, the returned list has following elements:

- conditional, the "fixed effects" terms from the model
- random, the "random effects" terms from the model
- zero_inflated, the "fixed effects" terms from the zero-inflation component of the model
- zero_inflated_random, the "random effects" terms from the zero-inflation component of the model
- dispersion, the dispersion terms
- instruments, for fixed-effects regressions like ivreg, felm or plm, the instrumental variables
- correlation, for models with correlation-component like gls, the variables used to describe the correlation structure

Examples
```r
data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
find_predictors(m)
```

Description
Return the name of the grouping factors from mixed effects models.

Usage
```r
find_random(x, split_nested = FALSE, flatten = FALSE)
```

Arguments

- `x` A fitted mixed model.
- `split_nested` Logical, if TRUE, terms from nested random effects will be returned as separated elements, not as single string with colon. See 'Examples'.
- `flatten` Logical, if TRUE, the values are returned as character vector, not as list. Duplicated values are removed.

Value
A list of character vectors that represent the name(s) of the random effects (grouping factors). Depending on the model, the returned list has following elements:

- random, the "random effects" terms from the conditional part of model
- zero_inflated_random, the "random effects" terms from the zero-inflation component of the model
Examples

```r
library(lme4)
data(sleepstudy)
sleepstudy$mygrp <- sample(1:5, size = 180, replace = TRUE)
sleepstudy$mysubgrp <- NA
for (i in 1:5) {
  filter_group <- sleepstudy$mygrp == i
  sleepstudy$mysubgrp[filter_group] <- sample(1:30, size = sum(filter_group), replace = TRUE)
}

m <- lmer(Reaction ~ Days + (1 | mygrp / mysubgrp) + (1 | Subject),
          data = sleepstudy)

find_random(m)
find_random(m, split_nested = TRUE)
```

find_random_slopes

Find names of random slopes

Description

Return the name of the random slopes from mixed effects models.

Usage

```r
find_random_slopes(x)
```

Arguments

- `x` A fitted mixed model.

Value

A list of character vectors with the name(s) of the random slopes, or NULL if model has no random slopes. Depending on the model, the returned list has following elements:

- `random`, the random slopes from the conditional part of model
- `zero_inflated_random`, the random slopes from the zero-inflation component of the model

Examples

```r
library(lme4)
data(sleepstudy)

m <- lmer(Reaction ~ Days + (1 + Days | Subject), data = sleepstudy)
find_random_slopes(m)
```
find_response  

Find name of the response variable

**Description**
Returns the name(s) of the response variable(s) from a model object.

**Usage**
find_response(x, combine = TRUE)

**Arguments**
- **x**
  - A fitted model.
- **combine**
  - Logical, if TRUE and the response is a matrix-column, the name of the response matches the notation in formula, and would for instance also contain patterns like "cbind(...)". Else, the original variable names from the matrix-column are returned. See 'Examples'.

**Value**
The name(s) of the response variable(s) from x as character vector.

**Examples**
```r
library(lme4)
data(cbpp)
cbpp$trials <- cbpp$size - cbpp$incidence
m <- glm(cbind(incidence, trials) ~ period, data = cbpp, family = binomial)
find_response(m, combine = TRUE)
find_response(m, combine = FALSE)
```

find_statistic  

Find statistic for model

**Description**
Returns the statistic for a regression model (t-statistic, z-statistic, etc.).
Small helper that checks if a model is a regression model object and return the statistic used.

**Usage**
find_statistic(x, ...)

**Examples**
```r
```
Arguments
x  An object.
... Currently not used.

Value
A character describing the type of statistic. If there is no statistic available with a distribution, NULL will be returned.

Examples
# regression model object
data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
find_statistic(m)

find_terms  Find all model terms

Description
Returns a list with the names of all terms, including response value and random effects, "as is". This means, on-the-fly transformations or arithmetic expressions like log(), I(), as.factor() etc. are preserved.

Usage
find_terms(x, flatten = FALSE, ...)

Arguments
x  A fitted model.
flatten Logical, if TRUE, the values are returned as character vector, not as list. Duplicated values are removed.
... Currently not used.

Value
A list with (depending on the model) following elements (character vectors):
- response, the name of the response variable
- conditional, the names of the predictor variables from the conditional model (as opposed to the zero-inflated part of a model)
- random, the names of the random effects (grouping factors)
- zero_inflated, the names of the predictor variables from the zero-inflated part of the model
- zero_inflated_random, the names of the random effects (grouping factors)
- dispersion, the name of the dispersion terms
- instruments, the names of instrumental variables
Note

The difference to `find_variables` is that `find_terms()` may return a variable multiple times in case of multiple transformations (see examples below), while `find_variables()` returns each variable name only once.

Examples

```r
library(lme4)
data(sleepstudy)
m <- lmer(
  log(Reaction) ~ Days + I(Days^2) + (1 + Days + exp(Days) | Subject),
  data = sleepstudy
)
find_terms(m)
```

---

**find_variables**  
*Find names of all variables*

Description

Returns a list with the names of all variables, including response value and random effects.

Usage

```r
find_variables(
  x, 
  effects = c("all", "fixed", "random"),
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion", 
               "instruments", "smooth_terms"),
  flatten = FALSE
)
```

Arguments

- `x` A fitted model.
- `effects` Should variables for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
- `component` Should all predictor variables, predictor variables for the conditional model, the zero-inflated part of the model, the dispersion term or the instrumental variables be returned? Applies to models with zero-inflated and/or dispersion formula, or to models with instrumental variable (so called fixed-effects regressions). May be abbreviated. Note that the `conditional` component is also called `count` or `mean` component, depending on the model.
- `flatten` Logical, if TRUE, the values are returned as character vector, not as list. Duplicated values are removed.
Value

A list with (depending on the model) following elements (character vectors):

- response, the name of the response variable
- conditional, the names of the predictor variables from the conditional model (as opposed to the zero-inflated part of a model)
- random, the names of the random effects (grouping factors)
- zero_inflated, the names of the predictor variables from the zero-inflated part of the model
- zero_inflated_random, the names of the random effects (grouping factors)
- dispersion, the name of the dispersion terms
- instruments, the names of instrumental variables

Note

The difference to `find_terms` is that `find_variables()` returns each variable name only once, while `find_terms()` may return a variable multiple times in case of transformations or when arithmetic expressions were used in the formula.

Examples

```r
library(lme4)
data(cbpp)
data(sleepstudy)
# some data preparation...
cbpp$s trials <- cbpp$ size - cbpp$ incidence
sleepstudy$mygrp <- sample(1:5, size = 180, replace = TRUE)
sleepstudy$mysubgrp <- NA
for (i in 1:5) {
  filter_group <- sleepstudy$mygrp == i
  sleepstudy$mysubgrp[filter_group] <-
    sample(1:30, size = sum(filter_group), replace = TRUE)
}

m1 <- glmer(
  cbind(incidence, size - incidence) ~ period + (1 | herd),
  data = cbpp,
  family = binomial
)
find_variables(m1)

m2 <- lmer(
  Reaction ~ Days + (1 | mygrp / mysubgrp) + (1 | Subject),
  data = sleepstudy
)
find_variables(m2)
find_variables(m2, flatten = TRUE)
```
find_weights

Find names of model weights

Description

Returns the name of the variable that describes the weights of a model.

Usage

find_weights(x, ...)

Arguments

x
A fitted model.

... Currently not used.

Value

The name of the weighting variable as character vector, or NULL if no weights were specified.

Examples

data(mtcars)
mtcars$weight <- rnorm(nrow(mtcars), 1, .3)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars, weights = weight)
find_weights(m)

format_ci

Confidence/Credible Interval (CI) Formatting

Description

Confidence/Credible Interval (CI) Formatting

Usage

format_ci(
  CI_low,
  CI_high,
  ci = 0.95,
  digits = 2,
  brackets = TRUE,
  width = NULL,
  width_low = width,
  width_high = width
)
Arguments

- **CI_low**: Lower CI bound.
- **CI_high**: Upper CI bound.
- **ci**: CI level in percentage.
- **digits**: Number of significant digits.
- **brackets**: Logical, if TRUE (default), values are encompassed in square brackets.
- **width**: Minimum width of the returned string. If not NULL and width is larger than the string’s length, leading whitespaces are added to the string. If width = "auto", width will be set to the length of the longest string.
- **width_low**, **width_high**: Like width, but only applies to the lower or higher confidence interval value. This can be used when the values for the lower and upper CI are of very different length.

Value

A formatted string.

Examples

```r
format_ci(1.20, 3.57, ci = 0.90)
format_ci(1.20, 3.57, ci = NULL)
format_ci(1.20, 3.57, ci = NULL, brackets = FALSE)
format_ci(c(1.205645, 23.4), c(3.57, -1.35), ci = 0.90)
format_ci(c(1.20, NA, NA), c(3.57, -1.35, NA), ci = 0.90)

# automatic alignment of width, useful for printing multiple CIs in columns
x <- format_ci(c(1.205, 23.4, 100.43), c(3.57, -13.35, 9.4))
cat(x, sep = "\n")

x <- format_ci(c(1.205, 23.4, 100.43), c(3.57, -13.35, 9.4), width = "auto")
cat(x, sep = "\n")
```

---

**Dataframe and Tables Pretty Formatting**

Description

Dataframe and Tables Pretty Formatting

Usage

```r
format_table(
  x,
  sep = " | ",
  header = "-",
)```
format_value

- digits = 2,
- protect_integers = TRUE,
- missing = "",
- width = NULL

Arguments

- x: A data frame.
- sep: Column separator.
- header: Header separator. Can be NULL.
- digits: Number of significant digits.
- protect_integers: Should integers be kept as integers (i.e., without decimals)?
- missing: Value by which NA values are replaced. By default, an empty string (i.e. "") is returned for NA.
- width: Minimum width of the returned string. If not NULL and width is larger than the string’s length, leading whitespaces are added to the string.

Value

A data frame in character format.

Examples

```r
cat(format_table(iris))
cat(format_table(iris, sep = " ", header = "x", digits = 1))
```

---

**Format Value**

Numeric Values Formatting

**Description**

Numeric Values Formatting

**Usage**

```r
format_value(
  x,
  digits = 2,
  protect_integers = FALSE,
  missing = "",
  width = NULL,
  as_percent = FALSE,
  ...
)
```
get_data

Arguments

- **x**: Numeric value.
- **digits**: Number of significant digits.
- **protect_integers**: Should integers be kept as integers (i.e., without decimals)?
- **missing**: Value by which NA values are replaced. By default, an empty string (i.e., "") is returned for NA.
- **width**: Minimum width of the returned string. If not NULL and width is larger than the string’s length, leading whitespaces are added to the string.
- **as_percent**: Logical, if TRUE, value is formatted as percentage value.
- **...**: Arguments passed to or from other methods.

Value

A formatted string.

Examples

```r
format_value(1.20)
format_value(1.2)
format_value(1.2012313)
format_value(c(0.0045, 234, -23))
format_value(c(0.0045, .12, .34))
format_value(c(0.0045, .12, .34), as_percent = TRUE)

format_value(as.factor(c("A", "B", "A")))
format_value(iris$Species)

format_value(3)
format_value(3, protect_integers = TRUE)

format_value(iris)
```

---

get_data *Get the data that was used to fit the model*

Description

This function tries to get the data that was used to fit the model and returns it as data frame.

Usage

```r
get_data(x, ...)
```

```r
## S3 method for class 'gee'
get_data(x, effects = c("all", "fixed", "random"), ...)
```
## S3 method for class 'rqss'
get_data(x, component = c("all", "conditional", "smooth_terms"), ...)  

## S3 method for class 'hurdle'
get_data(
  x,
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion"),
  ...
)

## S3 method for class 'glmmTMB'
get_data(
  x,
  effects = c("all", "fixed", "random"),
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion"),
  ...
)

## S3 method for class 'merMod'
get_data(x, effects = c("all", "fixed", "random"), ...)  

## S3 method for class 'glmmadmb'
get_data(x, effects = c("all", "fixed", "random"), ...)  

## S3 method for class 'rlmerMod'
get_data(x, effects = c("all", "fixed", "random"), ...)  

## S3 method for class 'clmm'
get_data(x, effects = c("all", "fixed", "random"), ...)  

## S3 method for class 'mixed'
get_data(x, effects = c("all", "fixed", "random"), ...)  

## S3 method for class 'lme'
get_data(x, effects = c("all", "fixed", "random"), ...)  

## S3 method for class 'MixMod'
get_data(
  x,
  effects = c("all", "fixed", "random"),
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion"),
  ...
)

## S3 method for class 'brmsfit'
get_data(
  x,
get_data

effects = c("all", "fixed", "random"),
component = c("all", "conditional", "zi", "zero_inflated"),
...
)

## S3 method for class 'stanreg'
get_data(x, effects = c("all", "fixed", "random"), ...)

## S3 method for class 'MCMCglmm'
get_data(x, effects = c("all", "fixed", "random"), ...)

Arguments

x A fitted model.
...
Currently not used.
effects Should model data for fixed effects, random effects or both be returned? Only
applies to mixed models.
component Should all predictor variables, predictor variables for the conditional model, the
zero-inflated part of the model, the dispersion term or the instrumental variables
be returned? Applies to models with zero-inflated and/or dispersion formula, or
to models with instrumental variable (so called fixed-effects regressions). May
be abbreviated. Note that the conditional component is also called count or
mean component, depending on the model.

Value

The data that was used to fit the model.

Note

Unlike model.frame(), which may contain transformed variables (e.g. if poly() or scale() was
used inside the formula to specify the model), get_data() aims at returning the "original", untrans-
formed data.

Examples

data(cbpp, package = "lme4")
cbpp$trials <- cbpp$size - cbpp$incidence
m <- glm(cbind(incidence, trials) ~ period, data = cbpp, family = binomial)
head(get_data(m))
get_parameters  Get model parameters

Description

Returns the coefficients (or posterior samples for Bayesian models) from a model.

Usage

get_parameters(x, ...)

## S3 method for class 'rqss'
get_parameters(x, component = c("all", "conditional", "smooth_terms"), ...)

## S3 method for class 'cgam'
get_parameters(x, component = c("all", "conditional", "smooth_terms"), ...)

## S3 method for class 'betareg'
get_parameters(x, component = c("all", "conditional", "precision"), ...)

## S3 method for class 'DirichletRegModel'
get_parameters(x, component = c("all", "conditional", "precision"), ...)

## S3 method for class 'clm2'
get_parameters(x, component = c("all", "conditional", "scale"), ...)

## S3 method for class 'coxme'
get_parameters(x, effects = c("fixed", "random"), ...)

## S3 method for class 'merMod'
get_parameters(x, effects = c("fixed", "random"), ...)

## S3 method for class 'lme'
get_parameters(x, effects = c("fixed", "random"), ...)

## S3 method for class 'mixed'
get_parameters(x, effects = c("fixed", "random"), ...)

## S3 method for class 'MixMod'
get_parameters(
    x,
    effects = c("fixed", "random"),
    component = c("all", "conditional", "zi", "zero_inflated", "dispersion"),
    ...
)

## S3 method for class 'glmmTMB'
get_parameters(
  x,
  effects = c("fixed", "random"),
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion"),
  ...
)

## S3 method for class 'BBmm'
get_parameters(x, effects = c("fixed", "random"), ...)

## S3 method for class 'glimML'
get_parameters(x, effects = c("fixed", "random", "all"), ...)

## S3 method for class 'gamm'
get_parameters(x, component = c("all", "conditional", "smooth_terms"), ...)

## S3 method for class 'Gam'
get_parameters(x, component = c("all", "conditional", "smooth_terms"), ...)

## S3 method for class 'gam'
get_parameters(x, component = c("all", "conditional", "smooth_terms"), ...)

## S3 method for class 'vgam'
get_parameters(x, component = c("all", "conditional", "smooth_terms"), ...)

## S3 method for class 'zeroinfl'
get_parameters(
  x,
  component = c("all", "conditional", "zi", "zero_inflated"),
  ...
)

## S3 method for class 'aovlist'
get_parameters(x, effects = c("fixed", "random", "all"), ...)

## S3 method for class 'MCMCglmm'
get_parameters(x, effects = c("fixed", "random", "all"), ...)

## S3 method for class 'BFBayesFactor'
get_parameters(
  x,
  effects = c("all", "fixed", "random"),
  component = c("all", "extra"),
  iterations = 4000,
  progress = FALSE,
  ...
)
get_parameters

## S3 method for class 'stanmvreg'
get_parameters(
  x,
  effects = c("fixed", "random", "all"),
  parameters = NULL,
  ...
)

## S3 method for class 'brmsfit'
get_parameters(
  x,
  effects = c("fixed", "random", "all"),
  component = c("all", "conditional", "zi", "zero_inflated", "dispersion", "simplex", "sigma", "smooth_terms"),
  parameters = NULL,
  ...
)

## S3 method for class 'stanreg'
get_parameters(
  x,
  effects = c("fixed", "random", "all"),
  parameters = NULL,
  ...
)

## S3 method for class 'sim.merMod'
get_parameters(
  x,
  effects = c("fixed", "random", "all"),
  parameters = NULL,
  ...
)

### Arguments

- **x**: A fitted model.
- **...**: Currently not used.
- **component**: Should all parameters, parameters for the conditional model, the zero-inflated part of the model, the dispersion term or the instrumental variables be returned? Applies to models with zero-inflated and/or dispersion formula, or to models with instrumental variable (so called fixed-effects regressions). May be abbreviated. Note that the conditional component is also called count or mean component, depending on the model.
- **effects**: Should parameters for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
- **iterations**: Number of posterior draws.
progress Display progress.
parameters Regular expression pattern that describes the parameters that should be returned.

Details

In most cases when models either return different "effects" (fixed, random) or "components" (conditional, zero-inflated, ...), the arguments effects and component can be used.

get_parameters() is comparable to coef(), however, the coefficients are returned as data frame (with columns for names and point estimates of coefficients). For Bayesian models, the posterior samples of parameters are returned.

Value

- for non-Bayesian models and if effects = "fixed", a data frame with two columns: the parameter names and the related point estimates
- if effects = "random", a list of data frames with the random effects (as returned by ranef()), unless the random effects have the same simplified structure as fixed effects (e.g. for models from MCMCglmm)
- for Bayesian models, the posterior samples from the requested parameters as data frame
- for Anova (aov()) with error term, a list of parameters for the conditional and the random effects parameters
- for models with smooth terms or zero-inflation component, a data frame with three columns: the parameter names, the related point estimates and the component

Examples

data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
get_parameters(m)

Description

Returns the data from all predictor variables (fixed effects).

Usage

get_predictors(x)

Arguments

x A fitted model.
get_priors

Value

The data from all predictor variables, as data frame.

Examples

```r
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
head(get_predictors(m))
```

---

get_priors

Get summary of priors used for a model

Description

Provides a summary of the prior distributions used for the parameters in a given model.

Usage

```r
get_priors(x, ...)
```

Arguments

- `x` A Bayesian model.
- `...` Currently not used.

Value

A data frame with a summary of the prior distributions used for the parameters in a given model.

Examples

```r
## Not run:
library(rstanarm)
model <- stan_glm(Sepal.Width ~ Species * Petal.Length, data = iris)
get_priors(model)
## End(Not run)
```
get_random

Get the data from random effects

Description
Returns the data from all random effects terms.

Usage
get_random(x)

Arguments
x A fitted mixed model.

Value
The data from all random effects terms, as data frame. Or NULL if model has no random effects.

Examples
library(lme4)
data(sleepstudy)
# prepare some data...
sleepstudy$mygrp <- sample(1:5, size = 180, replace = TRUE)
sleepstudy$mysubgrp <- NA
for (i in 1:5) {
  filter_group <- sleepstudy$mygrp == i
  sleepstudy$mysubgrp[filter_group] <-
    sample(1:30, size = sum(filter_group), replace = TRUE)
}

m <- lmer(Reaction ~ Days + (1 | mygrp / mysubgrp) + (1 | Subject),
  data = sleepstudy)

head(get_random(m))

get_response

Get the values from the response variable

Description
Returns the values the response variable(s) from a model object. If the model is a multivariate
response model, a data frame with values from all response variables is returned.
get_statistic

Usage

get_response(x, select = NULL)

Arguments

x A fitted model.

select Optional name(s) of response variables for which to extract values. Can be used in case of regression models with multiple response variables.

Value

The values of the response variable, as vector, or a data frame if x has more than one defined response variable.

Examples

library(lme4)
data(cbpp)
data(mtcars)
cbpp$trials <- cbpp$size - cbpp$incidence

m <- glm(cbind(incidence, trials) ~ period, data = cbpp, family = binomial)
head(get_response(m))
get_response(m, select = "incidence")

m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
get_response(m)

get_statistic

Get statistic associated with estimates

Description

Returns the statistic (t, z, ...) for model estimates. In most cases, this is the related column from coef(summary()).

Usage

get_statistic(x, ...)

## Default S3 method:
get_statistic(x, column_index = 3, ...)

## S3 method for class 'glmmTMB'
get_statistic(
x,
component = c("all", "conditional", "zi", "zero_inflated"),
...
get_varcov

## S3 method for class 'clm2'
get_statistic(x, component = c("all", "conditional", "scale"), ...)

## S3 method for class 'gee'
get_statistic(x, robust = FALSE, ...)

## S3 method for class 'betareg'
get_statistic(x, component = c("all", "conditional", "precision"), ...)

## S3 method for class 'DirichletRegModel'
get_statistic(x, component = c("all", "conditional", "precision"), ...)

Arguments

x A model.

... Currently not used.

column_index For model objects that have no defined get_statistic() method yet, the de-
default method is called. This method tries to extract the statistic column from
col summaries), where the index of the column that is being pulled is column_index. Defaults to 3, which is the default statistic column for most models’ summary-
output.

component Should all parameters, parameters for the conditional model, or for the zero-
inflated part of the model be returned? Applies to models with zero-inflated
component. component may be one of "conditional", "zi", "zero-inflated" or "all" (default). For models with smooth terms, component = "smooth_terms" is also possible. May be abbreviated. Note that the conditional component is also called count or mean component, depending on the model.

robust Logical, if TRUE, test statistic based on robust standard errors is returned.

Value

A data frame with the model’s parameter names and the related test statistic.

Examples

data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
get_statistic(m)

get_varcov Get variance-covariance matrix from models

Description

Returns the variance-covariance, as retrieved by stats::vcov(), but works for more model objects that probably don’t provide a vcov()-method.
Usage

\texttt{get_varcov(x, \ldots)}

---

\texttt{## S3 method for class \textquote{\texttt{betareg}}}
\texttt{get_varcov(x, component = c("conditional", \textquote{\texttt{precision}}, \textquote{\texttt{all}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{DirichletRegModel}}}
\texttt{get_varcov(x, component = c("conditional", \textquote{\texttt{precision}}, \textquote{\texttt{all}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{clm2}}}
\texttt{get_varcov(x, component = c("\textquote{\texttt{all}}", \textquote{\texttt{conditional}}, \textquote{\texttt{scale}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{truncreg}}}
\texttt{get_varcov(x, component = c("\textquote{\texttt{all}}", \textquote{\texttt{conditional}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{gamlss}}}
\texttt{get_varcov(x, component = c("\textquote{\texttt{conditional}}", \textquote{\texttt{all}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{hurdle}}}
\texttt{get_varcov(x, component = c("\textquote{\texttt{conditional}}", \textquote{\texttt{zero-inflated}}, \textquote{\texttt{zi}}, \textquote{\texttt{all}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{MixMod}}}
\texttt{get_varcov(x, component = c("\textquote{\texttt{conditional}}", \textquote{\texttt{zero-inflated}}, \textquote{\texttt{zi}}, \textquote{\texttt{all}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{glmmTMB}}}
\texttt{get_varcov(x, component = c("\textquote{\texttt{conditional}}", \textquote{\texttt{zero-inflated}}, \textquote{\texttt{zi}}, \textquote{\texttt{all}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{brmsfit}}}
\texttt{get_varcov(x, component = c("\textquote{\texttt{conditional}}", \textquote{\texttt{zero-inflated}}, \textquote{\texttt{zi}}, \textquote{\texttt{all}}), \ldots)}

\texttt{## S3 method for class \textquote{\texttt{mixor}}}
\texttt{get_varcov(x, effects = c("\textquote{\texttt{all}}", \textquote{\texttt{fixed}}, \textquote{\texttt{random}}), \ldots)}

Arguments

\texttt{x} \hspace{1em} A model.

\texttt{\ldots} \hspace{1em} Currently not used.

\texttt{component} \hspace{1em} Should the complete variance-covariance matrix of the model be returned, or only for specific model components only (like count or zero-inflated model parts)? Applies to models with zero-inflated component, or models with precision (e.g. betareg) component. component may be one of "conditional", "zi", "zero-inflated", "precision", or "all". May be abbreviated. Note that the \texttt{conditional} component is also called \texttt{count} or \texttt{mean} component, depending on the model.

\texttt{effects} \hspace{1em} Should the complete variance-covariance matrix of the model be returned, or only for specific model parameters only? Currently only applies to models of class \texttt{mixor}. 
Value

The variance-covariance matrix, as matrix-object.

Note

get_varcov() tries to return the nearest positive definite matrix in case of a negative variance-covariance matrix.

Examples

data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
get_varcov(m)

description

This function extracts the different variance components of a mixed model and returns the result as list. Functions like get_variance_residual(x) or get_variance_fixed(x) are shortcuts for get_variance(x, component = "residual") etc.

Usage

generate_variance(
x,
component = c("all", "fixed", "random", "residual", "distribution", "dispersion",
"intercept", "slope", "rho0l"),
verbose = TRUE,
...)

generate_variance_residual(x, ...)

generate_variance_fixed(x, ...)

generate_variance_random(x, ...)

generate_variance_distribution(x, ...)

generate_variance_dispersion(x, ...)

generate_variance_intercept(x, ...)

generate_variance_slope(x, ...)

generate_correlation_slope_intercept(x, ...)

get_variance

Arguments

- **x**: A mixed effects model.
- **component**: Character value, indicating the variance component that should be returned. By default, all variance components are returned. The distribution-specific ("distribution") and residual ("residual") variance are the most computational intensive components, and hence may take a few seconds to calculate.
- **verbose**: Toggle off warnings.

Details

This function returns different variance components from mixed models, which are needed, for instance, to calculate r-squared measures or the intraclass-correlation coefficient (ICC).

**Fixed effects variance**: The fixed effects variance, $\sigma_f^2$, is the variance of the matrix-multiplication $\beta \ast X$ (parameter vector by model matrix).

**Random effects variance**: The random effect variance, $\sigma_\tau^2$, represents the mean random effect variance of the model. Since this variance reflect the "average" random effects variance for mixed models, it is also appropriate for models with more complex random effects structures, like random slopes or nested random effects. Details can be found in Johnson 2014, in particular equation 10. For simple random-intercept models, the random effects variance equals the random-intercept variance.

**Distribution-specific variance**: The distribution-specific variance, $\sigma_d^2$, depends on the model family. For Gaussian models, it is $\sigma^2$ (i.e. `sigma(model)^2`). For models with binary outcome, it is $\pi^2/3$ for logit-link and 1 for probit-link. For all other models, the distribution-specific variance is based on lognormal approximation, $\log(1 + \text{var}(x)/\mu^2)$ (see Nakagawa et al. 2017). The expected variance of a zero-inflated model is computed according to Zuur et al. 2012, p277.

**Variance for the additive overdispersion term**: The variance for the additive overdispersion term, $\sigma_\epsilon^2$, represents "the excess variation relative to what is expected from a certain distribution" (Nakagawa et al. 2017). In (most? many?) cases, this will be 0.

**Residual variance**: The residual variance, $\sigma_\epsilon^2$, is simply $\sigma_d^2 + \sigma_\epsilon^2$.

**Random intercept variance**: The random intercept variance, or between-subject variance ($\tau_{00}$), is obtained from `VarCorr()`. It indicates how much groups or subjects differ from each other, while the residual variance $\sigma_\epsilon^2$ indicates the within-subject variance.

**Random slope variance**: The random slope variance ($\tau_{11}$) is obtained from `VarCorr()`. This measure is only available for mixed models with random slopes.

**Random slope-intercept correlation**: The random slope-intercept correlation ($\rho_{01}$) is obtained from `VarCorr()`. This measure is only available for mixed models with random intercepts and slopes.
Value

A list with following elements:

- `var_fixed`, variance attributable to the fixed effects
- `var_random`, (mean) variance of random effects
- `var_residual`, residual variance (sum of dispersion and distribution)
- `var_distribution`, distribution-specific variance
- `var_dispersion`, variance due to additive dispersion
- `var_intercept`, the random-intercept-variance, or between-subject-variance ($\tau_{00}$)
- `var_slope`, the random-slope-variance ($\tau_{11}$)
- `cor_slope_intercept`, the random-slope-intercept-correlation ($\rho_{01}$)

Note

This function supports models of class `merMod` (including models from `blme`, `clmm`, `gllmm`, `glmmadmb`, `glmmTMB`, `MixMod`, `lme`, `mixed`, `rlmerMod`, `stanreg` or `wbm`). Support for objects of class `MixMod` (GLMMadaptive) or `lme` (nlme) is experimental and may not work for all models.

References


Examples

```r
## Not run:
library(lme4)
data(sleepstudy)
m <- lmer(Reaction ~ Days + (1 + Days | Subject), data = sleepstudy)
get_variance(m)
get_variance_fixed(m)
get_variance_residual(m)

## End(Not run)```
**get_weights**  
*Get the values from model weights*

---

**Description**

Returns weighting variable of a model.

**Usage**

```r
get_weights(x, ...)
```

**Arguments**

- `x` A fitted model.
- `...` Currently not used.

**Value**

The weighting variable, or NULL if no weights were specified.

**Examples**

```r
data(mtcars)
mtcars$weight <- rnorm(nrow(mtcars), 1, .3)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars, weights = weight)
get_weights(m)
```

---

**has_intercept**  
*Checks if model has an intercept*

---

**Description**

Checks if model has an intercept.

**Usage**

```r
has_intercept(x)
```

**Arguments**

- `x` A model object.

**Value**

TRUE if `x` has an intercept, FALSE otherwise.
Examples

```r
model <- lm(mpg ~ 0 + gear, data = mtcars)
has_intercept(model)

model <- lm(mpg ~ gear, data = mtcars)
has_intercept(model)

library(lme4)
model <- lmer(Reaction ~ 0 + Days + (Days | Subject), data = sleepstudy)
has_intercept(model)

model <- lmer(Reaction ~ Days + (Days | Subject), data = sleepstudy)
has_intercept(model)
```

---

**is_model**  
Checks if an object is a regression model object

**Description**

Small helper that checks if a model is a regression model object.

**Usage**

```r
is_model(x)
```

**Arguments**

- `x`  
  An object.

**Details**

This function returns `TRUE` if `x` is a model object.

**Value**

A logical, `TRUE` if `x` is a (supported) model object.

**Examples**

```r
data(mtcars)
m <- lm(mpg ~ wt + cyl + vs, data = mtcars)
is_model(m)
is_model(mtcars)
```
is_model_supported

Checks if an object is a regression model object supported in \texttt{insight} package.

Description

Small helper that checks if a model is a supported (regression) model object. \texttt{supported_models()} prints a list of currently supported model classes.

Usage

\begin{verbatim}
is_model_supported(x)
supported_models()
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} An object.
\end{itemize}

Details

This function returns \texttt{TRUE} if \texttt{x} is a model object that works with the package’s functions. A list of supported models can also be found here: \url{https://github.com/easystats/insight}.

Value

A logical, \texttt{TRUE} if \texttt{x} is a (supported) model object.

Examples

\begin{verbatim}
data(mtcars)
m <- \texttt{lm(mpg ~ wt + cyl + vs, data = mtcars)}

is_model_supported(m)
is_model_supported(mtcars)
\end{verbatim}

is_multivariate

Checks if an object stems from a multivariate response model

Description

Small helper that checks if a model is a multivariate response model, i.e. a model with multiple outcomes.

Usage

\begin{verbatim}
is_multivariate(x)
\end{verbatim}
Arguments

\(x\) A model object, or an object returned by a function from this package.

Value

A logical, \texttt{TRUE} if either \(x\) is a model object and is a multivariate response model, or \texttt{TRUE} if a return value from a function of \texttt{insight} is from a multivariate response model.

Examples

```r
## Not run:
library(rstanarm)
data("pbcLong")
model <- stan_mvmer(
  formula = list(
    logBili ~ year + (1 | id),
    albumin ~ sex + year + (year | id)
  ),
  data = pbcLong,
  chains = 1, cores = 1, seed = 12345, iter = 1000
)

f <- find_formula(model)
is_multivariate(model)
is_multivariate(f)
```

## End(Not run)

---

### is_nullmodel

Checks if model is a null-model (intercept-only)

Description

Checks if model is a null-model (intercept-only), i.e. if the conditional part of the model has no predictors.

Usage

\texttt{is_nullmodel(x)}

Arguments

\(x\) A model object.

Value

\texttt{TRUE} if \(x\) is a null-model, \texttt{FALSE} otherwise.
Examples

```r
model <- lm(mpg ~ 1, data = mtcars)
is_nullmodel(model)

model <- lm(mpg ~ gear, data = mtcars)
is_nullmodel(model)

library(lme4)
model <- lmer(Reaction ~ 1 + (Days | Subject), data = sleepstudy)
is_nullmodel(model)

model <- lmer(Reaction ~ Days + (Days | Subject), data = sleepstudy)
is_nullmodel(model)
```

Description

Returns the link-function from a model object.

Usage

```r
link_function(x, ...)
```

## S3 method for class 'gamlss'
link_function(x, what = c("mu", "sigma", "nu", "tau"), ...)

## S3 method for class 'betareg'
link_function(x, what = c("mean", "precision"), ...)

## S3 method for class 'DirichletRegModel'
link_function(x, what = c("mean", "precision"), ...)

Arguments

- **x**: A fitted model.
- **...**: Currently not used.
- **what**: For gamlss models, indicates for which distribution parameter the link (inverse) function should be returned; for betareg or DirichletRegModel, can be "mean" or "precision".

Value

A function, describing the link-function from a model-object. For multivariate-response models, a list of functions is returned.
Examples

```r
# example from stats::glm
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
m <- glm(counts ~ outcome + treatment, family = poisson())

link_function(m)(.3)
# same as
log(.3)
```

Description

Returns the link-inverse function from a model object.

Usage

```r
link_inverse(x, ...)
```

## S3 method for class 'betareg'
```r
link_inverse(x, what = c("mean", "precision"), ...)
```

## S3 method for class 'DirichletRegModel'
```r
link_inverse(x, what = c("mean", "precision"), ...)
```

## S3 method for class 'gamlss'
```r
link_inverse(x, what = c("mu", "sigma", "nu", "tau"), ...)
```

Arguments

- `x` A fitted model.
- `...` Currently not used.
- `what` For gamlss models, indicates for which distribution parameter the link (inverse) function should be returned; for betareg or DirichletRegModel, can be "mean" or "precision".

Value

A function, describing the inverse-link function from a model-object. For multivariate-response models, a list of functions is returned.
Examples

# example from ?stats::glm
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
m <- glm(counts ~ outcome + treatment, family = poisson())

link_inverse(m)(.3)
# same as
exp(.3)

---

model_info

Access information from model objects

Description

Retrieve information from model objects.

Usage

model_info(x, ...)

Arguments

x  A fitted model.
...

Currently not used.

Details

model_info() returns a list with information about the model for many different model objects. Following information is returned, where all values starting with is_ are logicals.

- is_binomial: family is binomial (but not negative binomial)
- is_poisson: family is poisson
- is_negbin: family is negative binomial
- is_count: model is a count model (i.e. family is either poisson or negative binomial)
- is_beta: family is beta
- is_betabinomial: family is beta-binomial
- is_dirichlet: family is dirichlet
- is_exponential: family is exponential (e.g. Gamma or Weibull)
- is_logit: model has logit link
- is_probit: model has probit link
- is_linear: family is gaussian
- is_tweedie: family is tweedie
• is_ordinal: family is ordinal, multinomial, or cumulative link
• is_cumulative: family is ordinal, multinomial, or cumulative link
• is_multinomial: family is multinomial or categorical link
• is_categorical: family is categorical link
• is_censored: model is a censored model (has a censored response, including survival models)
• is_truncated: model is a truncated model (has a truncated response)
• is_survival: model is a survival model
• is_zero_inflated: model has zero-inflation component
• is_hurdle: model has zero-inflation component and is a hurdle-model (truncated family distribution)
• is_mixed: model is a mixed effects model (with random effects)
• is_multivariate: model is a multivariate response model (currently only works for \textit{brmsfit} objects)
• is_trial: model response contains additional information about the trials
• is_bayesian: model is a Bayesian model
• is_anova: model is an Anova object
• link_function: the link-function
• family: the family-object
• n_obs: number of observations
• model_terms: a list with all model terms, including terms such as random effects or from zero-inflated model parts.

\textbf{Value}

A list with information about the model, like family, link-function etc. (see ’Details’).

\textbf{Examples}

```r
ldose <- rep(0:5, 2)
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
sex <- factor(rep(c("M", "F"), c(6, 6)))
SF <- cbind(numdead, numalive = 20 - numdead)
dat <- data.frame(ldose, sex, SF, stringsAsFactors = FALSE)
m <- glm(SF ~ sex * ldose, family = binomial)

model_info(m)
## Not run:
library(glmmTMB)
data("Salamanders")
m <- glmmTMB(
  count ~ spp + cover + mined + (1 | site),
  ziformula = ~ spp + mined,
  dispformula = ~DOY,
  data = Salamanders,
)```


\texttt{family = \texttt{nbinom2}}

\texttt{)}

\texttt{## End(Not run)}

\texttt{model\_info(m)}

\begin{verbatim}
n\_obs

\end{verbatim}

\begin{verbatim}
\texttt{n\_obs} \quad \textit{Get number of observations from a model}

\end{verbatim}

\textbf{Description}

This method returns the number of observation that were used to fit the model, as numeric value.

\textbf{Usage}

\texttt{n\_obs(x, \ldots)}

\texttt{## S3 method for class 'svyolr'}
\texttt{n\_obs(x, weighted = FALSE, \ldots)}

\texttt{## S3 method for class 'stanmvreg'}
\texttt{n\_obs(x, select = NULL, \ldots)}

\textbf{Arguments}

\begin{itemize}
\item \texttt{x} \quad \text{A fitted model.}
\item \texttt{\ldots} \quad \text{Currently not used.}
\item \texttt{weighted} \quad \text{For survey designs, returns the weighted sample size.}
\item \texttt{select} \quad \text{Optional name(s) of response variables for which to extract values. Can be used in case of regression models with multiple response variables.}
\end{itemize}

\textbf{Value}

The number of observations used to fit the model, or NULL if this information is not available.

\textbf{Examples}

\begin{verbatim}
data(mtcars)
m \leftarrow \texttt{lm(mpg - wt + cyl + vs, data = mtcars)}
n\_obs(m)
\end{verbatim}
**print_color**  
*Coloured console output*

**Description**

Convenient function that allows coloured output in the console. Mainly implemented to reduce package dependencies.

**Usage**

```r
print_color(text, color)
print_colour(text, colour)
```

**Arguments**

- `text`  
The text to print.
- `color, colour`  
Character vector, indicating the colour for printing. May be one of "red", "yellow", "green", "blue", "violet", "cyan" or "grey". Formatting is also possible with "bold" or "italic".

**Details**

This function prints `text` directly to the console using `cat()`, so no string is returned.

**Value**

Nothing.

**Examples**

```r
print_color("I'm blue dabedi dabedei", "blue")
```

---

**print_parameters**  
*Prepare summary statistics of model parameters for printing*

**Description**

This function takes a data frame, typically a data frame with information on summaries of model parameters like `hdi` or `equivalence_test`, as input and splits this information into several parts, depending on the model. See details below.
print_parameters

Usage

print_parameters(
  x,
  ..., 
  split_by = c("Effects", "Component", "Group", "Response")
)

Arguments

x A fitted model, or a data frame returned by `clean_parameters`.

... One or more objects (data frames), which contain information about the model parameters and related statistics (like confidence intervals, HDI, ROPE, ...).

split_by split_by should be a character vector with one or more of the following elements: "Effects", "Component", "Response" and "Group". These are the column names returned by `clean_parameters`, which is used to extract the information from which the group or component model parameters belong. If `NULL`, the merged data frame is returned. Else, the data frame is split into a list, split by the values from those columns defined in `split_by`.

Details

This function prepares data frames that contain information about model parameters for clear printing.

First, `x` is required, which should either be a model object or a prepared data frame as returned by `clean_parameters`. If `x` is a model, `clean_parameters()` is called on that model object to get information with which model components the parameters are associated.

Then, `...` take one or more data frames that also contain information about parameters from the same model, but also have additional information provided by other methods. For instance, a data frame in `...` might be the result of `hdi`, where we have a) a `Parameters` column and b) columns with the HDI values.

Now we have a data frame with model parameters and information about the association to the different model components, a data frame with model parameters, and some summary statistics. `print_parameters()` then merges these data frames, so the statistic of interest (in our example: the HDI) is also associated with the different model components. The data frame is split into a list, so for a clear printing. Users can loop over this list and print each component for a better overview. Further, parameter names are "cleaned", if necessary, also for a cleaner print. See also 'Examples'.

Value

A data frame or a list of data frames (if `split_by` is not `NULL`). If a list is returned, the element names reflect the model components where the extracted information in the data frames belong to, e.g. `random.zero_inflated.Intercept: persons`. This is the data frame that contains the parameters for the random effects from group-level "persons" from the zero-inflated model component.
## Examples

```r
## Not run:
library(bayestestR)
model <- download_model("brms_zi_2")
x <- hdi(model, effects = "all", component = "all")

# hdi() returns a data frame; here we use only the information on
# parameter names and HDI values
tmp <- as.data.frame(x)[, 1:4]
tmp

# Based on the "split_by" argument, we get a list of data frames that
# is split into several parts that reflect the model components.
print_parameters(model, tmp)

# This is the standard print()-method for "bayestestR::hdi"-objects.
# For printing methods, it is easy to print complex summary statistics
# in a clean way to the console by splitting the information into
# different model components.
x

## End(Not run)
```
Index

all_models_equal, 3
all_models_same_class
   (all_models_equal), 3
  clean_names, 4
  clean_parameters, 5, 49
color_if, 6
colour_if (color_if), 6
download_model, 7
equivalence_test, 48
find_algorithm, 8
find_formula, 9
find_interactions, 10
find_parameters, 11, 14
find_predictors, 14
find_random, 15
find_random_slopes, 16
find_response, 17
find_statistic, 17
find_terms, 18, 20
find_variables, 19, 19
find_weights, 21
format_ci, 21
format_table, 22
format_value, 23
get_correlation_slope_intercept
   (get_variance), 36
get_data, 24
get_parameters, 27
get_parameters.bayesx
   (find_parameters), 11
get_predictors, 30
get_priors, 31
get_random, 32
get_response, 32
get_statistic, 33
get_varcov, 34
get_variance, 36
get_variance_distribution
   (get_variance), 36
get_variance_fixed (get_variance), 36
get_variance_intercept (get_variance), 36
get_variance_random (get_variance), 36
get_variance_residual (get_variance), 36
get_variance_slope (get_variance), 36
get_weights, 39
has_intercept, 39
hdi, 48, 49
is_model, 40
is_model_supported, 41
is_multivariate, 41
is_nullmodel, 42
link_function, 43
link_inverse, 44
model_info, 45
n_obs, 47
print_color, 48
print_colour (print_color), 48
print_parameters, 5, 48
supported_models (is_model_supported), 41