Package 'lvmisc'

October 13, 2022

Title Veras Miscellaneous

Version 0.1.1

Description Contains a collection of useful functions for basic data computation and manipulation, wrapper functions for generating 'ggplot2' graphics, including statistical model diagnostic plots, methods for computing statistical models quality measures (such as AIC, BIC, r squared, root mean squared error) and general utilities.

License MIT + file LICENSE

URL https://lveras.com/lvmisc/

BugReports https://github.com/verasls/lvmisc/issues

Imports cowplot, dplyr (>= 1.0.0), ggplot2, glue, grDevices, methods, purrr, rlang (>= 0.4.6), rsample, stats, tibble, tidyselect, vctrs (>= 0.3.0)

Suggests covr, devtools, forcats, fs, git2r, knitr, lme4, rmarkdown, testthat, usethis, vdiffr, withr

Encoding UTF-8

RoxygenNote 7.1.1

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Lucas Veras [aut, cre] (<https://orcid.org/0000-0003-0562-5803>)

Maintainer Lucas Veras <lucasdsveras@gmail.com>

Repository CRAN

Date/Publication 2021-04-05 15:20:02 UTC

R topics documented:

abort_argument *Abort based on issues with function argument*

Description

Create a custom error condition created with `rlang::abort()` with a - hopefully - more useful error message and metadata.

Usage

```
abort_argument_type(arg, must, not)

abort_argument_class(arg, must, not)

abort_argument_length(arg, must, not)

abort_argument_diff_length(arg1, arg2)

abort_argument_value(arg, valid_values)
```

Arguments

`arg`	A character string with the argument name.
`must`	A character string specifying a condition the argument must fulfill.
`not`	Either a character string specifying a condition the argument must not fulfill or the bare (unquoted) argument name. In the last case, the function evaluates the argument type (`abort_argument_type()`) or length (`abort_argument_length()`) and displays the result in the error message.
`arg1, arg2`	A character string with the argument name.
`valid_values`	A character vector with the valid values.

Value

Each function returns a classed error condition. `abort_argument_type()` returns a `error_argument_type` class, `abort_argument_length()` returns a `error_argument_length` class, `abort_argument_diff_length()` returns a `error_argument_diff_length` class and `abort_argument_value()` returns a `error_argument_value` class.

See Also

`abort_column_not_found()`, `abort_no_method_for_class()`

`abort_column_not_found`

Abort based on column not being found in a data frame

Description

Creates a custom error condition created with `rlang::abort()` with a - hopefully - more useful error message and metadata.

Usage

```
abort_column_not_found(data, col_name)
```

Arguments

`data`	A data frame.
`col_name`	A character vector with the column name.

Value

Returns an error condition of classerror_column_not_found.

See Also

`abort_argument_type()`, `abort_argument_class()`, `abort_argument_length()`, `abort_argument_diff_length()`, `abort_no_method_for_class()`, `abort_package_not_installed()`

`abort_no_method_for_class`

Abort method if class is not implemented

Description

Creates a custom error condition created with `rlang::abort()` with a - hopefully - more useful error message and metadata.

Usage

```
abort_no_method_for_class(fun, class, ...)
```

Arguments

`fun`	A character vector with the function name.
`class`	A character vector with the class name.
`...`	Extra message to be added to the error message. Must be character string.

Value

Returns an error condition of classerror_no_method_for_class.

See Also

abort_argument_type(), abort_argument_class(), abort_argument_length(), abort_argument_diff_length(), abort_column_not_found(), abort_package_not_installed()

---

abort_package_not_installed

Abort if required package is not installed

Description

Creates a custom error condition created with rlang::abort() with a - hopefully - more useful error message and metadata.

Usage

```
abort_package_not_installed(package)
```

Arguments

package	A character string with the required package name.

Value

Returns an error condition of classerror_package_not_installed.

See Also

abort_argument_type(), abort_argument_class(), abort_argument_length(), abort_argument_diff_length(), abort_column_not_found(), abort_no_method_for_class()

---

accuracy

Model accuracy

Description

Computes some common model accuracy indices, such as the R squared, mean absolute error, mean absolute percent error and root mean square error.

Usage

```
accuracy(model, na.rm = FALSE)

## Default S3 method:
accuracy(model, na.rm = FALSE)

## S3 method for class 'lvmisc_cv'
accuracy(model, na.rm = FALSE)

## S3 method for class 'lm'
accuracy(model, na.rm = FALSE)

## S3 method for class 'lmerMod'
accuracy(model, na.rm = FALSE)
```

Arguments

model	An object of class lvmisc_cv or an object containing a model.
na.rm	A logical value indicating whether or not to strip NA values to compute the indices. Defaults to FALSE.

Details

The method for the lm class (or for the lvmisc_cv class of a lm) returns a data frame with the columns AIC (Akaike information criterion), BIC (Bayesian information criterion), R2 (R squared), R2_adj (adjusted R squared), MAE (mean absolute error), MAPE (mean absolute percent error) and RMSE (root mean square error).

The method for the lmerMod (or for the lvmisc_cv class of a lmerMod) returns a data frame with the columns R2_marg and R2_cond instead of the columns R2 and R2_adj. All the other columns are the same as the method for lm. R2_marg is the marginal R squared, which considers only the variance by the fixed effects of a mixed model, and R2_cond is the conditional R squared, which considers both fixed and random effects variance.

Value

An object of class lvmisc_accuracy. See "Details" for more information.

Examples

```
mtcars <- tibble::as_tibble(mtcars, rownames = "car")
m <- stats::lm(disp ~ mpg, mtcars)
cv <- loo_cv(m, mtcars, car, keep = "used")

accuracy(m)
accuracy(cv)
```

bias *Bias*

Description

Computes the bias (mean error) between the input vectors.

Usage

```
bias(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

A double scalar with the bias value.

See Also

mean_error(), loa()

Examples

```
actual <- runif(10)
predicted <- runif(10)

bias(actual, predicted)
```

bmi *Compute body mass index (BMI)*

Description

bmi calculates the BMI in kilograms per meter squared.

Usage

```
bmi(mass, height)
```

Arguments

`mass, height`	A numerical vector with body mass and height data. `mass` unit must be kilograms and `height` unit must be meters. If the `height` unit is centimeters, it is converted to meters before BMI computation and a warning is shown.

Value

Returns a double vector with the element-wise body mass index (BMI).

See Also

`bmi_cat()`

Examples

```
mass <- sample(50:100, 20)
height <- rnorm(20, mean = 1.7, sd = 0.2)

bmi(mass, height)
```

`bmi_cat` *Classify body mass index (BMI) category*

Description

`bmi_cat` returns the element-wise BMI category as factor with 6 levels:

- Underweight (18.5 < BMI)
- Normal weight (18.5 $\leq$ BMI < 25)
- Overweight (25 $\leq$ BMI < 30)
- Obesity class I (30 $\leq$ BMI < 35)
- Obesity class II (35 $\leq$ BMI < 40)
- Obesity class III (BMI $\geq$ 40)

Usage

```
bmi_cat(bmi)
```

Arguments

`bmi`	A numeric vector with BMI data. `BMI` unit must be meters per square meter.

Value

A vector of class `factor` with 6 levels: "Underweight", "Normal weight", "Overweight", "Obesity class I", "Obesity class II" and "Obesity class III".

See Also

bmi()

Examples

```
mass <- sample(50:100, 20)
height <- rnorm(20, mean = 1.7, sd = 0.2)
bmi <- bmi(mass, height)

bmi_cat(bmi)
```

---

center_variable *Center variable*

Description

Center a variable by subtracting the mean from each element. Centering can be performed by the grand mean when by = NULL (the default), or by group means when by is a factor variable.

Usage

```
center_variable(variable, scale = FALSE, by = NULL)
```

Arguments

variable	A numeric vector.
scale	A logical vector. If scale = TRUE, the centered values of variable are divided by their standard deviation.
by	A vector with the factor class.

Value

A numeric vector.

Examples

```
df <- data.frame(
  id = 1:20,
  group = as.factor(sample(c("A", "B"), 20, replace = TRUE)),
  body_mass = rnorm(20, mean = 65, sd = 12)
)

df$body_mass_centered <- center_variable(df$body_mass, by = df$group)
df
```

cl *Clear the console*

Description

Clear the console by printing 50 times the new line character (`"\n"`).

Usage

```
cl()
```

Value

Prints to console. Called by its side-effects.

clean_observations *Clean observations*

Description

Replace valid observations by `NA`s when a given subject has more then `max_na` missing values.

Usage

```
clean_observations(data, id, var, max_na)
```

Arguments

`data`	A data frame, or data frame extension (e.g. a tibble).
`id`	The bare (unquoted) name of the column that identifies each subject.
`var`	The bare (unquoted) name of the column to be cleaned.
`max_na`	An integer indicating the maximum number of `NA`s per subject.

Value

The original `data` with the `var` observations matching the `max_na` criterion replaced by `NA`.

Examples

```
set.seed(10)

data <- data.frame(
  id = rep(1:5, each = 4),
  time = rep(1:4, 5),
  score = sample(c(1:5, rep(NA, 2)), 20, replace = TRUE)
)

clean_observations(data, id, score, 1)
```

compare_accuracy *Compare models accuracy*

Description

Computes some common model accuracy indices of several different models at once, allowing model comparison.

Usage

```
compare_accuracy(..., rank_by = NULL, quiet = FALSE)
```

Arguments

...	A list of models. The models can be of the same or of different classes, including lvmisc_cv class.
rank_by	A character string with the name of an accuracy index to rank the models by.
quiet	A logical indicating whether or not to show any warnings. If FALSE (the default) no warnings are shown.

Value

A data.frame with a model per row and an index per column.

Examples

```
m1 <- lm(Sepal.Length ~ Species, data = iris)
m2 <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
m3 <- lm(Sepal.Length ~ Species * Petal.Length, data = iris)
compare_accuracy(m1, m2, m3)

if (require(lme4, quietly = TRUE)) {
  mtcars <- tibble::as_tibble(mtcars, rownames = "cars")
  m1 <- lm(Sepal.Length ~ Species, data = iris)
  m2 <- lmer(
    Sepal.Length ~ Sepal.Width + Petal.Length + (1 | Species), data = iris
  )
  m3 <- lm(disp ~ mpg * hp, mtcars)
  cv3 <- loo_cv(m3, mtcars, cars)
  compare_accuracy(m1, m2, cv3, rank_by = "AIC")
}
```

create_proj *Create a project*

Description

Creates a project structure, including sub-directories, and initialization of a git repository.

Usage

```
create_proj(
  path,
  sub_dirs = "default",
  use_git = TRUE,
  use_gitignore = "default",
  use_readme = TRUE
)
```

Arguments

path	A path to a directory that does not exist.
sub_dirs	A character vector. If sub_dirs = "default", it creates 'code/', 'data/', 'docs/', 'figures/' and 'tables/' sub-directories. Otherwise, it creates the sub-directories specified in the character vector.
use_git	A logical value indicating whether or not to initialize a git repository. Defaults to TRUE.
use_gitignore	A character vector. If use_gitignore = "default", it adds a .gitignore file with the files generated by your operating system and by R, as well as some common file extensions. The default .gitignore is as generated by gitignore.io. To create a custom .gitignore, add the files to be ignored in a character vector. If you do not want to create a .gitignore file, set use_gitignore = NULL.
use_readme	A logical value. If TRUE (default), adds an empty 'README.md' file.

Value

Path to the newly created project, invisibly.

divide_by_quantile *Divide variable based on quantiles*

Description

Creates a factor based on equally spaced quantiles of a variable.

Usage

```
divide_by_quantile(data, n, na.rm = TRUE)
```

Arguments

data	A numeric vector.
n	An integer specifying the number of levels in the factor to be created.
na.rm	A logical vector indicating whether the NA values should be removed before the quantiles are computed.

Value

A vector of class factor indicating in which quantile the element in data belongs.

See Also

stats::quantile().

Examples

```
x <- c(sample(1:20, 9), NA)
divide_by_quantile(x, 3)
```

---

error *Error*

Description

Computes the element-wise error between the input vectors.

Usage

```
error(actual, predicted)
```

Arguments

actual	A numeric vector with the actual values
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.

Value

Returns a double vector with the element-wise error values.

See Also

error_pct(), error_abs(), error_abs_pct(), error_sqr().

Examples

```
actual <- runif(10)
predicted <- runif(10)

error(actual, predicted)
```

`error_abs` *Absolute error*

Description

Computes the element-wise absolute errors between the input vectors.

Usage

```
error_abs(actual, predicted)
```

Arguments

`actual`	A numeric vector with the actual values
`predicted`	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in `actual`.

Value

Returns a double vector with the element-wise absolute error values.

See Also

`error()`, `error_pct()`, `error_abs_pct()`, `error_sqr()`.

Examples

```
actual <- runif(10)
predicted <- runif(10)

error_abs(actual, predicted)
```

error_abs_pct *Absolute percent error*

Description

Computes the element-wise absolute percent errors between the input vectors.

Usage

```
error_abs_pct(actual, predicted)
```

Arguments

actual	A numeric vector with the actual values
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.

Value

Returns a double vector with the element-wise absolute percent error values.

A vector of the class lvmisc_percent with the element-wise absolute percent error values.

See Also

error(), error_pct(), error_abs(), error_sqr().

Examples

```
actual <- runif(10)
predicted <- runif(10)

error_abs_pct(actual, predicted)
```

error_pct *Percent error*

Description

Computes the element-wise percent error between the input vectors.

Usage

```
error_pct(actual, predicted)
```

Arguments

actual	A numeric vector with the actual values
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.

Value

Returns a double vector with the element-wise percent error values.

A vector of the class lvmisc_percent with the element-wise percent error values.

See Also

error(), error_abs(), error_abs_pct(), error_sqr().

Examples

```
actual <- runif(10)
predicted <- runif(10)

error_pct(actual, predicted)
```

---

error_sqr *Squared error*

Description

Computes the element-wise squared errors between the input vectors.

Usage

```
error_sqr(actual, predicted)
```

Arguments

actual	A numeric vector with the actual values
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.

Value

Returns a double vector with the element-wise squared error values.

See Also

error(), error_pct(), error_abs(), error_abs_pct().

Examples

```
actual <- runif(10)
predicted <- runif(10)

error_sqr(actual, predicted)
```

`is_outlier` *Check whether value is outlier*

Description

`is_outlier` returns a logical vector indicating whether a value is an outlier based on the rule of 1.5 times the interquartile range above the third quartile or below the first quartile.

Usage

```
is_outlier(x, na.rm = FALSE)
```

Arguments

`x`	A numerical vector
`na.rm`	A logical value indicating whether `NA` values should be stripped before the computation proceeds. Defaults to `FALSE`.

Value

A logical vector.

See Also

`stats::IQR()`, `stats::quantile()`

Examples

```
x <- c(1:8, NA, 15)
is_outlier(x, na.rm = TRUE)
```

loa *Limits of agreement*

Description

Computes the Bland-Altman limits of agreement between the input vectors.

Usage

```
loa(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

A named list with the lower and upper limits of agreement values, respectively.

See Also

mean_error(), bias()

Examples

```
actual <- runif(10)
predicted <- runif(10)

loa(actual, predicted)
```

loo_cv *Leave-one-out cross-validation*

Description

Cross-validates the model using the leave-one-out approach. In this method each subject's data is separated into a testing data set, and all other subject's are kept in the training data set, with as many resamples as the number of subjects in the original data set. It computes the model's predicted value in the testing data set for each subject.

Usage

```
loo_cv(model, data, id, keep = "all")

## Default S3 method:
loo_cv(model, data, id, keep = "all")

## S3 method for class 'lm'
loo_cv(model, data, id, keep = "all")

## S3 method for class 'lmerMod'
loo_cv(model, data, id, keep = "all")
```

Arguments

model	An object containing a model.
data	A data frame.
id	The bare (unquoted) name of the column which identifies subjects.
keep	A character string which controls which columns are present in the output. Can be one of three options: • "all": The default. Retain all variables in the original data frame plus the ".actual" and ".predicted" columns. • "used": Keeps only the "id" column of the original data frame, plus the ".actual" and ".predicted" columns. • "none": Returns just the ".actual" and '".predicted" columns.

Value

Returns an object of class lvmisc_cv. A tibble containing the ".actual" and ".predicted" columns.

Examples

```
mtcars$car <- row.names(mtcars)
m <- stats::lm(disp ~ mpg, mtcars)
loo_cv(m, mtcars, car, keep = "used")
```

---

lt *Last error*

Description

lt() prints the last error and the full backtrace and le() returns the last error with a simplified backtrace. These functions are just wrappers to rlang::last_trace() and rlang::last_error() respectively.

Usage

```
lt()

le()
```

Value

An object of class `rlang_trace`.

An object of class `rlang_error`.

`lunique` *Number of elements in a vector.*

Description

`lunique` returns the number of non-`NA` unique elements and `lna` returns the number of `NA`s.

Usage

```
lunique(x)

lna(x)
```

Arguments

`x`	A vector.

Value

A non-negative integer.

See Also

`length()`, `unique()`, `is.na()`

Examples

```
x <- sample(c(1:3, NA), 10, replace = TRUE)
lunique(x)
lna(x)
```

mean_error *Mean error*

Description

Computes the average error between the input vectors.

Usage

```
mean_error(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

Returns a double scalar with the mean error value.

See Also

mean_error_pct(), mean_error_abs(), mean_error_abs_pct(), mean_error_sqr(), mean_error_sqr_root()

Examples

```
actual <- runif(10)
predicted <- runif(10)

mean_error(actual, predicted)
```

mean_error_abs *Mean absolute error*

Description

Computes the average absolute error between the input vectors.

Usage

```
mean_error_abs(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

Returns a double scalar with the mean absolute error value.

See Also

mean_error(), mean_error_pct(), mean_error_abs_pct(), mean_error_sqr(), mean_error_sqr_root()

Examples

```
actual <- runif(10)
predicted <- runif(10)

mean_error_abs(actual, predicted)
```

---

mean_error_abs_pct *Mean absolute percent error*

Description

Computes the average absolute percent error between the input vectors.

Usage

```
mean_error_abs_pct(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

Returns a double scalar with the mean absolute percent error value.

A vector of the class lvmisc_percent.

See Also

mean_error(), mean_error_abs(), mean_error_pct(), mean_error_sqr(), mean_error_sqr_root()

Examples

```
actual <- runif(10)
predicted <- runif(10)

mean_error_abs_pct(actual, predicted)
```

---

mean_error_pct *Mean percent error*

Description

Computes the average percent error between the input vectors.

Usage

```
mean_error_pct(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

Returns a double scalar with the mean percent error value.

A vector of the class lvmisc_percent.

See Also

mean_error(), mean_error_abs(), mean_error_abs_pct(), mean_error_sqr(), mean_error_sqr_root()

Examples

```
actual <- runif(10)
predicted <- runif(10)

mean_error_pct(actual, predicted)
```

mean_error_sqr *Mean square error*

Description

Computes the average square error between the input vectors.

Usage

```
mean_error_sqr(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

Returns a double scalar with the mean square error value.

See Also

mean_error(), mean_error_abs(), mean_error_pct(), mean_error_abs_pct(), mean_error_sqr_root()

Examples

```
actual <- runif(10)
predicted <- runif(10)

mean_error_sqr(actual, predicted)
```

mean_error_sqr_root *Root mean square error*

Description

Computes the root mean square error between the input vectors.

Usage

```
mean_error_sqr_root(actual, predicted, na.rm = FALSE)
```

Arguments

actual	A numeric vector with the actual values.
predicted	A numeric vector with the predicted values. Each element in this vector must be a prediction for the corresponding element in actual.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Value

Returns a double scalar with the root mean square error value.

See Also

mean_error(), mean_error_abs(), mean_error_pct(), mean_error_abs_pct(), mean_error_sqr()

Examples

```
actual <- runif(10)
predicted <- runif(10)

mean_error_sqr_root(actual, predicted)
```

---

notin *Value matching*

Description

Value matching

Usage

```
x %!in% table
```

Arguments

x	Vector with the values to be matched.
table	Vector with the values to be matched against.

Value

A logical vector indicating which values are not in table.

See Also

match().

Examples

```
x <- 8:12
x %!in% 1:10
```

---

pa *Print all rows of a data frame or tibble*

Description

Shortcut to print all rows of a data frame or tibble. Useful to inspect the whole tibble, as it prints by default only the first 20 rows.

Usage

```
pa(data)
```

Arguments

data	A data frame or tibble.

Value

Prints data and returns it invisibly.

See Also

print() and printing tibbles.

Examples

```
df <- dplyr::starwars
pa(df)
```

---

percent percent *vector*

Description

Creates a double vector that represents percentages. When printed, it is multiplied by 100 and suffixed with %.

Usage

```
percent(x = double())

is_percent(x)

as_percent(x)
```

Arguments

x	• For `percent()`: A numeric vector • For `is_percent()`: An object to test. • For `as_percent()`: An object to cast.

Value

An S3 vector of class `lvmisc_percent`.

Examples

```
percent(c(0.25, 0.5, 0.75))
```

---

percent_change *Computes the percent change*

Description

`percent_change` returns the element-wise percent change between two numeric vectors.

Usage

```
percent_change(baseline, followup)
```

Arguments

`baseline, followup`	A numeric vector with data to compute the percent change.

Value

A vector of class `lvmisc_percent`.

See Also

`percent()`, {`error_pct()`}

Examples

```
baseline <- sample(20:40, 10)
followup <- baseline * runif(10, min = 0.5, max = 1.5)

percent_change(baseline, followup)
```

plots *Quick plotting*

Description

These functions are intended to be used to quickly generate simple exploratory plots using the package ggplot2.

Usage

```
plot_scatter(data, x, y, ...)

plot_line(data, x, y, ...)

plot_hist(data, x, bin_width = NULL, ...)

plot_qq(data, x, ...)
```

Arguments

data	A data frame.
x, y	x and y aesthetics as the bare (unquoted) name of a column in data.
...	Additional arguments to be passed to the ggplot2::aes() function.
bin_width	The width of the bins in a histogram. When NULL (default), it uses the number of bins in bins (defaults to 30). You can also use one of the character strings "Sturges", "scott" or "FD" to use one of the methods to determine the bin width as in grDevices::nclass.*()

Value

A ggplot object.

Examples

```
plot_scatter(mtcars, disp, mpg, color = factor(cyl))
plot_line(Orange, age, circumference, colour = Tree)
plot_hist(iris, Petal.Width, bin_width = "FD")
plot_qq(mtcars, mpg)
```

plot_bland_altman *Create a Bland-Altman plot*

Description

Create a Bland-Altman plot as described by Bland & Altman (1986).

Usage

```
plot_bland_altman(x, ...)
```

Arguments

x	An object of class `lvmisc_cv` or an object containing a model.
...	Additional arguments to be passed to `ggplot2::aes()`.

Value

A `ggplot` object.

References

- Bland, J.M. & Altman, D.G. (1986). Statistical methods for assessing agreement between two methods of clinical measurement. Lancet, 8(1), 307-10. doi: 10.1016/S01406736(86)908378

Examples

```
mtcars <- tibble::as_tibble(mtcars, rownames = "car")
m <- stats::lm(disp ~ mpg, mtcars)
cv <- loo_cv(m, mtcars, car)
plot_bland_altman(cv, colour = as.factor(am))
```

plot_model *Plot model diagnostics*

Description

Plotting functions for some common model diagnostics.

Usage

```
plot_model(model)

plot_model_residual_fitted(model)

plot_model_scale_location(model)

plot_model_qq(model)

plot_model_cooks_distance(model)

plot_model_multicollinearity(model)
```

Arguments

model	An object containing a model.

Details

plot_model_residual_fitted() plots the model residuals versus the fitted values. plot_model_scale_location() plots the square root of absolute value of the model residuals versus the fitted values. plot_model_qq() plots a QQ plot of the model standardized residuals. plot_model_cooks_distance() plots a bat chart of each observation Cook's distance value. plot_model_multicollinearity() plots a bar chart of the variance inflation factor (VIF) for each of the model terms. plot_model() returns a plot grid with all the applicable plot diagnostics to a given model.

Value

A ggplot object.

Examples

```
m <- lm(disp ~ mpg + hp + cyl + mpg:cyl, mtcars)
plot_model(m)
plot_model_residual_fitted(m)
plot_model_scale_location(m)
plot_model_qq(m)
plot_model_cooks_distance(m)
plot_model_multicollinearity(m)
```

---

r2 *Compute R squared*

Description

Returns the R squared values according to the model class.

Usage

```
r2(model)

## Default S3 method:
r2(model)

## S3 method for class 'lm'
r2(model)

## S3 method for class 'lmerMod'
r2(model)
```

Arguments

model	An object containing a model.

Details

R squared computations.

Value

If the model is a linear model, it returns a `data.frame` with the R squared and adjusted R squared values. If the model is a linear mixed model it return a `data.frame` with the marginal and conditional R squared values as described by Nakagawa and Schielzeth (2013). See the formulas for the computations in "Details".

R squared

$$R^2 = \frac{var(\hat{y})}{var(\epsilon)}$$

Where $var(\hat{y})$ is the variance explained by the model and $var(\epsilon)$ is the residual variance.

Adjusted R squared

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

Where n is the number of data points and p is the number of predictors in the model.

Marginal R squared

$$R^2_{marg} = \frac{var(f)}{var(f) + var(r) + var(\epsilon)}$$

Where $var(f)$ is the variance of the fixed effects, $var(r)$ is the variance of the random effects and $var(\epsilon)$ is the residual variance.

Conditional R squared

$$R^2_{cond} = \frac{var(f) + var(r)}{var(f) + var(r) + var(\epsilon)}$$

References

- Nakagawa, S., & Schielzeth, H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution, 4(2), 133–142. doi: 10.1111/j.2041210x.2012.00261.x.

Examples

```
m1 <- lm(Sepal.Length ~ Species, data = iris)
r2(m1)
if (require(lme4, quietly = TRUE)) {
  m2 <- lmer(
    Sepal.Length ~ Sepal.Width + Petal.Length + (1 | Species), data = iris
  )
  r2(m2)
}
```

---

repeat_baseline_values

Repeat baseline levels

Description

Returns a vector with the length equal to the number of rows in the `data` with the baseline value of the `var` repeated for every `time` value of each `id`.

Usage

```
repeat_baseline_values(data, var, id, time, baseline_level, repeat_NA = TRUE)
```

Arguments

`data`	A data frame.
`var`	The bare (unquoted) name of the column with the values to be repeated.
`id`	The bare (unquoted) name of the column that identifies each subject.
`time`	The bare (unquoted) name of the column with the time values.
`baseline_level`	The value of `time` corresponding the baseline.
`repeat_NA`	A logical vector indicating whether or not `NA` values in the `var` will correspond to `NA` values in return vector. Defaults to `TRUE`.

Value

A vector of the same lenght and class of `var`.

Examples

```
df <- data.frame(
 id = rep(1:5, each = 4),
 time = rep(1:4, 5),
 score = rnorm(20, mean = 10, sd = 2)
)

df$baseline_score <- repeat_baseline_values(df, score, id, time, 1)
df
```

---

tb *Capture a backtrace*

Description

Captures the sequence of calls that lead to the current function. It is just a wrapper to `rlang::trace_back()`.

Usage

```
tb(...)
```

Arguments

`...`	Passed to `rlang::trace_back()`.

Value

An object of class `rlang_trace`.

---

vif *Variance inflation factor*

Description

Computes the variance inflation factor (VIF). The VIF is a measure of how much the variance of a regression coefficient is increased due to collinearity.

Usage

```
vif(model)

## Default S3 method:
vif(model)

## S3 method for class 'lm'
vif(model)

## S3 method for class 'lmerMod'
vif(model)
```

Arguments

model	An object containing a model.

Details

VIF interpretation:

As a rule of thumb for the interpretation of the VIF value, a VIF less than 5 indicates a low correlation of a given model term with the others, a VIF between 5 and 10 indicates a moderate correlation and a VIF greater than 10 indicates a high correlation.

Value

It returns a `data.frame` with three columns: the name of the model term, the VIF value and its classification (see "Details").

References

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (eds.). (2013). An introduction to statistical learning: with applications in R. New York: Springer.

Examples

```
m <- lm(disp ~ mpg + cyl + mpg:cyl, mtcars)
vif(m)
```

Index