Package 'modelbased'

April 12, 2021

Type Package

Title Estimation of Model-Based Predictions, Contrasts and Means

Version 0.6.0

Maintainer Dominique Makowski <dom.makowski@gmail.com>

Description Implements a general interface for model-based estimations for a wide variety of models (see support list of insight; Lüdecke, Waggoner & Makowski (2019) <doi:10.21105/joss.01412>), used in the computation of marginal means, contrast analysis and predictions.

License GPL-3

URL https://easystats.github.io/modelbased/

BugReports https://github.com/easystats/modelbased/issues

Imports bayestestR (>= 0.8.3), emmeans (>= 1.5.3), graphics, insight (>= 0.13.1), parameters (>= 0.12.0), stats, utils

Suggests brms, coda, dplyr, effectsize, gamm4, gganimate, ggplot2, glmmTMB, knitr, lme4, logspline, MASS, Matrix, merTools, mgcv, rmarkdown, rstanarm, see, testthat, spelling

Encoding UTF-8

RoxygenNote 7.1.1

Config/testthat/edition 3

Config/testthat/parallel true

Language en-US

NeedsCompilation no

Author Dominique Makowski [aut, cre] (<https://orcid.org/0000-0001-5375-9967>, @Dom_Makowski),
Daniel Lüdecke [aut] (<https://orcid.org/0000-0002-8895-3206>),
Mattan S. Ben-Shachar [aut] (<https://orcid.org/0000-0002-4287-4801>),
Indrajeet Patil [ctb] (<https://orcid.org/0000-0003-1995-6531>, @patilindrajeets)

Repository CRAN

Date/Publication 2021-04-12 07:40:11 UTC

R topics documented:

`as.numeric_ifnumeric` *Convert to Numeric if Possible*

Description

Tries to convert vector to numeric if possible. Otherwise, leaves it as is.

Usage

```
as.numeric_ifnumeric(x)
```

Arguments

`x`	A vector to be converted.

Value

Numeric

Examples

```
as.numeric_ifnumeric(c("1", "2"))
as.numeric_ifnumeric(c("1", "2", "A"))
```

data_restoretype *Restore the type of columns according to a reference data.frame*

Description

Restore the type of columns according to a reference data.frame

Usage

```
data_restoretype(data, reference_data = NULL)
```

Arguments

data	The data.
reference_data	A reference dataframe from which to find the correct column types.

Examples

```
library(modelbased)

x <- data.frame(
  Sepal.Length = c("1", "3", "2"),
  Species = c("setosa", "versicolor", "setosa")
)
fixed <- data_restoretype(x, reference_data = iris)
summary(fixed)
```

estimate_contrasts *Estimate contrasts between factor levels*

Description

Contrast analysis. See the documentation for your object's class:

- Frequentist models
- Bayesian models

Usage

```
estimate_contrasts(
  model,
  levels = NULL,
  fixed = NULL,
  modulate = NULL,
  transform = "none",
  length = 10,
```

```
  standardize = TRUE,
  standardize_robust = FALSE,
  ...
)
```

Arguments

model	A statistical model.
levels	A character vector or formula specifying the names of the predictors over which to estimate means or contrasts.
fixed	A character vector indicating the names of the predictors to be "fixed" (i.e., maintained), so that the estimation is made at these values.
modulate	A character vector indicating the names of a numeric variable along which the means or the contrasts will be estimated. Adjust its length using `length`.
transform	Can be `"none"` (default for contrasts), `"response"` (default for means), `"mu"`, `"unlink"`, `"log"`. `"none"` will leave the values on scale of the linear predictors. `"response"` will transform them on scale of the response variable. Thus for a logistic model, `"none"` will give estimations expressed in log-odds (probabilities on logit scale) and `"response"` in terms of probabilities.
length	Length of the spread numeric variables.
standardize	If `TRUE`, adds standardized differences or coefficients.
standardize_robust	Robust standardization through `MAD` (Median Absolute Deviation, a robust estimate of SD) instead of regular `SD`.
...	Arguments passed to or from other methods.

Value

A data frame of estimated contrasts.

---

estimate_contrasts.lm *Estimate contrasts*

Description

Estimate contrasts

Usage

```
## S3 method for class 'lm'
estimate_contrasts(
  model,
  levels = NULL,
  fixed = NULL,
  modulate = NULL,
```

```
  transform = "none",
  length = 10,
  standardize = TRUE,
  standardize_robust = FALSE,
  ci = 0.95,
  adjust = "holm",
  ...
)
```

Arguments

model	A Bayesian model.
levels	A character vector or formula specifying the names of the predictors over which to estimate means or contrasts.
fixed	A character vector indicating the names of the predictors to be "fixed" (i.e., maintained), so that the estimation is made at these values.
modulate	A character vector indicating the names of a numeric variable along which the means or the contrasts will be estimated. Adjust its length using `length`.
transform	Can be `"none"` (default for contrasts), `"response"` (default for means), `"mu"`, `"unlink"`, `"log"`. `"none"` will leave the values on scale of the linear predictors. `"response"` will transform them on scale of the response variable. Thus for a logistic model, `"none"` will give estimations expressed in log-odds (probabilities on logit scale) and `"response"` in terms of probabilities.
length	Length of the spread numeric variables.
standardize	If `TRUE`, adds standardized differences or coefficients.
standardize_robust	Robust standardization through `MAD` (Median Absolute Deviation, a robust estimate of SD) instead of regular `SD`.
ci	Credible Interval (CI) level. Default to 0.89 (89%). See `ci` for further details.
adjust	The p-values adjustment method for multi-comparisons. Can be one of "holm" (default), "tukey", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" or "none". See the p-value adjustment section in the `emmeans::test` documentation.
...	Arguments passed to or from other methods.

Value

A dataframe of estimated contrasts.

Examples

```
library(modelbased)

model <- lm(Sepal.Width ~ Species, data = iris)
estimate_contrasts(model)

model <- lm(Sepal.Width ~ Species * Petal.Width, data = iris)
```

```
estimate_contrasts(model)
estimate_contrasts(model, fixed = "Petal.Width")
estimate_contrasts(model, modulate = "Petal.Width", length = 4)
estimate_contrasts(model, levels = "Petal.Width", length = 4)

if (require("lme4")) {
  data <- iris
  data$Petal.Length_factor <- ifelse(data$Petal.Length < 4.2, "A", "B")

  model <- lmer(Sepal.Width ~ Species + (1 | Petal.Length_factor), data = data)
  estimate_contrasts(model)
}
```

`estimate_contrasts.stanreg` *Estimate contrasts*

Description

Estimate contrasts

Usage

```
## S3 method for class 'stanreg'
estimate_contrasts(
  model,
  levels = NULL,
  fixed = NULL,
  modulate = NULL,
  transform = "none",
  length = 10,
  standardize = TRUE,
  standardize_robust = FALSE,
  centrality = "median",
  ci = 0.95,
  ci_method = "hdi",
  test = c("pd", "rope"),
  rope_range = "default",
  rope_ci = 1,
  ...
)
```

Arguments

`model`	A Bayesian model.
`levels`	A character vector or formula specifying the names of the predictors over which to estimate means or contrasts.

fixed	A character vector indicating the names of the predictors to be "fixed" (i.e., maintained), so that the estimation is made at these values.
modulate	A character vector indicating the names of a numeric variable along which the means or the contrasts will be estimated. Adjust its length using `length`.
transform	Can be `"none"` (default for contrasts), `"response"` (default for means), `"mu"`, `"unlink"`, `"log"`. `"none"` will leave the values on scale of the linear predictors. `"response"` will transform them on scale of the response variable. Thus for a logistic model, `"none"` will give estimations expressed in log-odds (probabilities on logit scale) and `"response"` in terms of probabilities.
length	Length of the spread numeric variables.
standardize	If `TRUE`, adds standardized differences or coefficients.
standardize_robust	Robust standardization through `MAD` (Median Absolute Deviation, a robust estimate of SD) instead of regular `SD`.
centrality	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: `"median"`, `"mean"`, `"MAP"` or `"all"`.
ci	Credible Interval (CI) level. Default to 0.89 (89%). See `ci` for further details.
ci_method	The type of index used for Credible Interval. Can be `"HDI"` (default, see `hdi`), `"ETI"` (see `eti`) or `"SI"` (see `si`).
test	The indices of effect existence to compute. Character (vector) or list with one or more of these options: `"p_direction"` (or `"pd"`), `"rope"`, `"p_map"`, `"equivalence_test"` (or `"equitest"`), `"bayesfactor"` (or `"bf"`) or `"all"` to compute all tests. For each "test", the corresponding **bayestestR** function is called (e.g. `rope` or `p_direction`) and its results included in the summary output.
rope_range	ROPE's lower and higher bounds. Should be a list of two values (e.g., `c(-0.1,0.1)`) or `"default"`. If `"default"`, the bounds are set to `x +-0.1*SD(response)`.
rope_ci	The Credible Interval (CI) probability, corresponding to the proportion of HDI, to use for the percentage in ROPE.
...	Arguments passed to or from other methods.

Value

A data frame of estimated contrasts.

Examples

```
library(modelbased)

data <- mtcars
data$cyl <- as.factor(data$cyl)
data$am <- as.factor(data$am)
## Not run:
if (require("rstanarm")) {
  model <- stan_glm(mpg ~ cyl * am, data = data, refresh = 0)
  estimate_contrasts(model)
  estimate_contrasts(model, fixed = "am")
```

```
  model <- stan_glm(mpg ~ cyl * wt, data = data, refresh = 0)
  estimate_contrasts(model)
  estimate_contrasts(model, fixed = "wt")
  estimate_contrasts(model, modulate = "wt", length = 4)
  estimate_contrasts(model, levels = "wt", length = 4)

 model <- stan_glm(Sepal.Width ~ Species + Petal.Width + Petal.Length, data = iris, refresh = 0)
 estimate_contrasts(model, fixed = "Petal.Width", modulate = "Petal.Length", test = "bf")
}

if (require("brms")) {
  model <- brm(mpg ~ cyl * am, data = data, refresh = 0)
  estimate_contrasts(model)
}

## End(Not run)
```

`estimate_means` *Estimate average value of response variable at each factor levels*

Description

Estimate average value of response variable at each factor levels

Usage

```
estimate_means(
  model,
  levels = NULL,
  fixed = NULL,
  modulate = NULL,
  transform = "response",
  length = 10,
  centrality = "median",
  ci = 0.95,
  ci_method = "hdi",
  ...
)
```

Arguments

`model`	A statistical model.
`levels`	A character vector or formula specifying the names of the predictors over which to estimate means or contrasts.
`fixed`	A character vector indicating the names of the predictors to be "fixed" (i.e., maintained), so that the estimation is made at these values.

modulate	A character vector indicating the names of a numeric variable along which the means or the contrasts will be estimated. Adjust its length using length.
transform	Can be "none" (default for contrasts), "response" (default for means), "mu", "unlink", "log". "none" will leave the values on scale of the linear predictors. "response" will transform them on scale of the response variable. Thus for a logistic model, "none" will give estimations expressed in log-odds (probabilities on logit scale) and "response" in terms of probabilities.
length	Length of the spread numeric variables.
centrality, ci, ci_method	Arguments for Bayesian models.
...	Arguments passed to or from other methods.

Value

A dataframe of estimated marginal means.

Examples

```
library(modelbased)

model <- lm(Petal.Length ~ Sepal.Width * Species, data = iris)

estimate_means(model)
estimate_means(model, fixed = "Sepal.Width")
estimate_means(model, levels = c("Species", "Sepal.Width"), length = 2)
estimate_means(model, levels = "Species=c('versicolor', 'setosa')")
estimate_means(model, levels = "Sepal.Width=c(2, 4)")
estimate_means(model, levels = c("Species", "Sepal.Width=0"))
estimate_means(model, modulate = "Sepal.Width", length = 5)
estimate_means(model, modulate = "Sepal.Width=c(2, 4)")
## Not run:
if (require("lme4")) {
  data <- iris
  data$Petal.Length_factor <- ifelse(data$Petal.Length < 4.2, "A", "B")

 model <- lmer(Petal.Length ~ Sepal.Width + Species + (1 | Petal.Length_factor), data = data)
  estimate_means(model)
  estimate_means(model, modulate = "Sepal.Width", length = 3)
}

## End(Not run)

data <- mtcars
data$cyl <- as.factor(data$cyl)
data$am <- as.factor(data$am)

if (require("rstanarm")) {
  model <- stan_glm(mpg ~ cyl * am, data = data, refresh = 0)
  estimate_means(model)

  model <- stan_glm(mpg ~ cyl * wt, data = data, refresh = 0)
```

```
  estimate_means(model)
  estimate_means(model, modulate = "wt")
  estimate_means(model, fixed = "wt")
}


## Not run:
if (require("brms")) {
  model <- brm(mpg ~ cyl * am, data = data, refresh = 0)
  estimate_means(model)
}

## End(Not run)
```

`estimate_relation` *Generates predictions from models*

Description

`estimate_link` is a shortcut to `estimate_response` with `data = "grid"`. `estimate_response` would be used in the context of generating actual predictions for the existing or new data, whereas `estimate_link` is more relevant in the context of visualisation and plotting. There are many control parameters that are not listed here but can be used, such as the arguments from `visualisation_matrix` (used when `data = "grid"`) and from `insight::get_predicted()` (the function to compute predictions used internally).

Usage

```
estimate_relation(
  model,
  data = "grid",
  ci = 0.95,
  keep_iterations = FALSE,
  ...
)

estimate_link(model, data = "grid", ci = 0.95, keep_iterations = FALSE, ...)

estimate_prediction(
  model,
  data = NULL,
  ci = 0.95,
  keep_iterations = FALSE,
  ...
)

estimate_response(model, data = NULL, ci = 0.95, keep_iterations = FALSE, ...)
```

Arguments

model	A statistical model.
data	A data frame with model's predictors to estimate the response. If NULL, the model's data is used. If "grid", the model matrix is obtained (through visualisation_matrix).
ci	The interval level (default 0.95, i.e., 95% CI).
keep_iterations	Only relevant for Bayesian models or simulated models. If TRUE, will keep all prediction iterations (draws). You can reshape them by running bayestestR::reshape_iterations().
...	You can add all the additional control arguments from visualisation_matrix (used when data = "grid") and insight::get_predicted().

Value

A dataframe of predicted values.

Examples

```
library(modelbased)

# Linear Models
model <- lm(mpg ~ wt, data = mtcars)
estimate_response(model)
estimate_relation(model)

# Logistic Models
model <- glm(vs ~ wt, data = mtcars, family = "binomial")
estimate_response(model)
estimate_relation(model)

# Mixed models
if (require("lme4")) {
  model <- lmer(mpg ~ wt + (1 | gear), data = mtcars)
  estimate_response(model)
  estimate_relation(model)
}

# Bayesian models
if (require("rstanarm")) {
  model <- rstanarm::stan_glm(mpg ~ wt, data = mtcars, refresh = 0, iter = 200)
  estimate_response(model)
  estimate_relation(model)
}
```

---

estimate_slopes *Estimate the slopes of a numeric predictor (over different factor levels)*

Description

See the documentation for your object's class:

- Frequentist models
- Bayesian models (stanreg and brms)

Usage

```
estimate_slopes(
  model,
  trend = NULL,
  levels = NULL,
  transform = "response",
  standardize = TRUE,
  standardize_robust = FALSE,
  ci = 0.95,
  ...
)

## S3 method for class 'glmmTMB'
estimate_slopes(
  model,
  trend = NULL,
  levels = NULL,
  transform = "response",
  standardize = TRUE,
  standardize_robust = FALSE,
  ci = 0.95,
  component = c("conditional", "zero_inflated", "zi"),
  ...
)
```

Arguments

model	A Bayesian model.
trend	A character vector indicating the name of the numeric variable for which to compute the slopes.
levels	A character vector indicating the variables over which the slope will be computed. If NULL (default), it will select all the remaining predictors.
transform	Can be "none" (default for contrasts), "response" (default for means), "mu", "unlink", "log". "none" will leave the values on scale of the linear predictors. "response" will transform them on scale of the response variable. Thus for a logistic model, "none" will give estimations expressed in log-odds (probabilities on logit scale) and "response" in terms of probabilities.
standardize	If TRUE, adds standardized differences or coefficients.
standardize_robust	Robust standardization through MAD (Median Absolute Deviation, a robust estimate of SD) instead of regular SD.

ci	Credible Interval (CI) level. Default to 0.89 (89%). See ci for further details.
...	Arguments passed to or from other methods.
component	A character vector indicating the model component for which estimation is requested. Only applies to models from **glmmTMB**. Use "conditional" for the count-model or "zero_inflate" or "zi" for the zero-inflation model.

Value

A data frame of slopes.

`estimate_slopes.lm` *Estimate the slopes of a numeric predictor (over different factor levels)*

Description

Estimate the slopes of a numeric predictor (over different factor levels)

Usage

```
## S3 method for class 'lm'
estimate_slopes(
  model,
  trend = NULL,
  levels = NULL,
  transform = "response",
  standardize = TRUE,
  standardize_robust = FALSE,
  ci = 0.95,
  ...
)
```

Arguments

model	A Bayesian model.
trend	A character vector indicating the name of the numeric variable for which to compute the slopes.
levels	A character vector indicating the variables over which the slope will be computed. If NULL (default), it will select all the remaining predictors.
transform	Can be "none" (default for contrasts), "response" (default for means), "mu", "unlink", "log". "none" will leave the values on scale of the linear predictors. "response" will transform them on scale of the response variable. Thus for a logistic model, "none" will give estimations expressed in log-odds (probabilities on logit scale) and "response" in terms of probabilities.
standardize	If TRUE, adds standardized differences or coefficients.

standardize_robust
: Robust standardization through MAD (Median Absolute Deviation, a robust estimate of SD) instead of regular SD.

ci
: Credible Interval (CI) level. Default to 0.89 (89%). See ci for further details.

...
: Arguments passed to or from other methods.

Examples

```
library(modelbased)

model <- lm(Sepal.Width ~ Species * Petal.Length, data = iris)
estimate_slopes(model)
```

---

estimate_slopes.stanreg

Estimate the slopes of a numeric predictor (over different factor levels)

Description

Estimate the slopes of a numeric predictor (over different factor levels)

Usage

```
## S3 method for class 'stanreg'
estimate_slopes(
  model,
  trend = NULL,
  levels = NULL,
  transform = "response",
  standardize = TRUE,
  standardize_robust = FALSE,
  ci = 0.95,
  centrality = "median",
  ci_method = "hdi",
  test = c("pd", "rope"),
  rope_range = "default",
  rope_ci = 1,
  ...
)
```

Arguments

model
: A Bayesian model.

trend
: A character vector indicating the name of the numeric variable for which to compute the slopes.

levels
: A character vector indicating the variables over which the slope will be computed. If NULL (default), it will select all the remaining predictors.

transform	Can be "none" (default for contrasts), "response" (default for means), "mu", "unlink", "log". "none" will leave the values on scale of the linear predictors. "response" will transform them on scale of the response variable. Thus for a logistic model, "none" will give estimations expressed in log-odds (probabilities on logit scale) and "response" in terms of probabilities.
standardize	If TRUE, adds standardized differences or coefficients.
standardize_robust	Robust standardization through MAD (Median Absolute Deviation, a robust estimate of SD) instead of regular SD.
ci	Credible Interval (CI) level. Default to 0.89 (89%). See ci for further details.
centrality	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: "median", "mean", "MAP" or "all".
ci_method	The type of index used for Credible Interval. Can be "HDI" (default, see hdi), "ETI" (see eti) or "SI" (see si).
test	The indices of effect existence to compute. Character (vector) or list with one or more of these options: "p_direction" (or "pd"), "rope", "p_map", "equivalence_test" (or "equitest"), "bayesfactor" (or "bf") or "all" to compute all tests. For each "test", the corresponding **bayestestR** function is called (e.g. rope or p_direction) and its results included in the summary output.
rope_range	ROPE's lower and higher bounds. Should be a list of two values (e.g., c(-0.1,0.1)) or "default". If "default", the bounds are set to x +-0.1*SD(response).
rope_ci	The Credible Interval (CI) probability, corresponding to the proportion of HDI, to use for the percentage in ROPE.
...	Arguments passed to or from other methods.

Examples

```
library(modelbased)

if (require("rstanarm")) {
  model <- stan_glm(Sepal.Width ~ Species * Petal.Length, data = iris)
  estimate_slopes(model)
}
```

---

estimate_smooth *Describe the smooth term (for GAMs) or non-linear predictors*

Description

This function summarises the smooth term trend in terms of linear segments. Using the aproximative derivative, it separates a non-linear vector into quasi-linear segments (in which the trend is either positive or negative). Each of this segment its characterized by its beginning, end, size (in proportion, relative to the total size) trend (the linear regression coefficient) and linearity (the R2 of the linear regression).

Usage

```
estimate_smooth(model, smooth = NULL, levels = NULL, ...)
```

Arguments

model	A Bayesian model.
smooth	A character indicating the name of the "smooth" term
levels	A character vector indicating the variables over which the slope will be computed. If NULL (default), it will select all the remaining predictors.
...	Arguments passed to or from other methods.

Value

A dataframe of linear description of non-linear terms.

Examples

```
library(modelbased)

if (require("rstanarm")) {
  model <- stan_gamm4(Sepal.Width ~ s(Petal.Length), data = iris, refresh = 0)
  estimate_smooth(model)

  model <- stan_glm(Sepal.Width ~ poly(Petal.Length, 2), data = iris)
  estimate_smooth(model)

  model <- stan_gamm4(Sepal.Width ~ Species + s(Petal.Length), data = iris)
  estimate_smooth(model)

  model <- stan_glm(Sepal.Width ~ Species * poly(Petal.Length, 2), data = iris)
  estimate_smooth(model)
  estimate_smooth(model, levels = "Species")
}
```

---

find_inversions *Find points of inversion*

Description

Find points of inversion of a curve.

Usage

```
find_inversions(x)
```

Arguments

x	A numeric vector.

Value

Vector of inversion points.

Examples

```
x <- sin(seq(0, 4 * pi, length.out = 100))
plot(x, type = "b")
find_inversions(x)
```

---

smoothing *Smoothing a vector or a time series*

Description

Smoothing a vector or a time series. For data.frames, the function will smooth all numeric variables stratified by factor levels (i.e., will smooth within each factor level combination).

Usage

```
smoothing(x, method = "loess", strength = 0.25, ...)
```

Arguments

x	A numeric vector.
method	Can be "loess" (default) or "smooth". A loess smoothing can be slow.
strength	This argument only applies to smooth_method = "loess". Degree of smoothing passed to span (see loess).
...	Arguments passed to or from other methods.

Value

A smoothed vector or data frame.

Examples

```
x <- sin(seq(0, 4 * pi, length.out = 100)) + rnorm(100, 0, 0.2)
plot(x, type = "l")
lines(smoothing(x, method = "smooth"), type = "l", col = "blue")
lines(smoothing(x, method = "loess"), type = "l", col = "red")

x <- sin(seq(0, 4 * pi, length.out = 10000)) + rnorm(10000, 0, 0.2)
plot(x, type = "l")
lines(smoothing(x, method = "smooth"), type = "l", col = "blue")
lines(smoothing(x, method = "loess"), type = "l", col = "red")
```

visualisation_matrix *Create a reference grid*

Description

Create a reference matrix, useful for visualisation, with evenly spread and combined values.

Usage

```
visualisation_matrix(
  x,
  target = "all",
  length = 10,
  factors = "reference",
  numerics = "mean",
  preserve_range = FALSE,
  standardize = FALSE,
  standardize_robust = FALSE,
  reference = x,
  na.rm = TRUE,
  ...
)
```

Arguments

x	An object from which to construct the reference grid.
target	Can be "all" or list of characters indicating columns of interest. Can also contain assignments (e.g., `target = "Sepal.Length = 2"` or `target = c("Sepal.Length = 2","Species = 'setosa'")` - note the usage of single and double quotes to assign strings within strings). The remaining variables will be fixed.
length	Length of numeric target variables.
factors	Type of summary for factors. Can be "combination" (include all unique values), "reference" (set at the reference level) or "mode" (set at the most common level).
numerics	Type of summary for numeric values. Can be "combination" (include all unique values), any function ("mean", "median", ...) or a value (e.g., `numerics = 0`).
preserve_range	In the case of combinations between numeric variables and factors, setting `preserve_range = TRUE` removes observations where the value of the numeric variable is originally not present in the range of its factor level.
standardize	The numeric target value is spread as deviations from the mean, with the central value being the mean (or the median if `standardize_robust` is TRUE). For instance, if `x` is a vector of mean 1 and SD 2.5, and a standardized grid is required of length 3, the result will be `c(Mean-1*SD,Mean,Mean+1*SD)`, i.e., `c(-1.5,1,3.5)`. Each value represents deviations (in terms of SD or MAD) from the central value. This needs the `length` argument to be an even integer, so that the central value represent the mean.

standardize_robust	Standardization based on median and MAD (a robust equivalent of the SD).
reference	The reference vector from which to compute the mean and SD.
na.rm	Remove NaNs.
...	Arguments passed to or from other methods.

Value

Reference grid data frame.

Examples

```
library(modelbased)

visualisation_matrix(iris, target = "Sepal.Length")
visualisation_matrix(iris, target = "Sepal.Length", factors = "combinations")
visualisation_matrix(iris, target = c("Sepal.Length", "Species"), length = 3)
visualisation_matrix(iris, target = c("Sepal.Length", "Species"), numerics = 0)
visualisation_matrix(iris, target = c("Sepal.Length = 3", "Species"))
visualisation_matrix(iris, target = c("Sepal.Length = c(3, 1)", "Species = 'setosa'"))
visualisation_matrix(iris, target = "Sepal.Length", standardize = TRUE, length = 3)
```

---

zero_crossings *Find zero crossings of a vector*

Description

Find zero crossings of a vector, i.e., indices when the numeric variable crosses 0.

Usage

```
zero_crossings(x)
```

Arguments

x	A numeric vector.

Value

Vector of zero crossings.

See Also

Based on the `uniroot.all` function from the rootSolve package.

Examples

```
x <- sin(seq(0, 4 * pi, length.out = 100))
plot(x)
zero_crossings(x)
```

Index