Package ‘modelbased’

March 12, 2020

Type Package

Title Estimation of Model-Based Predictions, Contrasts and Means

Version 0.1.2

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

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

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

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.

Imports insight (>= 0.7.1), bayestestR (>= 0.4.0), parameters (>= 0.3.0), emmeans, graphics, stats, utils

Suggests coda, covr, dplyr, effectsize, gganimate, ggplot2, knitr, lme4, MASS, merTools, rmarkdown, rstanarm, brms, see, testthat

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Author Dominique Makowski [aut, cre] (<https://orcid.org/0000-0001-5375-9967>),
Daniel Lüdecke [aut] (<https://orcid.org/0000-0002-8895-3206>),
Mattan S. Ben-Shachar [aut] (<https://orcid.org/0000-0002-4287-4801>)

Repository CRAN

Date/Publication 2020-03-12 06:00:03 UTC

R topics documented:

as.numeric_ifnumeric 2
estimate_contrasts 3
estimate_contrasts.lm 4

estimate_contrasts.stanreg 5
estimate_means 7
estimate_means.lm 8
estimate_means.stanreg 9
estimate_response 11
estimate_response.glm 12
estimate_response.stanreg 14
estimate_slopes 17
estimate_slopes.lm 18
estimate_slopes.stanreg 19
estimate_smooth 20
estimate_smooth.stanreg 21
find_inversions 22
reshape_draws 23
smoothing 24
visualisation_matrix 24
zero_crossings 26

Index **27**

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"))
```

`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)


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 = 0.89,
  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)

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

  model <- stan_glm(Sepal.Width ~ Species * Petal.Length_factor, data = data)
  estimate_contrasts(model)
  estimate_contrasts(model, fixed = "Petal.Length_factor")

  model <- stan_glm(Sepal.Width ~ Species * Petal.Width, data = iris)
  estimate_contrasts(model)
  estimate_contrasts(model, fixed = "Petal.Width")
  estimate_contrasts(model, modulate = "Petal.Width", length = 4)

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

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

Description

See the documentation for your object's class:

- Frequentist models
- Bayesian models

Usage

```
estimate_means(
  model,
  levels = NULL,
  fixed = NULL,
  modulate = NULL,
  transform = "response",
  length = 10,
  ...
)
```

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.

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

Value

A dataframe of estimated marginal means.

`estimate_means.lm` *Estimate marginal means*

Description

Estimate marginal means

Usage

```
## S3 method for class 'lm'
estimate_means(
  model,
  levels = NULL,
  fixed = NULL,
  modulate = NULL,
  transform = "response",
  length = 10,
  ci = 0.95,
  ...
)
```

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.

`ci` Confidence Interval (CI) level. Default to 0.95 (95%).

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

Value

A data frame of estimated marginal means.

Examples

```
library(modelbased)

model <- lm(Petal.Length ~ Sepal.Width + Species, data = iris)
estimate_means(model)
estimate_means(model, modulate = "Sepal.Width")

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")
}
```

`estimate_means.stanreg` *Estimate marginal means*

Description

Estimate marginal means

Usage

```
## S3 method for class 'stanreg'
estimate_means(
  model,
  levels = NULL,
  fixed = NULL,
  modulate = NULL,
  transform = "response",
  length = 10,
  centrality = "median",
  ci = 0.89,
  ci_method = "hdi",
  ...
)
```

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.

`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`).

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

Value

A dataframe of estimated marginal means.

Examples

```
library(modelbased)

data <- iris
```

```
data$Petal.Length_factor <- ifelse(data$Petal.Length < 4.2, "A", "B")

if (require("rstanarm")) {
  model <- stan_glm(Sepal.Width ~ Species * Petal.Length_factor, data = data)
  estimate_means(model)

  model <- stan_glm(Petal.Length ~ Sepal.Width * Species, data = iris)
  estimate_means(model)
  estimate_means(model, modulate = "Sepal.Width")
  estimate_means(model, fixed = "Sepal.Width")
}
```

estimate_response   *Generates predictions*

Description

See the documentation for your object's class:

- Bayesian models (stanreg and brms)
- Frequentist models

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.

Usage

```
estimate_response(
  model,
  data = NULL,
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  ...
)

estimate_link(
  model,
  data = "grid",
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  ...
)
```

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).
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.
random	Should it take the random effects into account? Can be TRUE, FALSE or a formula indicating which group-level parameters to condition on when making predictions. The data argument may include new levels of the grouping factors that were specified when the model was estimated, in which case the resulting posterior predictions marginalize over the relevant variables (see posterior_predict.stanreg).
length	Passed to visualisation_matrix if data = "grid".
preserve_range	Passed to visualisation_matrix if data = "grid".
...	Arguments passed to or from other methods.

Value

A data frame of predicted values.

---

estimate_response.glm *Generates predictions for Frequentist models*

Description

Generates predictions for Frequentist models

Usage

```
## S3 method for class 'glm'
estimate_response(
  model,
  data = NULL,
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  predict = "response",
  ci = 0.95,
  ...
)

## S3 method for class 'glm'
```

```
estimate_link(
  model,
  data = "grid",
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  predict = "link",
  ci = 0.95,
  ...
)
```

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`).

`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.

`random` Should it take the random effects into account? Can be `TRUE`, `FALSE` or a formula indicating which group-level parameters to condition on when making predictions. The data argument may include new levels of the grouping factors that were specified when the model was estimated, in which case the resulting posterior predictions marginalize over the relevant variables (see `posterior_predict.stanreg`).

`length` Passed to `visualisation_matrix` if `data = "grid"`.

`preserve_range` Passed to `visualisation_matrix` if `data = "grid"`.

`predict` Can be "response" (default) or "link". The former predicts the the outcome per se, while the latter predicts the link function (i.e., the regression "line"), equivalent to estimating the `fit`. In other words, `estimate_response(model,predict="link")` is equivalent to `estimate_link(model)`.

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

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

Value

A dataframe of predicted values.

Examples

```
library(modelbased)

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

```
if (require("lme4")) {
  model <- lmer(Sepal.Width ~ Petal.Length + (1 | Species), data = iris)
  estimate_response(model)
  estimate_link(model)
}
```

estimate_response.stanreg

Generates predictions for Bayesian models

Description

Generates predictions for Bayesian models

Usage

```
## S3 method for class 'stanreg'
estimate_response(
  model,
  data = NULL,
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  predict = "response",
  keep_draws = FALSE,
  draws = NULL,
  seed = NULL,
  centrality = "median",
  ci = 0.89,
  ci_method = "hdi",
  ...
)

## S3 method for class 'stanreg'
estimate_link(
  model,
  data = "grid",
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  predict = "link",
  keep_draws = FALSE,
  draws = NULL,
  seed = NULL,
  centrality = "median",
  ci = 0.89,
  ci_method = "hdi",
  ...
)

## S3 method for class 'data.frame'
estimate_response(
  model,
  data = NULL,
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  predict = "link",
  keep_draws = FALSE,
  draws = NULL,
  seed = NULL,
  centrality = "median",
  ci = 0.89,
  ci_method = "hdi",
  ...
)

## S3 method for class 'data.frame'
estimate_link(
  model,
  data = "grid",
  transform = "response",
  random = FALSE,
  length = 25,
  preserve_range = TRUE,
  predict = "link",
  keep_draws = FALSE,
  draws = NULL,
  seed = NULL,
  centrality = "median",
  ci = 0.89,
  ci_method = "hdi",
  ...
)
```

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`).

`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.

random — Should it take the random effects into account? Can be TRUE, FALSE or a formula indicating which group-level parameters to condition on when making predictions. The data argument may include new levels of the grouping factors that were specified when the model was estimated, in which case the resulting posterior predictions marginalize over the relevant variables (see posterior_predict.stanreg).

length — Passed to visualisation_matrix if data = "grid".

preserve_range — Passed to visualisation_matrix if data = "grid".

predict — Can be "response" (default) or "link". The former predicts the the outcome per se, while the latter predicts the link function (i.e., the regression "line"), equivalent to estimating the fit. In other words, estimate_response(model,predict="link") is equivalent to estimate_link(model).

keep_draws — If FALSE, will summarise the posterior the obtained distributions. If TRUE, will keep all prediction iterations (draws).

draws — An integer indicating the number of draws to return. The default and maximum number of draws is the size of the posterior sample contained in the model.

seed — An optional seed to use.

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).

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

Value

A dataframe of predicted values.

Examples

```
library(modelbased)

if (require("rstanarm") && require("brms")) {
  model <- stan_glm(Sepal.Width ~ Species * Petal.Length, data = iris)
  estimate_response(model)
  estimate_link(model)

  model <- stan_glmer(Sepal.Width ~ Petal.Length + (1 | Species), data = iris)
  estimate_response(model)
  estimate_link(model)

  library(brms)
  model <- brms::brm(Sepal.Width ~ Petal.Length, data = iris)
  estimate_response(model)
  estimate_link(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,
  ...
)
```

Arguments

model	A statistical model.
trend	A character indicating the name of the numeric variable for which to compute the slopes.
levels	A character vectors 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.
...	Arguments passed to or from other methods.

Value

A dataframe 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 statistical model.

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

`levels` A character vectors 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,
  centrality = "median",
  ci = 0.89,
  ci_method = "hdi",
  test = c("pd", "rope"),
  rope_range = "default",
  rope_ci = 1,
  ...
)
```

Arguments

`model`	A statistical model.
`trend`	A character indicating the name of the numeric variable for which to compute the slopes.
`levels`	A character vectors 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`.
`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.

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 summarise 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 characterised 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,
  length = 200,
  transform = "response",
  ...
)
```

Arguments

`model` A statistical model.

`smooth` A character indicating the name of the "smooth" term.

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

`length` Passed to `visualisation_matrix` if `data = "grid"`.

`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.

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

Details

See the documentation for your object's class:

- Bayesian models (stanreg and brms)

Value

A dataframe of linear description of non-linear terms.

---

estimate_smooth.stanreg

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

Description

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

Usage

```
## S3 method for class 'stanreg'
estimate_smooth(
  model,
  smooth = NULL,
  levels = NULL,
  length = 200,
  transform = "response",
  centrality = "median",
  ...
)
```

Arguments

`model` A statistical model.

`smooth` A character indicating the name of the "smooth" term.

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

`length` Passed to `visualisation_matrix` if `data = "grid"`.

`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.

`centrality` The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: `"median"`, `"mean"`, `"MAP"` or `"all"`.

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

Examples

```
library(modelbased)

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

---

reshape_draws *Reshape estimations with Bayesian posterior draws to long format*

Description

Reshape data.frame of estimations with Bayesian posterior draws to long format.

Usage

```
reshape_draws(draws)
```

Arguments

draws	data.frame containing posterior draws obtained from `estimate_response` or `estimate_link`.

Value

Data frame of reshaped draws in long format.

Examples

```
if (require("rstanarm")) {
  model <- stan_glm(Sepal.Width ~ Species * Petal.Length, data = iris)
  estimates <- estimate_response(model, keep_draws = TRUE, draws = 200)
  reshape_draws(estimates)
}
```

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

as.numeric_ifnumeric, 2

Bayesian models, *3*, *7*
Bayesian models (stanreg and brms), *11*, *17*, *21*

ci, *5*, *6*, *10*, *13*, *16*, *18*, *20*

estimate_contrasts, 3
estimate_contrasts.lm, 4
estimate_contrasts.stanreg, 5
estimate_link *(*estimate_response*)*, 11
estimate_link.data.frame *(*estimate_response.stanreg*)*, 14
estimate_link.glm *(*estimate_response.glm*)*, 12
estimate_link.stanreg *(*estimate_response.stanreg*)*, 14
estimate_means, 7
estimate_means.lm, 8
estimate_means.stanreg, 9
estimate_response, 11
estimate_response.data.frame *(*estimate_response.stanreg*)*, 14
estimate_response.glm, 12
estimate_response.stanreg, 14
estimate_slopes, 17
estimate_slopes.lm, 18
estimate_slopes.stanreg, 19
estimate_smooth, 20
estimate_smooth.stanreg, 21
eti, *6*, *10*, *16*, *20*

find_inversions, 22
Frequentist models, *3*, *7*, *11*, *17*

hdi, *6*, *10*, *16*, *20*

loess, *24*

p_direction, *6*, *20*

reshape_draws, 23
rope, *6*, *20*

si, *6*, *10*, *16*, *20*
smoothing, 24

visualisation_matrix, *12*, *13*, *15*, *16*, *21*, *22*, 24

zero_crossings, 26