Package 'bayestestR'

October 20, 2019

Type Package

Title Understand and Describe Bayesian Models and Posterior Distributions

Version 0.4.0

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

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

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

Description Provides utilities to describe posterior distributions and Bayesian models. It includes point-estimates such as Maximum A Posteriori (MAP), measures of dispersion (Highest Density Interval - HDI; Kruschke, 2015 <doi:10.1016/C2012-0-00477-2>) and indices used for null-hypothesis testing (such as ROPE percentage, pd and Bayes factors).

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.0)

Imports insight (>= 0.6.0), methods, stats, utils

Suggests BayesFactor, bridgesampling, brms, broom, covr, dplyr, emmeans, GGally, ggplot2, ggridges, KernSmooth, knitr, MASS, mclust, lme4, logspline, performance, rmarkdown, rstan, rstanarm, see, stringr, testthat, tidyr

RoxygenNote 6.1.1

Language en-GB

VignetteBuilder knitr

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>), Michael D. Wilson [aut] (<https://orcid.org/0000-0003-4143-7308>), Paul-Christian Bürkner [rev], Tristan Mahr [rev] (<https://orcid.org/0000-0002-8890-5116>),

Henrik Singmann [ctb] (<https://orcid.org/0000-0002-4842-3657>),
Quentin F. Gronau [ctb] (<https://orcid.org/0000-0001-5510-6943>)

Repository CRAN

Date/Publication 2019-10-20 07:20:02 UTC

R topics documented:

area_under_curve *Area under the Curve (AUC)*

Description

Based on the DescTools AUC function. It can calculate the area under the curve with a naive algorithm or a more elaborated spline approach. The curve must be given by vectors of xy-coordinates. This function can handle unsorted x values (by sorting x) and ties for the x values (by ignoring duplicates).

Usage

```
area_under_curve(x, y, method = c("trapezoid", "step", "spline"), ...)

auc(x, y, method = c("trapezoid", "step", "spline"), ...)
```

Arguments

x	Vector of x values.
y	Vector of y values.
method	Method to compute the Area Under the Curve (AUC). Can be "trapezoid" (default), "step" or "spline". If "trapezoid", the curve is formed by connecting all points by a direct line (composite trapezoid rule). If "step" is chosen then a stepwise connection of two points is used. For calculating the area under a spline interpolation the splinefun function is used in combination with integrate.
...	Arguments passed to or from other methods.

See Also

DescTools

Examples

```
library(bayestestR)
posterior <- distribution_normal(1000)

dens <- estimate_density(posterior)
dens <- dens[dens$x > 0, ]
x <- dens$x
y <- dens$y

area_under_curve(x, y, method = "trapezoid")
area_under_curve(x, y, method = "step")
area_under_curve(x, y, method = "spline")
```

as.data.frame.density *Coerce to a Data Frame*

Description

Coerce to a Data Frame

Usage

```
## S3 method for class 'density'
as.data.frame(x, ...)
```

Arguments

x	any R object.
...	additional arguments to be passed to or from methods.

as.numeric.map_estimate *Convert to Numeric*

Description

Convert to Numeric

Usage

```
## S3 method for class 'map_estimate'
as.numeric(x, ...)

## S3 method for class 'p_direction'
as.numeric(x, ...)

## S3 method for class 'p_map'
as.numeric(x, ...)

## S3 method for class 'p_rope'
as.numeric(x, ...)

## S3 method for class 'p_significance'
as.numeric(x, ...)
```

Arguments

x	object to be coerced or tested.
...	further arguments passed to or from other methods.

bayesfactor *Bayes Factors (BF)*

Description

This function compte the Bayes factors (BFs) that are appropriate to the input. For vectors or single models, it will compute BFs for single parameters, or is hypothesis is specified, BFs for restricted models. For multiple models, it will return the BF corresponding to comparison between models and if a model comparison is passed, it will compute the inclusion BF.

For a complete overview of these functions, read the Bayes factor vignette.

Usage

```
bayesfactor(..., prior = NULL, direction = "two-sided", null = 0,
  hypothesis = NULL, effects = c("fixed", "random", "all"),
  verbose = TRUE, denominator = 1, match_models = FALSE,
  prior_odds = NULL)
```

Arguments

...	A numeric vector, model object(s), or the output from bayesfactor_models.
prior	An object representing a prior distribution (see Details).
direction	Test type (see details). One of 0, "two-sided" (default, two tailed), -1, "left" (left tailed) or 1, "right" (right tailed).
null	Value of the null, either a scaler (for point-null) or a a range (for a interval-null).
hypothesis	Deprecated in favour of null.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
verbose	Toggle off warnings.
denominator	Either an integer indicating which of the models to use as the denominator, or a model to be used as a denominator. Ignored for BFBayesFactor.
match_models	See details.
prior_odds	Optional vector of prior odds for the models. See BayesFactor::priorOdds<-.

Value

Some type of Bayes factor, depending on the input. See bayesfactor_parameters, bayesfactor_models or bayesfactor_inclusion

Examples

```
library(bayestestR)

# Vectors
prior <- distribution_normal(1000, mean = 0, sd = 1)
posterior <- distribution_normal(1000, mean = .5, sd = .3)

bayesfactor(posterior, prior = prior)
## Not run:
# rstanarm models
# ---------------
library(rstanarm)
model <- stan_lmer(extra ~ group + (1 | ID), data = sleep)
bayesfactor(model)

## End(Not run)

# Frequentist models
# ---------------
m0 <- lm(extra ~ 1, data = sleep)
m1 <- lm(extra ~ group, data = sleep)
m2 <- lm(extra ~ group + ID, data = sleep)

comparison <- bayesfactor(m0, m1, m2)
comparison

bayesfactor(comparison)
```

---

bayesfactor_inclusion *Inclusion Bayes Factors for testing predictors across Bayesian models*

Description

Inclusion Bayes Factors for testing predictors across Bayesian models

Usage

```
bayesfactor_inclusion(models, match_models = FALSE, prior_odds = NULL,
  ...)
```

Arguments

models	An object of class bayesfactor_models or BFBayesFactor.
match_models	See details.
prior_odds	Optional vector of prior odds for the models. See BayesFactor::priorOdds<-.
...	Arguments passed to or from other methods.

Details

Inclusion Bayes factors answer the question: Are the observed data more probable under models with a particular effect, than they are under models without that particular effect? In other words, on average - are models with effect X more likely to have produced the observed data than models without effect X?

For more info, see the Bayes factors vignette.

Match Models: If `match_models=FALSE` (default), Inclusion BFs are computed by comparing all models with a predictor against all models without that predictor. If `TRUE`, comparison is restricted to models that (1) do not include any interactions with the predictor of interest; (2) for interaction predictors, averaging is done only across models that containe the main effect from which the interaction predictor is comprised.

Value

a data frame containing the prior and posterior probabilities, and BF for each effect.

Note

Random effects in the `lme` style will be displayed as interactions: i.e., `(X|G)` will become `1:G` and `X:G`.

Author(s)

Mattan S. Ben-Shachar

References

- Hinne, M., Gronau, Q. F., van den Bergh, D., and Wagenmakers, E. (2019, March 25). A conceptual introduction to Bayesian Model Averaging. doi: 10.31234/osf.io/wgb64
- Clyde, M. A., Ghosh, J., & Littman, M. L. (2011). Bayesian adaptive sampling for variable selection and model averaging. Journal of Computational and Graphical Statistics, 20(1), 80-101.
- Mathot. S. (2017). Bayes like a Baws: Interpreting Bayesian Repeated Measures in JASP [Blog post]. Retrieved from https://www.cogsci.nl/blog/interpreting-bayesian-repeated-measures-in-jasp

Examples

```
library(bayestestR)

# Using bayesfactor_models:
# -----------------------------
mo0 <- lm(Sepal.Length ~ 1, data = iris)
mo1 <- lm(Sepal.Length ~ Species, data = iris)
mo2 <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
mo3 <- lm(Sepal.Length ~ Species * Petal.Length, data = iris)
```

```
BFmodels <- bayesfactor_models(mo1, mo2, mo3, denominator = mo0)
bayesfactor_inclusion(BFmodels)
## Not run:
# BayesFactor
# ------------------------------
library(BayesFactor)

BF <- generalTestBF(len ~ supp * dose, ToothGrowth, progress = FALSE)

bayesfactor_inclusion(BF)

# compare only matched models:
bayesfactor_inclusion(BF, match_models = TRUE)

## End(Not run)
```

`bayesfactor_models` *Bayes Factors (BF) for model comparison*

Description

This function computes or extracts Bayes factors from fitted models.

Usage

```
bayesfactor_models(..., denominator = 1, verbose = TRUE)
```

Arguments

`...`	Fitted models (see details), all fit on the same data, or a single `BFBayesFactor` object (see 'Details').
`denominator`	Either an integer indicating which of the models to use as the denominator, or a model to be used as a denominator. Ignored for `BFBayesFactor`.
`verbose`	Toggle off warnings.

Details

If the passed models are supported by **insight** the DV of all models will be tested for equality (else this is assumed to be true), and the models' terms will be extracted (allowing for follow-up analysis with `bayesfactor_inclusion`).

- For `brmsfit` or `stanreg` models, Bayes factors are computed using the **bridgesampling** package.
  - `brmsfit` models must have been fitted with `save_all_pars = TRUE`.
  - `stanreg` models must have been fitted with a defined `diagnostic_file`.
- For `BFBayesFactor`, `bayesfactor_models()` is mostly a wraparoud `BayesFactor::extractBF()`.

- For all other model types (supported by **insight**), BIC approximations are used to compute Bayes factors.

In order to correctly and precisely estimate Bayes factors, a rule of thumb are the 4 P's: **P**roper **P**riors and **P**lentiful **P**osterior (i.e. probably at leat 40,000 samples instead of the default of 4,000).

A Bayes factor greater than 1 can be interpereted as evidence against the compared-to model (the denominator). One convention is that a Bayes factor greater than 3 can be considered as "substantial" evidence against the denominator model (and vice versa, a Bayes factor smaller than 1/3 indicates substantial evidence in favor of the denominator model) (*Wetzels et al. 2011*).

See also the Bayes factors vignette.

Value

A data frame containing the models' formulas (reconstructed fixed and random effects) and their BFs, that prints nicely.

Author(s)

Mattan S. Ben-Shachar

References

- Gronau, Q. F., Wagenmakers, E. J., Heck, D. W., and Matzke, D. (2019). A simple method for comparing complex models: Bayesian model comparison for hierarchical multinomial processing tree models using Warp-III bridge sampling. Psychometrika, 84(1), 261-284.
- Kass, R. E., and Raftery, A. E. (1995). Bayes Factors. Journal of the American Statistical Association, 90(430), 773-795.
- Robert, C. P. (2016). The expected demise of the Bayes factor. Journal of Mathematical Psychology, 72, 33–37.
- Wagenmakers, E. J. (2007). A practical solution to the pervasive problems of p values. Psychonomic bulletin & review, 14(5), 779-804.
- Wetzels, R., Matzke, D., Lee, M. D., Rouder, J. N., Iverson, G. J., and Wagenmakers, E.-J. (2011). Statistical Evidence in Experimental Psychology: An Empirical Comparison Using 855 t Tests. Perspectives on Psychological Science, 6(3), 291–298. doi: 10.1177/1745691611406923

Examples

```
# With lm objects:
# ----------------
lm1 <- lm(Sepal.Length ~ 1, data = iris)
lm2 <- lm(Sepal.Length ~ Species, data = iris)
lm3 <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
lm4 <- lm(Sepal.Length ~ Species * Petal.Length, data = iris)
bayesfactor_models(lm1, lm2, lm3, lm4, denominator = 1)
bayesfactor_models(lm2, lm3, lm4, denominator = lm1) # same result
bayesfactor_models(lm1, lm2, lm3, lm4, denominator = lm1) # same result
```

```
# With lmerMod objects:
# ---------------------
library(lme4)
lmer1 <- lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)
lmer2 <- lmer(Sepal.Length ~ Petal.Length + (Petal.Length | Species), data = iris)
lmer3 <- lmer(
  Sepal.Length ~ Petal.Length + (Petal.Length | Species) + (1 | Petal.Width),
  data = iris
)
bayesfactor_models(lmer1, lmer2, lmer3, denominator = 1)
bayesfactor_models(lmer1, lmer2, lmer3, denominator = lmer1)
## Not run:
# rstanarm models
# ---------------------
# (note that a unique diagnostic_file MUST be specified in order to work)
library(rstanarm)
stan_m0 <- stan_glm(Sepal.Length ~ 1,
  data = iris,
  family = gaussian(),
  diagnostic_file = file.path(tempdir(), "df0.csv")
)
stan_m1 <- stan_glm(Sepal.Length ~ Species,
  data = iris,
  family = gaussian(),
  diagnostic_file = file.path(tempdir(), "df1.csv")
)
stan_m2 <- stan_glm(Sepal.Length ~ Species + Petal.Length,
  data = iris,
  family = gaussian(),
  diagnostic_file = file.path(tempdir(), "df2.csv")
)
bayesfactor_models(stan_m1, stan_m2, denominator = stan_m0)


# brms models
# --------------------
# (note the save_all_pars MUST be set to TRUE in order to work)
library(brms)
brm1 <- brm(Sepal.Length ~ 1, data = iris, save_all_pars = TRUE)
brm2 <- brm(Sepal.Length ~ Species, data = iris, save_all_pars = TRUE)
brm3 <- brm(
  Sepal.Length ~ Species + Petal.Length,
  data = iris,
  save_all_pars = TRUE
)

bayesfactor_models(brm1, brm2, brm3, denominator = 1)


# BayesFactor
# ---------------------------
library(BayesFactor)
data(puzzles)
```

```
BF <- anovaBF(RT ~ shape * color + ID,
  data = puzzles,
  whichRandom = "ID", progress = FALSE
)
BF
bayesfactor_models(BF) # basically the same

## End(Not run)
```

`bayesfactor_parameters` *Savage-Dickey density ratio Bayes Factor (BF)*

Description

This method computes Bayes factors against the null (either a point or an interval), bases on prior and posterior samples of a single parameter. This Bayes factor indicates the degree by which the mass of the posterior distribution has shifted further away from or closer to the null value(s) (relative to the prior distribution), thus indicating if the null value has become less or more likely given the observed data.

When the null is an interval, the Bayes factor is computed by comparing the prior and posterior odds of the parameter falling within or outside the null; When the null is a point, a Savage-Dickey density ratio is computed, which is also an approximation of a Bayes factor comparing the marginal likelihoods of the model against a model in which the tested parameter has been restricted to the point null.

For info on specifying correct priors for factors with more than 2 levels, see the Bayes factors vignette.

For more info, see the Bayes factors vignette.

Usage

```
bayesfactor_parameters(posterior, prior = NULL,
  direction = "two-sided", null = 0, verbose = TRUE, ...)

bayesfactor_savagedickey(posterior, prior = NULL,
  direction = "two-sided", null = 0, verbose = TRUE,
  hypothesis = NULL, ...)

## S3 method for class 'numeric'
bayesfactor_parameters(posterior, prior = NULL,
  direction = "two-sided", null = 0, verbose = TRUE, ...)

## S3 method for class 'stanreg'
bayesfactor_parameters(posterior, prior = NULL,
```

```
  direction = "two-sided", null = 0, verbose = TRUE,
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), ...)

## S3 method for class 'brmsfit'
bayesfactor_parameters(posterior, prior = NULL,
  direction = "two-sided", null = 0, verbose = TRUE,
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), ...)

## S3 method for class 'emmGrid'
bayesfactor_parameters(posterior, prior = NULL,
  direction = "two-sided", null = 0, verbose = TRUE, ...)

## S3 method for class 'data.frame'
bayesfactor_parameters(posterior, prior = NULL,
  direction = "two-sided", null = 0, verbose = TRUE, ...)
```

Arguments

posterior	A numerical vector, stanreg / brmsfit object, emmGrid or a data frame - representing a posterior distribution(s) from (see Details).
prior	An object representing a prior distribution (see Details).
direction	Test type (see details). One of 0, "two-sided" (default, two tailed), -1, "left" (left tailed) or 1, "right" (right tailed).
null	Value of the null, either a scaler (for point-null) or a a range (for a interval-null).
verbose	Toggle off warnings.
...	Currently not used.
hypothesis	Deprecated in favour of null.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

This method is used to compute Bayes factors based on prior and posterior distributions. When posterior is a model (stanreg, brmsfit), posterior and prior samples are extracted for each parameter, and Savage-Dickey Bayes factors are computed for each parameter.

NOTE: For brmsfit models, the model must have been fitted with *custom (non-default)* priors. See example below.

Setting the correct prior**:** It is important to provide the correct prior for meaningful results.

- When posterior is a numerical vector, prior should also be a numerical vector.

- When `posterior` is a `data.frame`, `prior` should also be a `data.frame`, with matching column order.
- When `posterior` is a `stanreg` or `brmsfit` model:
  - `prior` can be set to `NULL`, in which case prior samples are drawn internally.
  - `prior` can also be a model equvilant to `posterior` but with samples from the priors *only*.
- When `posterior` is an `emmGrid` object:
  - `prior` should be the `stanreg` or `brmsfit` model used to create the `emmGrid` objects.
  - `prior` can also be an `emmGrid` object equvilant to `posterior` but created with a model of priors samples *only*.

One-sided Tests (setting an order restriction): One sided tests (controlled by `direction`) are conducted by restricting the prior and posterior of the non-null values (the "alternative") to one side of the null only (*Morey & Wagenmakers, 2013*). For example, if we have a prior hypothesis that the parameter should be positive, the alternative will be restricted to the region to the right of the null (point or interval).

Interpreting Bayes Factors: A Bayes factor greater than 1 can be interpereted as evidence against the null, at which one convention is that a Bayes factor greater than 3 can be considered as "substantial" evidence against the null (and vice versa, a Bayes factor smaller than 1/3 indicates substantial evidence in favor of the null-model) (*Wetzels et al. 2011*).

Value

A data frame containing the Bayes factor representing evidence *against* the null.

Author(s)

Mattan S. Ben-Shachar

References

- Wagenmakers, E. J., Lodewyckx, T., Kuriyal, H., and Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage-Dickey method. Cognitive psychology, 60(3), 158-189.
- Wetzels, R., Matzke, D., Lee, M. D., Rouder, J. N., Iverson, G. J., and Wagenmakers, E.-J. (2011). Statistical Evidence in Experimental Psychology: An Empirical Comparison Using 855 t Tests. Perspectives on Psychological Science, 6(3), 291–298. doi: 10.1177/1745691611406923
- Heck, D. W. (2019). A caveat on the Savage–Dickey density ratio: The case of computing Bayes factors for regression parameters. British Journal of Mathematical and Statistical Psychology, 72(2), 316-333.
- Morey, R. D., & Wagenmakers, E. J. (2014). Simple relation between Bayesian order-restricted and point-null hypothesis tests. Statistics & Probability Letters, 92, 121-124.
- Morey, R. D., & Rouder, J. N. (2011). Bayes factor approaches for testing interval null hypotheses. Psychological methods, 16(4), 406.

Examples

```
library(bayestestR)

prior <- distribution_normal(1000, mean = 0, sd = 1)
posterior <- distribution_normal(1000, mean = .5, sd = .3)

bayesfactor_parameters(posterior, prior)
## Not run:
# rstanarm models
# ---------------
library(rstanarm)
contrasts(sleep$group) <- contr.bayes # see vingette
stan_model <- stan_lmer(extra ~ group + (1 | ID), data = sleep)
bayesfactor_parameters(stan_model)
bayesfactor_parameters(stan_model, null = rope_range(stan_model))

# emmGrid objects
# ---------------
library(emmeans)
group_diff <- pairs(emmeans(stan_model, ~group))
bayesfactor_parameters(group_diff, prior = stan_model)

# brms models
# -----------
library(brms)
contrasts(sleep$group) <- contr.bayes # see vingette
my_custom_priors <-
  set_prior("student_t(3, 0, 1)", class = "b") +
  set_prior("student_t(3, 0, 1)", class = "sd", group = "ID")

brms_model <- brm(extra ~ group + (1 | ID),
  data = sleep,
  prior = my_custom_priors
)
bayesfactor_parameters(brms_model)

## End(Not run)
```

`bayesfactor_restricted` *Bayes Factors (BF) for Order Restricted Models*

Description

This method computes Bayes factors for comparing a model with an order restrictions on its parameters with the fully unrestricted model. *Note that this method should only be used for confirmatory analyses*.

For info on specifying correct priors for factors with more than 2 levels, see the Bayes factors vignette.

For more info, see the Bayes factors vignette.

Usage

```
bayesfactor_restricted(posterior, hypothesis, prior = NULL,
  verbose = TRUE, ...)

## S3 method for class 'stanreg'
bayesfactor_restricted(posterior, hypothesis,
  prior = NULL, verbose = TRUE, effects = c("fixed", "random",
  "all"), ...)

## S3 method for class 'brmsfit'
bayesfactor_restricted(posterior, hypothesis,
  prior = NULL, verbose = TRUE, effects = c("fixed", "random",
  "all"), ...)

## S3 method for class 'emmGrid'
bayesfactor_restricted(posterior, hypothesis,
  prior = NULL, verbose = TRUE, ...)
```

Arguments

`posterior`	A `stanreg` / `brmsfit` object, `emmGrid` or a data frame - representing a posterior distribution(s) from (see Details).
`hypothesis`	A character vector specifying the restrictions as logical conditions (see examples below).
`prior`	An object representing a prior distribution (see Details).
`verbose`	Toggle off warnings.
`...`	Currently not used.
`effects`	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.

Details

This method is used to compute Bayes factors for order-restricted models vs un-restricted models by setting an order restriction on the prior and posterior distributions (*Morey & Wagenmakers, 2013*).

(Though it is possible to use `bayesfactor_restricted` to test interval restrictions, it is more suitable for testing order restrictions (see examples)).

When `posterior` is a model (`stanreg`, `brmsfit`), posterior and prior samples are extracted for each parameter, and Savage-Dickey Bayes factors are computed for each parameter.

NOTE: For `brmsfit` models, the model must have been fitted with *custom (non-default)* priors. See example below.

Setting the correct `prior`: It is important to provide the correct `prior` for meaningful results.

- When `posterior` is a `data.frame`, `prior` should also be a `data.frame`, with matching column order.
- When `posterior` is a `stanreg` or `brmsfit` model:
  - `prior` can be set to `NULL`, in which case prior samples are drawn internally.
  - `prior` can also be a model equvilant to `posterior` but with samples from the priors *only*.
- When `posterior` is an `emmGrid` object:
  - `prior` should be the `stanreg` or `brmsfit` model used to create the `emmGrid` objects.
  - `prior` can also be an `emmGrid` object equvilant to `posterior` but created with a model of priors samples *only*.

Interpreting Bayes Factors: A Bayes factor greater than 1 can be interpereted as evidence against the null, at which one convention is that a Bayes factor greater than 3 can be considered as "substantial" evidence against the null (and vice versa, a Bayes factor smaller than 1/3 indicates substantial evidence in favor of the null-hypothesis) (*Wetzels et al. 2011*).

Value

A data frame containing the Bayes factor representing evidence *against* the un-restricted model.

References

- Morey, R. D., & Wagenmakers, E. J. (2014). Simple relation between Bayesian order-restricted and point-null hypothesis tests. Statistics & Probability Letters, 92, 121-124.
- Morey, R. D., & Rouder, J. N. (2011). Bayes factor approaches for testing interval null hypotheses. Psychological methods, 16(4), 406.
- Morey, R. D. (Jan, 2015). Multiple Comparisons with BayesFactor, Part 2 – order restrictions. Retrived from https://richarddmorey.org/category/order-restrictions/.

Examples

```
library(bayestestR)
prior <- data.frame(
  X = rnorm(100),
  X1 = rnorm(100),
  X3 = rnorm(100)
)

posterior <- data.frame(
  X = rnorm(100, .4),
  X1 = rnorm(100, -.2),
  X3 = rnorm(100)
)

hyps <- c(
  "X > X1 & X1 > X3",
  "X > X1"
)
```

```
bayesfactor_restricted(posterior, hypothesis = hyps, prior = prior)
## Not run:
# rstanarm models
# ---------------
library(rstanarm)
fit_stan <- stan_glm(mpg ~ wt + cyl + am,
  data = mtcars
)
hyps <- c(
  "am > 0 & cyl < 0",
  "cyl < 0",
  "wt - cyl > 0"
)
bayesfactor_restricted(fit_stan, hypothesis = hyps)

# emmGrid objects
# ---------------
library(emmeans)

# replicating http://bayesfactor.blogspot.com/2015/01/multiple-comparisons-with-bayesfactor-2.html
disgust_data <- read.table(url("http://www.learnbayes.org/disgust_example.txt"), header = TRUE)

contrasts(disgust_data$condition) <- contr.bayes # see vignette
fit_model <- stan_glm(score ~ condition, data = disgust_data, family = gaussian())

em_condition <- emmeans(fit_model, ~condition)
hyps <- c("lemon < control & control < sulfur")

bayesfactor_restricted(em_condition, prior = fit_model, hypothesis = hyps)
# > # Bayes Factor (Order-Restriction)
# >
# >                           Hypothesis P(Prior) P(Posterior) Bayes Factor
# >  lemon < control & control < sulfur     0.17         0.75         4.49
# > ---
# > Bayes factors for the restricted movel vs. the un-restricted model.

## End(Not run)
```

---

check_prior *Check if Prior is Informative*

Description

Performs a simple test to check whether the prior is informative to the posterior. This idea, and the accompanying heuristics, were discussed in this blogpost.

Usage

```
check_prior(model, method = "gelman", ...)
```

Arguments

model	A stanreg, stanfit, or brmsfit object.
method	Can be "gelman" or "lakeland". For the "gelman" method, if the SD of the posterior is more than 0.1 times the SD of the prior, then the prior is considered as informative. For the "lakeland" method, the prior is considered as informative if the posterior falls within the 95% HDI of the prior.
...	Currently not used.

References

https://statmodeling.stat.columbia.edu/2019/08/10/

Examples

```
library(bayestestR)
library(rstanarm)

model <- stan_glm(mpg ~ wt + am, data = mtcars, chains = 1, refresh = 0)
check_prior(model, method = "gelman")
check_prior(model, method = "lakeland")
```

---

ci *Confidence/Credible/Compatibility Interval (CI)*

Description

Compute Confidence/Credible/Compatibility Intervals (CI) for Bayesian and frequentist models. The Documentation is accessible for:

Usage

```
ci(x, ...)

## S3 method for class 'numeric'
ci(x, ci = 0.89, method = "ETI", verbose = TRUE,
  ...)

## S3 method for class 'data.frame'
ci(x, ci = 0.89, method = "ETI", verbose = TRUE,
  ...)

## S3 method for class 'emmGrid'
ci(x, ci = 0.89, method = "ETI", verbose = TRUE,
  ...)

## S3 method for class 'sim.merMod'
ci(x, ci = 0.89, method = "ETI",
```

```
  effects = c("fixed", "random", "all"), parameters = NULL,
  verbose = TRUE, ...)

## S3 method for class 'sim'
ci(x, ci = 0.89, method = "ETI", parameters = NULL,
  verbose = TRUE, ...)

## S3 method for class 'stanreg'
ci(x, ci = 0.89, method = "ETI",
  effects = c("fixed", "random", "all"), parameters = NULL,
  verbose = TRUE, ...)

## S3 method for class 'brmsfit'
ci(x, ci = 0.89, method = "ETI",
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), parameters = NULL, verbose = TRUE,
  ...)

## S3 method for class 'BFBayesFactor'
ci(x, ci = 0.89, method = "ETI",
  verbose = TRUE, ...)

## S3 method for class 'MCMCglmm'
ci(x, ci = 0.89, method = "ETI", verbose = TRUE,
  ...)
```

Arguments

x	A stanreg or brmsfit model, or a vector representing a posterior distribution.
...	Currently not used.
ci	Value or vector of probability of the CI (between 0 and 1) to be estimated. Default to .89 (89%) for Bayesian models and .95 (95%) for frequentist models.
method	Can be 'ETI' (default) or 'HDI'.
verbose	Toggle off warnings.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

- Bayesian models
- Frequentist models

Value

A data frame with following columns:

- `Parameter` The model parameter(s), if `x` is a model-object. If `x` is a vector, this column is missing.
- `CI` The probability of the credible interval.
- `CI_low`, `CI_high` The lower and upper credible interval limits for the parameters.

Note

When it comes to interpretation, we recommend thinking of the CI in terms of an "uncertainty" or "compatibility" interval, the latter being defined as “Given any value in the interval and the background assumptions, the data should not seem very surprising” (*Gelman & Greenland 2019*).

References

Gelman A, Greenland S. Are confidence intervals better termed "uncertainty intervals"? BMJ 2019;l5381. doi: 10.1136/bmj.l5381

Examples

```
library(bayestestR)

posterior <- rnorm(1000)
ci(posterior, method = "ETI")
ci(posterior, method = "HDI")

df <- data.frame(replicate(4, rnorm(100)))
ci(df, method = "ETI", ci = c(.80, .89, .95))
ci(df, method = "HDI", ci = c(.80, .89, .95))

library(rstanarm)
model <- stan_glm(mpg ~ wt, data = mtcars, chains = 2, iter = 200, refresh = 0)
ci(model, method = "ETI", ci = c(.80, .89))
ci(model, method = "HDI", ci = c(.80, .89))
## Not run:
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
ci(model, method = "ETI")
ci(model, method = "HDI")

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
ci(bf, method = "ETI")
ci(bf, method = "HDI")

library(emmeans)
model <- emtrends(model, ~1, "wt")
ci(model, method = "ETI")
ci(model, method = "HDI")
```

```
## End(Not run)
```

`contr.bayes` *Orthonormal Contrast Matrices for Bayesian Estimation*

Description

Returns a design or model matrix of orthonormal contrasts such that the marginal prior on all effects is identical. Implementation from Singmann \& Gronau's `bfrms`, following the description in Rouder, Morey, Speckman, \& Province (2012, p. 363).

Usage

```
contr.bayes(n, contrasts = TRUE)
```

Arguments

`n`	a vector of levels for a factor, or the number of levels.
`contrasts`	logical indicating whether contrasts should be computed.

Details

Though using this factor coding scheme might obscure the interpretation of parameters, it is essential for correct estimation of Bayes factors for contrasts and multi-level order restrictions. See info on specifying correct priors for factors with more than 2 levels in the Bayes factors vignette.

Value

A `matrix` with n rows and k columns, with k=n-1 if contrasts is `TRUE` and k=n if contrasts is `FALSE`.

References

Rouder, J. N., Morey, R. D., Speckman, P. L., \& Province, J. M. (2012). Default Bayes factors for ANOVA designs. *Journal of Mathematical Psychology*, 56(5), 356-374. https://doi.org/10.1016/j.jmp.2012.08.001

Examples

```
## Not run:
contr.bayes(2) # Q_2 in Rouder et al. (2012, p. 363)
#            [,1]
# [1,] -0.7071068
# [2,]  0.7071068

contr.bayes(5) # equivalent to Q_5 in Rouder et al. (2012, p. 363)
#            [,1]       [,2]       [,3]       [,4]
# [1,]  0.0000000  0.8944272  0.0000000  0.0000000
# [2,]  0.0000000 -0.2236068 -0.5000000  0.7071068
# [3,]  0.7071068 -0.2236068 -0.1666667 -0.4714045
```

```
# [4,] -0.7071068 -0.2236068 -0.1666667 -0.4714045
# [5,]  0.0000000 -0.2236068  0.8333333  0.2357023

## check decomposition
Q3 <- contr.bayes(3)
Q3 %*% t(Q3)
#            [,1]       [,2]       [,3]
# [1,]  0.6666667 -0.3333333 -0.3333333
# [2,] -0.3333333  0.6666667 -0.3333333
# [3,] -0.3333333 -0.3333333  0.6666667
## 2/3 on diagonal and -1/3 on off-diagonal elements

## End(Not run)
```

`convert_bayesian_as_frequentist` *Convert (refit) a Bayesian model to frequentist*

Description

Refit Bayesian model as frequentist. Can be useful for comparisons.

Usage

```
convert_bayesian_as_frequentist(model, data = NULL)

bayesian_as_frequentist(model, data = NULL)
```

Arguments

`model`	A Bayesian model.
`data`	Data used by the model. If `NULL`, will try to extract it from the model.

Examples

```
# Rstanarm ----------------------
library(rstanarm)

# Simple regressions
model <- stan_glm(Sepal.Length ~ Petal.Length * Species,
  data = iris, chains = 2, refresh = 0
)
bayesian_as_frequentist(model)

model <- stan_glm(vs ~ mpg,
  family = "binomial",
  data = mtcars, chains = 2, refresh = 0
```

```
)
bayesian_as_frequentist(model)

# Mixed models
model <- stan_glmer(Sepal.Length ~ Petal.Length + (1 | Species),
  data = iris, chains = 2, refresh = 0
)
bayesian_as_frequentist(model)

model <- stan_glmer(vs ~ mpg + (1 | cyl),
  family = "binomial",
  data = mtcars, chains = 2, refresh = 0
)
bayesian_as_frequentist(model)
```

`density_at` *Density Probability at a Given Value*

Description

Compute the density value at a given point of a distribution (i.e., the value of the `y` axis of a value `x` of a distribution).

Usage

```
density_at(posterior, x, precision = 2^10, method = "kernel", ...)
```

Arguments

`posterior`	Vector representing a posterior distribution.
`x`	The value of which to get the approximate probability.
`precision`	Number of points of density data. See the `n` parameter in density.
`method`	Density estimation method. Can be `"kernel"` (default), `"logspline"` or `"KernSmooth"`.
`...`	Currently not used.

Examples

```
library(bayestestR)
posterior <- distribution_normal(n = 10)
density_at(posterior, 0)
density_at(posterior, c(0, 1))
```

describe_posterior *Describe Posterior Distributions*

Description

Compute indices relevant to describe and characterise the posterior distributions.

Usage

```
describe_posterior(posteriors, centrality = "median",
  dispersion = FALSE, ci = 0.89, ci_method = "hdi",
  test = c("p_direction", "rope"), rope_range = "default",
  rope_ci = 0.89, ...)

## S3 method for class 'numeric'
describe_posterior(posteriors, centrality = "median",
  dispersion = FALSE, ci = 0.89, ci_method = "hdi",
  test = c("p_direction", "rope"), rope_range = "default",
  rope_ci = 0.89, bf_prior = NULL, ...)

## S3 method for class 'stanreg'
describe_posterior(posteriors, centrality = "median",
  dispersion = FALSE, ci = 0.89, ci_method = "hdi",
  test = c("p_direction", "rope"), rope_range = "default",
  rope_ci = 0.89, bf_prior = NULL, diagnostic = c("ESS", "Rhat"),
  priors = FALSE, effects = c("fixed", "random", "all"),
  parameters = NULL, ...)

## S3 method for class 'MCMCglmm'
describe_posterior(posteriors, centrality = "median",
  dispersion = FALSE, ci = 0.89, ci_method = "hdi",
  test = c("p_direction", "rope"), rope_range = "default",
  rope_ci = 0.89, diagnostic = "ESS", parameters = NULL, ...)

## S3 method for class 'brmsfit'
describe_posterior(posteriors, centrality = "median",
  dispersion = FALSE, ci = 0.89, ci_method = "hdi",
  test = c("p_direction", "rope"), rope_range = "default",
  rope_ci = 0.89, bf_prior = NULL, diagnostic = c("ESS", "Rhat"),
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), parameters = NULL, ...)

## S3 method for class 'BFBayesFactor'
describe_posterior(posteriors,
  centrality = "median", dispersion = FALSE, ci = 0.89,
  ci_method = "hdi", test = c("p_direction", "rope", "bf"),
  rope_range = "default", rope_ci = 0.89, priors = TRUE, ...)
```

Arguments

posteriors	A vector, dataframe or model of posterior draws.
centrality	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: "median", "mean", "MAP" or "all".
dispersion	Logical, if TRUE, computes indices of dispersion related to the estimate(s) (SD and MAD for mean and median, respectively).
ci	Value or vector of probability of the CI (between 0 and 1) to be estimated. Default to .89 (89%) for Bayesian models and .95 (95%) for frequentist models.
ci_method	The type of index used for Credible Interval. Can be "HDI" (default, see hdi) or "ETI" (see eti).
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.
...	Additional arguments to be passed to or from methods.
bf_prior	Distribution representing a prior for the computation of Bayes factors. Used if the input is a posterior, otherwise (in the case of models) ignored.
diagnostic	Diagnostic metrics to compute. Character (vector) or list with one or more of these options: "ESS", "Rhat", "MCSE" or "all".
priors	Add the prior used for each parameter.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

One or more components of point estimates (like posterior mean or median), intervals and tests can be ommitted from the summary output by setting the related argument to NULL. For example, test = NULL and centrality = NULL would only return the HDI (or CI).

References

- Comparison of Point-Estimates
- Region of Practical Equivalence (ROPE)
- Bayes factors

Examples

```
library(bayestestR)

x <- rnorm(1000)
describe_posterior(x)
describe_posterior(x, centrality = "all", dispersion = TRUE, test = "all")
describe_posterior(x, ci = c(0.80, 0.90))

df <- data.frame(replicate(4, rnorm(100)))
describe_posterior(df)
describe_posterior(df, centrality = "all", dispersion = TRUE, test = "all")
describe_posterior(df, ci = c(0.80, 0.90))

# rstanarm models
# ---------------------------------------------
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
describe_posterior(model)
describe_posterior(model, centrality = "all", dispersion = TRUE, test = "all")
describe_posterior(model, ci = c(0.80, 0.90))

# emmeans estimates
# ---------------------------------------------
library(emmeans)
describe_posterior(emtrends(model, ~1, "wt"))
## Not run:
# brms models
# ---------------------------------------------
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
describe_posterior(model)
describe_posterior(model, centrality = "all", dispersion = TRUE, test = "all")
describe_posterior(model, ci = c(0.80, 0.90))

# BayesFactor objects
# ---------------------------------------------
library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
describe_posterior(bf)
describe_posterior(bf, centrality = "all", dispersion = TRUE, test = "all")
describe_posterior(bf, ci = c(0.80, 0.90))

## End(Not run)
```

---

describe_prior *Describe Priors*

Description

Returns a summary of the priors used in the model.

Usage

```
describe_prior(model, ...)
```

Arguments

model	A Bayesian model.
...	Currently not used.

Examples

```
## Not run:
library(bayestestR)

# rstanarm models
# ---------------------------------------------
library(rstanarm)
model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
describe_prior(model)

# brms models
# ---------------------------------------------
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
describe_prior(model)

# BayesFactor objects
# ---------------------------------------------
library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
describe_prior(bf)

## End(Not run)
```

---

diagnostic_posterior *Posteriors Sampling Diagnostic*

Description

Extract diagnostic metrics (Effective Sample Size (ESS), Rhat and Monte Carlo Standard Error MCSE).

Usage

```
diagnostic_posterior(posteriors, diagnostic = c("ESS", "Rhat"), ...)

## S3 method for class 'stanreg'
diagnostic_posterior(posteriors, diagnostic = "all",
```

```
  effects = c("fixed", "random", "all"), parameters = NULL, ...)

## S3 method for class 'brmsfit'
diagnostic_posterior(posteriors, diagnostic = "all",
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), parameters = NULL, ...)
```

Arguments

posteriors	A stanreg or brms model.
diagnostic	Diagnostic metrics to compute. Character (vector) or list with one or more of these options: "ESS", "Rhat", "MCSE" or "all".
...	Currently not used.
effects	Should parameters for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned.
component	Should all parameters, parameters for the conditional model, the zero-inflated part of the model, the dispersion term or the instrumental variables be returned? Applies to models with zero-inflated and/or dispersion formula, or to models with instrumental variable (so called fixed-effects regressions). May be abbreviated.

Details

Effective Sample (ESS) should be as large as possible, altough for most applications, an effective sample size greater than 1000 is sufficient for stable estimates (Bürkner, 2017). The ESS corresponds to the number of independent samples with the same estimation power as the N autocorrelated samples. It is is a measure of "how much independent information there is in autocorrelated chains" (*Kruschke 2015, p182-3*).

Rhat should be the closest to 1. It should not be larger than 1.1 (Gelman and Rubin, 1992) or 1.01 (Vehtari et al., 2019). The split R-hat statistic quantifies the consistency of an ensemble of Markov chains.

Monte Carlo Standard Error (MCSE) is another measure of accuracy of the chains. It is defined as standard deviation of the chains divided by their effective sample size (the formula for `mcse()` is from Kruschke 2015, p. 187). The MCSE "provides a quantitative suggestion of how big the estimation noise is".

References

- Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. Statistical science, 7(4), 457-472.
- Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., \& Bürkner, P. C. (2019). Rank-normalization, folding, and localization: An improved Rhat for assessing convergence of MCMC. arXiv preprint arXiv:1903.08008.
- Kruschke, J. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.

Examples

```
# rstanarm models
# -----------------------------------------------
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
diagnostic_posterior(model)
## Not run:
# brms models
# -----------------------------------------------
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
diagnostic_posterior(model)

## End(Not run)
```

`distribution` *Empirical Distributions*

Description

Generate a sequence of n-quantiles, i.e., a sample of size n with a near-perfect distribution.

Usage

```
distribution(type = "normal", ...)

distribution_normal(n, mean = 0, sd = 1, random = FALSE, ...)

distribution_binomial(n, size = 1, prob = 0.5, random = FALSE, ...)

distribution_cauchy(n, location = 0, scale = 1, random = FALSE, ...)

distribution_poisson(n, lambda = 1, random = FALSE, ...)

distribution_student(n, df, ncp, random = FALSE, ...)

distribution_chisquared(n, df, ncp = 0, random = FALSE, ...)

distribution_uniform(n, min = 0, max = 1, random = FALSE, ...)

distribution_beta(n, shape1, shape2, ncp = 0, random = FALSE, ...)

distribution_gamma(n, shape, scale = 1, random = FALSE, ...)

distribution_custom(n, type = "norm", ..., random = FALSE)

distribution_mixture_normal(n, mean = c(-3, 3), sd = 1,
```

```
  random = FALSE, ...)

rnorm_perfect(n, mean = 0, sd = 1)
```

Arguments

type	Can be "normal" (default), "cauchy", "poisson", "gamma", "chisquared", "uniform", "student" or "beta".
...	Arguments passed to or from other methods.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
mean	vector of means.
sd	vector of standard deviations.
random	Generate near-perfect or random (simple wrappers for the base R r* functions) distributions.
size	number of trials (zero or more).
prob	probability of success on each trial.
location	location and scale parameters.
scale	location and scale parameters.
lambda	vector of (non-negative) means.
df	degrees of freedom (> 0, maybe non-integer). df = Inf is allowed.
ncp	non-centrality parameter δ; currently except for rt(), only for abs(ncp) <= 37.62. If omitted, use the central t distribution.
min	lower and upper limits of the distribution. Must be finite.
max	lower and upper limits of the distribution. Must be finite.
shape1	non-negative parameters of the Beta distribution.
shape2	non-negative parameters of the Beta distribution.
shape	shape and scale parameters. Must be positive, scale strictly.

Examples

```
library(bayestestR)
x <- distribution(n = 10)
plot(density(x))
```

`effective_sample` *Effective Sample Size (ESS)*

Description

This function returns the effective sample size (ESS).

Usage

```
effective_sample(model, ...)

## S3 method for class 'brmsfit'
effective_sample(model, effects = c("fixed", "random",
  "all"), component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL, ...)

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

## S3 method for class 'MCMCglmm'
effective_sample(model, effects = c("fixed", "random",
  "all"), parameters = NULL, ...)
```

Arguments

`model`	A `stanreg`, `stanfit`, or `brmsfit` object.
`...`	Currently not used.
`effects`	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
`component`	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.
`parameters`	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like `lp__` or `prior_`) are filtered by default, so only parameters that typically appear in the `summary()` are returned. Use `parameters` to select specific parameters for the output.

Details

Effective Sample (ESS) should be as large as possible, altough for most applications, an effective sample size greater than 1,000 is sufficient for stable estimates (Bürkner, 2017). The ESS corresponds to the number of independent samples with the same estimation power as the N autocorrelated samples. It is is a measure of "how much independent information there is in autocorrelated chains" (*Kruschke 2015, p182-3*).

Value

A data frame with two columns: Parameter name and effective sample size (ESS).

References

- Kruschke, J. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.
- Bürkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. Journal of Statistical Software, 80(1), 1-28

Examples

```
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
effective_sample(model)
```

---

equivalence_test *Test for Practical Equivalence*

Description

Perform a **Test for Practical Equivalence** for Bayesian and frequentist models.

Usage

```
equivalence_test(x, ...)

## Default S3 method:
equivalence_test(x, ...)

## S3 method for class 'numeric'
equivalence_test(x, range = "default", ci = 0.89,
  verbose = TRUE, ...)

## S3 method for class 'data.frame'
equivalence_test(x, range = "default", ci = 0.89,
  verbose = TRUE, ...)

## S3 method for class 'emmGrid'
equivalence_test(x, range = "default", ci = 0.89,
  verbose = TRUE, ...)

## S3 method for class 'BFBayesFactor'
equivalence_test(x, range = "default",
  ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'stanreg'
```

```
equivalence_test(x, range = "default", ci = 0.89,
  parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'brmsfit'
equivalence_test(x, range = "default", ci = 0.89,
  parameters = NULL, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg or brmsfit model.
...	Currently not used.
range	ROPE's lower and higher bounds. Should be a vector of length two (e.g., c(-0.1,0.1)) or "default". If "default", the range is set to c(-0.1,0.1) if input is a vector, and based on rope_range() if a Bayesian model is provided.
ci	The Credible Interval (CI) probability, corresponding to the proportion of HDI, to use for the percentage in ROPE.
verbose	Toggle off warnings.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.

Details

Documentation is accessible for:

- Bayesian models
- Frequentist models

For Bayesian models, the **Test for Practical Equivalence** is based on the *"HDI+ROPE decision rule"* (*Kruschke, 2014, 2018*) to check whether parameter values should be accepted or rejected against an explicitly formulated "null hypothesis" (i.e., a ROPE). In other words, it checks the percentage of the 89% HDI that is the null region (the ROPE). If this percentage is sufficiently low, the null hypothesis is rejected. If this percentage is sufficiently high, the null hypothesis is accepted.

Using the ROPE and the HDI, *Kruschke (2018)* suggests using the percentage of the 95% (or 89%, considered more stable) HDI that falls within the ROPE as a decision rule. If the HDI is completely outside the ROPE, the "null hypothesis" for this parameter is "rejected". If the ROPE completely covers the HDI, i.e., all most credible values of a parameter are inside the region of practical equivalence, the null hypothesis is accepted. Else, it's undecided whether to accept or reject the null hypothesis. If the full ROPE is used (i.e., 100% of the HDI), then the null hypothesis is rejected or accepted if the percentage of the posterior within the ROPE is smaller than to 2.5% or greater than 97.5%. Desirable results are low proportions inside the ROPE (the closer to zero the better).

Some attention is required for finding suitable values for the ROPE limits (argument range). See 'Details' in rope_range() for further information.

Multicollinearity: Non-independent covariates

When parameters show strong correlations, i.e. when covariates are not independent, the joint parameter distributions may shift towards or away from the ROPE. In such cases, the test for practical equivalence may have inappropriate results. Collinearity invalidates ROPE and hypothesis testing based on univariate marginals, as the probabilities are conditional on independence. Most problematic are the results of the "undecided" parameters, which may either move further towards "rejection" or away from it (*Kruschke 2014, 340f*).

`equivalence_test()` performs a simple check for pairwise correlations between parameters, but as there can be collinearity between more than two variables, a first step to check the assumptions of this hypothesis testing is to look at different pair plots. An even more sophisticated check is the projection predictive variable selection (*Piironen and Vehtari 2017*).

Value

A data frame with following columns:

- `Parameter` The model parameter(s), if `x` is a model-object. If `x` is a vector, this column is missing.
- `CI` The probability of the HDI.
- `ROPE_low`, `ROPE_high` The limits of the ROPE. These values are identical for all parameters.
- `ROPE_Percentage` The proportion of the HDI that lies inside the ROPE.
- `ROPE_Equivalence` The "test result", as character. Either "rejected", "accepted" or "undecided".
- `HDI_low` , `HDI_high` The lower and upper HDI limits for the parameters.

Note

There is a `print()`-method with a `digits`-argument to control the amount of digits in the output, and there is a `plot()`-method to visualize the results from the equivalence-test (for models only).

References

- Kruschke, J. K. (2018). Rejecting or accepting parameter values in Bayesian estimation. Advances in Methods and Practices in Psychological Science, 1(2), 270-280. doi: 10.1177/2515245918771304
- Kruschke, J. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press
- Piironen, J., & Vehtari, A. (2017). Comparison of Bayesian predictive methods for model selection. Statistics and Computing, 27(3), 711–735. doi: 10.1007/s112220169649y

Examples

```
library(bayestestR)

equivalence_test(x = rnorm(1000, 0, 0.01), range = c(-0.1, 0.1))
equivalence_test(x = rnorm(1000, 0, 1), range = c(-0.1, 0.1))
equivalence_test(x = rnorm(1000, 1, 0.01), range = c(-0.1, 0.1))
```

```
equivalence_test(x = rnorm(1000, 1, 1), ci = c(.50, .99))

# print more digits
test <- equivalence_test(x = rnorm(1000, 1, 1), ci = c(.50, .99))
print(test, digits = 4)
## Not run:
library(rstanarm)
model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
equivalence_test(model)
equivalence_test(model, ci = c(.50, 1))

# plot result
test <- equivalence_test(model)
plot(test)

library(emmeans)
equivalence_test(emtrends(model, ~1, "wt"))

library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
equivalence_test(model)
equivalence_test(model, ci = c(.50, .99))

ibrary(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
equivalence_test(bf)
equivalence_test(bf, ci = c(.50, .99))

## End(Not run)
```

`estimate_density` *Density Estimation*

Description

This function is a wrapper over different methods of density estimation. By default, it uses the base R density with by default uses a different smoothing bandwidth ("SJ") from the legacy default implemented the base R density function ("nrd0"). However, Deng \& Wickham suggest that method = "KernSmooth" is the fastest and the most accurate.

Usage

```
estimate_density(x, method = "kernel", precision = 2^10,
  extend = FALSE, extend_scale = 0.1, bw = "SJ", ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg, brmsfit or a BayesFactor model.

method	Density estimation method. Can be "kernel" (default), "logspline" or "KernSmooth".
precision	Number of points of density data. See the n parameter in density.
extend	Extend the range of the x axis by a factor of extend_scale.
extend_scale	Ratio of range by which to extend the x axis. A value of 0.1 means that the x axis will be extended by 1/10 of the range of the data.
bw	the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel. (Note this differs from the reference books cited below, and from S-PLUS.) bw can also be a character string giving a rule to choose the bandwidth. See bw.nrd. The default, "nrd0", has remained the default for historical and compatibility reasons, rather than as a general recommendation, where e.g., "SJ" would rather fit, see also Venables and Ripley (2002). The specified (or computed) value of bw is multiplied by adjust.
...	Currently not used.

References

Deng, H., & Wickham, H. (2011). Density estimation in R. Electronic publication.

Examples

```
library(bayestestR)

set.seed(1)
x <- rnorm(250, 1)

# Methods
density_kernel <- estimate_density(x, method = "kernel")
density_logspline <- estimate_density(x, method = "logspline")
density_KernSmooth <- estimate_density(x, method = "KernSmooth")
density_mixture <- estimate_density(x, method = "mixture")

hist(x, prob = TRUE)
lines(density_kernel$x, density_kernel$y, col = "black", lwd = 2)
lines(density_logspline$x, density_logspline$y, col = "red", lwd = 2)
lines(density_KernSmooth$x, density_KernSmooth$y, col = "blue", lwd = 2)
lines(density_mixture$x, density_mixture$y, col = "green", lwd = 2)

# Extension
density_extended <- estimate_density(x, extend = TRUE)
density_default <- estimate_density(x, extend = FALSE)

hist(x, prob = TRUE)
lines(density_extended$x, density_extended$y, col = "red", lwd = 3)
lines(density_default$x, density_default$y, col = "black", lwd = 3)

df <- data.frame(replicate(4, rnorm(100)))
head(estimate_density(df))
```

```
# rstanarm models
# ---------------------------------------------
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
head(estimate_density(model))

library(emmeans)
head(estimate_density(emtrends(model, ~1, "wt")))
## Not run:
# brms models
# ---------------------------------------------
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
estimate_density(model)

## End(Not run)
```

---

eti *Equal-Tailed Interval (ETI)*

Description

Compute the **Equal-Tailed Interval (ETI)** of posterior distributions using the quantiles method. The probability of being below this interval is equal to the probability of being above it. The ETI can be used in the context of uncertainty characterisation of posterior distributions as **Credible Interval (CI)**.

Usage

```
eti(x, ...)

## S3 method for class 'numeric'
eti(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'data.frame'
eti(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'MCMCglmm'
eti(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'sim.merMod'
eti(x, ci = 0.89, effects = c("fixed", "random",
  "all"), parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'sim'
eti(x, ci = 0.89, parameters = NULL, verbose = TRUE,
```

```
  ...)

## S3 method for class 'emmGrid'
eti(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'stanreg'
eti(x, ci = 0.89, effects = c("fixed", "random",
  "all"), parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'brmsfit'
eti(x, ci = 0.89, effects = c("fixed", "random",
  "all"), component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'BFBayesFactor'
eti(x, ci = 0.89, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg, brmsfit or a BayesFactor model.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .89 (89%).
verbose	Toggle off warnings.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

Unlike equal-tailed intervals (see eti()) that typically exclude 2.5% from each tail of the distribution and always include the median, the HDI is *not* equal-tailed and therefore always includes the mode(s) of posterior distributions.

By default, hdi() and eti() return the 89% intervals (ci = 0.89), deemed to be more stable than, for instance, 95% intervals (*Kruschke, 2014*). An effective sample size of at least 10.000 is recommended if 95% intervals should be computed (*Kruschke, 2014, p. 183ff*). Moreover, 89 indicates the arbitrariness of interval limits - its only remarkable property is being the highest prime number that does not exceed the already unstable 95% threshold (*McElreath, 2015*).

A 90% equal-tailed interval (ETI) has 5% of the distribution on either side of its limits. It indicates the 5th percentile and the 95h percentile. In symmetric distributions, the two methods of computing credible intervals, the ETI and the HDI, return similar results.

This is not the case for skewed distributions. Indeed, it is possible that parameter values in the ETI have lower credibility (are less probable) than parameter values outside the ETI. This property seems undesirable as a summary of the credible values in a distribution.

On the other hand, the ETI range does change when transformations are applied to the distribution (for instance, for a log odds scale to probabilities): the lower and higher bounds of the transformed distribution will correspond to the transformed lower and higher bounds of the original distribution. On the contrary, applying transformations to the distribution will change the resulting HDI.

Value

A data frame with following columns:

- `Parameter` The model parameter(s), if `x` is a model-object. If `x` is a vector, this column is missing.
- `CI` The probability of the credible interval.
- `CI_low`, `CI_high` The lower and upper credible interval limits for the parameters.

Examples

```
library(bayestestR)

posterior <- rnorm(1000)
eti(posterior)
eti(posterior, ci = c(.80, .89, .95))

df <- data.frame(replicate(4, rnorm(100)))
eti(df)
eti(df, ci = c(.80, .89, .95))

library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
eti(model)
eti(model, ci = c(.80, .89, .95))

library(emmeans)
eti(emtrends(model, ~1, "wt"))
## Not run:
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
eti(model)
eti(model, ci = c(.80, .89, .95))

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
eti(bf)
eti(bf, ci = c(.80, .89, .95))
```

```
## End(Not run)
```

---

hdi *Highest Density Interval (HDI)*

Description

Compute the **Highest Density Interval (HDI)** of posterior distributions. All points within this interval have a higher probability density than points outside the interval. The HDI can be used in the context of uncertainty characterisation of posterior distributions as **Credible Interval (CI)**.

Usage

```
hdi(x, ...)

## S3 method for class 'numeric'
hdi(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'data.frame'
hdi(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'MCMCglmm'
hdi(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'sim.merMod'
hdi(x, ci = 0.89, effects = c("fixed", "random",
  "all"), parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'sim'
hdi(x, ci = 0.89, parameters = NULL, verbose = TRUE,
  ...)

## S3 method for class 'emmGrid'
hdi(x, ci = 0.89, verbose = TRUE, ...)

## S3 method for class 'stanreg'
hdi(x, ci = 0.89, effects = c("fixed", "random",
  "all"), parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'brmsfit'
hdi(x, ci = 0.89, effects = c("fixed", "random",
  "all"), component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'BFBayesFactor'
hdi(x, ci = 0.89, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg, brmsfit or a BayesFactor model.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .89 (89%).
verbose	Toggle off warnings.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

Unlike equal-tailed intervals (see eti()) that typically exclude 2.5% from each tail of the distribution and always include the median, the HDI is *not* equal-tailed and therefore always includes the mode(s) of posterior distributions.

By default, hdi() and eti() return the 89% intervals (ci = 0.89), deemed to be more stable than, for instance, 95% intervals (*Kruschke, 2014*). An effective sample size of at least 10.000 is recommended if 95% intervals should be computed (*Kruschke, 2014, p. 183ff*). Moreover, 89 indicates the arbitrariness of interval limits - its only remarkable property is being the highest prime number that does not exceed the already unstable 95% threshold (*McElreath, 2015*).

A 90% equal-tailed interval (ETI) has 5% of the distribution on either side of its limits. It indicates the 5th percentile and the 95h percentile. In symmetric distributions, the two methods of computing credible intervals, the ETI and the HDI, return similar results.

This is not the case for skewed distributions. Indeed, it is possible that parameter values in the ETI have lower credibility (are less probable) than parameter values outside the ETI. This property seems undesirable as a summary of the credible values in a distribution.

On the other hand, the ETI range does change when transformations are applied to the distribution (for instance, for a log odds scale to probabilities): the lower and higher bounds of the transformed distribution will correspond to the transformed lower and higher bounds of the original distribution. On the contrary, applying transformations to the distribution will change the resulting HDI.

Value

A data frame with following columns:

- Parameter The model parameter(s), if x is a model-object. If x is a vector, this column is missing.

- `CI` The probability of the credible interval.
- `CI_low`, `CI_high` The lower and upper credible interval limits for the parameters.

Author(s)

Credits go to ggdistribute and HDInterval.

References

- Kruschke, J. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.
- McElreath, R. (2015). Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC.

Examples

```
library(bayestestR)

posterior <- rnorm(1000)
hdi(posterior, ci = .89)
hdi(posterior, ci = c(.80, .90, .95))

df <- data.frame(replicate(4, rnorm(100)))
hdi(df)
hdi(df, ci = c(.80, .90, .95))

library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
hdi(model)
hdi(model, ci = c(.80, .90, .95))

library(emmeans)
hdi(emtrends(model, ~1, "wt"))
## Not run:
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
hdi(model)
hdi(model, ci = c(.80, .90, .95))

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
hdi(bf)
hdi(bf, ci = c(.80, .90, .95))

## End(Not run)
```

map_estimate *Maximum A Posteriori probability estimate (MAP)*

Description

Find the **Highest Maximum A Posteriori probability estimate (MAP)** of a posterior, i.e., the value associated with the highest probability density (the "peak" of the posterior distribution). In other words, it is an estimation of the *mode* for continuous parameters. Note that this function relies on estimate_density, which by default uses a different smoothing bandwidth ("SJ") compared to the legacy default implemented the base R density function ("nrd0").

Usage

```
map_estimate(x, precision = 2^10, method = "kernel", ...)

## S3 method for class 'numeric'
map_estimate(x, precision = 2^10, method = "kernel",
  ...)

## S3 method for class 'stanreg'
map_estimate(x, precision = 2^10, method = "kernel",
  effects = c("fixed", "random", "all"), parameters = NULL, ...)

## S3 method for class 'brmsfit'
map_estimate(x, precision = 2^10, method = "kernel",
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), parameters = NULL, ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg, brmsfit or a BayesFactor model.
precision	Number of points of density data. See the n parameter in density.
method	Density estimation method. Can be "kernel" (default), "logspline" or "KernSmooth".
...	Currently not used.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Value

A numeric value if `posterior` is a vector. If `posterior` is a model-object, returns a data frame with following columns:

- `Parameter` The model parameter(s), if `x` is a model-object. If `x` is a vector, this column is missing.
- `MAP_Estimate` The MAP estimate for the posterior or each model parameter.

Examples

```
library(bayestestR)

posterior <- rnorm(10000)
map_estimate(posterior)

plot(density(posterior))
abline(v = map_estimate(posterior), col = "red")
## Not run:
library(rstanarm)
model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
map_estimate(model)

library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
map_estimate(model)

## End(Not run)
```

---

mcse *Monte-Carlo Standard Error (MCSE)*

Description

This function returns the Monte Carlo Standard Error (MCSE).

Usage

```
mcse(model, ...)

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

Arguments

model	A stanreg, stanfit, or brmsfit object.
...	Currently not used.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.

Details

Monte Carlo Standard Error (MCSE) is another measure of accuracy of the chains. It is defined as standard deviation of the chains divided by their effective sample size (the formula for mcse() is from Kruschke 2015, p. 187). The MCSE "provides a quantitative suggestion of how big the estimation noise is".

References

Kruschke, J. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.

Examples

```
library(bayestestR)
library(rstanarm)

model <- stan_glm(mpg ~ wt + am, data = mtcars, chains = 1, refresh = 0)
mcse(model)
```

---

overlap *Overlap Coefficient*

Description

A method to calculate the overlap coefficient between two empirical distributions (that can be used as a measure of similarity between two samples).

Usage

```
overlap(x, y, method_density = "kernel", method_auc = "trapezoid",
  precision = 2^10, extend = TRUE, extend_scale = 0.1, ...)
```

Arguments

x	Vector of x values.
y	Vector of x values.
method_density	Density estimation method. See estimate_density.
method_auc	Area Under the Curve (AUC) estimation method. See area_under_curve.
precision	Number of points of density data. See the n parameter in density.
extend	Extend the range of the x axis by a factor of extend_scale.
extend_scale	Ratio of range by which to extend the x axis. A value of 0.1 means that the x axis will be extended by 1/10 of the range of the data.
...	Currently not used.

Examples

```
library(bayestestR)

x <- distribution_normal(1000, 2, 0.5)
y <- distribution_normal(1000, 0, 1)

overlap(x, y)
plot(overlap(x, y))
```

---

pd_to_p *Convert between Probability of Direction (pd) and p-value.*

Description

Enables a conversion between sProbability of Direction (pd) and p-value.

Usage

```
pd_to_p(pd, direction = "two-sided", ...)

p_to_pd(p, direction = "two-sided", ...)

convert_p_to_pd(p, direction = "two-sided", ...)

convert_pd_to_p(pd, direction = "two-sided", ...)
```

Arguments

pd	A Probability of Direction (pd) value (between 0 and 1).
direction	What type of p-value is requested or provided. Can be "two-sided" (default, two tailed) or "one-sided" (one tailed).
...	Arguments passed to or from other methods.
p	A p-value.

Examples

```
pd_to_p(pd = 0.95)
pd_to_p(pd = 0.95, direction = "one-sided")
```

`point_estimate` *Point-estimates of posterior distributions*

Description

Compute various point-estimates, such as the mean, the median or the MAP, to describe posterior distributions.

Usage

```
point_estimate(x, centrality = "median", dispersion = FALSE, ...)

## S3 method for class 'stanreg'
point_estimate(x, centrality = "median",
  dispersion = FALSE, effects = c("fixed", "random", "all"),
  parameters = NULL, ...)

## S3 method for class 'brmsfit'
point_estimate(x, centrality = "median",
  dispersion = FALSE, effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL, ...)

## S3 method for class 'BFBayesFactor'
point_estimate(x, centrality = "median",
  dispersion = FALSE, ...)
```

Arguments

`x`	Vector representing a posterior distribution. Can also be a `stanreg`, `brmsfit` or a `BayesFactor` model.
`centrality`	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: `"median"`, `"mean"`, `"MAP"` or `"all"`.
`dispersion`	Logical, if `TRUE`, computes indices of dispersion related to the estimate(s) (`SD` and `MAD` for `mean` and `median`, respectively).
`...`	Additional arguments to be passed to or from methods.
`effects`	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
`parameters`	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like `lp__` or `prior_`) are filtered by default, so only parameters that typically appear in the `summary()` are returned. Use `parameters` to select specific parameters for the output.

component — Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

References

Vignette In-Depth 1: Comparison of Point-Estimates

Examples

```
library(bayestestR)

point_estimate(rnorm(1000))
point_estimate(rnorm(1000), centrality = "all", dispersion = TRUE)
point_estimate(rnorm(1000), centrality = c("median", "MAP"))

df <- data.frame(replicate(4, rnorm(100)))
point_estimate(df, centrality = "all", dispersion = TRUE)
point_estimate(df, centrality = c("median", "MAP"))
## Not run:
# rstanarm models
# -----------------------------------------------
library(rstanarm)
model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
point_estimate(model, centrality = "all", dispersion = TRUE)
point_estimate(model, centrality = c("median", "MAP"))


# emmeans estimates
# -----------------------------------------------
library(emmeans)
point_estimate(emtrends(model, ~1, "wt"), centrality = c("median", "MAP"))

# brms models
# -----------------------------------------------
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
point_estimate(model, centrality = "all", dispersion = TRUE)
point_estimate(model, centrality = c("median", "MAP"))

# BayesFactor objects
# -----------------------------------------------
library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
point_estimate(bf, centrality = "all", dispersion = TRUE)
point_estimate(bf, centrality = c("median", "MAP"))

## End(Not run)
```

p_direction *Probability of Direction (pd)*

Description

Compute the **Probability of Direction** (***pd***, also known as the Maximum Probability of Effect - *MPE*). It varies between 50% and 100% (*i.e.*, `0.5` and 1) and can be interpreted as the probability (expressed in percentage) that a parameter (described by its posterior distribution) is strictly positive or negative (whichever is the most probable). It is mathematically defined as the proportion of the posterior distribution that is of the median's sign. Altough differently expressed, this index is fairly similar (*i.e.*, is strongly correlated) to the frequentist **p-value**.

Usage

```
p_direction(x, ...)

pd(x, ...)

## S3 method for class 'numeric'
p_direction(x, method = "direct", ...)

## S3 method for class 'data.frame'
p_direction(x, method = "direct", ...)

## S3 method for class 'MCMCglmm'
p_direction(x, method = "direct", ...)

## S3 method for class 'emmGrid'
p_direction(x, method = "direct", ...)

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

## S3 method for class 'brmsfit'
p_direction(x, effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL, method = "direct", ...)

## S3 method for class 'BFBayesFactor'
p_direction(x, method = "direct", ...)
```

Arguments

`x`	Vector representing a posterior distribution. Can also be a Bayesian model (`stanreg`, `brmsfit` or `BayesFactor`).
`...`	Currently not used.

`method`	Can be `"direct"` or one of methods of density estimation, such as `"kernel"`, `"logspline"` or `"KernSmooth"`. If `"direct"` (default), the computation is based on the raw ratio of samples superior and inferior to 0. Else, the result is based on the Area under the Curve (AUC) of the estimated density function.
`effects`	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
`parameters`	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like `lp__` or `prior_`) are filtered by default, so only parameters that typically appear in the `summary()` are returned. Use `parameters` to select specific parameters for the output.
`component`	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

What is the *pd*?

The Probability of Direction (pd) is an index of effect existence, ranging from 50% to 100%, representing the certainty with which an effect goes in a particular direction (*i.e.*, is positive or negative). Beyond its simplicity of interpretation, understanding and computation, this index also presents other interesting properties:

- It is independent from the model: It is solely based on the posterior distributions and does not require any additional information from the data or the model.
- It is robust to the scale of both the response variable and the predictors.
- It is strongly correlated with the frequentist p-value, and can thus be used to draw parallels and give some reference to readers non-familiar with Bayesian statistics.

Relationship with the p-value

In most cases, it seems that the *pd* has a direct correspondance with the frequentist one-sided *p*-value through the formula $p_{onesided} = 1 - \frac{p_d}{100}$ and to the two-sided p-value (the most commonly reported one) through the formula $p_{twosided} = 2 * (1 - \frac{p_d}{100})$. Thus, a two-sided p-value of respectively `.1`, `.05`, `.01` and `.001` would correspond approximately to a *pd* of 95%, 97.5%, 99.5% and 99.95%.

Methods of computation

The most simple and direct way to compute the *pd* is to 1) look at the median's sign, 2) select the portion of the posterior of the same sign and 3) compute the percentage that this portion represents. This "simple" method is the most straigtfoward, but its precision is directly tied to the number of posterior draws. The second approach relies on density estimation. It starts by estimating the density function (for which many methods are available), and then computing the area under the curve (AUC) of the density curve on the other side of 0.

Value

Values between 0.5 and 1 corresponding to the probability of direction (pd).

Examples

```
library(bayestestR)

# Simulate a posterior distribution of mean 1 and SD 1
# ----------------------------------------------------
posterior <- rnorm(1000, mean = 1, sd = 1)
p_direction(posterior)
p_direction(posterior, method = "kernel")

# Simulate a dataframe of posterior distributions
# -----------------------------------------------
df <- data.frame(replicate(4, rnorm(100)))
p_direction(df)
p_direction(df, method = "kernel")

# rstanarm models
# -----------------------------------------------
library(rstanarm)
model <- rstanarm::stan_glm(mpg ~ wt + cyl,
  data = mtcars,
  chains = 2, refresh = 0
)
p_direction(model)
p_direction(model, method = "kernel")
## Not run:
# emmeans
# -----------------------------------------------
library(emmeans)
p_direction(emtrends(model, ~1, "wt"))

# brms models
# -----------------------------------------------
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
p_direction(model)
p_direction(model, method = "kernel")

# BayesFactor objects
# -----------------------------------------------
library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
p_direction(bf)
p_direction(bf, method = "kernel")

## End(Not run)
```

---

p_map *Bayesian p-value based on the density at the Maximum A Posteriori (MAP)*

Description

Compute a Bayesian equivalent of the *p*-value, related to the odds that a parameter (described by its posterior distribution) has against the null hypothesis (*h0*) using Mills' (2014, 2017) *Objective Bayesian Hypothesis Testing* framework. It corresponds to the density value at 0 divided by the density at the Maximum A Posteriori (MAP).

Usage

```
p_map(x, precision = 2^10, method = "kernel", ...)

## S3 method for class 'numeric'
p_map(x, precision = 2^10, method = "kernel", ...)

## S3 method for class 'stanreg'
p_map(x, precision = 2^10, method = "kernel",
  effects = c("fixed", "random", "all"), parameters = NULL, ...)

## S3 method for class 'brmsfit'
p_map(x, precision = 2^10, method = "kernel",
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), parameters = NULL, ...)

## S3 method for class 'BFBayesFactor'
p_map(x, precision = 2^10, method = "kernel",
  ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg, brmsfit or a BayesFactor model.
precision	Number of points of density data. See the n parameter in density.
method	Density estimation method. Can be "kernel" (default), "logspline" or "KernSmooth".
...	Currently not used.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

Note that this method is sensitive to the density estimation method (see the secion in the examples below).

References

- Mills, J. A. (2018). Objective Bayesian Precise Hypothesis Testing. University of Cincinnati.

See Also

Jeff Mill's talk

Examples

```
library(bayestestR)

p_map(rnorm(1000, 0, 1))
p_map(rnorm(1000, 10, 1))

library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
p_map(model)

library(emmeans)
p_map(emtrends(model, ~1, "wt"))
## Not run:
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
p_map(model)

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
p_map(bf)

## End(Not run)


# ----------------------------------------
# Robustness to density estimation method
set.seed(333)
data <- data.frame()
for (iteration in 1:250) {
  x <- rnorm(1000, 1, 1)
  result <- data.frame(
    "Kernel" = p_map(x, method = "kernel"),
    "KernSmooth" = p_map(x, method = "KernSmooth"),
    "logspline" = p_map(x, method = "logspline")
  )
  data <- rbind(data, result)
}
data$KernSmooth <- data$Kernel - data$KernSmooth
data$logspline <- data$Kernel - data$logspline

summary(data$KernSmooth)
summary(data$logspline)
boxplot(data[c("KernSmooth", "logspline")])
```

p_rope *ROPE-based p-value*

Description

The ROPE-based *p*-value is an exploratory and non-validated index representing the maximum percentage of HDI that does not contain (or is entirely contained, in which case the value is prefixed with a negative sign), in the negligible values space defined by the ROPE. It differs from the ROPE percentage, *i.e.*, from the proportion of a given CI in the ROPE, as it represents the maximum CI values needed to reach a ROPE proportion of 0% or 100%. Whether the index reflects the ROPE reaching 0% or 100% is indicated through the sign: a negative sign is added to indicate that the probability corresponds to the probability of a not significant effect (a percentage in ROPE of 100%). For instance, a ROPE-based *p* of 97% means that there is a probability of .97 that a parameter (described by its posterior distribution) is outside the ROPE. In other words, the 97% HDI is the maximum HDI level for which the percentage in ROPE is 0%. On the contrary, a ROPE-based p of -97% indicates that there is a probability of .97 that the parameter is inside the ROPE (percentage in ROPE of 100%). A value close to 0% would indicate that the mode of the distribution falls perfectly at the edge of the ROPE, in which case the percentage of HDI needed to be on either side of the ROPE becomes infinitely small. Negative values do not refer to negative values *per se*, simply indicating that the value corresponds to non-significance rather than significance.

Usage

```
p_rope(x, ...)

## S3 method for class 'numeric'
p_rope(x, range = "default", precision = 0.1, ...)

## S3 method for class 'data.frame'
p_rope(x, range = "default", precision = 0.1, ...)

## S3 method for class 'emmGrid'
p_rope(x, range = "default", precision = 0.1, ...)

## S3 method for class 'BFBayesFactor'
p_rope(x, range = "default", precision = 0.1,
  ...)

## S3 method for class 'stanreg'
p_rope(x, range = "default", precision = 0.1,
  effects = c("fixed", "random", "all"), parameters = NULL, ...)

## S3 method for class 'brmsfit'
p_rope(x, range = "default", precision = 0.1,
  effects = c("fixed", "random", "all"), component = c("conditional",
  "zi", "zero_inflated", "all"), parameters = NULL, ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg or brmsfit model.
...	Currently not used.
range	ROPE's lower and higher bounds. Should be a vector of length two (e.g., c(-0.1,0.1)) or "default". If "default", the range is set to c(-0.1,0.1) if input is a vector, and based on rope_range() if a Bayesian model is provided.
precision	The precision by which to explore the ROPE space (in percentage). Lower values increase the precision of the returned p value but can be quite computationaly costly.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Examples

```
library(bayestestR)

# precision = 1 is used to speed up examples...

p_rope(
  x = rnorm(1000, mean = 1, sd = 1),
  range = c(-0.1, 0.1),
  precision = 1
)

df <- data.frame(replicate(4, rnorm(100)))
p_rope(df, precision = 1)

library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
p_rope(model, precision = 1)

library(emmeans)
p_rope(emtrends(model, ~1, "wt"))
## Not run:
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
p_rope(model)

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
p_rope(bf)
```

```
## End(Not run)
```

`p_significance` *Practical Significance (ps)*

Description

Compute the probability of **Practical Significance** (***ps***), which can be conceptualized as a unidirectional equivalence test. It returns the probability that effect is above a given threshold corresponding to a negligible effect in the median's direction. Mathematically, it is defined as the proportion of the posterior distribution of the median sign above the threshold.

Usage

```
p_significance(x, ...)

## S3 method for class 'numeric'
p_significance(x, threshold = "default", ...)

## S3 method for class 'data.frame'
p_significance(x, threshold = "default", ...)

## S3 method for class 'MCMCglmm'
p_significance(x, threshold = "default", ...)

## S3 method for class 'emmGrid'
p_significance(x, threshold = "default", ...)

## S3 method for class 'stanreg'
p_significance(x, threshold = "default",
  effects = c("fixed", "random", "all"), parameters = NULL,
  verbose = TRUE, ...)
```

Arguments

`x`	Vector representing a posterior distribution. Can also be a `stanreg` or `brmsfit` model.
`...`	Currently not used.
`threshold`	The threshold value that separates significant from negligible effect. If `"default"`, the range is set to `0.1` if input is a vector, and based on `rope_range()` if a Bayesian model is provided.
`effects`	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.

parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
verbose	Toggle off warnings.

Value

Values between 0.5 and 1 corresponding to the probability of practical significance (ps).

Examples

```
library(bayestestR)

# Simulate a posterior distribution of mean 1 and SD 1
# ----------------------------------------------------
posterior <- rnorm(1000, mean = 1, sd = 1)
p_significance(posterior)

# Simulate a dataframe of posterior distributions
# -----------------------------------------------
df <- data.frame(replicate(4, rnorm(100)))
p_significance(df)

# rstanarm models
# -----------------------------------------------
library(rstanarm)
model <- rstanarm::stan_glm(mpg ~ wt + cyl,
  data = mtcars,
  chains = 2, refresh = 0
)
p_significance(model)
```

---

reshape_ci *Reshape CI between wide/long formats*

Description

Reshape CI between wide/long formats.

Usage

```
reshape_ci(x)
```

Arguments

x	A data.frame containing CI_low and CI_high.

Examples

```
library(bayestestR)

x <- data.frame(replicate(4, rnorm(100)))
x <- ci(x, ci = c(0.68, 0.89, 0.95))
reshape_ci(x)
reshape_ci(reshape_ci(x))

x <- data.frame(replicate(4, rnorm(100)))
x <- describe_posterior(x, ci = c(0.68, 0.89, 0.95))
reshape_ci(x)
reshape_ci(reshape_ci(x))
```

---

rope *Region of Practical Equivalence (ROPE)*

Description

Compute the proportion (in percentage) of the HDI (default to the 90% HDI) of a posterior distribution that lies within a region of practical equivalence.

Usage

```
rope(x, ...)

## Default S3 method:
rope(x, ...)

## S3 method for class 'numeric'
rope(x, range = "default", ci = 0.89,
  ci_method = "HDI", verbose = TRUE, ...)

## S3 method for class 'data.frame'
rope(x, range = "default", ci = 0.89,
  ci_method = "HDI", verbose = TRUE, ...)

## S3 method for class 'emmGrid'
rope(x, range = "default", ci = 0.89,
  verbose = TRUE, ...)

## S3 method for class 'BFBayesFactor'
rope(x, range = "default", ci = 0.89,
  verbose = TRUE, ...)

## S3 method for class 'MCMCglmm'
rope(x, range = "default", ci = 0.89,
  verbose = TRUE, ...)
```

```
## S3 method for class 'stanreg'
rope(x, range = "default", ci = 0.89,
  ci_method = "HDI", effects = c("fixed", "random", "all"),
  parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'brmsfit'
rope(x, range = "default", ci = 0.89,
  ci_method = "HDI", effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a stanreg or brmsfit model.
...	Currently not used.
range	ROPE's lower and higher bounds. Should be a vector of length two (e.g., c(-0.1,0.1)) or "default". If "default", the range is set to c(-0.1,0.1) if input is a vector, and based on rope_range() if a Bayesian model is provided.
ci	The Credible Interval (CI) probability, corresponding to the proportion of HDI, to use for the percentage in ROPE.
ci_method	The type of interval to use to quantify the percentage in ROPE. Can be 'HDI' (default) or 'ETI'. See ci.
verbose	Toggle off warnings.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to **brms**-models.

Details

Statistically, the probability of a posterior distribution of being different from 0 does not make much sense (the probability of a single value null hypothesis in a continuous distribution is 0). Therefore, the idea underlining ROPE is to let the user define an area around the null value enclosing values that are *equivalent to the null* value for practical purposes (*Kruschke 2010, 2011, 2014*).

Kruschke (2018) suggests that such null value could be set, by default, to the -0.1 to 0.1 range of a standardized parameter (negligible effect size according to Cohen, 1988). This could be generalized: For instance, for linear models, the ROPE could be set as `0 +/-.1 * sd(y)`. This ROPE range can be automatically computed for models using the rope_range function.

Kruschke (2010, 2011, 2014) suggests using the proportion of the 95% (or 89%, considered more stable) HDI that falls within the ROPE as an index for "null-hypothesis" testing (as understood under the Bayesian framework, see `equivalence_test()`).

Sensitivity to parameter's scale

It is important to consider the unit (i.e., the scale) of the predictors when using an index based on the ROPE, as the correct interpretation of the ROPE as representing a region of practical equivalence to zero is dependent on the scale of the predictors. Indeed, the percentage in ROPE depend on the unit of its parameter. In other words, as the ROPE represents a fixed portion of the response's scale, its proximity with a coefficient depends on the scale of the coefficient itself.

Multicollinearity: Non-independent covariates

When parameters show strong correlations, i.e. when covariates are not independent, the joint parameter distributions may shift towards or away from the ROPE. Collinearity invalidates ROPE and hypothesis testing based on univariate marginals, as the probabilities are conditional on independence. Most problematic are parameters that only have partial overlap with the ROPE region. In case of collinearity, the (joint) distributions of these parameters may either get an increased or decreased ROPE, which means that inferences based on `rope()` are inappropriate (*Kruschke 2014, 340f*).

`rope()` performs a simple check for pairwise correlations between parameters, but as there can be collinearity between more than two variables, a first step to check the assumptions of this hypothesis testing is to look at different pair plots. An even more sophisticated check is the projection predictive variable selection (*Piironen and Vehtari 2017*).

References

- Cohen, J. (1988). Statistical power analysis for the behavioural sciences.
- Kruschke, J. K. (2010). What to believe: Bayesian methods for data analysis. Trends in cognitive sciences, 14(7), 293-300. doi: 10.1016/j.tics.2010.05.001.
- Kruschke, J. K. (2011). Bayesian assessment of null values via parameter estimation and model comparison. Perspectives on Psychological Science, 6(3), 299-312. doi: 10.1177/1745691611406925.
- Kruschke, J. K. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press. doi: 10.1177/2515245918771304.
- Kruschke, J. K. (2018). Rejecting or accepting parameter values in Bayesian estimation. Advances in Methods and Practices in Psychological Science, 1(2), 270-280. doi: 10.1177/2515245918771304.
- Piironen, J., & Vehtari, A. (2017). Comparison of Bayesian predictive methods for model selection. Statistics and Computing, 27(3), 711–735. doi: 10.1007/s112220169649y

Examples

```
library(bayestestR)
```

```
rope(x = rnorm(1000, 0, 0.01), range = c(-0.1, 0.1))
rope(x = rnorm(1000, 0, 1), range = c(-0.1, 0.1))
rope(x = rnorm(1000, 1, 0.01), range = c(-0.1, 0.1))
rope(x = rnorm(1000, 1, 1), ci = c(.90, .95))

library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
rope(model)
rope(model, ci = c(.90, .95))

library(emmeans)
rope(emtrends(model, ~1, "wt"), ci = c(.90, .95))
## Not run:
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
rope(model)
rope(model, ci = c(.90, .95))

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
rope(bf)
rope(bf, ci = c(.90, .95))

## End(Not run)
```

`rope_range` *Find Default Equivalence (ROPE) Region Bounds*

Description

This function attempts at automatically finding suitable "default" values for the Region Of Practical Equivalence (ROPE).

Usage

```
rope_range(x, ...)
```

Arguments

`x`	A `stanreg`, `brmsfit` or `BFBayesFactor` object.
`...`	Currently not used.

Details

Kruschke (2018) suggests that the region of practical equivalence could be set, by default, to a range from `-0.1` to `0.1` of a standardized parameter (negligible effect size according to Cohen, 1988).

- For **linear models (lm)**, this can be generalised to $[-0.1 * SD_y, 0.1 * SD_y]$.

- For **logistic models**, the parameters expressed in log odds ratio can be converted to standardized difference through the formula $\pi/\sqrt{3}$, resulting in a range of `-0.18` to `0.18`.
- For other models with **binary outcome**, it is strongly recommended to manually specify the rope argument. Currently, the same default is applied that for logistic models.
- For **t-tests**, the standard deviation of the response is used, similarly to linear models (see above).
- For **correlations**, `-0.05,0.05` is used, i.e., half the value of a negligible correlation as suggested by Cohen's (1988) rules of thumb.
- For all other models, `-0.1,0.1` is used to determine the ROPE limits, but it is strongly advised to specify it manually.

References

Kruschke, J. K. (2018). Rejecting or accepting parameter values in Bayesian estimation. Advances in Methods and Practices in Psychological Science, 1(2), 270-280. doi: 10.1177/2515245918771304.

Examples

```
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
rope_range(model)
## Not run:
library(rstanarm)
model <- stan_glm(vs ~ mpg, data = mtcars, family = "binomial")
rope_range(model)

library(brms)
model <- brm(mpg ~ wt + cyl, data = mtcars)
rope_range(model)

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
rope_range(bf)

## End(Not run)
```

---

sensitivity_to_prior *Sensitivity to Prior*

Description

Computes the sensitivity to priors specification. This represents the proportion of change in some indices when the model is fitted with an antagonistic prior (a prior of same shape located on the opposite of the effect).

Usage

```
sensitivity_to_prior(model, index = "Median", magnitude = 10, ...)
```

Arguments

model	A Bayesian model (stanreg or brmsfit).
index	The indices from which to compute the sensitivity. Can be one or multiple names of the columns returned by describe_posterior. The case is important here (e.g., write 'Median' instead of 'median').
magnitude	This represent the magnitude by which to shift the antagonistic prior (to test the sensitivity). For instance, a magnitude of 10 (default) means that the mode wil be updated with a prior located at 10 standard deviations from its original location.
...	Arguments passed to or from other methods.

See Also

DescTools

Examples

```
## Not run:
library(bayestestR)

# rstanarm models
# ---------------------------------------------
library(rstanarm)
model <- rstanarm::stan_glm(mpg ~ wt, data = mtcars)
sensitivity_to_prior(model)

model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
sensitivity_to_prior(model, index = c("Median", "MAP"))


# brms models
# ---------------------------------------------
library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
# sensitivity_to_prior(model)

## End(Not run)
```

---

simulate_correlation *Data Simulation*

Description

Simulate data with specific characteristics.

Usage

```
simulate_correlation(n = 100, r = 0.5, mean = 0, sd = 1,
  names = NULL, ...)

simulate_ttest(n = 100, d = 0.5, names = NULL, ...)
```

Arguments

n	The number of observations to be generated.
r	A value or vector corresponding to the desired correlation coefficients.
mean	A value or vector corresponding to the mean of the variables.
sd	A value or vector corresponding to the SD of the variables.
names	A character vector of desired variable names.
...	Arguments passed to or from other methods.
d	A value or vector corresponding to the desired difference between the groups.

Examples

```
# Correlation ------------------------------
data <- simulate_correlation(r = 0.5)
plot(data$V1, data$V2)
cor.test(data$V1, data$V2)
summary(lm(V2 ~ V1, data = data))

# Specify mean and SD
data <- simulate_correlation(r = 0.5, n = 50, mean = c(0, 1), sd = c(0.7, 1.7))
cor.test(data$V1, data$V2)
round(c(mean(data$V1), sd(data$V1)), 1)
round(c(mean(data$V2), sd(data$V2)), 1)
summary(lm(V2 ~ V1, data = data))

# Generate multiple variables
cor_matrix <- matrix(c(
  1.0, 0.2, 0.4,
  0.2, 1.0, 0.3,
  0.4, 0.3, 1.0
),
nrow = 3
)

data <- simulate_correlation(r = cor_matrix, names = c("y", "x1", "x2"))
cor(data)
summary(lm(y ~ x1, data = data))

# t-test -----------------------------
data <- simulate_ttest(n = 30, d = 0.3)
plot(data$V1, data$V0)
round(c(mean(data$V1), sd(data$V1)), 1)
```

```
diff(t.test(data$V1 ~ data$V0)$estimate)
summary(lm(V1 ~ V0, data = data))
summary(glm(V0 ~ V1, data = data, family = "binomial"))
```

`simulate_prior` *Returns Priors of a Model as Empirical Distributions*

Description

Transforms priors information to actual distributions.

Usage

```
simulate_prior(model, n = 1000, ...)
```

Arguments

`model`	A `stanreg`, `stanfit`, or `brmsfit` object.
`n`	Size of the simulated prior distributions.
`...`	Currently not used.

Examples

```
library(bayestestR)
library(rstanarm)

model <- stan_glm(mpg ~ wt + am, data = mtcars, chains = 1, refresh = 0)
simulate_prior(model)
```

`update.bayesfactor_models`
Update bayesfactor_models

Description

Update bayesfactor_models

Usage

```
## S3 method for class 'bayesfactor_models'
update(object, subset = NULL,
  reference = NULL, ...)
```

Arguments

object	A bayesfactor_models object.
subset	Vector of model indices to keep or remove.
reference	Index of model to rereference to, or "top" to reference to the best model, or "bottom" to reference to the worst model.
...	Currently not used.

Examples

```
library(lme4)
lmer1 <- lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)
lmer2 <- lmer(Sepal.Length ~ Petal.Length + (Petal.Length | Species), data = iris)
lmer3 <- lmer(
  Sepal.Length ~ Petal.Length + (Petal.Length | Species) + (1 | Petal.Width),
  data = iris
)

m <- bayesfactor_models(lmer1, lmer2, lmer3, denominator = 1)
m

update(m, reference = "bottom")
```

Index