Package 'bayestestR'

February 7, 2025

Type Package

Title Understand and Describe Bayesian Models and Posterior
Distributions

Version 0.15.2

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

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). References: Makowski et al. (2021) <doi:10.21105/joss.01541>.

Depends R (>= 3.6)

Imports insight (>= 1.0.1), datawizard (>= 1.0.0), graphics, methods,
stats, utils

Suggests BayesFactor (>= 0.9.12-4.4), bayesQR, bayesplot, betareg, BH,
blavaan, bridgesampling, brms, collapse, curl, effectsize,
emmeans, gamm4, ggdist, ggplot2, glmmTMB, httr2, KernSmooth,
knitr, lavaan, lme4, logspline (>= 2.1.21), marginaleffects (>=
0.24.0), MASS, mclust, mediation, modelbased, ordbetareg,
parameters, patchwork, performance, quadprog, posterior,
RcppEigen, rmarkdown, rstan, rstanarm, see (>= 0.8.5),
testthat, tweedie, withr

License GPL-3

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

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

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/testthat/parallel true

Config/rcmdcheck/ignore-inconsequential-notes true

Config/Needs/website easystats/easystatstemplate

Config/Needs/check stan-dev/cmdstanr

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>),
     Indrajeet Patil [aut] (<https://orcid.org/0000-0003-1995-6531>),
     Micah K. Wilson [aut] (<https://orcid.org/0000-0003-4143-7308>),
     Brenton M. Wiernik [aut] (<https://orcid.org/0000-0001-9560-6336>),
     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>),
     Sam Crawley [ctb] (<https://orcid.org/0000-0002-7847-0411>)

Repository CRAN

Date/Publication 2025-02-07 09:10:02 UTC

Contents

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_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 scalar (for point-null) or a range (for a interval-null).
hypothesis	A character vector specifying the restrictions as logical conditions (see examples below).
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().

Note

There is also a plot()-method implemented in the **see**-package.

Examples

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

prior <- distribution_normal(1000, mean = 0, sd = 1)
```

```
posterior <- distribution_normal(1000, mean = 0.5, sd = 0.3)

bayesfactor(posterior, prior = prior, verbose = FALSE)

# rstanarm models
# ---------------
model <- suppressWarnings(rstanarm::stan_lmer(extra ~ group + (1 | ID), data = sleep))
bayesfactor(model, verbose = FALSE)

# 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)

## End(Not run)
```

---

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

Description

The bf_* function is an alias of the main function.

For more info, see the Bayes factors vignette.

Usage

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

bf_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?

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

Value

a data frame containing the prior and posterior probabilities, and log(BF) for each effect (Use `as.numeric()` to extract the non-log Bayes factors; see examples).

Interpreting Bayes Factors

A Bayes factor greater than 1 can be interpreted 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*).

Note

Random effects in the `lmer` style are converted to interaction terms: i.e., `(X|G)` will become the terms `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.

See Also

`weighted_posteriors()` for Bayesian parameter averaging.

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)
(bf_inc <- bayesfactor_inclusion(BFmodels))

as.numeric(bf_inc)


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

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

---

bayesfactor_models   *Bayes Factors (BF) for model comparison*

Description

This function computes or extracts Bayes factors from fitted models.

The bf_* function is an alias of the main function.

Usage

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

bf_models(..., denominator = 1, verbose = TRUE)

## Default S3 method:
bayesfactor_models(..., denominator = 1, verbose = TRUE)

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

## S3 method for class 'bayesfactor_models'
as.matrix(x, ...)
```

Arguments

...	Fitted models (see details), all fit on the same data, or a single BFBayesFactor object (see 'Details'). Ignored in as.matrix(), update(). If the following named arguments are present, they are passed to insight::get_loglikelihood() (see details):

- estimator (defaults to "ML")
- check_response (defaults to FALSE)

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.
object, x	A bayesfactor_models() object.
subset	Vector of model indices to keep or remove.
reference	Index of model to reference to, or "top" to reference to the best model, or "bottom" to reference to the worst model.

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_pars = save_pars(all = TRUE).
  - stanreg models must have been fitted with a defined diagnostic_file.

- For BFBayesFactor, bayesfactor_models() is mostly a wraparound BayesFactor::extractBF().

- For all other model types, Bayes factors are computed using the BIC approximation. Note that BICs are extracted from using insight::get_loglikelihood, see documentation there for options for dealing with transformed responses and REML estimation.

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**osteriors. How many? The number of posterior samples needed for testing is substantially larger than for estimation (the default of 4000 samples may not be enough in many cases). A conservative rule of thumb is to obtain 10 times more samples than would be required for estimation (*Gronau, Singmann, & Wagenmakers, 2017*). If less than 40,000 samples are detected, bayesfactor_models() gives a warning.

See also the Bayes factors vignette.

Value

A data frame containing the models' formulas (reconstructed fixed and random effects) and their log(BF)s (Use as.numeric() to extract the non-log Bayes factors; see examples), that prints nicely.

Interpreting Bayes Factors

A Bayes factor greater than 1 can be interpreted 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*).

Note

There is also a plot()-method implemented in the **see**-package.

Author(s)

Mattan S. Ben-Shachar

References

- Gronau, Q. F., Singmann, H., & Wagenmakers, E. J. (2017). Bridgesampling: An R package for estimating normalizing constants. arXiv preprint arXiv:1710.08162.
- 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(mpg ~ 1, data = mtcars)
lm2 <- lm(mpg ~ hp, data = mtcars)
lm3 <- lm(mpg ~ hp + drat, data = mtcars)
lm4 <- lm(mpg ~ hp * drat, data = mtcars)
(BFM <- 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

update(BFM, reference = "bottom")
as.matrix(BFM)
as.numeric(BFM)

lm2b <- lm(sqrt(mpg) ~ hp, data = mtcars)
# Set check_response = TRUE for transformed responses
bayesfactor_models(lm2b, denominator = lm2, check_response = TRUE)
```

```
# With lmerMod objects:
# ---------------------
lmer1 <- lme4::lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)
lmer2 <- lme4::lmer(Sepal.Length ~ Petal.Length + (Petal.Length | Species), data = iris)
lmer3 <- lme4::lmer(
  Sepal.Length ~ Petal.Length + (Petal.Length | Species) + (1 | Petal.Width),
  data = iris
)
bayesfactor_models(lmer1, lmer2, lmer3,
  denominator = 1,
  estimator = "REML"
)


# rstanarm models
# ---------------------
# (note that a unique diagnostic_file MUST be specified in order to work)
stan_m0 <- suppressWarnings(rstanarm::stan_glm(Sepal.Length ~ 1,
  data = iris,
  family = gaussian(),
  diagnostic_file = file.path(tempdir(), "df0.csv")
))
stan_m1 <- suppressWarnings(rstanarm::stan_glm(Sepal.Length ~ Species,
  data = iris,
  family = gaussian(),
  diagnostic_file = file.path(tempdir(), "df1.csv")
))
stan_m2 <- suppressWarnings(rstanarm::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, verbose = FALSE)


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

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


# BayesFactor
# --------------------------
data(puzzles)
BF <- BayesFactor::anovaBF(RT ~ shape * color + ID,
  data = puzzles,
```

```
    whichRandom = "ID", progress = FALSE
  )
  BF
  bayesfactor_models(BF) # basically the same
```

---

bayesfactor_parameters

Bayes Factors (BF) for a Single Parameter

Description

This method computes Bayes factors against the null (either a point or an interval), based 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 interval (Morey & Rouder, 2011; Liao et al., 2020); 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 (Wagenmakers et al., 2010; Heck, 2019).

Note that the `logspline` package is used for estimating densities and probabilities, and must be installed for the function to work.

`bayesfactor_pointnull()` and `bayesfactor_rope()` are wrappers around `bayesfactor_parameters` with different defaults for the null to be tested against (a point and a range, respectively). Aliases of the main functions are prefixed with `bf_*`, like `bf_parameters()` or `bf_pointnull()`.

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

Usage

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

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

bayesfactor_rope(
  posterior,
  prior = NULL,
  direction = "two-sided",
  null = rope_range(posterior, verbose = FALSE),
  ...,
  verbose = TRUE
)

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

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

bf_rope(
  posterior,
  prior = NULL,
  direction = "two-sided",
  null = rope_range(posterior, verbose = FALSE),
  ...,
  verbose = TRUE
)

## 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,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "location", "smooth_terms", "sigma", "zi",
    "zero_inflated", "all"),
  parameters = NULL,
  ...,
  verbose = TRUE
)

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

## S3 method for class 'blavaan'
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,
    rvar_col = NULL,
    ...,
    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 scalar (for point-null) or a range (for a interval-null).
...	Arguments passed to and from other methods. (Can be used to pass arguments to internal logspline::logspline().)
verbose	Toggle off warnings.
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.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().

Details

This method is used to compute Bayes factors based on prior and posterior distributions.

One-sided & Dividing 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, 2014*). 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). For example, for a Bayes factor comparing the "null" of 0-0.1 to the alternative >0.1, we would set bayesfactor_parameters(null = c(0, 0.1), direction = ">").

It is also possible to compute a Bayes factor for **dividing** hypotheses - that is, for a null and alternative that are complementary, opposing one-sided hypotheses (*Morey & Wagenmakers, 2014*). For example, for a Bayes factor comparing the "null" of <0 to the alternative >0, we would set bayesfactor_parameters(null = c(-Inf, 0)).

Value

A data frame containing the (log) Bayes factor representing evidence *against* the null (Use `as.numeric()` to extract the non-log Bayes factors; see examples).

Setting the correct `prior`

For the computation of Bayes factors, the model priors must be proper priors (at the very least they should be *not flat*, and it is preferable that they be *informative*); As the priors for the alternative get wider, the likelihood of the null value(s) increases, to the extreme that for completely flat priors the null is infinitely more favorable than the alternative (this is called *the Jeffreys-Lindley-Bartlett paradox*). Thus, you should only ever try (or want) to compute a Bayes factor when you have an informed prior.

(Note that by default, `brms::brm()` uses flat priors for fixed-effects; See example below.)

It is important to provide the correct `prior` for meaningful results, to match the `posterior`-type input:

- **A numeric vector** - `prior` should also be a *numeric vector*, representing the prior-estimate.
- **A data frame** - `prior` should also be a *data frame*, representing the prior-estimates, in matching column order.
    - If `rvar_col` is specified, `prior` should be *the name of an* `rvar` *column* that represents the prior-estimates.
- **Supported Bayesian model** (`stanreg`**,** `brmsfit`**, etc.)**
    - `prior` should be *a model an equivalent model with MCMC samples from the priors* **only**. See [unupdate()](#).
    - If `prior` is set to `NULL`, [unupdate()](#) is called internally (not supported for `brmsfit_multiple` model).
- **Output from a** {`marginaleffects`} **function** - `prior` should also be *an equivalent output* from a {marginaleffects} function based on a prior-model (See [unupdate()](#)).
- **Output from an** {`emmeans`} **function**
    - `prior` should also be *an equivalent output* from an {emmeans} function based on a prior-model (See [unupdate()](#)).
    - `prior` can also be *the original (posterior) model*, in which case the function will try to "unupdate" the estimates (not supported if the estimates have undergone any transformations – `"log"`, `"response"`, etc. – or any `regriding`).

Interpreting Bayes Factors

A Bayes factor greater than 1 can be interpreted 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*).

Note

There is also a [plot()-method](#) implemented in the **see**-package.

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.

- 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.

- Liao, J. G., Midya, V., & Berg, A. (2020). Connecting and contrasting the Bayes factor and a modified ROPE procedure for testing interval null hypotheses. The American Statistician, 1-19.

- 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

```
library(bayestestR)
prior <- distribution_normal(1000, mean = 0, sd = 1)
posterior <- distribution_normal(1000, mean = .5, sd = .3)
(BF_pars <- bayesfactor_parameters(posterior, prior, verbose = FALSE))

as.numeric(BF_pars)



# rstanarm models
# ---------------
contrasts(sleep$group) <- contr.equalprior_pairs # see vingette
stan_model <- suppressWarnings(stan_lmer(
  extra ~ group + (1 | ID),
  data = sleep,
  refresh = 0
))
bayesfactor_parameters(stan_model, verbose = FALSE)
bayesfactor_parameters(stan_model, null = rope_range(stan_model))

# emmGrid objects
# ---------------
group_diff <- pairs(emmeans(stan_model, ~group, data = sleep))
bayesfactor_parameters(group_diff, prior = stan_model, verbose = FALSE)
```

```
# Or
# group_diff_prior <- pairs(emmeans(unupdate(stan_model), ~group))
# bayesfactor_parameters(group_diff, prior = group_diff_prior, verbose = FALSE)



# brms models
# -----------
## Not run:
contrasts(sleep$group) <- contr.equalprior_pairs # see vignette
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 <- suppressWarnings(brm(extra ~ group + (1 | ID),
  data = sleep,
  prior = my_custom_priors,
  refresh = 0
))
bayesfactor_parameters(brms_model, verbose = FALSE)

## 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*.

The bf_* function is an alias of the main function.

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

Usage

```
bayesfactor_restricted(posterior, ...)

bf_restricted(posterior, ...)

## S3 method for class 'stanreg'
bayesfactor_restricted(
  posterior,
  hypothesis,
```

```
  prior = NULL,
  verbose = TRUE,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  ...
)

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

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

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

## S3 method for class 'data.frame'
bayesfactor_restricted(
  posterior,
  hypothesis,
  prior = NULL,
  rvar_col = NULL,
  ...
)

## S3 method for class 'bayesfactor_restricted'
as.logical(x, which = c("posterior", "prior"), ...)
```

Arguments

posterior	A stanreg / brmsfit object, emmGrid or a data frame - representing a posterior distribution(s) from (see Details).
...	Currently not used.
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.
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.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().
x	An object of class bayesfactor_restricted
which	Should the logical matrix be of the posterior or prior distribution(s)?

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).

Value

A data frame containing the (log) Bayes factor representing evidence *against* the un-restricted model (Use as.numeric() to extract the non-log Bayes factors; see examples). (A bool_results attribute contains the results for each sample, indicating if they are included or not in the hypothesized restriction.)

Setting the correct prior

For the computation of Bayes factors, the model priors must be proper priors (at the very least they should be *not flat*, and it is preferable that they be *informative*); As the priors for the alternative get wider, the likelihood of the null value(s) increases, to the extreme that for completely flat priors the null is infinitely more favorable than the alternative (this is called *the Jeffreys-Lindley-Bartlett paradox*). Thus, you should only ever try (or want) to compute a Bayes factor when you have an informed prior.

(Note that by default, brms::brm() uses flat priors for fixed-effects; See example below.)

It is important to provide the correct prior for meaningful results, to match the posterior-type input:

- **A numeric vector** - `prior` should also be a *numeric vector*, representing the prior-estimate.
- **A data frame** - `prior` should also be a *data frame*, representing the prior-estimates, in matching column order.
  - If `rvar_col` is specified, `prior` should be *the name of an* `rvar` *column* that represents the prior-estimates.
- **Supported Bayesian model** (`stanreg`, `brmsfit`**, etc.**)
  - `prior` should be *a model an equivalent model with MCMC samples from the priors* **only**. See `unupdate()`.
  - If `prior` is set to `NULL`, `unupdate()` is called internally (not supported for `brmsfit_multiple` model).
- **Output from a** {`marginaleffects`} **function** - `prior` should also be *an equivalent output* from a {marginaleffects} function based on a prior-model (See `unupdate()`).
- **Output from an** {`emmeans`} **function**
  - `prior` should also be *an equivalent output* from an {emmeans} function based on a prior-model (See `unupdate()`).
  - `prior` can also be *the original (posterior) model*, in which case the function will try to "unupdate" the estimates (not supported if the estimates have undergone any transformations – `"log"`, `"response"`, etc. – or any `regriding`).

Interpreting Bayes Factors

A Bayes factor greater than 1 can be interpreted 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*).

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. Retrieved from https://richarddmorey.org/category/order-restrictions/.

Examples

```
set.seed(444)
library(bayestestR)
prior <- data.frame(
  A = rnorm(500),
  B = rnorm(500),
  C = rnorm(500)
)

posterior <- data.frame(
  A = rnorm(500, .4, 0.7),
```

```
    B = rnorm(500, -.2, 0.4),
    C = rnorm(500, 0, 0.5)
)

hyps <- c(
  "A > B & B > C",
  "A > B & A > C",
  "C > A"
)


(b <- bayesfactor_restricted(posterior, hypothesis = hyps, prior = prior))

bool <- as.logical(b, which = "posterior")
head(bool)



see::plots(
  plot(estimate_density(posterior)),
  # distribution **conditional** on the restrictions
  plot(estimate_density(posterior[bool[, hyps[1]], ])) + ggplot2::ggtitle(hyps[1]),
  plot(estimate_density(posterior[bool[, hyps[2]], ])) + ggplot2::ggtitle(hyps[2]),
  plot(estimate_density(posterior[bool[, hyps[3]], ])) + ggplot2::ggtitle(hyps[3]),
  guides = "collect"
)



# rstanarm models
# ---------------
data("mtcars")

fit_stan <- rstanarm::stan_glm(mpg ~ wt + cyl + am,
  data = mtcars, refresh = 0
)
hyps <- c(
  "am > 0 & cyl < 0",
  "cyl < 0",
  "wt - cyl > 0"
)

bayesfactor_restricted(fit_stan, hypothesis = hyps)



# emmGrid objects
# ---------------
# replicating http://bayesfactor.blogspot.com/2015/01/multiple-comparisons-with-bayesfactor-2.html
data("disgust")
contrasts(disgust$condition) <- contr.equalprior_pairs # see vignette
fit_model <- rstanarm::stan_glm(score ~ condition, data = disgust, family = gaussian())
```

```
em_condition <- emmeans::emmeans(fit_model, ~condition, data = disgust)
hyps <- c("lemon < control & control < sulfur")

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

---

bci   *Bias Corrected and Accelerated Interval (BCa)*

Description

Compute the **Bias Corrected and Accelerated Interval (BCa)** of posterior distributions.

Usage

```
bci(x, ...)

bcai(x, ...)

## S3 method for class 'numeric'
bci(x, ci = 0.95, verbose = TRUE, ...)

## S3 method for class 'data.frame'
bci(x, ci = 0.95, rvar_col = NULL, verbose = TRUE, ...)

## S3 method for class 'MCMCglmm'
bci(x, ci = 0.95, verbose = TRUE, ...)

## S3 method for class 'sim.merMod'
bci(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'sim'
bci(x, ci = 0.95, parameters = NULL, verbose = TRUE, ...)
```

```
## S3 method for class 'emmGrid'
bci(x, ci = 0.95, verbose = TRUE, ...)

## S3 method for class 'slopes'
bci(x, ci = 0.95, verbose = TRUE, ...)

## S3 method for class 'stanreg'
bci(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
bci(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'BFBayesFactor'
bci(x, ci = 0.95, verbose = TRUE, ...)

## S3 method for class 'get_predicted'
bci(x, ci = 0.95, use_iterations = FALSE, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, `methods("hdi")`) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the `.numeric` or `.data.frame` methods.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to `.95` (95%).
verbose	Toggle off warnings.

rvar_col   A single character - the name of an rvar column in the data frame to be pro-
                    cessed. See example in p_direction().

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 param-
                    eters 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.

use_iterations   Logical, if TRUE and x is a get_predicted object, (returned by insight::get_predicted()),
                    the function is applied to the iterations instead of the predictions. This only ap-
                    plies to models that return iterations for predicted values (e.g., brmsfit models).

Details

Unlike equal-tailed intervals (see eti()) that typically exclude 2.5% from each tail of the distribu-
tion and always include the median, the HDI is *not* equal-tailed and therefore always includes the
mode(s) of posterior distributions. While this can be useful to better represent the credibility mass
of a distribution, the HDI also has some limitations. See spi() for details.

The 95% or 89% Credible Intervals (CI) are two reasonable ranges to characterize the uncertainty
related to the estimation (see here for a discussion about the differences between these two values).

The 89% intervals (ci = 0.89) are deemed to be more stable than, for instance, 95% intervals (*Kr-
uschke, 2014*). An effective sample size of at least 10.000 is recommended if one wants to estimate
95% intervals with high precision (*Kruschke, 2014, p. 183ff*). Unfortunately, the default number of
posterior samples for most Bayes packages (e.g., rstanarm or brms) is only 4.000 (thus, you might
want to increase it when fitting your model). 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*).

However, 95% has some advantages too. For instance, it shares (in the case of a normal posterior
distribution) an intuitive relationship with the standard deviation and it conveys a more accurate
image of the (artificial) bounds of the distribution. Also, because it is wider, it makes analyses more
conservative (i.e., the probability of covering 0 is larger for the 95% CI than for lower ranges such
as 89%), which is a good thing in the context of the reproducibility crisis.

A 95% equal-tailed interval (ETI) has 2.5% of the distribution on either side of its limits. It indi-
cates the 2.5th percentile and the 97.5h 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.

References

DiCiccio, T. J. and B. Efron. (1996). Bootstrap Confidence Intervals. Statistical Science. 11(3): 189–212. 10.1214/ss/1032280214

See Also

Other ci: `ci()`, `eti()`, `hdi()`, `si()`, `spi()`

Examples

```
posterior <- rnorm(1000)
bci(posterior)
bci(posterior, ci = c(0.80, 0.89, 0.95))
```

| bic_to_bf | *Convert BIC indices to Bayes Factors via the BIC-approximation method.* |

Description

The difference between two Bayesian information criterion (BIC) indices of two models can be used to approximate Bayes factors via:

$$BF_{10} = e^{(BIC_0 - BIC_1)/2}$$

Usage

```
bic_to_bf(bic, denominator, log = FALSE)
```

Arguments

bic	A vector of BIC values.
denominator	The BIC value to use as a denominator (to test against).
log	If `TRUE`, return the `log(BF)`.

Value

The Bayes Factors corresponding to the BIC values against the denominator.

References

Wagenmakers, E. J. (2007). A practical solution to the pervasive problems of p values. Psychonomic bulletin & review, 14(5), 779-804

Examples

```
bic1 <- BIC(lm(Sepal.Length ~ 1, data = iris))
bic2 <- BIC(lm(Sepal.Length ~ Species, data = iris))
bic3 <- BIC(lm(Sepal.Length ~ Species + Petal.Length, data = iris))
bic4 <- BIC(lm(Sepal.Length ~ Species * Petal.Length, data = iris))

bic_to_bf(c(bic1, bic2, bic3, bic4), denominator = bic1)
```

---

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 *Gelman et al. 2017*.

Usage

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

Arguments

model	A stanreg, stanfit, brmsfit, blavaan, or MCMCglmm 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.
simulate_priors	
	Should prior distributions be simulated using simulate_prior() (default; faster) or sampled via unupdate() (slower, more accurate).
...	Currently not used.

Value

A data frame with two columns: The parameter names and the quality of the prior (which might be "informative", "uninformative") or "not determinable" if the prior distribution could not be determined).

References

Gelman, A., Simpson, D., and Betancourt, M. (2017). The Prior Can Often Only Be Understood in the Context of the Likelihood. Entropy, 19(10), 555. doi:10.3390/e19100555

Examples

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

# An extreme example where both methods diverge:
model <- rstanarm::stan_glm(mpg ~ wt,
  data = mtcars[1:3, ],
  prior = normal(-3.3, 1, FALSE),
  prior_intercept = normal(0, 1000, FALSE),
  refresh = 0
)
check_prior(model, method = "gelman")
check_prior(model, method = "lakeland")
# can provide visual confirmation to the Lakeland method
plot(si(model, verbose = FALSE))
```

---

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

Description

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

Usage

```
ci(x, ...)

## S3 method for class 'numeric'
ci(x, ci = 0.95, method = "ETI", verbose = TRUE, BF = 1, ...)

## S3 method for class 'data.frame'
ci(x, ci = 0.95, method = "ETI", BF = 1, rvar_col = NULL, verbose = TRUE, ...)

## S3 method for class 'sim.merMod'
ci(
  x,
  ci = 0.95,
  method = "ETI",
  effects = c("fixed", "random", "all"),
  parameters = NULL,
  verbose = TRUE,
  ...
```

```
)

## S3 method for class 'sim'
ci(x, ci = 0.95, method = "ETI", parameters = NULL, verbose = TRUE, ...)

## S3 method for class 'stanreg'
ci(
  x,
  ci = 0.95,
  method = "ETI",
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  BF = 1,
  ...
)

## S3 method for class 'brmsfit'
ci(
  x,
  ci = 0.95,
  method = "ETI",
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  verbose = TRUE,
  BF = 1,
  ...
)

## S3 method for class 'BFBayesFactor'
ci(x, ci = 0.95, method = "ETI", verbose = TRUE, BF = 1, ...)

## S3 method for class 'MCMCglmm'
ci(x, ci = 0.95, 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 `0.95` (95%).
method	Can be "ETI" (default), "HDI", "BCI", "SPI" or "SI".
verbose	Toggle off warnings.
BF	The amount of support required to be included in the support interval.

rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().
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*).

There is also a plot()-method implemented in the **see**-package.

References

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

See Also

Other ci: bci(), eti(), hdi(), si(), spi()

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(0.80, 0.89, 0.95))
ci(df, method = "HDI", ci = c(0.80, 0.89, 0.95))

model <- suppressWarnings(
  stan_glm(mpg ~ wt, data = mtcars, chains = 2, iter = 200, refresh = 0)
)
ci(model, method = "ETI", ci = c(0.80, 0.89))
ci(model, method = "HDI", ci = c(0.80, 0.89))


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


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

---

contr.equalprior   *Contrast Matrices for Equal Marginal Priors in Bayesian Estimation*

Description

Build contrasts for factors with equal marginal priors on all levels. The 3 functions give the same orthogonal contrasts, but are scaled differently to allow different prior specifications (see 'Details'). Implementation from Singmann & Gronau's bfrms, following the description in Rouder, Morey, Speckman, & Province (2012, p. 363).

Usage

```
contr.equalprior(n, contrasts = TRUE, sparse = FALSE)

contr.equalprior_pairs(n, contrasts = TRUE, sparse = FALSE)

contr.equalprior_deviations(n, contrasts = TRUE, sparse = FALSE)
```

Arguments

n	a vector of levels for a factor, or the number of levels.
contrasts	a logical indicating whether contrasts should be computed.
sparse	logical indicating if the result should be sparse (of class dgCMatrix), using package **Matrix**.

Details

When using `stats::contr.treatment`, each dummy variable is the difference between each level and the reference level. While this is useful if setting different priors for each coefficient, it should not be used if one is trying to set a general prior for differences between means, as it (as well as `stats::contr.sum` and others) results in unequal marginal priors on the means the the difference between them.

```
library(brms)

data <- data.frame(
  group = factor(rep(LETTERS[1:4], each = 3)),
  y = rnorm(12)
)

contrasts(data$group) # R's default contr.treatment
#>   B C D
#> A 0 0 0
#> B 1 0 0
#> C 0 1 0
#> D 0 0 1

model_prior <- brm(
  y ~ group, data = data,
  sample_prior = "only",
  # Set the same priors on the 3 dummy variable
  # (Using an arbitrary scale)
  prior = set_prior("normal(0, 10)", coef = c("groupB", "groupC", "groupD"))
)

est <- emmeans::emmeans(model_prior, pairwise ~ group)

point_estimate(est, centr = "mean", disp = TRUE)
#> Point Estimate
#>
#> Parameter |  Mean |    SD
#> ------------------------
#> A         | -0.01 |  6.35
#> B         | -0.10 |  9.59
#> C         |  0.11 |  9.55
#> D         | -0.16 |  9.52
#> A - B     |  0.10 |  9.94
#> A - C     | -0.12 |  9.96
#> A - D     |  0.15 |  9.87
#> B - C     | -0.22 | 14.38
#> B - D     |  0.05 | 14.14
#> C - D     |  0.27 | 14.00
```

We can see that the priors for means aren't all the same (A having a more narrow prior), and likewise

for the pairwise differences (priors for differences from A are more narrow).

The solution is to use one of the methods provided here, which *do* result in marginally equal priors on means differences between them. Though this will obscure the interpretation of parameters, setting equal priors on means and differences is important for they are useful for specifying equal priors on all means in a factor and their differences correct estimation of Bayes factors for contrasts and order restrictions of multi-level factors (where k>2). See info on specifying correct priors for factors with more than 2 levels in the Bayes factors vignette.

NOTE: When setting priors on these dummy variables, always:

1. Use priors that are **centered on 0**! Other location/centered priors are meaningless!

2. Use **identically-scaled priors** on all the dummy variables of a single factor!

`contr.equalprior` returns the original orthogonal-normal contrasts as described in Rouder, Morey, Speckman, & Province (2012, p. 363). Setting `contrasts` = FALSE returns the $I_n - \frac{1}{n}$ matrix.

`contr.equalprior_pairs`:

Useful for setting priors in terms of pairwise differences between means - the scales of the priors defines the prior distribution of the pair-wise differences between all pairwise differences (e.g., A - B, B - C, etc.).

```
contrasts(data$group) <- contr.equalprior_pairs
contrasts(data$group)
#>          [,1]        [,2]        [,3]
#> A  0.0000000  0.6123724  0.0000000
#> B -0.1893048 -0.2041241  0.5454329
#> C -0.3777063 -0.2041241 -0.4366592
#> D  0.5670111 -0.2041241 -0.1087736
```

```
model_prior <- brm(
  y ~ group, data = data,
  sample_prior = "only",
  # Set the same priors on the 3 dummy variable
  # (Using an arbitrary scale)
  prior = set_prior("normal(0, 10)", coef = c("group1", "group2", "group3"))
)
```

```
est <- emmeans(model_prior, pairwise ~ group)
```

```
point_estimate(est, centr = "mean", disp = TRUE)
#> Point Estimate
#>
#> Parameter | Mean |    SD
#> ------------------------
#> A         | -0.31 |  7.46
#> B         | -0.24 |  7.47
#> C         | -0.34 |  7.50
#> D         | -0.30 |  7.25
#> A - B     | -0.08 | 10.00
#> A - C     |  0.03 | 10.03
```

```
#> A - D      | -0.01 |  9.85
#> B - C      |  0.10 | 10.28
#> B - D      |  0.06 |  9.94
#> C - D      | -0.04 | 10.18
```

All means have the same prior distribution, and the distribution of the differences matches the prior we set of `"normal(0, 10)"`. Success!

`contr.equalprior_deviations`:

Useful for setting priors in terms of the deviations of each mean from the grand mean - the scales of the priors defines the prior distribution of the distance (above, below) the mean of one of the levels might have from the overall mean. (See examples.)

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

```
contr.equalprior(2) # Q_2 in Rouder et al. (2012, p. 363)

contr.equalprior(5) # equivalent to Q_5 in Rouder et al. (2012, p. 363)

## check decomposition
Q3 <- contr.equalprior(3)
Q3 %*% t(Q3) ## 2/3 on diagonal and -1/3 on off-diagonal elements
```

---

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, REML = TRUE)

bayesian_as_frequentist(model, data = NULL, REML = TRUE)
```

Arguments

`model`	A Bayesian model.
`data`	Data used by the model. If `NULL`, will try to extract it from the model.
`REML`	For mixed effects, should models be estimated using restricted maximum likelihood (REML) (`TRUE`, default) or maximum likelihood (`FALSE`)?

Examples

```
# Rstanarm --------------------
# Simple regressions
model <- rstanarm::stan_glm(Sepal.Length ~ Species,
  data = iris, chains = 2, refresh = 0
)
bayesian_as_frequentist(model)

model <- rstanarm::stan_glm(vs ~ mpg,
  family = "binomial",
  data = mtcars, chains = 2, refresh = 0
)
bayesian_as_frequentist(model)

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

model <- rstanarm::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 characterize the posterior distributions.

Usage

```
describe_posterior(posterior, ...)

## S3 method for class 'numeric'
describe_posterior(
  posterior,
  centrality = "median",
  dispersion = FALSE,
  ci = 0.95,
  ci_method = "eti",
  test = c("p_direction", "rope"),
  rope_range = "default",
  rope_ci = 0.95,
  keep_iterations = FALSE,
  bf_prior = NULL,
  BF = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'data.frame'
describe_posterior(
  posterior,
  centrality = "median",
```

```
  dispersion = FALSE,
  ci = 0.95,
  ci_method = "eti",
  test = c("p_direction", "rope"),
  rope_range = "default",
  rope_ci = 0.95,
  keep_iterations = FALSE,
  bf_prior = NULL,
  BF = 1,
  rvar_col = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'stanreg'
describe_posterior(
  posterior,
  centrality = "median",
  dispersion = FALSE,
  ci = 0.95,
  ci_method = "eti",
  test = c("p_direction", "rope"),
  rope_range = "default",
  rope_ci = 0.95,
  keep_iterations = FALSE,
  bf_prior = NULL,
  diagnostic = c("ESS", "Rhat"),
  priors = FALSE,
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  BF = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
describe_posterior(
  posterior,
  centrality = "median",
  dispersion = FALSE,
  ci = 0.95,
  ci_method = "eti",
  test = c("p_direction", "rope"),
  rope_range = "default",
  rope_ci = 0.95,
  keep_iterations = FALSE,
```

```
    bf_prior = NULL,
    diagnostic = c("ESS", "Rhat"),
    effects = c("fixed", "random", "all"),
    component = c("conditional", "zi", "zero_inflated", "all", "location",
      "distributional", "auxiliary"),
    parameters = NULL,
    BF = 1,
    priors = FALSE,
    verbose = TRUE,
    ...
)
```

Arguments

posterior	A vector, data frame or model of posterior draws. **bayestestR** supports a wide range of models (see methods("describe_posterior")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric method.
...	Additional arguments to be passed to or from methods.
centrality	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: "median", "mean", "MAP" (see map_estimate()), "trimmed" (which is just mean(x, trim = threshold)), "mode" or "all".
dispersion	Logical, if TRUE, computes indices of dispersion related to the estimate(s) (SD and MAD for mean and median, respectively). Dispersion is not available for "MAP" or "mode" centrality indices.
ci	Value or vector of probability of the CI (between 0 and 1) to be estimated. Default to 0.95 (95%).
ci_method	The type of index used for Credible Interval. Can be "ETI" (default, see eti()), "HDI" (see hdi()), "BCI" (see bci()), "SPI" (see spi()), or "SI" (see si()).
test	The indices of effect existence to compute. Character (vector) or list with one or more of these options: "p_direction" (or "pd"), "rope", "p_map", "p_significance" (or "ps"), "p_rope", "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 vector of two values (e.g., c(-0.1, 0.1)), "default" or a list of numeric vectors of the same length as numbers of parameters. 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.
keep_iterations	If TRUE, will keep all iterations (draws) of bootstrapped or Bayesian models. They will be added as additional columns named iter_1, iter_2, ... You can reshape them to a long format by running reshape_iterations().
bf_prior	Distribution representing a prior for the computation of Bayes factors / SI. Used if the input is a posterior, otherwise (in the case of models) ignored.

BF   The amount of support required to be included in the support interval.

verbose   Toggle off warnings.

rvar_col   A single character - the name of an rvar column in the data frame to be pro-
                    cessed. See example in p_direction().

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.

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 param-
                    eters that typically appear in the summary() are returned. Use parameters to
                    select specific parameters for the output.

Details

One or more components of point estimates (like posterior mean or median), intervals and tests can
be omitted 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

- Makowski, D., Ben-Shachar, M. S., Chen, S. H. A., and Lüdecke, D. (2019). *Indices of Effect
  Existence and Significance in the Bayesian Framework*. Frontiers in Psychology 2019;10:2767.
  doi:10.3389/fpsyg.2019.02767

- Region of Practical Equivalence (ROPE)

- Bayes factors

Examples

```
library(bayestestR)

if (require("logspline")) {
  x <- rnorm(1000)
  describe_posterior(x, verbose = FALSE)
  describe_posterior(x,
    centrality = "all",
    dispersion = TRUE,
    test = "all",
    verbose = FALSE
  )
  describe_posterior(x, ci = c(0.80, 0.90), verbose = FALSE)

  df <- data.frame(replicate(4, rnorm(100)))
  describe_posterior(df, verbose = FALSE)
```

```
  describe_posterior(
    df,
    centrality = "all",
    dispersion = TRUE,
    test = "all",
    verbose = FALSE
  )
  describe_posterior(df, ci = c(0.80, 0.90), verbose = FALSE)

  df <- data.frame(replicate(4, rnorm(20)))
  head(reshape_iterations(
    describe_posterior(df, keep_iterations = TRUE, verbose = FALSE)
  ))
}

# rstanarm models
# --------------------------------------------
if (require("rstanarm") && require("emmeans")) {
  model <- suppressWarnings(
    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))
  describe_posterior(model, rope_range = list(c(-10, 5), c(-0.2, 0.2), "default"))

  # emmeans estimates
  # --------------------------------------------
  describe_posterior(emtrends(model, ~1, "wt"))
}

# BayesFactor objects
# --------------------------------------------
if (require("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))
}
```

---

describe_prior   *Describe Priors*

Description

Returns a summary of the priors used in the model.

Usage

```
describe_prior(model, ...)

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

Arguments

model	A Bayesian model.
...	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.

Examples

```
library(bayestestR)

# rstanarm models
# ----------------------------------------------
if (require("rstanarm")) {
  model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
  describe_prior(model)
}

# brms models
# ----------------------------------------------
if (require("brms")) {
  model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
  describe_prior(model)
}

# BayesFactor objects
# ----------------------------------------------
if (require("BayesFactor")) {
  bf <- ttestBF(x = rnorm(100, 1, 1))
```

```
  describe_prior(bf)
}
```

---

diagnostic_draws   *Diagnostic values for each iteration*

Description

Returns the accumulated log-posterior, the average Metropolis acceptance rate, divergent transitions, treedepth rather than terminated its evolution normally.

Usage

```
diagnostic_draws(posterior, ...)
```

Arguments

posterior   A stanreg, stanfit, brmsfit, or blavaan object.

...   Currently not used.

Examples

```
set.seed(333)

if (require("brms", quietly = TRUE)) {
  model <- suppressWarnings(brm(mpg ~ wt * cyl * vs,
    data = mtcars,
    iter = 100, control = list(adapt_delta = 0.80),
    refresh = 0
  ))
  diagnostic_draws(model)
}
```

---

diagnostic_posterior   *Posteriors Sampling Diagnostic*

Description

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

Usage

```
diagnostic_posterior(posterior, ...)

## Default S3 method:
diagnostic_posterior(posterior, diagnostic = c("ESS", "Rhat"), ...)

## S3 method for class 'stanreg'
diagnostic_posterior(
  posterior,
  diagnostic = "all",
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  ...
)

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

Arguments

posterior	A `stanreg`, `stanfit`, `brmsfit`, or `blavaan` object.
...	Currently not used.
diagnostic	Diagnostic metrics to compute. Character (vector) or list with one or more of these options: `"ESS"`, `"Rhat"`, `"MCSE"` or `"all"`.
effects	Should variables for fixed effects (`"fixed"`), random effects (`"random"`) or both (`"all"`) be returned? Only applies to mixed models. May be abbreviated.
component	Which type of parameters to return, such as parameters for the conditional model, the zero-inflated part of the model, the dispersion term, the instrumental variables or marginal effects be returned? Applies to models with zero-inflated and/or dispersion formula, or to models with instrumental variables (so called fixed-effects regressions), or models with marginal effects (from **mfx**). See details in section *Model Components* .May be abbreviated. Note that the *conditional* component also refers to the *count* or *mean* component - names may differ, depending on the modeling package. There are three convenient shortcuts (not applicable to *all* model classes):

- component = `"all"` returns all possible parameters.
- If component = `"location"`, location parameters such as `conditional`, `zero_inflated`, `smooth_terms`, or `instruments` are returned (everything

that are fixed or random effects - depending on the `effects` argument - but no auxiliary parameters).

- For `component = "distributional"` (or `"auxiliary"`), components like `sigma`, `dispersion`, `beta` or `precision` (and other auxiliary parameters) are returned.

parameters   Regular expression pattern that describes the parameters that should be returned.

Details

Effective Sample (ESS) should be as large as possible, although 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 Rhat 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., and 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
# ----------------------------------------------
model <- suppressWarnings(
  rstanarm::stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
)
diagnostic_posterior(model)

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

---

disgust   *Moral Disgust Judgment*

Description

A sample (simulated) dataset, used in tests and some examples.

Format

A data frame with 500 rows and 5 variables:

score Score on the questionnaire, which ranges from 0 to 50 with higher scores representing harsher moral judgment

condition one of three conditions, differing by the odor present in the room: a pleasant scent associated with cleanliness (lemon), a disgusting scent (sulfur), and a control condition in which no unusual odor is present

```
data("disgust")
head(disgust, n = 5)
#>   score condition
#> 1    13   control
#> 2    26   control
#> 3    30   control
#> 4    23   control
#> 5    34   control
```

Author(s)

Richard D. Morey

---

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

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

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

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

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

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

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

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

distribution_mixture_normal(n, mean = c(-3, 3), sd = 1, random = FALSE, ...)

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

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

distribution_nbinom(n, size, prob, mu, phi, random = FALSE, ...)

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

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

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

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

distribution_tweedie(n, xi = NULL, mu, phi, power = NULL, random = FALSE, ...)

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

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

Arguments

type	Can be any of the names from base R's Distributions, like "cauchy", "pois" or "beta".
...	Arguments passed to or from other methods.
n	the number of observations
random	Generate near-perfect or random (simple wrappers for the base R r* functions) distributions.
shape1, shape2	non-negative parameters of the Beta distribution.
ncp	non-centrality parameter.
size	number of trials (zero or more).
prob	probability of success on each trial.

`location, scale`	location and scale parameters.
`df`	degrees of freedom (non-negative, but can be non-integer).
`shape`	Shape parameter.
`mean`	vector of means.
`sd`	vector of standard deviations.
`mu`	the mean
`phi`	Corresponding to glmmTMB's implementation of nbinom distribution, where `size=mu/phi`.
`lambda`	vector of (non-negative) means.
`xi`	For tweedie distributions, the value of `xi` such that the variance is `var(Y) = phi * mu^xi`.
`power`	Alias for `xi`.
`min, max`	lower and upper limits of the distribution. Must be finite.

Details

When `random = FALSE`, these function return `q*(ppoints(n), ...)`.

Examples

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

x <- distribution(type = "gamma", n = 100, shape = 2)
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"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  ...
)
```

Arguments

model	A `stanreg`, `stanfit`, `brmsfit`, `blavaan`, or `MCMCglmm` 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 <- suppressWarnings(
  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 'data.frame'
equivalence_test(
  x,
  range = "default",
  ci = 0.95,
  rvar_col = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'stanreg'
equivalence_test(
  x,
  range = "default",
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
equivalence_test(
  x,
```

```
    range = "default",
    ci = 0.95,
    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 `"default"` or depending on the number of outcome variables a vector or a list. For models with one response, `range` can be:

  - a vector of length two (e.g., `c(-0.1, 0.1)`),
  - a list of numeric vector of the same length as numbers of parameters (see 'Examples').
  - a list of *named* numeric vectors, where names correspond to parameter names. In this case, all parameters that have no matching name in `range` will be set to `"default"`.

	In multivariate models, `range` should be a list with another list (one for each response variable) of numeric vectors . Vector names should correspond to the name of the response variables. If `"default"` and input is a vector, the range is set to `c(-0.1, 0.1)`. If `"default"` and input is a Bayesian model, [`rope_range()`](#) is used. See 'Examples'.
ci	The Credible Interval (CI) probability, corresponding to the proportion of HDI, to use for the percentage in ROPE.
rvar_col	A single character - the name of an `rvar` column in the data frame to be processed. See example in [`p_direction()`](#).
verbose	Toggle off warnings.
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

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. K. (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)

model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
equivalence_test(model)
# multiple ROPE ranges - asymmetric, symmetric, default
equivalence_test(model, range = list(c(10, 40), c(-5, -4), "default"))
# named ROPE ranges
equivalence_test(model, range = list(wt = c(-5, -4), `(Intercept)` = c(10, 40)))

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

equivalence_test(emmeans::emtrends(model, ~1, "wt", data = mtcars))

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

bf <- BayesFactor::ttestBF(x = rnorm(100, 1, 1))
# equivalence_test(bf)
```

---

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 and Wickham suggest that method = "KernSmooth" is the fastest and the most accurate.

Usage

```
estimate_density(x, ...)

## S3 method for class 'data.frame'
estimate_density(
  x,
  method = "kernel",
  precision = 2^10,
  extend = FALSE,
  extend_scale = 0.1,
  bw = "SJ",
  ci = NULL,
  select = NULL,
  by = NULL,
  at = NULL,
  rvar_col = NULL,
  ...
)
```

Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, methods("hdi")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric or .data.framemethods.
...	Currently not used.
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	See the eponymous argument in density. Here, the default has been changed for "SJ", which is recommended.

ci	The confidence interval threshold. Only used when method = "kernel". This feature is experimental, use with caution.
select	Character vector of column names. If NULL (the default), all numeric variables will be selected. Other arguments from datawizard::extract_column_names() (such as exclude) can also be used.
by	Optional character vector. If not NULL and input is a data frame, density estimation is performed for each group (subsets) indicated by by. See examples.
at	Deprecated in favour of by.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().

Note

There is also a plot()-method implemented in the **see**-package.

References

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

Examples

```
library(bayestestR)

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

# Basic usage
density_kernel <- estimate_density(x) # default method is "kernel"

hist(x, prob = TRUE)
lines(density_kernel$x, density_kernel$y, col = "black", lwd = 2)
lines(density_kernel$x, density_kernel$CI_low, col = "gray", lty = 2)
lines(density_kernel$x, density_kernel$CI_high, col = "gray", lty = 2)
legend("topright",
  legend = c("Estimate", "95% CI"),
  col = c("black", "gray"), lwd = 2, lty = c(1, 2)
)

# Other Methods
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)

# Multiple columns
head(estimate_density(iris))
head(estimate_density(iris, select = "Sepal.Width"))

# Grouped data
head(estimate_density(iris, by = "Species"))
head(estimate_density(iris$Petal.Width, by = iris$Species))

# rstanarm models
# -----------------------------------------------
library(rstanarm)
model <- suppressWarnings(
  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", data = mtcars)))

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

---

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 = 0.95, verbose = TRUE, ...)
```

```
## S3 method for class 'data.frame'
eti(x, ci = 0.95, rvar_col = NULL, verbose = TRUE, ...)

## S3 method for class 'stanreg'
eti(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
eti(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'get_predicted'
eti(x, ci = 0.95, use_iterations = FALSE, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, methods("hdi")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric or .data.framemethods.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .95 (95%).
verbose	Toggle off warnings.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in [p_direction()]().
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.

use_iterations   Logical, if `TRUE` and `x` is a `get_predicted` object, (returned by `insight::get_predicted()`), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., `brmsfit` 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. While this can be useful to better represent the credibility mass of a distribution, the HDI also has some limitations. See `spi()` for details.

The 95% or 89% Credible Intervals (CI) are two reasonable ranges to characterize the uncertainty related to the estimation (see here for a discussion about the differences between these two values).

The 89% intervals (`ci = 0.89`) are deemed to be more stable than, for instance, 95% intervals (*Kruschke, 2014*). An effective sample size of at least 10.000 is recommended if one wants to estimate 95% intervals with high precision (*Kruschke, 2014, p. 183ff*). Unfortunately, the default number of posterior samples for most Bayes packages (e.g., `rstanarm` or `brms`) is only 4.000 (thus, you might want to increase it when fitting your model). 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*).

However, 95% has some advantages too. For instance, it shares (in the case of a normal posterior distribution) an intuitive relationship with the standard deviation and it conveys a more accurate image of the (artificial) bounds of the distribution. Also, because it is wider, it makes analyses more conservative (i.e., the probability of covering 0 is larger for the 95% CI than for lower ranges such as 89%), which is a good thing in the context of the reproducibility crisis.

A 95% equal-tailed interval (ETI) has `2.5%` of the distribution on either side of its limits. It indicates the 2.5th percentile and the 97.5h 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.

See Also

Other ci: `bci()`, `ci()`, `hdi()`, `si()`, `spi()`

Examples

```
library(bayestestR)

posterior <- rnorm(1000)
eti(posterior)
eti(posterior, ci = c(0.80, 0.89, 0.95))

df <- data.frame(replicate(4, rnorm(100)))
eti(df)
eti(df, ci = c(0.80, 0.89, 0.95))

model <- suppressWarnings(
  rstanarm::stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
)
eti(model)
eti(model, ci = c(0.80, 0.89, 0.95))

eti(emmeans::emtrends(model, ~1, "wt", data = mtcars))

model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
eti(model)
eti(model, ci = c(0.80, 0.89, 0.95))

bf <- BayesFactor::ttestBF(x = rnorm(100, 1, 1))
eti(bf)
eti(bf, ci = c(0.80, 0.89, 0.95))
```

---

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 = 0.95, verbose = TRUE, ...)
```

```
## S3 method for class 'data.frame'
hdi(x, ci = 0.95, rvar_col = NULL, verbose = TRUE, ...)

## S3 method for class 'stanreg'
hdi(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
hdi(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'get_predicted'
hdi(x, ci = 0.95, use_iterations = FALSE, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, methods("hdi")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric or .data.framemethods.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .95 (95%).
verbose	Toggle off warnings.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().
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.

use_iterations   Logical, if `TRUE` and `x` is a get_predicted object, (returned by [insight::get_predicted()](#)), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., `brmsfit` 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. While this can be useful to better represent the credibility mass of a distribution, the HDI also has some limitations. See [spi()](#) for details.

The [95% or 89% Credible Intervals (CI)](#) are two reasonable ranges to characterize the uncertainty related to the estimation (see [here](#) for a discussion about the differences between these two values).

The 89% intervals (`ci = 0.89`) are deemed to be more stable than, for instance, 95% intervals (*Kruschke, 2014*). An effective sample size of at least 10.000 is recommended if one wants to estimate 95% intervals with high precision (*Kruschke, 2014, p. 183ff*). Unfortunately, the default number of posterior samples for most Bayes packages (e.g., `rstanarm` or `brms`) is only 4.000 (thus, you might want to increase it when fitting your model). 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*).

However, 95% has some [advantages too](#). For instance, it shares (in the case of a normal posterior distribution) an intuitive relationship with the standard deviation and it conveys a more accurate image of the (artificial) bounds of the distribution. Also, because it is wider, it makes analyses more conservative (i.e., the probability of covering 0 is larger for the 95% CI than for lower ranges such as 89%), which is a good thing in the context of the reproducibility crisis.

A 95% equal-tailed interval (ETI) has `2.5%` of the distribution on either side of its limits. It indicates the 2.5th percentile and the 97.5h 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.

Note

There is also a `plot()`-method implemented in the **see**-package.

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.

See Also

Other interval functions, such as `hdi()`, `eti()`, `bci()`, `spi()`, `si()`.

Other ci: `bci()`, `ci()`, `eti()`, `si()`, `spi()`

Examples

```
library(bayestestR)

posterior <- rnorm(1000)
hdi(posterior, ci = 0.89)
hdi(posterior, ci = c(0.80, 0.90, 0.95))

bayestestR::hdi(iris[1:4])
bayestestR::hdi(iris[1:4], ci = c(0.80, 0.90, 0.95))

model <- suppressWarnings(
  rstanarm::stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
)
bayestestR::hdi(model)
bayestestR::hdi(model, ci = c(0.80, 0.90, 0.95))

bayestestR::hdi(emmeans::emtrends(model, ~1, "wt", data = mtcars))

model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
bayestestR::hdi(model)
bayestestR::hdi(model, ci = c(0.80, 0.90, 0.95))

bf <- BayesFactor::ttestBF(x = rnorm(100, 1, 1))
bayestestR::hdi(bf)
bayestestR::hdi(bf, ci = c(0.80, 0.90, 0.95))
```

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, ...)

## 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"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  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,
  ...
)

## S3 method for class 'data.frame'
map_estimate(x, precision = 2^10, method = "kernel", rvar_col = NULL, ...)

## S3 method for class 'get_predicted'
map_estimate(
  x,
```

```
    precision = 2^10,
    method = "kernel",
    use_iterations = FALSE,
    verbose = TRUE,
    ...
)
```

Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, methods("hdi")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric or .data.framemethods.
...	Currently not used.
precision	Number of points of density data. See the n parameter in density.
method	Density estimation method. Can be "kernel" (default), "logspline" or "KernSmooth".
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.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().
use_iterations	Logical, if TRUE and x is a get_predicted object, (returned by insight::get_predicted()), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., brmsfit models).
verbose	Toggle off warnings.

Value

A numeric value if x is a vector. If x 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 = as.numeric(map_estimate(posterior)), col = "red")

model <- rstanarm::stan_glm(mpg ~ wt + cyl, data = mtcars)
map_estimate(model)

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

---

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"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  ...
)
```

Arguments

model	A stanreg, stanfit, brmsfit, blavaan, or MCMCglmm 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

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)

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

---

mediation   *Summary of Bayesian multivariate-response mediation-models*

Description

`mediation()` is a short summary for multivariate-response mediation-models, i.e. this function computes average direct and average causal mediation effects of multivariate response models.

Usage

```
mediation(model, ...)

## S3 method for class 'brmsfit'
mediation(
  model,
  treatment,
  mediator,
  response = NULL,
  centrality = "median",
  ci = 0.95,
  method = "ETI",
  ...
)

## S3 method for class 'stanmvreg'
```

```
mediation(
  model,
  treatment,
  mediator,
  response = NULL,
  centrality = "median",
  ci = 0.95,
  method = "ETI",
  ...
)
```

Arguments

`model`	A `brmsfit` or `stanmvreg` object.
`...`	Not used.
`treatment`	Character, name of the treatment variable (or direct effect) in a (multivariate response) mediator-model. If missing, `mediation()` tries to find the treatment variable automatically, however, this may fail.
`mediator`	Character, name of the mediator variable in a (multivariate response) mediator-model. If missing, `mediation()` tries to find the treatment variable automatically, however, this may fail.
`response`	A named character vector, indicating the names of the response variables to be used for the mediation analysis. Usually can be NULL, in which case these variables are retrieved automatically. If not NULL, names should match the names of the model formulas, `names(insight::find_response(model, combine = TRUE))`. This can be useful if, for instance, the mediator variable used as predictor has a different name from the mediator variable used as response. This might occur when the mediator is transformed in one model, but used "as is" as response variable in the other model. Example: The mediator `m` is used as response variable, but the centered version `m_center` is used as mediator variable. The second response variable (for the treatment model, with the mediator as additional predictor), `y`, is not transformed. Then we could use `response` like this: `mediation(model, response = c(m = "m_center", y = "y"))`.
`centrality`	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: `"median"`, `"mean"`, `"MAP"` (see `map_estimate()`), `"trimmed"` (which is just `mean(x, trim = threshold)`), `"mode"` or `"all"`.
`ci`	Value or vector of probability of the CI (between 0 and 1) to be estimated. Default to `0.95` (95%).
`method`	Can be `"ETI"` (default), `"HDI"`, `"BCI"`, `"SPI"` or `"SI"`.

Details

`mediation()` returns a data frame with information on the *direct effect* (mean value of posterior samples from `treatment` of the outcome model), *mediator effect* (mean value of posterior samples from `mediator` of the outcome model), *indirect effect* (mean value of the multiplication of the posterior samples from `mediator` of the outcome model and the posterior samples from `treatment`

of the mediation model) and the total effect (mean value of sums of posterior samples used for the direct and indirect effect). The *proportion mediated* is the indirect effect divided by the total effect.

For all values, the 89% credible intervals are calculated by default. Use ci to calculate a different interval.

The arguments treatment and mediator do not necessarily need to be specified. If missing, mediation() tries to find the treatment and mediator variable automatically. If this does not work, specify these variables.

The direct effect is also called *average direct effect* (ADE), the indirect effect is also called *average causal mediation effects* (ACME). See also *Tingley et al. 2014* and *Imai et al. 2010*.

Value

A data frame with direct, indirect, mediator and total effect of a multivariate-response mediation-model, as well as the proportion mediated. The effect sizes are median values of the posterior samples (use centrality for other centrality indices).

Note

There is an as.data.frame() method that returns the posterior samples of the effects, which can be used for further processing in the different **bayestestR** package.

References

- Imai, K., Keele, L. and Tingley, D. (2010) A General Approach to Causal Mediation Analysis, Psychological Methods, Vol. 15, No. 4 (December), pp. 309-334.
- Tingley, D., Yamamoto, T., Hirose, K., Imai, K. and Keele, L. (2014). mediation: R package for Causal Mediation Analysis, Journal of Statistical Software, Vol. 59, No. 5, pp. 1-38.

See Also

The **mediation** package for a causal mediation analysis in the frequentist framework.

Examples

```
library(mediation)
library(brms)
library(rstanarm)

# load sample data
data(jobs)
set.seed(123)

# linear models, for mediation analysis
b1 <- lm(job_seek ~ treat + econ_hard + sex + age, data = jobs)
b2 <- lm(depress2 ~ treat + job_seek + econ_hard + sex + age, data = jobs)
# mediation analysis, for comparison with Stan models
m1 <- mediate(b1, b2, sims = 1000, treat = "treat", mediator = "job_seek")

# Fit Bayesian mediation model in brms
```

```
f1 <- bf(job_seek ~ treat + econ_hard + sex + age)
f2 <- bf(depress2 ~ treat + job_seek + econ_hard + sex + age)
m2 <- brm(f1 + f2 + set_rescor(FALSE), data = jobs, refresh = 0)

# Fit Bayesian mediation model in rstanarm
m3 <- suppressWarnings(stan_mvmer(
  list(
    job_seek ~ treat + econ_hard + sex + age + (1 | occp),
    depress2 ~ treat + job_seek + econ_hard + sex + age + (1 | occp)
  ),
  data = jobs,
  refresh = 0
))

summary(m1)
mediation(m2, centrality = "mean", ci = 0.95)
mediation(m3, centrality = "mean", ci = 0.95)
```

---

model_to_priors   *Convert model's posteriors to priors (EXPERIMENTAL)*

Description

Convert model's posteriors to (normal) priors.

Usage

```
model_to_priors(model, scale_multiply = 3, ...)
```

Arguments

model	A Bayesian model.
scale_multiply	The SD of the posterior will be multiplied by this amount before being set as a prior to avoid overly narrow priors.
...	Other arguments for insight::get_prior() or [describe_posterior].

Examples

```
# brms models
# ---------------------------------------------
if (require("brms")) {
  formula <- brms::brmsformula(mpg ~ wt + cyl, center = FALSE)

  model <- brms::brm(formula, data = mtcars, refresh = 0)
  priors <- model_to_priors(model)
  priors <- brms::validate_prior(priors, formula, data = mtcars)
  priors
```

```
    model2 <- brms::brm(formula, data = mtcars, prior = priors, refresh = 0)
}
```

---

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 Probability of Direction (pd) and p-value.

Usage

```
pd_to_p(pd, ...)

## S3 method for class 'numeric'
pd_to_p(pd, direction = "two-sided", verbose = TRUE, ...)

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

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

convert_pd_to_p(pd, ...)
```

Arguments

pd	A Probability of Direction (pd) value (between 0 and 1). Can also be a data frame with a column named pd, p_direction, or PD, as returned by `p_direction()`. In this case, the column is converted to p-values and the new data frame is returned.
...	Arguments passed to or from other methods.
direction	What type of p-value is requested or provided. Can be "two-sided" (default, two tailed) or "one-sided" (one tailed).
verbose	Toggle off warnings.
p	A p-value.

Details

Conversion is done using the following equation (see *Makowski et al., 2019*):

When direction = "two-sided"

$$p = 2 \times (1 - p_d)$$

When direction = "one-sided"

$$p = 1 - p_d$$

Note that this conversion is only valid when the lowest possible values of pd is 0.5 - i.e., when the posterior represents continuous parameter space (see `p_direction()`). If any pd < 0.5 are detected, they are converted to a p of 1, and a warning is given.

Value

A p-value or a data frame with a p-value column.

References

Makowski, D., Ben-Shachar, M. S., Chen, S. H. A., and Lüdecke, D. (2019). *Indices of Effect Existence and Significance in the Bayesian Framework*. Frontiers in Psychology 2019;10:2767. doi:10.3389/fpsyg.2019.02767

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, ...)

## S3 method for class 'numeric'
point_estimate(x, centrality = "all", dispersion = FALSE, threshold = 0.1, ...)

## S3 method for class 'data.frame'
point_estimate(
  x,
  centrality = "all",
  dispersion = FALSE,
  threshold = 0.1,
  rvar_col = NULL,
  ...
)

## S3 method for class 'stanreg'
point_estimate(
  x,
  centrality = "all",
  dispersion = FALSE,
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
```

```
      "distributional", "auxiliary"),
    parameters = NULL,
    ...
)

## S3 method for class 'brmsfit'
point_estimate(
    x,
    centrality = "all",
    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 = "all", dispersion = FALSE, ...)

## S3 method for class 'get_predicted'
point_estimate(
    x,
    centrality = "all",
    dispersion = FALSE,
    use_iterations = FALSE,
    verbose = TRUE,
    ...
)
```

Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, methods("hdi")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric or .data.framemethods.
...	Additional arguments to be passed to or from methods.
centrality	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: "median", "mean", "MAP" (see map_estimate()), "trimmed" (which is just mean(x, trim = threshold)), "mode" or "all".
dispersion	Logical, if TRUE, computes indices of dispersion related to the estimate(s) (SD and MAD for mean and median, respectively). Dispersion is not available for "MAP" or "mode" centrality indices.
threshold	For centrality = "trimmed" (i.e. trimmed mean), indicates the fraction (0 to 0.5) of observations to be trimmed from each end of the vector before the mean is computed.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().

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.
use_iterations	Logical, if `TRUE` and `x` is a `get_predicted` object, (returned by `insight::get_predicted()`), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., `brmsfit` models).
verbose	Toggle off warnings.

Note

There is also a `plot()`-method implemented in the **see**-package.

References

Makowski, D., Ben-Shachar, M. S., Chen, S. H. A., and Lüdecke, D. (2019). *Indices of Effect Existence and Significance in the Bayesian Framework*. Frontiers in Psychology 2019;10:2767. doi:10.3389/fpsyg.2019.02767

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"))

# rstanarm models
# -----------------------------------------------
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
# -----------------------------------------------
point_estimate(
  emmeans::emtrends(model, ~1, "wt", data = mtcars),
  centrality = c("median", "MAP")
)
```

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

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

---

p_direction   *Probability of Direction (pd)*

Description

Compute the **Probability of Direction** (**pd**, also known as the Maximum Probability of Effect - *MPE*). This can be interpreted as the probability that a parameter (described by its posterior distribution) is strictly positive or negative (whichever is the most probable). Although differently expressed, this index is fairly similar (*i.e.*, is strongly correlated) to the frequentist **p-value** (see details).

Usage

```
p_direction(x, ...)

pd(x, ...)

## S3 method for class 'numeric'
p_direction(
  x,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  ...
)

## S3 method for class 'data.frame'
p_direction(
  x,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
```

```
  rvar_col = NULL,
  ...
)

## S3 method for class 'MCMCglmm'
p_direction(
  x,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  ...
)

## S3 method for class 'emmGrid'
p_direction(
  x,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  ...
)

## S3 method for class 'slopes'
p_direction(
  x,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  ...
)

## S3 method for class 'stanreg'
p_direction(
  x,
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  ...
)
```

```
## S3 method for class 'brmsfit'
p_direction(
  x,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  ...
)

## S3 method for class 'BFBayesFactor'
p_direction(
  x,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  ...
)

## S3 method for class 'get_predicted'
p_direction(
  x,
  method = "direct",
  null = 0,
  as_p = FALSE,
  remove_na = TRUE,
  use_iterations = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	A vector representing a posterior distribution, a data frame of posterior draws (samples be parameter). Can also be a Bayesian model.
...	Currently not used.
method	Can be "direct" or one of methods of [estimate_density()](), such as "kernel", "logspline" or "KernSmooth". See details.
null	The value considered as a "null" effect. Traditionally 0, but could also be 1 in the case of ratios of change (OR, IRR, ...).
as_p	If TRUE, the p-direction (pd) values are converted to a frequentist p-value using [pd_to_p()]().
remove_na	Should missing values be removed before computation? Note that Inf (infinity) are *not* removed.

rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in `p_direction()`.
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.
use_iterations	Logical, if `TRUE` and `x` is a `get_predicted` object, (returned by `insight::get_predicted()`), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., `brmsfit` models).
verbose	Toggle off warnings.

Value

Values between 0.5 and 1 *or* between 0 and 1 (see above) corresponding to the probability of direction (pd).

What is the *pd*?

The Probability of Direction (pd) is an index of effect existence, representing the certainty with which an effect goes in a particular direction (i.e., is positive or negative / has a sign), typically ranging from 0.5 to 1 (but see next section for cases where it can range between 0 and 1). Beyond its simplicity of interpretation, understanding and computation, this index also presents other interesting properties:

- Like other posterior-based indices, *pd* is solely based on the posterior distributions and does not require any additional information from the data or the model (e.g., such as priors, as in the case of Bayes factors).

- 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 (Makowski et al., 2019).

Relationship with the p-value

In most cases, it seems that the *pd* has a direct correspondence with the frequentist one-sided *p*-value through the formula (for two-sided *p*): $p = 2 \times (1 - p_d)$ 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%. See `pd_to_p()` for details.

Possible Range of Values

The largest value *pd* can take is 1 - the posterior is strictly directional. However, the smallest value *pd* can take depends on the parameter space represented by the posterior.

For a continuous parameter space, exact values of 0 (or any point null value) are not possible, and so 100% of the posterior has *some* sign, some positive, some negative. Therefore, the smallest the *pd* can be is 0.5 - with an equal posterior mass of positive and negative values. Values close to 0.5 *cannot* be used to support the null hypothesis (that the parameter does *not* have a direction) is a similar why to how large p-values cannot be used to support the null hypothesis (see `pd_to_p()`; Makowski et al., 2019).

For a discrete parameter space or a parameter space that is a mixture between discrete and continuous spaces, exact values of 0 (or any point null value) *are* possible! Therefore, the smallest the *pd* can be is 0 - with 100% of the posterior mass on 0. Thus values close to 0 can be used to support the null hypothesis (see van den Bergh et al., 2021).

Examples of posteriors representing discrete parameter space:

- When a parameter can only take discrete values.
- When a mixture prior/posterior is used (such as the spike-and-slab prior; see van den Bergh et al., 2021).
- When conducting Bayesian model averaging (e.g., `weighted_posteriors()` or `brms::posterior_average`).

Methods of computation

The *pd* is defined as:

$$p_d = max(Pr(\hat{\theta} < \theta_{null}), Pr(\hat{\theta} > \theta_{null}))$$

The most simple and direct way to compute the *pd* is to compute the proportion of positive (or larger than `null`) posterior samples, the proportion of negative (or smaller than `null`) posterior samples, and take the larger of the two. This "simple" method is the most straightforward, but its precision is directly tied to the number of posterior draws.

The second approach relies on density estimation: It starts by estimating the continuous-smooth density function (for which many methods are available), and then computing the area under the curve (AUC) of the density curve on either side of `null` and taking the maximum between them. Note the this approach assumes a continuous density function, and so **when the posterior represents a (partially) discrete parameter space, only the direct method *must* be used** (see above).

Note

There is also a `plot()`-method implemented in the **see**-package.

References

- Makowski, D., Ben-Shachar, M. S., Chen, S. A., & Lüdecke, D. (2019). Indices of effect existence and significance in the Bayesian framework. Frontiers in psychology, 10, 2767. doi:10.3389/fpsyg.2019.02767
- van den Bergh, D., Haaf, J. M., Ly, A., Rouder, J. N., & Wagenmakers, E. J. (2021). A cautionary note on estimating effect size. Advances in Methods and Practices in Psychological Science, 4(1). doi:10.1177/2515245921992035

See Also

`pd_to_p()` to convert between Probability of Direction (pd) and p-value.

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
# ----------------------------------------------
model <- rstanarm::stan_glm(mpg ~ wt + cyl,
  data = mtcars,
  chains = 2, refresh = 0
)
p_direction(model)
p_direction(model, method = "kernel")

# emmeans
# ----------------------------------------------
p_direction(emmeans::emtrends(model, ~1, "wt", data = mtcars))

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

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


# Using "rvar_col"
x <- data.frame(mu = c(0, 0.5, 1), sigma = c(1, 0.5, 0.25))
x$my_rvar <- posterior::rvar_rng(rnorm, 3, mean = x$mu, sd = x$sigma)
x
p_direction(x, rvar_col = "my_rvar")
```

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 the null (e.g., 0) divided by the density at the Maximum A Posteriori (MAP).

Usage

```
p_map(x, ...)

p_pointnull(x, ...)

## S3 method for class 'numeric'
p_map(x, null = 0, precision = 2^10, method = "kernel", ...)

## S3 method for class 'get_predicted'
p_map(
  x,
  null = 0,
  precision = 2^10,
  method = "kernel",
  use_iterations = FALSE,
  verbose = TRUE,
  ...
)

## S3 method for class 'data.frame'
p_map(x, null = 0, precision = 2^10, method = "kernel", rvar_col = NULL, ...)

## S3 method for class 'stanreg'
p_map(
  x,
  null = 0,
  precision = 2^10,
  method = "kernel",
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  ...
)
```

```
## S3 method for class 'brmsfit'
p_map(
  x,
  null = 0,
  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, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, methods("hdi")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric or .data.framemethods.
...	Currently not used.
null	The value considered as a "null" effect. Traditionally 0, but could also be 1 in the case of ratios of change (OR, IRR, ...).
precision	Number of points of density data. See the n parameter in density.
method	Density estimation method. Can be "kernel" (default), "logspline" or "KernSmooth".
use_iterations	Logical, if TRUE and x is a get_predicted object, (returned by insight::get_predicted()), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., brmsfit models).
verbose	Toggle off warnings.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().
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

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

Strengths and Limitations:

Strengths: Straightforward computation. Objective property of the posterior distribution.

Limitations: Limited information favoring the null hypothesis. Relates on density approximation. Indirect relationship between mathematical definition and interpretation. Only suitable for weak / very diffused priors.

References

- Makowski D, Ben-Shachar MS, Chen SHA, Lüdecke D (2019) Indices of Effect Existence and Significance in the Bayesian Framework. Frontiers in Psychology 2019;10:2767. doi:10.3389/fpsyg.2019.02767

- 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))

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

p_map(suppressWarnings(
  emmeans::emtrends(model, ~1, "wt", data = mtcars)
))

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

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

# --------------------------------------
# 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 = as.numeric(p_map(x, method = "kernel")),
    KernSmooth = as.numeric(p_map(x, method = "KernSmooth")),
    logspline = as.numeric(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   *Probability of being in the ROPE*

Description

Compute the proportion of the whole posterior distribution that doesn't lie within a region of practical equivalence (ROPE). It is equivalent to running rope(..., ci = 1).

Usage

```
p_rope(x, ...)

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

## S3 method for class 'data.frame'
p_rope(x, range = "default", rvar_col = NULL, verbose = TRUE, ...)

## S3 method for class 'stanreg'
p_rope(
  x,
  range = "default",
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
p_rope(
  x,
  range = "default",
  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 `"default"` or depending on the number of outcome variables a vector or a list. For models with one response, `range` can be: • a vector of length two (e.g., `c(-0.1, 0.1)`), • a list of numeric vector of the same length as numbers of parameters (see 'Examples'). • a list of *named* numeric vectors, where names correspond to parameter names. In this case, all parameters that have no matching name in `range` will be set to `"default"`. In multivariate models, `range` should be a list with another list (one for each response variable) of numeric vectors . Vector names should correspond to the name of the response variables. If `"default"` and input is a vector, the range is set to `c(-0.1, 0.1)`. If `"default"` and input is a Bayesian model, `rope_range()` is used. See 'Examples'.
verbose	Toggle off warnings.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in `p_direction()`.
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.

Examples

```
library(bayestestR)

p_rope(x = rnorm(1000, 0, 0.01), range = c(-0.1, 0.1))
p_rope(x = mtcars, range = c(-0.1, 0.1))
```

---

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 'get_predicted'
p_significance(
  x,
  threshold = "default",
  use_iterations = FALSE,
  verbose = TRUE,
  ...
)

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

## S3 method for class 'stanreg'
p_significance(
  x,
  threshold = "default",
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
p_significance(
  x,
  threshold = "default",
  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.
threshold	The threshold value that separates significant from negligible effect, which can have following possible values:

- "default", in which case the range is set to 0.1 if input is a vector, and based on rope_range() if a (Bayesian) model is provided.
- a single numeric value (e.g., 0.1), which is used as range around zero (i.e. the threshold range is set to -0.1 and 0.1, i.e. reflects a symmetric interval)
- a numeric vector of length two (e.g., c(-0.2, 0.1)), useful for asymmetric intervals
- a list of numeric vectors, where each vector corresponds to a parameter
- a list of *named* numeric vectors, where names correspond to parameter names. In this case, all parameters that have no matching name in threshold will be set to "default".

use_iterations	Logical, if TRUE and x is a get_predicted object, (returned by insight::get_predicted()), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., brmsfit models).
verbose	Toggle off warnings.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in p_direction().
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

p_significance() returns the proportion of a probability distribution (x) that is outside a certain range (the negligible effect, or ROPE, see argument threshold). If there are values of the distribution both below and above the ROPE, p_significance() returns the higher probability of a value being outside the ROPE. Typically, this value should be larger than 0.5 to indicate practical significance. However, if the range of the negligible effect is rather large compared to the range of the probability distribution x, p_significance() will be less than 0.5, which indicates no clear practical significance.

Value

Values between 0 and 1 corresponding to the probability of practical significance (ps).

Note

There is also a `plot()`-method implemented in the **see**-package.

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
# ---------------------------------------------
model <- rstanarm::stan_glm(mpg ~ wt + cyl,
  data = mtcars,
  chains = 2, refresh = 0
)
p_significance(model)
# multiple thresholds - asymmetric, symmetric, default
p_significance(model, threshold = list(c(-10, 5), 0.2, "default"))
# named thresholds
p_significance(model, threshold = list(wt = 0.2, `(Intercept)` = c(-10, 5)))
```

---

p_to_bf   *Convert p-values to (pseudo) Bayes Factors*

Description

Convert p-values to (pseudo) Bayes Factors. This transformation has been suggested by Wagenmakers (2022), but is based on a vast amount of assumptions. It might therefore be not reliable. Use at your own risks. For more accurate approximate Bayes factors, use `bic_to_bf()` instead.

Usage

```
p_to_bf(x, ...)

## S3 method for class 'numeric'
```

```
p_to_bf(x, log = FALSE, n_obs = NULL, ...)

## Default S3 method:
p_to_bf(x, log = FALSE, ...)
```

Arguments

x	A (frequentist) model object, or a (numeric) vector of p-values.
...	Other arguments to be passed (not used for now).
log	Wether to return log Bayes Factors. **Note:** The `print()` method always shows BF - the `"log_BF"` column is only accessible from the returned data frame.
n_obs	Number of observations. Either length 1, or same length as p.

Value

A data frame with the p-values and pseudo-Bayes factors (against the null).

References

- Wagenmakers, E.J. (2022). Approximate objective Bayes factors from p-values and sample size: The 3p(sqrt(n)) rule. Preprint available on ArXiv: https://psyarxiv.com/egydq

See Also

[bic_to_bf()](bic_to_bf()) for more accurate approximate Bayes factors.

Examples

```
data(iris)
model <- lm(Petal.Length ~ Sepal.Length + Species, data = iris)
p_to_bf(model)

# Examples that demonstrate comparison between
# BIC-approximated and pseudo BF
# -----------------------------------------
m0 <- lm(mpg ~ 1, mtcars)
m1 <- lm(mpg ~ am, mtcars)
m2 <- lm(mpg ~ factor(cyl), mtcars)

# In this first example, BIC-approximated BF and
# pseudo-BF based on p-values are close...

# BIC-approximated BF, m1 against null model
bic_to_bf(BIC(m1), denominator = BIC(m0))

# pseudo-BF based on p-values - dropping intercept
p_to_bf(m1)[-1, ]

# The second example shows that results from pseudo-BF are less accurate
# and should be handled wit caution!
bic_to_bf(BIC(m2), denominator = BIC(m0))
```

```
p_to_bf(anova(m2), n_obs = nrow(mtcars))
```

---

reshape_iterations   *Reshape estimations with multiple iterations (draws) to long format*

Description

Reshape a wide data.frame of iterations (such as posterior draws or bootsrapped samples) as columns to long format. Instead of having all iterations as columns (e.g., iter_1, iter_2, ...), will return 3 columns with the *_index (the previous index of the row), the *_group (the iteration number) and the *_value (the value of said iteration).

Usage

```
reshape_iterations(x, prefix = c("draw", "iter", "iteration", "sim"))

reshape_draws(x, prefix = c("draw", "iter", "iteration", "sim"))
```

Arguments

x	A data.frame containing posterior draws obtained from estimate_response or estimate_link.
prefix	The prefix of the draws (for instance, "iter_" for columns named as iter_1, iter_2, iter_3). If more than one are provided, will search for the first one that matches.

Value

Data frame of reshaped draws in long format.

Examples

```
if (require("rstanarm")) {
  model <- stan_glm(mpg ~ am, data = mtcars, refresh = 0)
  draws <- insight::get_predicted(model)
  long_format <- reshape_iterations(draws)
  head(long_format)
}
```

rope   *Region of Practical Equivalence (ROPE)*

Description

Compute the proportion of the HDI (default to the 89% HDI) of a posterior distribution that lies within a region of practical equivalence.

Usage

```
rope(x, ...)

## S3 method for class 'numeric'
rope(x, range = "default", ci = 0.95, ci_method = "ETI", verbose = TRUE, ...)

## S3 method for class 'data.frame'
rope(
  x,
  range = "default",
  ci = 0.95,
  ci_method = "ETI",
  rvar_col = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'stanreg'
rope(
  x,
  range = "default",
  ci = 0.95,
  ci_method = "ETI",
  effects = c("fixed", "random", "all"),
  component = c("location", "all", "conditional", "smooth_terms", "sigma",
    "distributional", "auxiliary"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'brmsfit'
rope(
  x,
  range = "default",
  ci = 0.95,
  ci_method = "ETI",
  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 `"default"` or depending on the number of outcome variables a vector or a list. For models with one response, `range` can be: • a vector of length two (e.g., `c(-0.1, 0.1)`), • a list of numeric vector of the same length as numbers of parameters (see 'Examples'). • a list of *named* numeric vectors, where names correspond to parameter names. In this case, all parameters that have no matching name in `range` will be set to `"default"`. In multivariate models, `range` should be a list with another list (one for each response variable) of numeric vectors . Vector names should correspond to the name of the response variables. If `"default"` and input is a vector, the range is set to `c(-0.1, 0.1)`. If `"default"` and input is a Bayesian model, [rope_range()](#) is used. See 'Examples'.
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.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in [p_direction()](#).
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.

ROPE

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*).

Strengths and Limitations

Strengths: Provides information related to the practical relevance of the effects.

Limitations: A ROPE range needs to be arbitrarily defined. Sensitive to the scale (the unit) of the predictors. Not sensitive to highly significant effects.

Note

There is also a `plot()`-method implemented in the **see**-package.

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.

- Makowski D, Ben-Shachar MS, Chen SHA, Lüdecke D (2019) Indices of Effect Existence and Significance in the Bayesian Framework. Frontiers in Psychology 2019;10:2767. doi:10.3389/fpsyg.2019.02767

- 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(0.90, 0.95))

library(rstanarm)
model <- suppressWarnings(
  stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
)
rope(model)
rope(model, ci = c(0.90, 0.95))

# multiple ROPE ranges
rope(model, range = list(c(-10, 5), c(-0.2, 0.2), "default"))

# named ROPE ranges
rope(model, range = list(gear = c(-3, 2), wt = c(-0.2, 0.2)))

library(emmeans)
rope(emtrends(model, ~1, "wt"), ci = c(0.90, 0.95))

library(brms)
model <- brm(mpg ~ wt + cyl, data = mtcars, refresh = 0)
rope(model)
rope(model, ci = c(0.90, 0.95))

library(brms)
model <- brm(
  bf(mvbind(mpg, disp) ~ wt + cyl) + set_rescor(rescor = TRUE),
  data = mtcars,
  refresh = 0
)
```

```
rope(model)
rope(model, ci = c(0.90, 0.95))

# different ROPE ranges for model parameters. For each response, a named
# list (with the name of the response variable) is required as list-element
# for the `range` argument.
rope(
  model,
  range = list(
    mpg = list(b_mpg_wt = c(-1, 1), b_mpg_cyl = c(-2, 2)),
    disp = list(b_disp_wt = c(-5, 5), b_disp_cyl = c(-4, 4))
  )
)

library(BayesFactor)
bf <- ttestBF(x = rnorm(100, 1, 1))
rope(bf)
rope(bf, ci = c(0.90, 0.95))
```

---

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, ...)

## Default S3 method:
rope_range(x, verbose = TRUE, ...)
```

Arguments

x	A `stanreg`, `brmsfit` or `BFBayesFactor` object, or a frequentist regression model.
...	Currently not used.
verbose	Toggle warnings.

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 models from **count data**, the residual variance is used. This is a rather experimental threshold and is probably often similar to `-0.1, 0.1`, but should be used with care!

- 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

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

model <- suppressWarnings(
  rstanarm::stan_glm(vs ~ mpg, data = mtcars, family = "binomial", refresh = 0)
)
rope_range(model)

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

model <- BayesFactor::ttestBF(mtcars[mtcars$vs == 1, "mpg"], mtcars[mtcars$vs == 0, "mpg"])
rope_range(model)

model <- lmBF(mpg ~ vs, data = mtcars)
rope_range(model)
```

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, ...)

## S3 method for class 'stanreg'
sensitivity_to_prior(model, index = "Median", magnitude = 10, ...)
```

Arguments

model	A Bayesian model (stanreg or brmsfit).
...	Arguments passed to or from other methods.
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.

See Also

DescTools

Examples

```
library(bayestestR)

# rstanarm models
# -----------------------------------------------
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"))
```

sexit   *Sequential Effect eXistence and sIgnificance Testing (SEXIT)*

Description

The SEXIT is a new framework to describe Bayesian effects, guiding which indices to use. Accordingly, the sexit() function returns the minimal (and optimal) required information to describe models' parameters under a Bayesian framework. It includes the following indices:

- Centrality: the median of the posterior distribution. In probabilistic terms, there is 50% of probability that the effect is higher and lower. See `point_estimate()`.

- Uncertainty: the 95% Highest Density Interval (HDI). In probabilistic terms, there is 95% of probability that the effect is within this confidence interval. See `ci()`.

- Existence: The probability of direction allows to quantify the certainty by which an effect is positive or negative. It is a critical index to show that an effect of some manipulation is not harmful (for instance in clinical studies) or to assess the direction of a link. See `p_direction()`.

- Significance: Once existence is demonstrated with high certainty, we can assess whether the effect is of sufficient size to be considered as significant (i.e., not negligible). This is a useful index to determine which effects are actually important and worthy of discussion in a given process. See `p_significance()`.

- Size: Finally, this index gives an idea about the strength of an effect. However, beware, as studies have shown that a big effect size can be also suggestive of low statistical power (see details section).

Usage

```
sexit(x, significant = "default", large = "default", ci = 0.95, ...)
```

Arguments

x   A vector representing a posterior distribution, a data frame of posterior draws (samples be parameter). Can also be a Bayesian model.

significant, large
                     The threshold values to use for significant and large probabilities. If left to 'default', will be selected through `sexit_thresholds()`. See the details section below.

ci   Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .95 (95%).

...   Currently not used.

Details

Rationale: The assessment of "significance" (in its broadest meaning) is a pervasive issue in science, and its historical index, the p-value, has been strongly criticized and deemed to have played an important role in the replicability crisis. In reaction, more and more scientists have tuned to Bayesian methods, offering an alternative set of tools to answer their questions. However, the Bayesian framework offers a wide variety of possible indices related to "significance", and the debate has been raging about which index is the best, and which one to report.

This situation can lead to the mindless reporting of all possible indices (with the hopes that with that the reader will be satisfied), but often without having the writer understanding and interpreting them. It is indeed complicated to juggle between many indices with complicated definitions and subtle differences.

SEXIT aims at offering a practical framework for Bayesian effects reporting, in which the focus is put on intuitiveness, explicitness and usefulness of the indices' interpretation. To that end, we suggest a system of description of parameters that would be intuitive, easy to learn and apply, mathematically accurate and useful for taking decision.

Once the thresholds for significance (i.e., the ROPE) and the one for a "large" effect are explicitly defined, the SEXIT framework does not make any interpretation, i.e., it does not label the effects, but just sequentially gives 3 probabilities (of direction, of significance and of being large, respectively) as-is on top of the characteristics of the posterior (using the median and HDI for centrality and uncertainty description). Thus, it provides a lot of information about the posterior distribution (through the mass of different 'sections' of the posterior) in a clear and meaningful way.

Threshold selection: One of the most important thing about the SEXIT framework is that it relies on two "arbitrary" thresholds (i.e., that have no absolute meaning). They are the ones related to effect size (an inherently subjective notion), namely the thresholds for significant and large effects. They are set, by default, to `0.05` and `0.3` of the standard deviation of the outcome variable (tiny and large effect sizes for correlations according to Funder and Ozer, 2019). However, these defaults were chosen by lack of a better option, and might not be adapted to your case. Thus, they are to be handled with care, and the chosen thresholds should always be explicitly reported and justified.

- For **linear models (lm)**, this can be generalised to $[0.05 * SD_y]$ and $[0.3 * SD_y]$ for significant and large effects, respectively.
- For **logistic models**, the parameters expressed in log odds ratio can be converted to standardized difference through the formula $\pi/\sqrt{3}$, resulting a threshold of `0.09` and `0.54`.
- 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 models from **count data**, the residual variance is used. This is a rather experimental threshold and is probably often similar to `0.05` and `0.3`, but should be used with care!
- For **t-tests**, the standard deviation of the response is used, similarly to linear models (see above).
- For **correlations**, `0.05` and `0.3` are used.
- For all other models, `0.05` and `0.3` are used, but it is strongly advised to specify it manually.

Examples: The three values for existence, significance and size provide a useful description of the posterior distribution of the effects. Some possible scenarios include:

- The probability of existence is low, but the probability of being large is high: it suggests that the posterior is very wide (covering large territories on both side of 0). The statistical power might be too low, which should warrant any confident conclusion.

- The probability of existence and significance is high, but the probability of being large is very small: it suggests that the effect is, with high confidence, not large (the posterior is mostly contained between the significance and the large thresholds).
- The 3 indices are very low: this suggests that the effect is null with high confidence (the posterior is closely centred around 0).

Value

A dataframe and text as attribute.

References

- Makowski, D., Ben-Shachar, M. S., & Lüdecke, D. (2019). bayestestR: Describing Effects and their Uncertainty, Existence and Significance within the Bayesian Framework. Journal of Open Source Software, 4(40), 1541. doi:10.21105/joss.01541
- Makowski D, Ben-Shachar MS, Chen SHA, Lüdecke D (2019) Indices of Effect Existence and Significance in the Bayesian Framework. Frontiers in Psychology 2019;10:2767. doi:10.3389/fpsyg.2019.02767

Examples

```
library(bayestestR)

s <- sexit(rnorm(1000, -1, 1))
s
print(s, summary = TRUE)

s <- sexit(iris)
s
print(s, summary = TRUE)

if (require("rstanarm")) {
  model <- suppressWarnings(rstanarm::stan_glm(mpg ~ wt * cyl,
    data = mtcars,
    iter = 400, refresh = 0
  ))
  s <- sexit(model)
  s
  print(s, summary = TRUE)
}
```

---

sexit_thresholds   *Find Effect Size Thresholds*

Description

This function attempts at automatically finding suitable default values for a "significant" (i.e., non-negligible) and "large" effect. This is to be used with care, and the chosen threshold should always be explicitly reported and justified. See the detail section in `sexit()` for more information.

Usage

```
sexit_thresholds(x, ...)
```

Arguments

x	Vector representing a posterior distribution. Can also be a `stanreg` or `brmsfit` model.
...	Currently not used.

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

```
sexit_thresholds(rnorm(1000))

if (require("rstanarm")) {
  model <- suppressWarnings(stan_glm(
    mpg ~ wt + gear,
    data = mtcars,
    chains = 2,
    iter = 200,
    refresh = 0
  ))
  sexit_thresholds(model)

  model <- suppressWarnings(
    stan_glm(vs ~ mpg, data = mtcars, family = "binomial", refresh = 0)
  )
  sexit_thresholds(model)
}

if (require("brms")) {
  model <- brm(mpg ~ wt + cyl, data = mtcars)
  sexit_thresholds(model)
}

if (require("BayesFactor")) {
  bf <- ttestBF(x = rnorm(100, 1, 1))
  sexit_thresholds(bf)
}
```

---

si   *Compute Support Intervals*

Description

A support interval contains only the values of the parameter that predict the observed data better than average, by some degree *k*; these are values of the parameter that are associated with an updating factor greater or equal than *k*. From the perspective of the Savage-Dickey Bayes factor, testing against a point null hypothesis for any value within the support interval will yield a Bayes factor smaller than *1/k*.

Usage

```
si(posterior, ...)

## S3 method for class 'numeric'
si(posterior, prior = NULL, BF = 1, verbose = TRUE, ...)

## S3 method for class 'stanreg'
si(
  posterior,
  prior = NULL,
  BF = 1,
  verbose = TRUE,
  effects = c("fixed", "random", "all"),
 component = c("location", "conditional", "all", "smooth_terms", "sigma", "auxiliary",
    "distributional"),
  parameters = NULL,
  ...
)

## S3 method for class 'brmsfit'
si(
  posterior,
  prior = NULL,
  BF = 1,
  verbose = TRUE,
  effects = c("fixed", "random", "all"),
 component = c("location", "conditional", "all", "smooth_terms", "sigma", "auxiliary",
    "distributional"),
  parameters = NULL,
  ...
)

## S3 method for class 'blavaan'
si(
  posterior,
```

```
  prior = NULL,
  BF = 1,
  verbose = TRUE,
  effects = c("fixed", "random", "all"),
 component = c("location", "conditional", "all", "smooth_terms", "sigma", "auxiliary",
    "distributional"),
  parameters = NULL,
  ...
)

## S3 method for class 'emmGrid'
si(posterior, prior = NULL, BF = 1, verbose = TRUE, ...)

## S3 method for class 'get_predicted'
si(
  posterior,
  prior = NULL,
  BF = 1,
  use_iterations = FALSE,
  verbose = TRUE,
  ...
)

## S3 method for class 'data.frame'
si(posterior, prior = NULL, BF = 1, rvar_col = NULL, verbose = TRUE, ...)
```

Arguments

posterior	A numerical vector, stanreg / brmsfit object, emmGrid or a data frame - representing a posterior distribution(s) from (see 'Details').
...	Arguments passed to and from other methods. (Can be used to pass arguments to internal logspline::logspline().)
prior	An object representing a prior distribution (see 'Details').
BF	The amount of support required to be included in the support interval.
verbose	Toggle off warnings.
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.
use_iterations	Logical, if TRUE and x is a get_predicted object, (returned by insight::get_predicted()), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., brmsfit models).

rvar_col	A single character - the name of an rvar column in the data frame to be pro-
	cessed. See example in p_direction().

Details

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

This method is used to compute support intervals based on prior and posterior distributions. For the computation of support intervals, the model priors must be proper priors (at the very least they should be *not flat*, and it is preferable that they be *informative* - note that by default, brms::brm() uses flat priors for fixed-effects; see example below).

Value

A data frame containing the lower and upper bounds of the SI.

Note that if the level of requested support is higher than observed in the data, the interval will be [NA,NA].

Choosing a value of BF

The choice of BF (the level of support) depends on what we want our interval to represent:

- A BF = 1 contains values whose credibility is not decreased by observing the data.
- A BF > 1 contains values who received more impressive support from the data.
- A BF < 1 contains values whose credibility has *not* been impressively decreased by observing the data. Testing against values outside this interval will produce a Bayes factor larger than 1/BF in support of the alternative. E.g., if an SI (BF = 1/3) excludes 0, the Bayes factor against the point-null will be larger than 3.

Setting the correct prior

For the computation of Bayes factors, the model priors must be proper priors (at the very least they should be *not flat*, and it is preferable that they be *informative*); As the priors for the alternative get wider, the likelihood of the null value(s) increases, to the extreme that for completely flat priors the null is infinitely more favorable than the alternative (this is called *the Jeffreys-Lindley-Bartlett paradox*). Thus, you should only ever try (or want) to compute a Bayes factor when you have an informed prior.

(Note that by default, brms::brm() uses flat priors for fixed-effects; See example below.)

It is important to provide the correct prior for meaningful results, to match the posterior-type input:

- **A numeric vector** - prior should also be a *numeric vector*, representing the prior-estimate.
- **A data frame** - prior should also be a *data frame*, representing the prior-estimates, in match-ing column order.
  - If rvar_col is specified, prior should be *the name of an* rvar *column* that represents the prior-estimates.

- **Supported Bayesian model** (stanreg, brmsfit**, etc.**)
  - prior should be *a model an equivalent model with MCMC samples from the priors* **only**. See unupdate().
  - If prior is set to NULL, unupdate() is called internally (not supported for brmsfit_multiple model).
- **Output from a** {marginaleffects} **function** - prior should also be *an equivalent output* from a {marginaleffects} function based on a prior-model (See unupdate()).
- **Output from an** {emmeans} **function**
  - prior should also be *an equivalent output* from an {emmeans} function based on a prior-model (See unupdate()).
  - prior can also be *the original (posterior) model*, in which case the function will try to "unupdate" the estimates (not supported if the estimates have undergone any transformations – "log", "response", etc. – or any regriding).

Note

There is also a plot()-method implemented in the **see**-package.

References

Wagenmakers, E., Gronau, Q. F., Dablander, F., & Etz, A. (2018, November 22). The Support Interval. doi:10.31234/osf.io/zwnxb

See Also

Other ci: bci(), ci(), eti(), hdi(), spi()

Examples

```
library(bayestestR)

prior <- distribution_normal(1000, mean = 0, sd = 1)
posterior <- distribution_normal(1000, mean = 0.5, sd = 0.3)

si(posterior, prior, verbose = FALSE)

# rstanarm models
# ---------------
library(rstanarm)
contrasts(sleep$group) <- contr.equalprior_pairs # see vignette
stan_model <- stan_lmer(extra ~ group + (1 | ID), data = sleep)
si(stan_model, verbose = FALSE)
si(stan_model, BF = 3, verbose = FALSE)

# emmGrid objects
# ---------------
library(emmeans)
group_diff <- pairs(emmeans(stan_model, ~group))
si(group_diff, prior = stan_model, verbose = FALSE)
```

```
# brms models
# -----------
library(brms)
contrasts(sleep$group) <- contr.equalprior_pairs # 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 <- suppressWarnings(brm(extra ~ group + (1 | ID),
  data = sleep,
  prior = my_custom_priors,
  refresh = 0
))
si(brms_model, verbose = FALSE)
```

---

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, ...)

simulate_difference(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"))

# Difference -------------------------------
data <- simulate_difference(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`, `brmsfit`, `blavaan`, or `MCMCglmm` object.
n	Size of the simulated prior distributions.
...	Currently not used.

See Also

`unupdate()` for directly sampling from the prior distribution (useful for complex priors and designs).

Examples

```
library(bayestestR)
if (require("rstanarm")) {
  model <- suppressWarnings(
    stan_glm(mpg ~ wt + am, data = mtcars, chains = 1, refresh = 0)
  )
  simulate_prior(model)
}
```

---

simulate_simpson   *Simpson's paradox dataset simulation*

Description

Simpson's paradox, or the Yule-Simpson effect, is a phenomenon in probability and statistics, in which a trend appears in several different groups of data but disappears or reverses when these groups are combined.

Usage

```
simulate_simpson(
  n = 100,
  r = 0.5,
  groups = 3,
  difference = 1,
  group_prefix = "G_"
)
```

Arguments

n	The number of observations for each group to be generated (minimum 4).
r	A value or vector corresponding to the desired correlation coefficients.
groups	Number of groups (groups can be participants, clusters, anything).
difference	Difference between groups.
group_prefix	The prefix of the group name (e.g., "G_1", "G_2", "G_3", ...).

Value

A dataset.

Examples

```
data <- simulate_simpson(n = 10, groups = 5, r = 0.5)

if (require("ggplot2")) {
  ggplot(data, aes(x = V1, y = V2)) +
    geom_point(aes(color = Group)) +
    geom_smooth(aes(color = Group), method = "lm") +
    geom_smooth(method = "lm")
}
```

---

spi   *Shortest Probability Interval (SPI)*

Description

Compute the **Shortest Probability Interval (SPI)** of posterior distributions. The SPI is a more computationally stable HDI. The implementation is based on the algorithm from the **SPIn** package.

Usage

```
spi(x, ...)

## S3 method for class 'numeric'
spi(x, ci = 0.95, verbose = TRUE, ...)

## S3 method for class 'data.frame'
spi(x, ci = 0.95, rvar_col = NULL, verbose = TRUE, ...)

## S3 method for class 'stanreg'
spi(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
```

```
    component = c("location", "all", "conditional", "smooth_terms", "sigma",
      "distributional", "auxiliary"),
    parameters = NULL,
    verbose = TRUE,
    ...
)

## S3 method for class 'brmsfit'
spi(
  x,
  ci = 0.95,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'get_predicted'
spi(x, ci = 0.95, use_iterations = FALSE, verbose = TRUE, ...)
```

Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model. **bayestestR** supports a wide range of models (see, for example, methods("hdi")) and not all of those are documented in the 'Usage' section, because methods for other classes mostly resemble the arguments of the .numeric or .data.frame methods.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .95 (95%).
verbose	Toggle off warnings.
rvar_col	A single character - the name of an rvar column in the data frame to be processed. See example in [p_direction()]().
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.
use_iterations	Logical, if TRUE and x is a get_predicted object, (returned by [insight::get_predicted()]()), the function is applied to the iterations instead of the predictions. This only applies to models that return iterations for predicted values (e.g., brmsfit models).

Details

The SPI is an alternative method to the HDI ([`hdi()`](#)) to quantify uncertainty of (posterior) distributions. The SPI is said to be more stable than the HDI, because, the *"HDI can be noisy (that is, have a high Monte Carlo error)"* (Liu et al. 2015). Furthermore, the HDI is sensitive to additional assumptions, in particular assumptions related to the different estimation methods, which can make the HDI less accurate or reliable.

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

The code to compute the SPI was adapted from the **SPIn** package, and slightly modified to be more robust for Stan models. Thus, credits go to Ying Liu for the original SPI algorithm and R implementation.

References

Liu, Y., Gelman, A., & Zheng, T. (2015). Simulation-efficient shortest probability intervals. Statistics and Computing, 25(4), 809–819. https://doi.org/10.1007/s11222-015-9563-8

See Also

Other ci: [`bci`](#)(), [`ci`](#)(), [`eti`](#)(), [`hdi`](#)(), [`si`](#)()

Examples

```
library(bayestestR)

posterior <- rnorm(1000)
spi(posterior)
spi(posterior, ci = c(0.80, 0.89, 0.95))

df <- data.frame(replicate(4, rnorm(100)))
spi(df)
spi(df, ci = c(0.80, 0.89, 0.95))

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

---

weighted_posteriors   *Generate posterior distributions weighted across models*

Description

Extract posterior samples of parameters, weighted across models. Weighting is done by comparing posterior model probabilities, via `bayesfactor_models()`.

Usage

```
weighted_posteriors(..., prior_odds = NULL, missing = 0, verbose = TRUE)

## S3 method for class 'data.frame'
weighted_posteriors(..., prior_odds = NULL, missing = 0, verbose = TRUE)

## S3 method for class 'stanreg'
weighted_posteriors(
  ...,
  prior_odds = NULL,
  missing = 0,
  verbose = TRUE,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL
)

## S3 method for class 'brmsfit'
weighted_posteriors(
  ...,
  prior_odds = NULL,
  missing = 0,
  verbose = TRUE,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL
)

## S3 method for class 'blavaan'
weighted_posteriors(
  ...,
  prior_odds = NULL,
  missing = 0,
  verbose = TRUE,
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL
)
```

```
## S3 method for class 'BFBayesFactor'
weighted_posteriors(
  ...,
  prior_odds = NULL,
  missing = 0,
  verbose = TRUE,
  iterations = 4000
)
```

Arguments

`...`	Fitted models (see details), all fit on the same data, or a single BFBayesFactor object.
`prior_odds`	Optional vector of prior odds for the models compared to the first model (or the denominator, for `BFBayesFactor` objects). For `data.frames`, this will be used as the basis of weighting.
`missing`	An optional numeric value to use if a model does not contain a parameter that appears in other models. Defaults to 0.
`verbose`	Toggle off warnings.
`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.
`iterations`	For `BayesFactor` models, how many posterior samples to draw.

Details

Note that across models some parameters might play different roles. For example, the parameter A plays a different role in the model Y ~ A + B (where it is a main effect) than it does in the model Y ~ A + B + A:B (where it is a simple effect). In many cases centering of predictors (mean subtracting for continuous variables, and effects coding via contr.sum or orthonormal coding via [contr.equalprior_pairs](#) for factors) can reduce this issue. In any case you should be mindful of this issue.

See [bayesfactor_models()](#) details for more info on passed models.

Note that for BayesFactor models, posterior samples cannot be generated from intercept only models.

This function is similar in function to brms::posterior_average.

Value

A data frame with posterior distributions (weighted across models) .

Note

For BayesFactor < 0.9.12-4.3, in some instances there might be some problems of duplicate columns of random effects in the resulting data frame.

References

- Clyde, M., Desimone, H., & Parmigiani, G. (1996). Prediction via orthogonalized model mixing. Journal of the American Statistical Association, 91(435), 1197-1208.
- 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
- Rouder, J. N., Haaf, J. M., & Vandekerckhove, J. (2018). Bayesian inference for psychology, part IV: Parameter estimation and Bayes factors. Psychonomic bulletin & review, 25(1), 102-113.
- van den Bergh, D., Haaf, J. M., Ly, A., Rouder, J. N., & Wagenmakers, E. J. (2019). A cautionary note on estimating effect size.

See Also

bayesfactor_inclusion() for Bayesian model averaging.

Examples

```
if (require("rstanarm") && require("see") && interactive()) {
  stan_m0 <- suppressWarnings(stan_glm(extra ~ 1,
    data = sleep,
    family = gaussian(),
    refresh = 0,
    diagnostic_file = file.path(tempdir(), "df0.csv")
  ))

  stan_m1 <- suppressWarnings(stan_glm(extra ~ group,
    data = sleep,
    family = gaussian(),
    refresh = 0,
    diagnostic_file = file.path(tempdir(), "df1.csv")
  ))

  res <- weighted_posteriors(stan_m0, stan_m1, verbose = FALSE)

  plot(eti(res))
}

## With BayesFactor
if (require("BayesFactor")) {
  extra_sleep <- ttestBF(formula = extra ~ group, data = sleep)

  wp <- weighted_posteriors(extra_sleep, verbose = FALSE)

  describe_posterior(extra_sleep, test = NULL, verbose = FALSE)
  # also considers the null
```

```
    describe_posterior(wp$delta, test = NULL, verbose = FALSE)
}


## weighted prediction distributions via data.frames
if (require("rstanarm") && interactive()) {
  m0 <- suppressWarnings(stan_glm(
    mpg ~ 1,
    data = mtcars,
    family = gaussian(),
    diagnostic_file = file.path(tempdir(), "df0.csv"),
    refresh = 0
  ))

  m1 <- suppressWarnings(stan_glm(
    mpg ~ carb,
    data = mtcars,
    family = gaussian(),
    diagnostic_file = file.path(tempdir(), "df1.csv"),
    refresh = 0
  ))

  # Predictions:
  pred_m0 <- data.frame(posterior_predict(m0))
  pred_m1 <- data.frame(posterior_predict(m1))

  BFmods <- bayesfactor_models(m0, m1, verbose = FALSE)

  wp <- weighted_posteriors(
    pred_m0, pred_m1,
    prior_odds = as.numeric(BFmods)[2],
    verbose = FALSE
  )

  # look at first 5 prediction intervals
  hdi(pred_m0[1:5])
  hdi(pred_m1[1:5])
  hdi(wp[1:5]) # between, but closer to pred_m1
}
```

Index