

# Package ‘bayestestR’

July 20, 2020

**Type** Package

**Title** Understand and Describe Bayesian Models and Posterior Distributions

**Version** 0.7.2

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

**URL** <https://easystats.github.io/bayestestR/>

**BugReports** <https://github.com/easystats/bayestestR/issues>

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

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.0)

**Imports** insight (>= 0.8.4), methods, stats, utils

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

**RoxygenNote** 7.1.1

**Language** en-GB

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Dominique Makowski [aut, cre] (<<https://orcid.org/0000-0001-5375-9967>>), Daniel Lüdtke [aut] (<<https://orcid.org/0000-0002-8895-3206>>), Mattan S. Ben-Shachar [aut] (<<https://orcid.org/0000-0002-4287-4801>>), Michael D. Wilson [aut] (<<https://orcid.org/0000-0003-4143-7308>>), Paul-Christian Bürkner [rev],

Tristan Mahr [rev] (<<https://orcid.org/0000-0002-8890-5116>>),  
 Henrik Singmann [ctb] (<<https://orcid.org/0000-0002-4842-3657>>),  
 Quentin F. Gronau [ctb] (<<https://orcid.org/0000-0001-5510-6943>>)

**Repository** CRAN

**Date/Publication** 2020-07-20 09:30:03 UTC

## R topics documented:

area_under_curve . . . . .	3
as.data.frame.density . . . . .	4
as.numeric.map_estimate . . . . .	4
bayesfactor . . . . .	5
bayesfactor_inclusion . . . . .	6
bayesfactor_models . . . . .	8
bayesfactor_parameters . . . . .	11
bayesfactor_restricted . . . . .	17
check_prior . . . . .	21
ci . . . . .	22
contr.bayes . . . . .	25
convert_bayesian_as_frequentist . . . . .	26
density_at . . . . .	27
describe_posterior . . . . .	28
describe_prior . . . . .	32
diagnostic_posterior . . . . .	33
distribution . . . . .	35
effective_sample . . . . .	37
equivalence_test . . . . .	39
estimate_density . . . . .	42
eti . . . . .	44
hdi . . . . .	48
map_estimate . . . . .	51
mcse . . . . .	53
mediation . . . . .	54
mhdior . . . . .	57
overlap . . . . .	59
pd_to_p . . . . .	61
point_estimate . . . . .	61
p_direction . . . . .	64
p_map . . . . .	67
p_rope . . . . .	70
p_significance . . . . .	71
reshape_ci . . . . .	74
rope . . . . .	74
rope_range . . . . .	78
sensitivity_to_prior . . . . .	79
si . . . . .	80
simulate_correlation . . . . .	84

*area\_under\_curve* 3

simulate_prior . . . . .	85
update.bayesfactor_models . . . . .	86
weighted_posteriors . . . . .	87

**Index** 91

---

*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 'mhdior'
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	<i>Bayes Factors (BF)</i>
-------------	---------------------------

---

## Description

This function compute 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 <code>bayesfactor_models</code> .
<code>prior</code>	An object representing a prior distribution (see 'Details').
<code>direction</code>	Test type (see 'Details'). One of 0, "two-sided" (default, two tailed), -1, "left" (left tailed) or 1, "right" (right tailed).
<code>null</code>	Value of the null, either a scalar (for point-null) or a range (for a interval-null).
<code>hypothesis</code>	A character vector specifying the restrictions as logical conditions (see examples below).
<code>effects</code>	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
<code>verbose</code>	Toggle off warnings.
<code>denominator</code>	Either an integer indicating which of the models to use as the denominator, or a model to be used as a denominator. Ignored for <code>BFBayesFactor</code> .
<code>match_models</code>	See details.
<code>prior_odds</code>	Optional vector of prior odds for the models. See <code>BayesFactor::priorOdds&lt;-</code> .

**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**

```
library(bayestestR)

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

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

## End(Not run)

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

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

bayesfactor(comparison)
```

---

bayesfactor\_inclusion *Inclusion Bayes Factors for testing predictors across Bayesian models*

---

**Description**

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 <code>bayesfactor_models</code> or <code>BFBayesFactor</code> .
match_models	See details.
prior_odds	Optional vector of prior odds for the models. See <code>BayesFactor::priorOdds&lt;-</code> .
...	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 contain the main effect terms from which the interaction term is comprised.

**Value**

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

**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](https://doi.org/10.31234/osf.io/wgb64)
- Clyde, M. A., Ghosh, J., & Littman, M. L. (2011). Bayesian adaptive sampling for variable selection and model averaging. *Journal of Computational and Graphical Statistics*, 20(1), 80-101.
- Mathot, S. (2017). Bayes like a Baws: Interpreting Bayesian Repeated Measures in JASP [Blog post]. Retrieved from <https://www.cogsci.nl/blog/interpreting-bayesian-repeated-measures-in-jasp>

**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)
bayesfactor_inclusion(BFmodels)
## Not run:
# BayesFactor
# -----
library(BayesFactor)

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

bayesfactor_inclusion(BF)

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

## End(Not run)
```

---

bayesfactor\_models      *Bayes Factors (BF) for model comparison*

---

**Description**

This function computes or extracts Bayes factors from fitted models.

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

**Usage**

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

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



## Arguments

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

## Details

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

- For `brmsfit` or `stanreg` models, Bayes factors are computed using the **bridgesampling** package.
  - `brmsfit` models must have been fitted with `save_all_pars = TRUE`.
  - `stanreg` models must have been fitted with a defined `diagnostic_file`.
- For `BFBayesFactor`, `bayesfactor_models()` is mostly a wraparound `BayesFactor::extractBF()`.
- For all other model types (supported by **insight**), BIC approximations are used to compute Bayes factors.

In order to correctly and precisely estimate Bayes factors, a rule of thumb are the 4 P's: **Proper Priors** and **Plentiful Posterior** (i.e. probably at least 40,000 samples instead of the default of 4,000).

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

See also [the Bayes factors vignette](#).

## Value

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

## Note

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

## Author(s)

Mattan S. Ben-Shachar

## References

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

## Examples

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

## Not run:
# With lmerMod objects:
# -----
if (require("lme4")) {
  lmer1 <- lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)
  lmer2 <- lmer(Sepal.Length ~ Petal.Length + (Petal.Length | Species), data = iris)
  lmer3 <- lmer(
    Sepal.Length ~ Petal.Length + (Petal.Length | Species) + (1 | Petal.Width),
    data = iris
  )
  bayesfactor_models(lmer1, lmer2, lmer3, denominator = 1)
  bayesfactor_models(lmer1, lmer2, lmer3, denominator = lmer1)
}

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

```

    data = iris,
    family = gaussian(),
    diagnostic_file = file.path(tempdir(), "df1.csv")
  )
  stan_m2 <- stan_glm(Sepal.Length ~ Species + Petal.Length,
    data = iris,
    family = gaussian(),
    diagnostic_file = file.path(tempdir(), "df2.csv")
  )
  bayesfactor_models(stan_m1, stan_m2, denominator = stan_m0)
}

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

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

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

## End(Not run)

```

---

 bayesfactor\_parameters

*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 = TRUE,
  ...
)
```

```
bf_parameters(  
  posterior,  
  prior = NULL,  
  direction = "two-sided",  
  null = 0,  
  verbose = TRUE,  
  ...  
)  
  
bf_pointull(  
  posterior,  
  prior = NULL,  
  direction = "two-sided",  
  null = 0,  
  verbose = TRUE,  
  ...  
)  
  
bf_rope(  
  posterior,  
  prior = NULL,  
  direction = "two-sided",  
  null = rope_range(posterior),  
  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,  
  verbose = TRUE,  
  effects = c("fixed", "random", "all"),  
  component = c("conditional", "zi", "zero_inflated", "all"),  
  parameters = NULL,  
  ...  
)
```

```

    ...
  )

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

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

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

```

### Arguments

posterior	A numerical vector, stanreg / brmsfit object, emmGrid or a data frame - representing a posterior distribution(s) from (see 'Details').
prior	An object representing a prior distribution (see 'Details').
direction	Test type (see 'Details'). One of 0, "two-sided" (default, two tailed), -1, "left" (left tailed) or 1, "right" (right tailed).
null	Value of the null, either a scalar (for point-null) or a range (for a interval-null).
verbose	Toggle off warnings.
...	Arguments passed to and from other methods. (Can be used to pass arguments to internal <a href="#">logspline</a> .)

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 <b>brms</b> -models.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> to select specific parameters for the output.

## Details

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

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.)

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

- When posterior is a numerical vector, prior should also be a numerical vector.
- When posterior is a `data.frame`, prior should also be a `data.frame`, with matching column order.
- When posterior is a `stanreg` or `brmsfit` model:
  - prior can be set to `NULL`, in which case prior samples are drawn internally.
  - prior can also be a model equivalent to posterior but with samples from the priors *only*.
- When posterior is an `emmGrid` object:
  - prior should be the `stanreg` or `brmsfit` model used to create the `emmGrid` objects.
  - prior can also be an `emmGrid` object equivalent to posterior but created with a model of priors samples *only*.
  - **Note:** When the `emmGrid` has undergone any transformations ("`log`", "`response`", etc.), or regriding, then prior must be an `emmGrid` object, as stated above.

**One-sided Tests (setting an order restriction):** One sided tests (controlled by `direction`) are conducted by restricting the prior and posterior of the non-null values (the "alternative") to one side of the null only (*Morey & Wagenmakers, 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).

**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*).

**Value**

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

**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](https://doi.org/10.1177/1745691611406923)

**Examples**

```
library(bayestestR)

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

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

# emmGrid objects
# -----
```



```

    group_diff <- pairs(emmeans(stan_model, ~group))
    bayesfactor_parameters(group_diff, prior = stan_model)
  }

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

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

## End(Not run)

```

---

 bayesfactor\_restricted

*Bayes Factors (BF) for Order Restricted Models*

---

## Description

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

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,
  hypothesis,
  prior = NULL,
  verbose = TRUE,
  ...
)

bf_restricted(posterior, hypothesis, prior = NULL, verbose = TRUE, ...)

```

```

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

```

### Arguments

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

## 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).

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.)

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

- When `posterior` is a `data.frame`, `prior` should also be a `data.frame`, with matching column order.
- When `posterior` is a `stanreg` or `brmsfit` model:
  - `prior` can be set to `NULL`, in which case prior samples are drawn internally.
  - `prior` can also be a model equivalent to `posterior` but with samples from the priors *only*.
- When `posterior` is an `emmGrid` object:
  - `prior` should be the `stanreg` or `brmsfit` model used to create the `emmGrid` objects.
  - `prior` can also be an `emmGrid` object equivalent to `posterior` but created with a model of priors samples *only*.
  - **Note:** When the `emmGrid` has undergone any transformations ("`log`", "`response`", etc.), or regriding, then `prior` must be an `emmGrid` object, as stated above.

**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-hypothesis) (*Wetzels et al. 2011*).

## Value

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

## References

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

**Examples**

```

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

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

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

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

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

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

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

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

```

```

}

## End(Not run)

```

---

check\_prior

*Check if Prior is Informative*


---

## Description

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

## Usage

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

## Arguments

model	A stanreg, stanfit, or brmsfit object.
method	Can be "gelman" or "lakeland". For the "gelman" method, if the SD of the posterior is more than 0.1 times the SD of the prior, then the prior is considered as informative. For the "lakeland" method, the prior is considered as informative if the posterior falls within the 95% HDI of the prior.
simulate_priors	Should prior distributions be simulated using simulate_prior (default; faster) or sampled (slower, more accurate).
...	Currently not used.

## References

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

## Examples

```

## Not run:
library(bayestestR)
if (require("rstanarm")) {
  model <- 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 <- 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")
}

```

```

  plot(si(model)) # can provide visual confirmation to the Lakeland method
}

## End(Not run)

```

---

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 = 0.89, method = "ETI", verbose = TRUE, BF = 1, ...)

## S3 method for class 'data.frame'
ci(x, ci = 0.89, method = "ETI", verbose = TRUE, BF = 1, ...)

## S3 method for class 'emmGrid'
ci(x, ci = 0.89, method = "ETI", verbose = TRUE, BF = 1, ...)

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

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

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

```

```

    verbose = TRUE,
    BF = 1,
    ...
)

## S3 method for class 'brmsfit'
ci(
  x,
  ci = 0.89,
  method = "ETI",
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  verbose = TRUE,
  BF = 1,
  ...
)

## S3 method for class 'BFBayesFactor'
ci(x, ci = 0.89, method = "ETI", verbose = TRUE, BF = 1, ...)

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

```

## Arguments

x	A stanreg or brmsfit model, or a vector representing a posterior distribution.
...	Currently not used.
ci	Value or vector of probability of the CI (between 0 and 1) to be estimated. Default to .89 (89%) for Bayesian models and .95 (95%) for frequentist models.
method	Can be 'ETI' (default), 'HDI' or 'SI'.
verbose	Toggle off warnings.
BF	The amount of support required to be included in the support interval.
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 <b>brms</b> -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. doi: [10.1136/bmj.l5381](https://doi.org/10.1136/bmj.l5381)

**Examples**

```
library(bayestestR)

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

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

## Not run:
if (require("rstanarm")) {
  model <- stan_glm(mpg ~ wt, data = mtcars, chains = 2, iter = 200, refresh = 0)
  ci(model, method = "ETI", ci = c(.80, .89))
  ci(model, method = "HDI", ci = c(.80, .89))
  ci(model, method = "SI")
}

if (require("brms")) {
  model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
  ci(model, method = "ETI")
  ci(model, method = "HDI")
  ci(model, method = "SI")
}

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



```
  ci(bf, method = "HDI")
}

if (require("emmeans")) {
  model <- emtrends(model, ~1, "wt")
  ci(model, method = "ETI")
  ci(model, method = "HDI")
  ci(model, method = "SI")
}

## End(Not run)
```

---

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

---

## Description

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

## Usage

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

## Arguments

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

## Details

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

## Value

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

## References

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

**Examples**

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

contr.bayes(5) # equivalent to Q_5 in Rouder et al. (2012, p. 363)
#           [,1]      [,2]      [,3]      [,4]
# [1,]  0.0000000  0.8944272  0.0000000  0.0000000
# [2,]  0.0000000 -0.2236068 -0.5000000  0.7071068
# [3,]  0.7071068 -0.2236068 -0.1666667 -0.4714045
# [4,] -0.7071068 -0.2236068 -0.1666667 -0.4714045
# [5,]  0.0000000 -0.2236068  0.8333333  0.2357023

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

## End(Not run)
```

---

```
convert_bayesian_as_frequentist
```

*Convert (refit) a Bayesian model to frequentist*

---

**Description**

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

**Usage**

```
convert_bayesian_as_frequentist(model, data = NULL)
```

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

**Arguments**

model            A Bayesian model.

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

**Examples**

```

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

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

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

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

```

---

density\_at

*Density Probability at a Given Value*


---

**Description**

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

**Usage**

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

**Arguments**

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

**Examples**

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

---

describe\_posterior      *Describe Posterior Distributions*

---

**Description**

Compute indices relevant to describe and characterize the posterior distributions.

**Usage**

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

## S3 method for class 'numeric'
describe_posterior(
  posteriors,
  centrality = "median",
  dispersion = FALSE,
  ci = 0.89,
  ci_method = "hdi",
  test = c("p_direction", "rope"),
  rope_range = "default",
  rope_ci = 0.89,
  bf_prior = NULL,
  BF = 1,
  ...
)

## S3 method for class 'stanreg'
describe_posterior(
  posteriors,
  centrality = "median",
  dispersion = FALSE,
```

```
    ci = 0.89,  
    ci_method = "hdi",  
    test = c("p_direction", "rope"),  
    rope_range = "default",  
    rope_ci = 0.89,  
    bf_prior = NULL,  
    diagnostic = c("ESS", "Rhat"),  
    priors = FALSE,  
    effects = c("fixed", "random", "all"),  
    parameters = NULL,  
    BF = 1,  
    ...  
  )
```

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

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

```
## S3 method for class 'brmsfit'
```

```

describe_posterior(
  posteriors,
  centrality = "median",
  dispersion = FALSE,
  ci = 0.89,
  ci_method = "hdi",
  test = c("p_direction", "rope"),
  rope_range = "default",
  rope_ci = 0.89,
  bf_prior = NULL,
  diagnostic = c("ESS", "Rhat"),
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  BF = 1,
  ...
)

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

```

### Arguments

<code>posteriors</code>	A vector, data frame or model of posterior draws.
<code>centrality</code>	The point-estimates (centrality indices) to compute. Character (vector) or list with one or more of these options: "median", "mean", "MAP" or "all".
<code>dispersion</code>	Logical, if TRUE, computes indices of dispersion related to the estimate(s) (SD and MAD for mean and median, respectively).
<code>ci</code>	Value or vector of probability of the CI (between 0 and 1) to be estimated. Default to .89 (89%) for Bayesian models and .95 (95%) for frequentist models.
<code>ci_method</code>	The type of index used for Credible Interval. Can be "HDI" (default, see <a href="#">hdi</a> ), "ETI" (see <a href="#">eti</a> ) or "SI" (see <a href="#">si</a> ).
<code>test</code>	The indices of effect existence to compute. Character (vector) or list with one or more of these options: "p_direction" (or "pd"), "rope", "p_map", "equivalence_test" (or "equitest"), "bayesfactor" (or "bf") or "all" to compute all tests. For each "test", the corresponding <b>bayestestR</b> function is called (e.g. <a href="#">rope</a> or <a href="#">p_direction</a> ) and its results included in the summary output.

rope_range	ROPE's lower and higher bounds. Should be a list of two values (e.g., <code>c(-0.1, 0.1)</code> ) or "default". If "default", the bounds are set to $x \pm 0.1 \times \text{SD}(\text{response})$ .
rope_ci	The Credible Interval (CI) probability, corresponding to the proportion of HDI, to use for the percentage in ROPE.
...	Additional arguments to be passed to or from methods.
bf_prior	Distribution representing a prior for the computation of Bayes factors / 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.
diagnostic	Diagnostic metrics to compute. Character (vector) or list with one or more of these options: "ESS", "Rhat", "MCSE" or "all".
priors	Add the prior used for each parameter.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> 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 <b>brms</b> -models.

## 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

- [Comparison of Point-Estimates](#)
- [Region of Practical Equivalence \(ROPE\)](#)
- [Bayes factors](#)

## Examples

```
library(bayestestR)

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

df <- data.frame(replicate(4, rnorm(100)))
describe_posterior(df)
describe_posterior(df, centrality = "all", dispersion = TRUE, test = "all")
describe_posterior(df, ci = c(0.80, 0.90))
## Not run:
```

```

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

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

# brms models
# -----
if (require("brms")) {
  model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
  describe_posterior(model)
  describe_posterior(model, centrality = "all", dispersion = TRUE, test = "all")
  describe_posterior(model, ci = c(0.80, 0.90))
}

# BayesFactor objects
# -----
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))
}

## End(Not run)

```

---

describe\_prior

*Describe Priors*


---

## Description

Returns a summary of the priors used in the model.

## Usage

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

## Arguments

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



**Examples**

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

# rstanarm models
# -----
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)
}

## End(Not run)
```

---

diagnostic\_posterior *Posteriors Sampling Diagnostic*

---

**Description**

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

**Usage**

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

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

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

### Arguments

posteriors	A stanreg or brms model.
diagnostic	Diagnostic metrics to compute. Character (vector) or list with one or more of these options: "ESS", "Rhat", "MCSE" or "all".
...	Currently not used.
effects	Should parameters for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned.
component	Should all parameters, parameters for the conditional model, the zero-inflated part of the model, the dispersion term, 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 <b>mf</b> x. May be abbreviated. Note that the <i>conditional</i> component is also called <i>count</i> or <i>mean</i> component, depending on the model.

### 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 a measure of “how much independent information there is in autocorrelated chains” (Kruschke 2015, p182-3).

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

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

### References

- Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4), 457-472.

- Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., & Bürkner, P. C. (2019). Rank-normalization, folding, and localization: An improved Rhat for assessing convergence of MCMC. arXiv preprint arXiv:1903.08008.
- Kruschke, J. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.

## Examples

```
## Not run:
# rstanarm models
# -----
if (require("rstanarm", quietly = TRUE)) {
  model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
  diagnostic_posterior(model)
}

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

## End(Not run)
```

---

distribution

*Empirical Distributions*

---

## Description

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

## Usage

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

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

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

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

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

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

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

```

distribution_uniform(n, min = 0, max = 1, random = FALSE, ...)
distribution_beta(n, shape1, shape2, ncp = 0, random = FALSE, ...)
distribution_tweedie(n, xi = NULL, mu, phi, power = NULL, random = FALSE, ...)
distribution_gamma(n, shape, scale = 1, random = FALSE, ...)
distribution_custom(n, type = "norm", ..., random = FALSE)
distribution_mixture_normal(n, mean = c(-3, 3), sd = 1, random = FALSE, ...)
rnorm_perfect(n, mean = 0, sd = 1)

```

### Arguments

type	Can be any of the names from base R's <a href="#">Distributions</a> , like "cauchy", "pois" or "beta".
...	Arguments passed to or from other methods.
n	the number of observations
mean	vector of means.
sd	vector of standard deviations.
random	Generate near-perfect or random (simple wrappers for the base R <code>r*</code> functions) distributions.
size	number of trials (zero or more).
prob	probability of success on each trial.
location	location and scale parameters.
scale	location and scale parameters.
lambda	vector of (non-negative) means.
df	degrees of freedom ( $> 0$ , maybe non-integer). <code>df = Inf</code> is allowed.
ncp	non-centrality parameter $\delta$ ; currently except for <code>rt()</code> , only for <code>abs(ncp) &lt;= 37.62</code> . If omitted, use the central t distribution.
min	lower and upper limits of the distribution. Must be finite.
max	lower and upper limits of the distribution. Must be finite.
shape1	non-negative parameters of the Beta distribution.
shape2	non-negative parameters of the Beta distribution.
xi	the value of $\xi$ such that the variance is $\text{var}[Y] = \phi\mu^\xi$
mu	the mean
phi	the dispersion
power	a synonym for $\xi$
shape	shape and scale parameters. Must be positive, scale strictly.

**Examples**

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

x <- distribution(type = "gamma", n = 100, shape = 2)
plot(density(x))
```

---

effective_sample	<i>Effective Sample Size (ESS)</i>
------------------	------------------------------------

---

**Description**

This function returns the effective sample size (ESS).

**Usage**

```
effective_sample(model, ...)

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

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

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

**Arguments**

model	A stanreg, stanfit, or brmsfit object.
...	Currently not used.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to <b>brms</b> -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, although 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 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**

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

## End(Not run)
```

---

equivalence_test	<i>Test for Practical Equivalence</i>
------------------	---------------------------------------

---

### Description

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

### Usage

```
equivalence_test(x, ...)

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

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

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

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

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

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

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

)

**Arguments**

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

**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](https://doi.org/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](https://doi.org/10.1007/s112220169649y)

### Examples

```
library(bayestestR)

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

```

equivalence_test(x = rnorm(1000, 1, 0.01), range = c(-0.1, 0.1))
equivalence_test(x = rnorm(1000, 1, 1), ci = c(.50, .99))

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

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

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

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

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

## End(Not run)

```

---

estimate\_density

*Density Estimation*


---

## Description

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

## Usage

```

estimate_density(
  x,
  method = "kernel",
  precision = 2^10,
  extend = FALSE,

```

```

    extend_scale = 0.1,
    bw = "SJ",
    ...
)

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

```

### Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model ( <code>stanreg</code> , <code>brmsfit</code> , <code>MCMCglmm</code> , <code>mcmc</code> or <code>bcplm</code> ) or a <code>BayesFactor</code> model.
method	Density estimation method. Can be "kernel" (default), "logspline" or "KernSmooth".
precision	Number of points of density data. See the <code>n</code> parameter in <code>density</code> .
extend	Extend the range of the x axis by a factor of <code>extend_scale</code> .
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 <code>density</code> . Here, the default has been changed for "SJ", which is recommended.
...	Currently not used.
group_by	Optional character vector. If not <code>NULL</code> and <code>x</code> is a data frame, density estimation is performed for each group (subset) indicated by <code>group_by</code> .

### 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, 1)

```

```

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

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

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

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

df <- data.frame(replicate(4, rnorm(100)))
head(estimate_density(df))
## Not run:
# rstanarm models
# -----
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
head(estimate_density(model))

library(emmeans)
head(estimate_density(emtrends(model, ~1, "wt")))

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

## End(Not run)

```

---

eti

*Equal-Tailed Interval (ETI)*


---

## Description

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

**Usage**

```
eti(x, ...)  
  
## S3 method for class 'numeric'  
eti(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'data.frame'  
eti(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'MCMCglmm'  
eti(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'bayesQR'  
eti(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'sim.merMod'  
eti(  
  x,  
  ci = 0.89,  
  effects = c("fixed", "random", "all"),  
  parameters = NULL,  
  verbose = TRUE,  
  ...  
)  
  
## S3 method for class 'sim'  
eti(x, ci = 0.89, parameters = NULL, verbose = TRUE, ...)  
  
## S3 method for class 'emmGrid'  
eti(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'stanreg'  
eti(  
  x,  
  ci = 0.89,  
  effects = c("fixed", "random", "all"),  
  parameters = NULL,  
  verbose = TRUE,  
  ...  
)  
  
## S3 method for class 'brmsfit'  
eti(  
  x,  
  ci = 0.89,  
  effects = c("fixed", "random", "all"),  
  component = c("conditional", "zi", "zero_inflated", "all"),  
  parameters = NULL,
```

```

    verbose = TRUE,
    ...
)

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

```

### Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model ( <code>stanreg</code> , <code>brmsfit</code> , <code>MCMCglmm</code> , <code>mcmc</code> or <code>bcplm</code> ) or a <code>BayesFactor</code> model.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .89 (89%).
verbose	Toggle off warnings.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> 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 <b>brms</b> -models.

### Details

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

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

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

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

On the other hand, the ETI range does change when transformations are applied to the distribution

(for instance, for a log odds scale to probabilities): the lower and higher bounds of the transformed distribution will correspond to the transformed lower and higher bounds of the original distribution. On the contrary, applying transformations to the distribution will change the resulting HDI.

## Value

A data frame with following columns:

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

## Examples

```
library(bayestestR)

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

df <- data.frame(replicate(4, rnorm(100)))
eti(df)
eti(df, ci = c(.80, .89, .95))
## Not run:
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
eti(model)
eti(model, ci = c(.80, .89, .95))

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

library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
eti(model)
eti(model, ci = c(.80, .89, .95))

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

## End(Not run)
```

---

hdi *Highest Density Interval (HDI)*

---

### Description

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

### Usage

```
hdi(x, ...)  
  
## S3 method for class 'numeric'  
hdi(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'data.frame'  
hdi(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'MCMCglmm'  
hdi(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'bayesQR'  
hdi(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'sim.merMod'  
hdi(  
  x,  
  ci = 0.89,  
  effects = c("fixed", "random", "all"),  
  parameters = NULL,  
  verbose = TRUE,  
  ...  
)  
  
## S3 method for class 'sim'  
hdi(x, ci = 0.89, parameters = NULL, verbose = TRUE, ...)  
  
## S3 method for class 'emmGrid'  
hdi(x, ci = 0.89, verbose = TRUE, ...)  
  
## S3 method for class 'stanreg'  
hdi(  
  x,  
  ci = 0.89,  
  effects = c("fixed", "random", "all"),  
  parameters = NULL,
```



```

    verbose = TRUE,
    ...
  )

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

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

```

### Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model ( <code>stanreg</code> , <code>brmsfit</code> , <code>MCMCglmm</code> , <code>mcmc</code> or <code>bcplm</code> ) or a <code>BayesFactor</code> model.
...	Currently not used.
ci	Value or vector of probability of the (credible) interval - CI (between 0 and 1) to be estimated. Default to .89 (89%).
verbose	Toggle off warnings.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> 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 <b>brms</b> -models.

### Details

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

By default, `hdi()` and `eti()` return the 89% intervals (`ci = 0.89`), deemed to be more stable than, for instance, 95% intervals (*Kruschke, 2014*). An effective sample size of at least 10.000 is recommended if 95% intervals should be computed (*Kruschke, 2014, p. 183ff*). Moreover, 89 indicates the arbitrariness of interval limits - its only remarkable property is being the highest prime number

that does not exceed the already unstable 95% threshold (McElreath, 2015).

A 90% equal-tailed interval (ETI) has 5% of the distribution on either side of its limits. It indicates the 5th percentile and the 95th 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.

### Examples

```
library(bayestestR)

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

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

```

## Not run:
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
hdi(model)
hdi(model, ci = c(.80, .90, .95))

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

library(brms)
model <- brms::brm(mpg ~ wt + cyl, data = mtcars)
hdi(model)
hdi(model, ci = c(.80, .90, .95))

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

## End(Not run)

```

---

map\_estimate

*Maximum A Posteriori probability estimate (MAP)*


---

## Description

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

## Usage

```

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

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

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

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

```

```

    parameters = NULL,
    ...
  )

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

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

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

```

### Arguments

x	Vector representing a posterior distribution, or a data frame of such vectors. Can also be a Bayesian model ( <code>stanreg</code> , <code>brmsfit</code> , <code>MCMCglmm</code> , <code>mcmc</code> or <code>bcplm</code> ) or a <code>BayesFactor</code> model.
precision	Number of points of density data. See the <code>n</code> parameter in <code>density</code> .
method	Density estimation method. Can be "kernel" (default), "logspline" or "KernSmooth".
...	Currently not used.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> 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 <b>brms</b> -models.

### Value

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

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

**Examples**

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

posterior <- rnorm(10000)
map_estimate(posterior)

plot(density(posterior))
abline(v = map_estimate(posterior), col = "red")

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

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

## End(Not run)
```

---

mcse

*Monte-Carlo Standard Error (MCSE)*


---

**Description**

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

**Usage**

```
mcse(model, ...)

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

**Arguments**

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

## Details

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

## References

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

## Examples

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

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

## End(Not run)
```

---

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 = 0.89,
  method = "ETI",
  ...
)

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

```

mediation(
  model,
  treatment,
  mediator,
  response = NULL,
  centrality = "median",
  ci = 0.89,
  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" or "all".
ci	Value or vector of probability of the CI (between 0 and 1) to be estimated. Default to .89 (89%) for Bayesian models and .95 (95%) for frequentist models.
method	Can be 'ETI' (default), 'HDI' 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

```
## Not run:
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, cores = 4, refresh = 0)

# Fit Bayesian mediation model in rstanarm
m3 <- stan_mvmer(
  list(job_seek ~ treat + econ_hard + sex + age + (1 | occp),
        depress2 ~ treat + job_seek + econ_hard + sex + age + (1 | occp)),
  data = jobs,
  cores = 4,
  refresh = 0
)

summary(m1)
mediation(m2, centrality = "mean", ci = .95)
mediation(m3, centrality = "mean", ci = .95)

## End(Not run)

```

---

mhdior

*Maximum HDI level inside/outside ROPE (MHDIOR)*


---

## Description

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

## Usage

```

mhdior(x, ...)

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

## S3 method for class 'data.frame'

```

```

mhdior(x, range = "default", precision = 0.1, ...)

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

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

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

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

```

## Arguments

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

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

## Examples

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

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

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

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

if (require("rstanarm")) {
  model <- stan_glm(
    mpg ~ wt + gear, data = mtcars,
    chains = 2,
    iter = 200,
    refresh = 0
  )
  mhdior(model, precision = 1)
}

if (require("emmeans")) {
  mhdior(emtrends(model, ~1, "wt"))
}

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

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

## End(Not run)
```

## 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 <a href="#">estimate_density</a> .
method_auc	Area Under the Curve (AUC) estimation method. See <a href="#">area_under_curve</a> .
precision	Number of points of density data. See the n parameter in <a href="#">density</a> .
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	<i>Convert between Probability of Direction (pd) and p-value.</i>
---------	---

---

**Description**

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

**Usage**

```
pd_to_p(pd, direction = "two-sided", ...)
p_to_pd(p, direction = "two-sided", ...)
convert_p_to_pd(p, direction = "two-sided", ...)
convert_pd_to_p(pd, direction = "two-sided", ...)
```

**Arguments**

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

**Examples**

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

---

point_estimate	<i>Point-estimates of posterior distributions</i>
----------------	---

---

**Description**

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

**Usage**

```

point_estimate(x, centrality = "all", dispersion = FALSE, ...)

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

```

**Arguments**

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

**Note**

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

**References**

[Vignette In-Depth 1: Comparison of Point-Estimates](#)

**Examples**

```
library(bayestestR)

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

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

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

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

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

## End(Not run)
```

---

p\_direction                      *Probability of Direction (pd)*

---

### Description

Compute the **Probability of Direction** (*pd*, also known as the Maximum Probability of Effect - *MPE*). It varies between 50% and 100% (*i.e.*, 0.5 and 1) and can be interpreted as the probability (expressed in percentage) that a parameter (described by its posterior distribution) is strictly positive or negative (whichever is the most probable). It is mathematically defined as the proportion of the posterior distribution that is of the median's sign. Although differently expressed, this index is fairly similar (*i.e.*, is strongly correlated) to the frequentist **p-value**. In some rare situations, especially when using when using model averaged posteriors (see [weighted\\_posteriors](#) or `brms::posterior_average`), this value may be lower than 0.5.

### Usage

```
p_direction(x, ...)

pd(x, ...)

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

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

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

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

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

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



```

    method = "direct",
    ...
)

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

```

### Arguments

x	Vector representing a posterior distribution. Can also be a Bayesian model (stanreg, brmsfit or BayesFactor).
...	Currently not used.
method	Can be "direct" or one of methods of <a href="#">density estimation</a> , such as "kernel", "logspline" or "KernSmooth". If "direct" (default), the computation is based on the raw ratio of samples superior and inferior to 0. Else, the result is based on the <a href="#">Area under the Curve (AUC)</a> of the estimated <a href="#">density</a> function.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like lp__ or prior_) are filtered by default, so only parameters that typically appear in the summary() are returned. Use parameters to select specific parameters for the output.
component	Should results for all parameters, parameters for the conditional model or the zero-inflated part of the model be returned? May be abbreviated. Only applies to <b>brms</b> -models.

### Details

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

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

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

**Methods of computation:** The most simple and direct way to compute the *pd* is to 1) look at the median's sign, 2) select the portion of the posterior of the same sign and 3) compute the percentage that this portion represents. This "simple" method is the most 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 density function (for which many methods are available), and then computing the [area under the curve](#) (AUC) of the density curve on the other side of 0.

**Strengths and Limitations:** **Strengths:** Straightforward computation and interpretation. Objective property of the posterior distribution. 1:1 correspondence with the frequentist p-value.

**Limitations:** Limited information favoring the null hypothesis.

### Value

Values between 0.5 and 1 corresponding to the probability of direction (pd). In some rare situations, especially when using when using model averaged posteriors (see `brms::weighted_posteriors` or `brms::posterior_average`), this value may be lower than 0.5.

### Note

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

### References

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

### 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")

## Not run:
# rstanarm models
# -----
if (require("rstanarm")) {
  model <- rstanarm::stan_glm(mpg ~ wt + cyl,
    data = mtcars,
    chains = 2, refresh = 0
  )
}
```

```

)
p_direction(model)
p_direction(model, method = "kernel")
}

# emmeans
# -----
if (require("emmeans")) {
  p_direction(emtrends(model, ~1, "wt"))
}

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

# BayesFactor objects
# -----
if (require("BayesFactor")) {
  bf <- tttestBF(x = rnorm(100, 1, 1))
  p_direction(bf)
  p_direction(bf, method = "kernel")
}

## End(Not run)

```

---

p\_map

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


---

## Description

Compute a Bayesian equivalent of the  $p$ -value, related to the odds that a parameter (described by its posterior distribution) has against the null hypothesis ( $h_0$ ) using Mills' (2014, 2017) *Objective Bayesian Hypothesis Testing* framework. It corresponds to the density value at 0 divided by the density at the Maximum A Posteriori (MAP).

## Usage

```

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

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

## S3 method for class 'stanreg'
p_map(
  x,

```

```

precision = 2^10,
method = "kernel",
effects = c("fixed", "random", "all"),
parameters = NULL,
...
)

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

```

### Arguments

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

### 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, Lüdtke D (2019) Indices of Effect Existence and Significance in the Bayesian Framework. *Frontiers in Psychology* 2019;10:2767. doi: [10.3389/fpsyg.2019.02767](https://doi.org/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))

## Not run:
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
p_map(model)

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

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

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

## End(Not run)

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

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

---

p\_rope

*Probability of not being in ROPE*


---

### Description

Compute the proportion of the 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, ...)

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

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

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

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

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

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

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

## S3 method for class 'brmsfit'
p_rope(
```

```

  x,
  range = "default",
  effects = c("fixed", "random", "all"),
  component = c("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  ...
)

```

### Arguments

x	Vector representing a posterior distribution. Can also be a stanreg or brmsfit model.
...	Currently not used.
range	ROPE's lower and higher bounds. Should be a vector of length two (e.g., <code>c(-0.1, 0.1)</code> ) or "default". If "default", the range is set to <code>c(-0.1, 0.1)</code> if input is a vector, and based on <code>rope_range()</code> if a Bayesian model is provided.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> 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 <b>brms</b> -models.

### 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 'emmGrid'
p_significance(x, threshold = "default", ...)

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

## 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**

<code>x</code>	Vector representing a posterior distribution. Can also be a stanreg or brmsfit model.
<code>...</code>	Currently not used.
<code>threshold</code>	The threshold value that separates significant from negligible effect. If "default", the range is set to 0.1 if input is a vector, and based on <a href="#">rope_range()</a> if a Bayesian model is provided.
<code>effects</code>	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
<code>parameters</code>	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp_</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> to select specific parameters for the output.
<code>verbose</code>	Toggle off warnings.



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

`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)

## Not run:
# rstanarm models
# -----
if (require("rstanarm")) {
  model <- rstanarm::stan_glm(mpg ~ wt + cyl,
    data = mtcars,
    chains = 2, refresh = 0
  )
  p_significance(model)
}

## End(Not run)
```

---

reshape_ci	<i>Reshape CI between wide/long formats</i>
------------	---

---

**Description**

Reshape CI between wide/long formats.

**Usage**

```
reshape_ci(x)
```

**Arguments**

x                    A data.frame containing CI\_low and CI\_high.

**Examples**

```
library(bayestestR)

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

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

---

rope	<i>Region of Practical Equivalence (ROPE)</i>
------	---

---

**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, ...)
```

## Default S3 method:

```
rope(x, ...)
```

## S3 method for class 'numeric'

```
rope(x, range = "default", ci = 0.89, ci_method = "HDI", verbose = TRUE, ...)
```

```

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

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

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

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

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

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

```

### Arguments

<code>x</code>	Vector representing a posterior distribution. Can also be a <code>stanreg</code> or <code>brmsfit</code> model.
<code>...</code>	Currently not used.
<code>range</code>	ROPE's lower and higher bounds. Should be a vector of length two (e.g., <code>c(-0.1, 0.1)</code> ) or "default". If "default", the range is set to <code>c(-0.1, 0.1)</code> if input is a vector, and based on <code>rope_range()</code> if a Bayesian model is provided.
<code>ci</code>	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 <a href="#">ci</a> .
verbose	Toggle off warnings.
effects	Should results for fixed effects, random effects or both be returned? Only applies to mixed models. May be abbreviated.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> 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 <b>brms</b> -models.

### Details

**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  $0 \pm .1 * sd(y)$ . 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](https://doi.org/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](https://doi.org/10.1177/1745691611406925).
- Kruschke, J. K. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press. doi: [10.1177/2515245918771304](https://doi.org/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](https://doi.org/10.1177/2515245918771304).
- Makowski D, Ben-Shachar MS, Chen SHA, Lüdtke D (2019) Indices of Effect Existence and Significance in the Bayesian Framework. *Frontiers in Psychology* 2019;10:2767. doi: [10.3389/fpsyg.2019.02767](https://doi.org/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](https://doi.org/10.1007/s112220169649y)

### Examples

```
library(bayestestR)

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

## Not run:
library(rstanarm)
model <- stan_glm(mpg ~ wt + gear, data = mtcars, chains = 2, iter = 200, refresh = 0)
rope(model)
rope(model, ci = c(.90, .95))

library(emmeans)
rope(emtrends(model, ~1, "wt"), ci = c(.90, .95))

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

```

rope(model)
rope(model, ci = c(.90, .95))

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

## End(Not run)

```

---

rope\_range

*Find Default Equivalence (ROPE) Region Bounds*


---

### Description

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

### Usage

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

### Arguments

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

### Details

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

- For **linear models (lm)**, this can be generalised to  $[-0.1 * SD_y, 0.1 * SD_y]$ .
- For **logistic models**, the parameters expressed in log odds ratio can be converted to standardized difference through the formula  $\pi/\sqrt{3}$ , resulting in a range of  $-0.18$  to  $0.18$ .
- For other models with **binary outcome**, it is strongly recommended to manually specify the rope argument. Currently, the same default is applied that for logistic models.
- For 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](https://doi.org/10.1177/2515245918771304).

## Examples

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

  model <- stan_glm(vs ~ mpg, data = mtcars, family = "binomial")
  rope_range(model)
}

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

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

## End(Not run)
```

---

sensitivity\_to\_prior *Sensitivity to Prior*

---

## Description

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

## Usage

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

**Arguments**

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

**See Also**

DescTools

**Examples**

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

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

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

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

## End(Not run)
```

**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  $1/k$ .

**For more info, in particular on specifying correct priors for factors with more than 2 levels, see [the Bayes factors vignette](#).**

## Usage

```
si(posterior, prior = NULL, BF = 1, verbose = TRUE, ...)

## 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("conditional", "zi", "zero_inflated", "all"),
  parameters = NULL,
  ...
)

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

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

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

BF	The amount of support required to be included in the support interval.
verbose	Toggle off warnings.
...	Arguments passed to and from other methods. (Can be used to pass arguments to internal <code>logspline</code> .)
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 <b>brms</b> -models.
parameters	Regular expression pattern that describes the parameters that should be returned. Meta-parameters (like <code>lp__</code> or <code>prior_</code> ) are filtered by default, so only parameters that typically appear in the <code>summary()</code> are returned. Use <code>parameters</code> to select specific parameters for the output.

### Details

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]`.

### Note

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

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

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

**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.

## References

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

## Examples

```
library(bayestestR)

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

si(posterior, prior)
## Not run:
# rstanarm models
# -----
library(rstanarm)
contrasts(sleep$group) <- contr.bayes # see vingette
stan_model <- stan_lmer(extra ~ group + (1 | ID), data = sleep)
si(stan_model)
si(stan_model, BF = 3)

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

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

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

## End(Not run)
```

---

simulate\_correlation *Data Simulation*

---

### Description

Simulate data with specific characteristics.

### Usage

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

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

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

## Examples

```

## Not run:
library(bayestestR)
if (require("rstanarm")) {
  model <- stan_glm(mpg ~ wt + am, data = mtcars, chains = 1, refresh = 0)
  simulate_prior(model)
}

## End(Not run)

```

---

```
update.bayesfactor_models  
      Update bayesfactor_models
```

---

## Description

Update bayesfactor\_models

## Usage

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

## Arguments

object	A <a href="#">bayesfactor_models</a> object.
subset	Vector of model indices to keep or remove.
reference	Index of model to rereference to, or "top" to reference to the best model, or "bottom" to reference to the worst model.
...	Currently not used.

## Examples

```
## Not run:  
library(lme4)  
lmer1 <- lmer(Sepal.Length ~ Petal.Length + (1 | Species), data = iris)  
lmer2 <- lmer(Sepal.Length ~ Petal.Length + (Petal.Length | Species), data = iris)  
lmer3 <- lmer(  
  Sepal.Length ~ Petal.Length + (Petal.Length | Species) + (1 | Petal.Width),  
  data = iris  
)  
  
m <- bayesfactor_models(lmer1, lmer2, lmer3, denominator = 1)  
m  
  
update(m, reference = "bottom")  
  
## End(Not run)
```

---

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 '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 (see 'Details').
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 <b>brms</b> -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  $Y \sim A + B$  (where it is a main effect) than it does in the model  $Y \sim A + B + A:B$  (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.bayes` 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) .

**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](https://doi.org/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")) {
  stan_m0 <- stan_glm(extra ~ 1, data = sleep,
                    family = gaussian(),
                    refresh=0,
                    diagnostic_file = file.path(tempdir(), "df0.csv"))

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

  res <- weighted_posteriors(stan_m0, stan_m1)

  plot(eti(res))
}

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

  wp <- weighted_posteriors(extra_sleep)

  describe_posterior(extra_sleep, test = NULL)
  describe_posterior(wp$delta, test = NULL) # also considers the null
}

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

  m1 <- 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)

wp <- weighted_posteriors(pred_m0, pred_m1,
                          prior_odds = BFmods$BF[2])

# 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

- 'ETI', [23](#), [55](#)
- 'HDI', [23](#), [55](#)
- 'SI', [23](#), [55](#)
  
- area under the curve, [66](#)
- Area under the Curve (AUC), [65](#)
- area\_under\_curve, [3](#), [60](#)
- as.data.frame.density, [4](#)
- as.numeric.map\_estimate, [4](#)
- as.numeric.mhdior
  - (as.numeric.map\_estimate), [4](#)
- as.numeric.p\_direction
  - (as.numeric.map\_estimate), [4](#)
- as.numeric.p\_map
  - (as.numeric.map\_estimate), [4](#)
- as.numeric.p\_significance
  - (as.numeric.map\_estimate), [4](#)
- auc (area\_under\_curve), [3](#)
  
- bayesfactor, [5](#)
- bayesfactor\_inclusion, [6](#), [6](#), [89](#)
- bayesfactor\_models, [6](#), [7](#), [8](#), [86–88](#)
- bayesfactor\_parameters, [6](#), [11](#)
- bayesfactor\_pointull
  - (bayesfactor\_parameters), [11](#)
- bayesfactor\_restricted, [17](#)
- bayesfactor\_ropo
  - (bayesfactor\_parameters), [11](#)
- bayesian\_as\_frequentist
  - (convert\_bayesian\_as\_frequentist), [26](#)
- bf\_inclusion (bayesfactor\_inclusion), [6](#)
- bf\_models (bayesfactor\_models), [8](#)
- bf\_parameters (bayesfactor\_parameters), [11](#)
- bf\_pointull (bayesfactor\_parameters), [11](#)
- bf\_restricted (bayesfactor\_restricted), [17](#)
- bf\_ropo (bayesfactor\_parameters), [11](#)
- BFs for restricted models, [5](#)
  
- BFs for single parameters, [5](#)
  
- check\_prior, [21](#)
- ci, [22](#), [76](#)
- comparison between models, [5](#)
- contr.bayes, [25](#), [88](#)
- convert\_bayesian\_as\_frequentist, [26](#)
- convert\_p\_to\_pd (pd\_to\_p), [61](#)
- convert\_pd\_to\_p (pd\_to\_p), [61](#)
  
- density, [51](#), [65](#)
- density estimation, [65](#), [66](#)
- density\_at, [27](#)
- describe\_posterior, [28](#)
- describe\_prior, [32](#)
- diagnostic\_posterior, [33](#)
- distribution, [35](#)
- distribution\_beta (distribution), [35](#)
- distribution\_binomial (distribution), [35](#)
- distribution\_cauchy (distribution), [35](#)
- distribution\_chisquared (distribution), [35](#)
- distribution\_custom (distribution), [35](#)
- distribution\_gamma (distribution), [35](#)
- distribution\_mixture\_normal (distribution), [35](#)
- distribution\_normal (distribution), [35](#)
- distribution\_poisson (distribution), [35](#)
- distribution\_student (distribution), [35](#)
- distribution\_tweedie (distribution), [35](#)
- distribution\_uniform (distribution), [35](#)
- Distributions, [36](#)
  
- effective\_sample, [37](#)
- equivalence\_test, [39](#)
- equivalence\_test(), [76](#)
- estimate\_density, [42](#), [51](#), [60](#)
- eti, [30](#), [44](#)
  
- HDI, [40](#), [46](#), [50](#), [57](#), [76](#)

hdi, [30](#), [48](#)

inclusion BF, [5](#)

logspline, [14](#), [82](#)

map\_estimate, [51](#)

mcse, [53](#)

mediation, [54](#)

mhdior, [57](#)

overlap, [59](#)

p\_direction, [30](#), [64](#)

p\_map, [67](#)

p\_pointnull (p\_map), [67](#)

p\_rope, [70](#)

p\_significance, [71](#)

p\_to\_pd (pd\_to\_p), [61](#)

pd (p\_direction), [64](#)

pd\_to\_p, [61](#), [65](#), [66](#)

point\_estimate, [61](#)

reshape\_ci, [74](#)

rnorm\_perfect (distribution), [35](#)

ROPE, [40](#), [57](#)

rope, [30](#), [74](#)

rope\_range, [76](#), [78](#)

rope\_range(), [40](#), [58](#), [71](#), [72](#), [75](#)

sensitivity\_to\_prior, [79](#)

si, [30](#), [80](#)

simulate\_correlation, [84](#)

simulate\_difference  
    (simulate\_correlation), [84](#)

simulate\_prior, [85](#)

simulate\_ttest (simulate\_correlation),  
    [84](#)

update.bayesfactor\_models, [86](#)

weighted\_posteriors, [8](#), [64](#), [87](#)