Package ‘BayesTools’

October 12, 2022

Title Tools for Bayesian Analyses
Version 0.2.13
Description Provides tools for conducting Bayesian analyses. The package contains functions for creating a wide range of prior distribution objects, mixing posterior samples from 'JAGS' and 'Stan' models, plotting posterior distributions, and etc...
The tools for working with prior distribution span from visualization, generating 'JAGS' and 'bridgesampling' syntax to basic functions such as rng, quantile, and distribution functions.
Maintainer František Bartoš <f.bartos96@gmail.com>
URL https://fbartos.github.io/BayesTools/
BugReports https://github.com/FBartos/BayesTools/issues
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.2.0
SystemRequirements JAGS >= 4.3.0 (https://mcmc-jags.sourceforge.io/)
Depends stats
Imports graphics, extraDistr, mvtnorm, coda, bridgesampling, parallel, ggrepplot2, Rdpack
Suggests scales, testthat, vdiffr, covr, knitr, rstan, rjags, runjags, BayesFactor, rmarkdown
RdMacros Rdpack
VignetteBuilder knitr
NeedsCompilation no
Author František Bartoš [aut, cre] (<https://orcid.org/0000-0002-0018-5573>)
Repository CRAN
Date/Publication 2022-09-14 16:50:03 UTC
R topics documented:

add_column .................................................. 3
BayesTools .................................................. 4
BayesTools_ensemble_tables .............................. 4
BayesTools_model_tables ................................. 6
check_input .................................................. 9
density.prior ............................................... 11
density_prior ............................................... 11
ensemble_inference ........................................ 13
format_BF .................................................. 14
geom_prior ............................................... 14
geom_prior_list ........................................... 16
inclusion_BF ............................................... 18
interpret ................................................... 18
is.prior .................................................. 19
JAGS_add_priors ........................................... 20
JAGS_bridgesampling ....................................... 21
JAGS_bridgesampling_posterior .......................... 23
JAGS_check_and_list ....................................... 23
JAGS_check_convergence .................................. 25
JAGS_diagnostics .......................................... 26
JAGS_evaluate_formula ..................................... 29
JAGS_fit ................................................... 29
JAGS_formula ............................................... 32
JAGS_get_inits ............................................... 33
JAGS_marglik_parameters .................................. 34
JAGS_marglik_priors ....................................... 35
JAGS_to_monitor ........................................... 35
kitchen_rolls ............................................. 36
lines.prior ............................................... 36
lines_prior_list ........................................... 38
mean.prior .................................................. 39
mix_posteriors ............................................. 40
parameter_names ............................................ 41
plot.prior ............................................... 42
plot_models ............................................... 44
plot_posterior ............................................. 45
plot_prior_list ............................................ 47
point ................................................... 49
print.BayesTools_table .................................... 50
print.prior ............................................... 50
prior ................................................... 51
prior_factor ............................................... 53
prior_functions ............................................ 55
prior_functions_methods ................................... 56
prior_informed ............................................. 57
prior_informed_medicine_names ........................... 59
**add_column**  

Add column to BayesTools table

**Description**

Adds column to a BayesTools table while not breaking formatting, attributes, etc...

**Usage**

```r
add_column(
  table,  
  column_title,  
  column_values,  
  column_position = NULL,  
  column_type = NULL
)
```

**Arguments**

- **table** BayesTools table
- **column_title** title of the new column
- **column_values** values of the new column
- **column_position** position of the new column (defaults to NULL which appends the column to the end)
- **column_type** type of values of the new column (important for formatting, defaults to NULL = the function tries to guess numeric/character based on the column_values but many more specific types are available)

**Value**

returns an object of `BayesTools_table` class.
BayesTools: Provides tools for conducting Bayesian analyses. The package contains functions for creating a wide range of prior distribution objects, mixing posterior samples from JAGS and Stan models, plotting posterior distributions, and etc... The tools for working with prior distribution span from visualization, generating JAGS and bridgesampling syntax to basic functions such as rng, quantile, and distribution functions.

Author(s)
František Bartoš <f.bartos96@gmail.com>

See Also
Useful links:
- [https://fbartos.github.io/BayesTools/](https://fbartos.github.io/BayesTools/)

Description
Creates estimate summaries based on posterior distributions created by `mix_posteriors`, inference summaries based on inference created by `ensemble_inference`, or ensemble summary/diagnostics based on a list of `models_inference` models.

Usage
```r
ensemble_estimates_table(
  samples, 
  parameters, 
  probs = c(0.025, 0.95), 
  title = NULL, 
  footnotes = NULL, 
  warnings = NULL, 
  transform_orthonormal = FALSE, 
  formula_prefix = TRUE 
)
```
ensemble_inference_table(
  inference,
  parameters,
  logBF = FALSE,
  BF01 = FALSE,
  title = NULL,
  footnotes = NULL,
  warnings = NULL
)

ensemble_summary_table(
  models,
  parameters,
  logBF = FALSE,
  BF01 = FALSE,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  remove_spike_0 = TRUE,
  short_name = FALSE
)

ensemble_diagnostics_table(
  models,
  parameters,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  remove_spike_0 = TRUE,
  short_name = FALSE
)

Arguments

samples        posterior samples created by mix_posteriors
parameters      character vector of parameters (or a named list with of character vectors for
                summary and diagnostics tables) specifying the parameters (and their grouping)
                for the summary table
probs          quantiles for parameter estimates
title           title to be added to the table
footnotes      footnotes to be added to the table
warnings       warnings to be added to the table
transform_orthonormal     whether factors with orthonormal prior distributions should be transformed to
                          differences from the grand mean
formula_prefix whether the parameter prefix from formula should be printed. Defaults to TRUE.
inference       model inference created by ensemble_inference
BayesTools_model_tables

logBF  whether the Bayes factor should be on log scale
BF01  whether the Bayes factor should be inverted
models  list of models_inference model objects, each of which containing a list of priors and inference object. The inference must be a named list with information about the model: model number m_number, marginal likelihood marglik, prior and posterior probability prior_prob and post_prob, inclusion Bayes factor inclusion_BF, and fit summary generated by runjags_estimates_table for the diagnostics table
remove_spike_0  whether prior distributions equal to spike at 0 should be removed from the prior_list
short_name  whether the prior distribution names should be shortened. Defaults to FALSE.

Value

ensemble_estimates_table returns a table with the model-averaged estimates, ensemble_inference_table returns a table with the prior and posterior probabilities and inclusion Bayes factors, ensemble_summary_table returns a table with overview of the models included in the ensemble, and ensemble_diagnostics_table returns an overview of the MCMC diagnostics for the models included in the ensemble. All of the tables are objects of class ‘BayesTools_table’.

See Also

ensemble_inference mix_posteriors BayesTools_model_tables

BayesTools_model_tables

Create BayesTools model tables

Description

Creates model summary based on a model objects or provides estimates table for a runjags fit.

Usage

model_summary_table(
  model,
  model_description = NULL,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  remove_spike_0 = TRUE,
  short_name = FALSE,
  formula_prefix = TRUE
)

runjags_estimates_table(
fun fit, transformations = NULL, title = NULL, footnotes = NULL, warnings = NULL, conditional = FALSE, remove_spike_0 = TRUE, transform_orthonormal = FALSE, formula_prefix = TRUE, remove_inclusion = FALSE, remove_parameters = NULL)

runjags_inference_table(fit, title = NULL, footnotes = NULL, warnings = NULL, formula_prefix = TRUE)

JAGS_estimates_table(fit, transformations = NULL, title = NULL, footnotes = NULL, warnings = NULL, conditional = FALSE, remove_spike_0 = TRUE, transform_orthonormal = FALSE, formula_prefix = TRUE, remove_inclusion = FALSE, remove_parameters = NULL)

JAGS_inference_table(fit, title = NULL, footnotes = NULL, warnings = NULL, formula_prefix = TRUE)

JAGS_summary_table(model, model_description = NULL, title = NULL, footnotes = NULL,
BayesTools_model_tables

```r
warnings = NULL,
remove_spike_0 = TRUE,
short_name = FALSE,
formula_prefix = TRUE
)

runjags_estimates_empty_table(title = NULL, footnotes = NULL, warnings = NULL)
runjags_inference_empty_table(title = NULL, footnotes = NULL, warnings = NULL)
stan_estimates_table(
    fit,
    transformations = NULL,
    title = NULL,
    footnotes = NULL,
    warnings = NULL
)
```

Arguments

- **model**
  - model object containing a list of priors and inference object. The inference must be a named list with information about the model: model number `m_number`, marginal likelihood `marglik`, prior and posterior probability `prior_prob` and `post_prob`, and model inclusion Bayes factor `inclusion_BF`.

- **model_description**
  - named list with additional description to be added to the table.

- **title**
  - title to be added to the table.

- **footnotes**
  - footnotes to be added to the table.

- **warnings**
  - warnings to be added to the table.

- **remove_spike_0**
  - whether prior distributions equal to spike at 0 should be removed from the `prior_list`.

- **short_name**
  - whether the prior distribution names should be shortened. Defaults to `FALSE`.

- **formula_prefix**
  - whether the parameter prefix from formula should be printed. Defaults to `TRUE`.

- **fit**
  - runjags model fit.

- **transformations**
  - named list of transformations to be applied to specific parameters.

- **conditional**
  - summarizes estimates conditional on being included in the model for spike and slab priors. Defaults to `FALSE`.

- **transform_orthonormal**
  - whether factors with orthonormal prior distributions should be transformed to differences from the grand mean.

- **remove_inclusion**
  - whether estimates of the inclusion probabilities should be excluded from the summary table. Defaults to `FALSE`.

- **remove_parameters**
  - parameters to be removed from the summary. Defaults to `NULL`, i.e., including all parameters.
Value

model_summary_table returns a table with overview of the fitted model, runjags_estimates_table returns a table with MCMC estimates, and runjags_estimates_empty_table returns an empty estimates table. All of the tables are objects of class 'BayesTools_table'.

See Also

BayesTools_ensemble_tables

Description

A set of convenience functions for checking objects/arguments to a function passed by a user.

Usage

check_bool(x, name, check_length = 1, allow_NULL = FALSE, call = "")

check_char(
  x,  
  name,  
  check_length = 1,  
  allow_values = NULL,  
  allow_NULL = FALSE,  
  call = ""  )

check_real(
  x,  
  name,  
  lower = -Inf,  
  upper = Inf,  
  allow_bound = TRUE,  
  check_length = 1,  
  allow_NULL = FALSE,  
  call = ""  )

check_int(
  x,  
  name,  
  lower = -Inf,  
  upper = Inf,  
  allow_bound = TRUE,  
  check_length = 1,  
  allow_NULL = FALSE,  
  call = ""  )
allow_NULL = FALSE,
call = ""
}

check_list(
  x,
  name,
  check_length = 0,
  check_names = NULL,
  all_objects = FALSE,
  allow_other = FALSE,
  allow_NULL = FALSE,
  call = ""
)

Arguments

x object to be checked
name name of the object that will be print in the error message.
check_length length of the object to be checked. Defaults to 1. Set to 0 in order to not check object length.
allow_NULL whether the object can be NULL. If so, no checks are executed.
call string to be placed as a prefix to the error call.
allow_values names of values allowed in a character vector. Defaults to NULL (do not check).
lower lower bound of allowed values. Defaults to -Inf (do not check).
upper upper bound of allowed values. Defaults to Inf (do not check).
allow_bound whether the values at the boundary are allowed. Defaults to TRUE.
check_names names of entries allowed in a list. Defaults to NULL (do not check).
all_objects whether all entries in check_names must be present. Defaults to FALSE.
allow_other whether additional entries then the specified in check_names might be present

Value

returns NULL, called for the input check.

Examples

# check whether the object is logical
check_bool(TRUE, name = "input")

# will throw an error on any other type
## Not run:
  check_bool("TRUE", name = "input")

## End(Not run)
**contr.orthonormal**

**Orthormal contrast matrix**

**Description**

Return a matrix of orthormal contrasts. Code is based on `stanova::contr.bayes` and corresponding to description by Rouder et al. (2012)

**Usage**

```
contr.orthonormal(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

**Value**

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

**References**


**Examples**

```
contr.orthonormal(c(1, 2))
contr.orthonormal(c(1, 2, 3))
```

---

**density.prior**

**Prior density**

**Description**

Computes density of a prior distribution across a range of values.
Usage

```r
## S3 method for class 'prior'
density(
  x,
  x_seq = NULL,
  x_range = NULL,
  x_range_quant = NULL,
  n_points = 1000,
  n_samples = 10000,
  force_samples = FALSE,
  individual = FALSE,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  truncate_end = TRUE,
  ...
)
```

Arguments

- **x**: a prior
- **x_seq**: sequence of x coordinates
- **x_range**: vector of length two with lower and upper range for the support (used if x_seq is unspecified)
- **x_range_quant**: quantile used for automatically obtaining x_range if both x_range and x_seq are unspecified. Defaults to 0.005 for all but Cauchy, Student-t, Gamma, and Inverse-gamma distributions that use 0.010.
- **n_points**: number of equally spaced points in the x_range if x_seq is unspecified
- **n_samples**: number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with force_samples = TRUE)
- **force_samples**: should prior be sampled instead of obtaining analytic solution whenever possible
- **individual**: should individual densities be returned (e.g., in case of weightfunction)
- **transformation**: transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:
  - **lin**: linear transformation in form of a + b*x
  - **tanh**: also known as Fisher's z transformation
  - **exp**: exponential transformation
  , or a list containing the transformation function fun, inverse transformation function inv, and the Jacobian of the transformation jac. See examples for details.
- **transformation_arguments**: a list with named arguments for the transformation
- **transformation_settings**: boolean indicating whether the settings the x_seq or x_range was specified on the transformed support
Description

Computes prior probabilities, posterior probabilities, and inclusion Bayes factors based either on (1) a list of models, vector of parameters, and a list of indicators the models represent the null or alternative hypothesis for each parameter, (2) on prior model odds, marginal likelihoods, and indicator whether the models represent the null or alternative hypothesis, or (3) list of models for each model.

Usage

compute_inference(prior_weights, margliks, is_null = NULL, conditional = FALSE)
ensemble_inference(model_list, parameters, is_null_list, conditional = FALSE)
models_inference(model_list)

Arguments

prior_weights vector of prior model odds
margliks vector of marginal likelihoods
is_null logical vector of indicators specifying whether the model corresponds to the null or alternative hypothesis (or an integer vector indexing models corresponding to the null hypothesis)
conditional whether prior and posterior model probabilities should be returned only for the conditional model. Defaults to FALSE
model_list list of models, each of which contains marginal likelihood estimated with bridge sampling marglik and prior model odds prior_weights
parameters vector of parameters names for which inference should be drawn
is_null_list list with entries for each parameter carrying either logical vector of indicators specifying whether the model corresponds to the null or alternative hypothesis (or an integer vector indexing models corresponding to the null hypothesis)
Value

compute_inference returns a named list of prior probabilities, posterior probabilities, and Bayes factors, ppoint gives the distribution function, ensemble_inference gives a list of named lists of inferences for each parameter, and models_inference returns a list of models, each expanded by the inference list.

See Also

mix_posteriors BayesTools_ensemble_tables

---

format_BF  

Format Bayes factor

Description

Formats Bayes factor

Usage

format_BF(BF, logBF = FALSE, BF01 = FALSE, inclusion = FALSE)

Arguments

- BF: Bayes factor(s)
- logBF: log(BF)
- BF01: 1/BF
- inclusion: whether the Bayes factor is an inclusion BF (for naming purposes)

Value

format_BF returns a formatted Bayes factor.

---

geom_prior  

Add prior object to a ggplot

Description

Add prior object to a ggplot
Usage

geom_prior(
  x,
  xlim = NULL,
  x_seq = NULL,
  x_range_quant = NULL,
  n_points = 1000,
  n_samples = 10000,
  force_samples = FALSE,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  show_parameter = if (individual) 1 else NULL,
  individual = FALSE,
  rescale_x = FALSE,
  scale_y2 = 1,
  ...
)

Arguments

x a prior
xlim plotting range of the prior
x_seq sequence of x coordinates
x_range_quant quantile used for automatically obtaining x_range if both x_range and x_seq are unspecified. Defaults to \(0.005\) for all but Cauchy, Student-t, Gamma, and Inverse-gamma distributions that use \(0.010\).
n_points number of equally spaced points in the x_range if x_seq is unspecified
n_samples number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with force_samples = TRUE)
force_samples should prior be sampled instead of obtaining analytic solution whenever possible
transformation transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:
  lin linear transformation in form of \(a + b*x\)
  tanh also known as Fisher’s z transformation
  exp exponential transformation
, or a list containing the transformation function fun, inverse transformation function inv, and the Jacobian of the transformation jac. See examples for details.
transformation_arguments a list with named arguments for the transformation
transformation_settings boolean indicating whether the settings the x_seq or x_range was specified on the transformed support
show_parameter which parameter should be returned in case of multiple parameters per prior. Useful when priors for the omega parameter are plotted and individual = TRUE.

individual should individual densities be returned (e.g., in case of weightfunction)

rescale_x allows to rescale x-axis in case a weightfunction is plotted.

scale_y2 scaling factor for a secondary axis

... additional arguments

Value

geom_prior_list returns an object of class 'ggplot'.

See Also

plot.prior() lines.prior()
Arguments

- `prior_list` list of prior distributions
- `xlim` x plotting range
- `x_seq` sequence of x coordinates
- `x_range_quant` quantile used for automatically obtaining x_range if both x_range and x_seq are unspecified. Defaults to 0.005 for all but Cauchy, Student-t, Gamma, and Inverse-gamma distributions that use 0.010.
- `n_points` number of equally spaced points in the x_range if x_seq is unspecified
- `n_samples` number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with `force_samples = TRUE`)
- `force_samples` should prior be sampled instead of obtaining analytic solution whenever possible
- `transformation` transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:
  - `lin` linear transformation in form of $a + b \times x$
  - `tanh` also known as Fisher’s z transformation
  - `exp` exponential transformation
  , or a list containing the transformation function `fun`, inverse transformation function `inv`, and the Jacobian of the transformation `jac`. See examples for details.
- `transformation_arguments` a list with named arguments for the transformation
- `transformation_settings` boolean indicating whether the settings the x_seq or x_range was specified on the transformed support
- `rescale_x` allows to rescale x-axis in case a weightfunction is plotted.
- `scale_y2` scaling factor for a secondary axis
- `prior_list_mu` list of priors for the mu parameter required when plotting PET-PEESE
- `...` additional arguments

Value

gem_prior_list returns an object of class ‘ggplot’.

See Also

`plot_prior_list()` `lines_prior_list()`
### inclusion_BF  
*Compute inclusion Bayes factors*

**Description**

Computes inclusion Bayes factors based on prior model probabilities, posterior model probabilities (or marginal likelihoods), and indicator whether the models represent the null or alternative hypothesis.

**Usage**

```r
inclusion_BF(prior_probs, post_probs, margliks, is_null)
```

**Arguments**

- `prior_probs`: vector of prior model probabilities
- `post_probs`: vector of posterior model probabilities
- `margliks`: vector of marginal likelihoods.
- `is_null`: logical vector of indicators whether the model corresponds to the null or alternative hypothesis (or an integer vector indexing models corresponding to the null hypothesis)

**Details**

Supplying `margliks` as the input is preferred since it is better at dealing with under/overflow (posterior probabilities are very close to either 0 or 1). In case that both the `post_probs` and `margliks` are supplied, the results are based on `margliks`.

**Value**

inclusion_BF returns a Bayes factor.

---

### interpret  
*Interpret ensemble inference and estimates*

**Description**

Provides textual summary for posterior distributions created by `mix_posteriors` and ensemble inference created by `ensemble_inference`.

**Usage**

```r
interpret(inference, samples, specification, method)
```
Arguments

- **inference**: model inference created by `ensemble_inference`
- **samples**: posterior samples created by `mix_posteriors`
- **specification**: list of lists specifying the generated text. Each inner list carries: (1) inference specifying the name of in the inference entry and optionally `inference_name` as a name to use in the text and `inference_BF_name` as a symbol to be used instead of the default "BF", (2) samples specifying the name of in the samples entry and optionally `samples_name` as a name to use in the text, `samples_units` as a unit text to be appended after the estimate, and `samples_conditional` specifying whether the estimate is conditional or model-averaged.
- **method**: character specifying name of the method to be appended at the beginning of each sentence.

Value

- `interpret` returns character.

See Also

- `ensemble_inference`  
- `mix_posteriors`  
- `BayesTools_model_tables`  
- `BayesTools_ensemble_tables`

---

is.prior  

Reports whether x is a a prior object

Description

Reports whether x is a a prior object. Note that point priors inherit the prior.simple property

Usage

- `is.prior(x)`
- `is.prior.point(x)`
- `is.prior.none(x)`
- `is.prior.simple(x)`
- `is.prior.discrete(x)`
- `is.prior.vector(x)`
- `is.prior.PET(x)`
- `is.prior.PEESE(x)`
is.prior.weightfunction(x)

is.prior.factor(x)

is.prior.orthonormal(x)

is.prior.dummy(x)

is.prior.spike_and_slab(x)

Arguments

x an object of test

Value

returns a boolean indicating whether the test object is a prior (of specific type).

Examples

# create some prior distributions
p0 <- prior(distribution = "point", parameters = list(location = 0))
p1 <- prior_PET(distribution = "normal", parameters = list(mean = 0, sd = 1))

is.prior(p0)
is.prior.simple(p0)
is.prior.point(p0)
is.prior.PET(p0)

is.prior(p1)
is.prior.simple(p1)
is.prior.point(p1)
is.prior.PET(p1)

JAGS_add_priors Add 'JAGS' prior

Description

Add priors to a 'JAGS' syntax.

Usage

JAGS_add_priors(syntax, prior_list)

Arguments

syntax JAGS model syntax
prior_list named list of prior distribution (names correspond to the parameter names)
Value

JAGS_add_priors returns a JAGS syntax.

Description

A wrapper around `bridge_sampler` that automatically computes likelihood part dependent on the prior distribution and prepares parameter samples. `log_posterior` must specify a function that takes two arguments - a named list of samples from the prior distributions and the data, and returns log likelihood of the model part.

Usage

```r
JAGS_bridgesampling(
  fit,
  log_posterior,
  data = NULL,
  prior_list = NULL,
  formula_list = NULL,
  formula_data_list = NULL,
  formula_prior_list = NULL,
  add_parameters = NULL,
  add_bounds = NULL,
  maxiter = 10000,
  silent = TRUE,
  ...
)
```

Arguments

- `fit` model fitted with either `runjags` posterior samples obtained with `rjags-package`
- `log_posterior` function that takes a named list of samples, the data, and additional list of parameters passed as ... as input and returns the log of the unnormalized posterior density of the model part
- `data` list containing data to fit the model (not including data for the formulas)
- `prior_list` named list of prior distribution (names correspond to the parameter names) of parameters not specified within the `formula_list`
- `formula_list` named list of formulas to be added to the model (names correspond to the parameter name created by each of the formula)
- `formula_data_list` named list of data frames containing data for each formula (names of the lists correspond to the parameter name created by each of the formula)
JAGS_bridgesampling

formula_prior_list
named list of named lists of prior distributions (names of the lists correspond
to the parameter name created by each of the formula and the names of the prior
distribution correspond to the parameter names) of parameters specified within
the formula

add_parameters vector of additional parameter names that should be used in bridgesampling but
were not specified in the prior_list

add_bounds list with two name vectors ("lb" and "up") containing lower and upper bounds
of the additional parameters that were not specified in the prior_list

maxiter maximum number of iterations for the bridge_sampler

silent whether the progress should be printed, defaults to TRUE

... additional argument to the bridge_sampler and log_posterior function

Value

JAGS_bridgesampling returns an object of class 'bridge'.

Examples

## Not run:
# simulate data
set.seed(1)
data <- list(  
  x = rnorm(10),  
  N = 10
)
data$x

# define priors
priors_list <- list(mu = prior("normal", list(0, 1)))

# define likelihood for the data
model_syntax <-  
"model{  
  for(i in 1:N){  
    x[i] ~ dnorm(mu, 1)
  }
}"

# fit the models
fit <- JAGS_fit(model_syntax, data, priors_list)

# define log posterior for bridge sampling
log_posterior <- function(parameters, data){  
  sum(dnorm(data$x, parameters$mu, 1, log = TRUE))
}

# get marginal likelihoods
marglik <- JAGS_bridgesampling(fit, log_posterior, data, priors_list)

## End(Not run)
Description

prepares posterior distribution for 'bridgesampling' by removing unnecessary parameters and attaching lower and upper bounds of parameters based on a list of prior distributions.

Usage

JAGS_bridgesampling_posterior(
  posterior,
  prior_list,
  add_parameters = NULL,
  add_bounds = NULL
)

Arguments

  posterior    matrix of mcmc samples from the posterior distribution
  prior_list   named list of prior distribution (names correspond to the parameter names) of parameters not specified within the formula_list
  add_parameters  vector of additional parameter names that should be used in bridgesampling but were not specified in the prior_list
  add_bounds    list with two name vectors ("lb" and "up") containing lower and upper bounds of the additional parameters that were not specified in the prior_list

Value

JAGS_bridgesampling_posterior returns a matrix of posterior samples with 'lb' and 'ub' attributes carrying the lower and upper boundaries.

Description

Checks and lists settings for the JAGS_fit function.
Usage

JAGS_check_and_list_fit_settings(
  chains,
  adapt,
  burnin,
  sample,
  thin,
  autofit,
  parallel,
  cores,
  silent,
  seed,
  check_mins = list(chains = 1, adapt = 50, burnin = 50, sample = 100, thin = 1),
  call = ""
)

JAGS_check_and_list_autofit_settings(
  autofit_control,
  skip_sample_extend = FALSE,
  call = ""
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chains</td>
<td>number of chains to be run, defaults to 4</td>
</tr>
<tr>
<td>adapt</td>
<td>number of samples used for adapting the MCMC chains, defaults to 500</td>
</tr>
<tr>
<td>burnin</td>
<td>number of burnin iterations of the MCMC chains, defaults to 1000</td>
</tr>
<tr>
<td>sample</td>
<td>number of sampling iterations of the MCMC chains, defaults to 4000</td>
</tr>
<tr>
<td>thin</td>
<td>thinning interval for the MCMC samples, defaults to 1</td>
</tr>
<tr>
<td>autofit</td>
<td>whether the models should be refitted until convergence criteria specified in autofit_control. Defaults to FALSE.</td>
</tr>
<tr>
<td>parallel</td>
<td>whether the chains should be run in parallel FALSE</td>
</tr>
<tr>
<td>cores</td>
<td>number of cores used for multithreading if parallel = TRUE, defaults to chains</td>
</tr>
<tr>
<td>silent</td>
<td>whether the function should proceed silently, defaults to TRUE</td>
</tr>
<tr>
<td>seed</td>
<td>seed for random number generation</td>
</tr>
<tr>
<td>check_mins</td>
<td>named list of minimal values for which should some input be checked. Defaults to:</td>
</tr>
<tr>
<td></td>
<td>chains 1</td>
</tr>
<tr>
<td></td>
<td>adapt 50</td>
</tr>
<tr>
<td></td>
<td>burnin 50</td>
</tr>
<tr>
<td></td>
<td>sample 100</td>
</tr>
<tr>
<td></td>
<td>thin 1</td>
</tr>
<tr>
<td>call</td>
<td>string to be placed as a prefix to the error call.</td>
</tr>
</tbody>
</table>

check_mins = list(chains = 1, adapt = 50, burnin = 50, sample = 100, thin = 1),
call = ""
)
JAGS_check_convergence

autofit_control

a list of arguments controlling the autofit function. Possible options are:

- **max_Rhat**: maximum R-hat error for the autofit function. Defaults to 1.05.
- **min_ESS**: minimum effective sample size. Defaults to 500.
- **max_error**: maximum MCMC error. Defaults to 0.01.
- **max_SD_error**: maximum MCMC error as the proportion of standard deviation of the parameters. Defaults to 0.05.
- **max_time**: list specifying the time and units after which the automatic fitting function is stopped. The units arguments need to correspond to units passed to `difftime` function.
- **sample_extend**: number of samples between each convergence check. Defaults to 1000.
- **skip_sample_extend**: whether `sample_extend` is allowed to be NULL and skipped in the check.

Value

`JAGS_check_and_list_fit_settings` invisibly returns a list of checked fit settings. `JAGS_check_and_list_autofit_settings` invisibly returns a list of checked autofit settings. Parameter names.

---

**JAGS_check_convergence**

Assess convergence of a runjags model

Description

Checks whether the supplied runjags-package model satisfied convergence criteria.

Usage

```r
JAGS_check_convergence(
  fit,
  prior_list,
  max_Rhat = 1.05,
  min_ESS = 500,
  max_error = 0.01,
  max_SD_error = 0.05
)
```

Arguments

- **fit**: a runjags model
- **prior_list**: named list of prior distribution (names correspond to the parameter names)
- **max_Rhat**: maximum R-hat error for the autofit function. Defaults to 1.05.
- **min_ESS**: minimum effective sample size. Defaults to 500.
- **max_error**: maximum MCMC error. Defaults to 1.01.
- **max_SD_error**: maximum MCMC error as the proportion of standard deviation of the parameters. Defaults to 0.05.
Value

JAGS_check_convergence returns a boolean indicating whether the model converged or not, with an attribute 'errors' carrying the failed convergence checks (if any).

See Also

JAGS_fit()

Examples

## Not run:
# simulate data
set.seed(1)
data <- list(
  x = rnorm(10),
  N = 10
)
data$x

# define priors
priors_list <- list(mu = prior("normal", list(0, 1)))

# define likelihood for the data
model_syntax <- "model{
  for(i in 1:N){
    x[i] ~ dnorm(mu, 1)
  }
}"

# fit the models
fit <- JAGS_fit(model_syntax, data, priors_list)
JAGS_check_convergence(fit, priors_list)

## End(Not run)

JAGS_diagnostics

Plot diagnostics of a 'JAGS' model

Description

Creates density plots, trace plots, and autocorrelation plots for a given parameter of a JAGS model.

Usage

JAGS_diagnostics(
  fit,
  parameter,
  type,
plot_type = "base",
xlim = NULL,
ylim = NULL,
lags = 30,
n_points = 1000,
transformations = NULL,
transform_orthonormal = FALSE,
short_name = FALSE,
parameter_names = FALSE,
formula_prefix = TRUE,
...
)

JAGS_diagnostics_density(  
  fit,
  parameter,
  plot_type = "base",
xlim = NULL,
n_points = 1000,
transformations = NULL,
transform_orthonormal = FALSE,
short_name = FALSE,
parameter_names = FALSE,
formula_prefix = TRUE,
...
)

JAGS_diagnostics_trace(  
  fit,
  parameter,
  plot_type = "base",
ylim = NULL,
transformations = NULL,
transform_orthonormal = FALSE,
short_name = FALSE,
parameter_names = FALSE,
formula_prefix = TRUE,
...
)

JAGS_diagnostics_autocorrelation(  
  fit,
  parameter,
  plot_type = "base",
lags = 30,
transformations = NULL,
transform_orthonormal = FALSE,
short_name = FALSE,
parameter_names = FALSE,
formula_prefix = TRUE,
...
)

Arguments

- **fit**
  - a JAGS model fitted via `JAGS_fit()`
- **parameter**
  - parameter to be plotted
- **type**
  - what type of model diagnostic should be plotted. The available options are "density", "trace", and "autocorrelation"
- **plot_type**
  - whether to use a base plot "base" or ggplot2 "ggplot" for plotting.
- **xlim**
  - x plotting range
- **ylim**
  - y plotting range
- **lags**
  - number of lags to be shown for the autocorrelation plot. Defaults to 30.
- **n_points**
  - number of equally spaced points in the x_range if x_seq is unspecified
- **transformations**
  - named list of transformations to be applied to specific parameters
- **transform_orthonormal**
  - whether factors with orthonormal prior distributions should be transformed to differences from the grand mean
- **short_name**
  - whether prior distribution names should be shorted
- **parameter_names**
  - whether parameter names should be printed
- **formula_prefix**
  - whether the parameter prefix from formula should be printed. Defaults to TRUE.
- **...**
  - additional arguments

Value

diagnostics returns either NULL if plot_type = "base" or an object/list of objects (depending on the number of parameters to be plotted) of class 'ggplot2' if plot_type = "ggplot2".

See Also

`JAGS_fit()` `JAGS_check_convergence()`
Evaluate JAGS formula using posterior samples

Description
Evaluates a JAGS formula on a posterior distribution obtained from a fitted model.

Usage
JAGS_evaluate_formula(fit, formula, parameter, data, prior_list)

Arguments
- fit: model fitted with either runjags posterior samples obtained with rjags-package
- formula: formula specifying the right hand side of the assignment (the left hand side is ignored)
- parameter: name of the parameter created with the formula
- data: data.frame containing predictors included in the formula
- prior_list: named list of prior distribution of parameters specified within the formula

Value
JAGS_evaluate_formula returns a matrix of the evaluated posterior samples on the supplied data.

See Also
JAGS_fit() JAGS_formula()

Fits a ‘JAGS’ model

Description
A wrapper around runjags that simplifies fitting ‘JAGS’ models with usage with pre-specified model part of the ‘JAGS’ syntax, data and list of prior distributions.
Usage

\[
\text{JAGS\_fit}( \\
\quad \text{model\_syntax}, \\
\quad \text{data} = \text{NULL}, \\
\quad \text{prior\_list} = \text{NULL}, \\
\quad \text{formula\_list} = \text{NULL}, \\
\quad \text{formula\_data\_list} = \text{NULL}, \\
\quad \text{formula\_prior\_list} = \text{NULL}, \\
\quad \text{chains} = 4, \\
\quad \text{adapt} = 500, \\
\quad \text{burnin} = 1000, \\
\quad \text{sample} = 4000, \\
\quad \text{thin} = 1, \\
\quad \text{autofit} = \text{FALSE}, \\
\quad \text{autofit\_control} = \text{list(max\_Rhat = 1.05, min\_ESS = 500, max\_error = 0.01, max\_SD\_error} \\
\quad \quad \quad \quad \quad = 0.05, \text{max\_time} = \text{list(time = 60, unit = "mins"), sample\_extend} = 1000), \\
\quad \text{parallel} = \text{FALSE}, \\
\quad \text{cores} = \text{chains}, \\
\quad \text{silent} = \text{TRUE}, \\
\quad \text{seed} = \text{NULL}, \\
\quad \text{add\_parameters} = \text{NULL}, \\
\quad \text{required\_packages} = \text{NULL} \\
\)
\]

Arguments

- **model\_syntax**: jags syntax for the model part
- **data**: list containing data to fit the model (not including data for the formulas)
- **prior\_list**: named list of prior distribution (names correspond to the parameter names) of parameters not specified within the formula\_list
- **formula\_list**: named list of formulas to be added to the model (names correspond to the parameter name created by each of the formula)
- **formula\_data\_list**: named list of data frames containing data for each formula (names of the lists correspond to the parameter name created by each of the formula)
- **formula\_prior\_list**: named list of named lists of prior distributions (names of the lists correspond to the parameter name created by each of the formula and the names of the prior distribution correspond to the parameter names) of parameters specified within the formula
- **chains**: number of chains to be run, defaults to 4
- **adapt**: number of samples used for adapting the MCMC chains, defaults to 500
- **burnin**: number of burnin iterations of the MCMC chains, defaults to 1000
- **sample**: number of sampling iterations of the MCMC chains, defaults to 4000
- **thin**: thinning interval for the MCMC samples, defaults to 1
autofit

whether the models should be refitted until convergence criteria specified in autofit_control. Defaults to FALSE.

autofit_control

a list of arguments controlling the autofit function. Possible options are:

max_Rhat maximum R-hat error for the autofit function. Defaults to 1.05.

min_ESS minimum effective sample size. Defaults to 500.

max_error maximum MCMC error. Defaults to 1.01.

max_SD_error maximum MCMC error as the proportion of standard deviation of the parameters. Defaults to 0.05.

max_time list specifying the time time and units after which the automatic fitting function is stopped. The units arguments need to correspond to units passed to difftime function.

sample_extend number of samples between each convergence check. Defaults to 1000.

parallel whether the chains should be run in parallel FALSE

cores number of cores used for multithreading if parallel = TRUE, defaults to chains

silent whether the function should proceed silently, defaults to TRUE

seed seed for random number generation

add_parameters vector of additional parameter names that should be used monitored but were not specified in the prior_list

required_packages character vector specifying list of packages containing JAGS models required for sampling (in case that the function is run in parallel or in detached R session). Defaults to NULL.

Value

JAGS_fit returns an object of class 'runjags'.

See Also

JAGS_check_convergence()

Examples

## Not run:
# simulate data
set.seed(1)
data <- list(
  x = rnorm(10),
  N = 10
)
data$x

# define priors
priors_list <- list(mu = prior("normal", list(0, 1)))
# define likelihood for the data
model_syntax <-
"model{
   for(i in 1:N){
      x[i] ~ dnorm(mu, 1)
   }
}
"

# fit the models
fit <- JAGS_fit(model_syntax, data, priors_list)

## End(Not run)

---

**JAGS_formula**

Create JAGS formula syntax and data object

**Description**

Creates a JAGS formula syntax, prepares data input, and returns modified prior list for further processing in the JAGS_fit function

**Usage**

```r
JAGS_formula(formula, parameter, data, prior_list)
```

**Arguments**

- `formula`: formula specifying the right hand side of the assignment (the left hand side is ignored)
- `parameter`: name of the parameter to be created with the formula
- `data`: data.frame containing predictors included in the formula
- `prior_list`: named list of prior distribution of parameters specified within the formula

**Value**

`JAGS_formula` returns a list containing the formula JAGS syntax, JAGS data object, and modified prior_list.

**See Also**

`JAGS_fit()`
**Examples**

```r
# simulate data
set.seed(1)
df <- data.frame(
  y = rnorm(60),
  x_cont = rnorm(60),
  x_bin = rbinom(60, 1, .5),
  x_fac3 = factor(rep(c("A", "B", "C"), 20), levels = c("A", "B", "C")),
  x_fac4 = factor(rep(c("A", "B", "C", "D"), 15), levels = c("A", "B", "C", "D"))
)

# specify priors
prior_list <- list(
  "intercept" = prior("normal", list(0, 1)),
  "x_cont" = prior("normal", list(0, .5)),
  "x_fac3" = prior_factor("normal", list(0, 1), contrast = "treatment"),
  "x_fac4" = prior_factor("mnormal", list(0, 1), contrast = "orthonormal"),
  "x_fac3:x_fac4" = prior_factor("mnormal", list(0, .5), contrast = "orthonormal")
)

# create the formula object
formula <- JAGS_formula(
  formula = ~ x_cont + x_fac3 * x_fac4,
  parameter = "mu", data = df, prior_list = prior_list)
```

---

**JAGS_get_inits**

*Create initial values for 'JAGS' model*

**Description**

Creates initial values for priors in a 'JAGS' model.

**Usage**

`JAGS_get_inits(prior_list, chains, seed)`

**Arguments**

- `prior_list` named list of prior distribution (names correspond to the parameter names)
- `chains` number of chains
- `seed` seed for random number generation

**Value**

`JAGS_add_priors` returns a list of JAGS initial values.
JAGS_marglik_parameters

Extract parameters for 'JAGS' priors

Description

Extracts transformed parameters from the prior part of a 'JAGS' model inside of a 'bridgesampling' function (returns them as a named list)

Usage

JAGS_marglik_parameters(samples, prior_list)

JAGS_marglik_parameters_formula(
  samples,
  formula_data_list,
  formula_prior_list,
  prior_list_parameters
)

Arguments

- **samples**: samples provided by bridgesampling function
- **prior_list**: named list of prior distribution (names correspond to the parameter names) of parameters not specified within the formula_list
- **formula_data_list**: named list of data frames containing data for each formula (names of the lists correspond to the parameter name created by each of the formula)
- **formula_prior_list**: named list of named lists of prior distributions (names of the lists correspond to the parameter name created by each of the formula and the names of the prior distribution correspond to the parameter names) of parameters specified within the formula
- **prior_list_parameters**: named list of prior distributions on model parameters (not specified within the formula but that might scale the formula parameters)

Value

*JAGS_marglik_parameters* returns a named list of (transformed) posterior samples.


**JAGS_marglik_priors**

*Compute marginal likelihood for 'JAGS' priors*

---

**Description**

Computes marginal likelihood for the prior part of a 'JAGS' model within 'bridgesampling' function.

**Usage**

```
JAGS_marglik_priors(samples, prior_list)
```

```
JAGS_marglik_priors_formula(samples, formula_prior_list)
```

**Arguments**

- `samples`: samples provided by bridgesampling function
- `prior_list`: named list of prior distribution (names correspond to the parameter names) of parameters not specified within the `formula_list`
- `formula_prior_list`: named list of named lists of prior distributions (names of the lists correspond to the parameter name created by each of the formula and the names of the prior distribution correspond to the parameter names) of parameters specified within the `formula`

**Value**

`JAGS_marglik_priors` returns a numeric value of likelihood evaluated at the current posterior sample.

---

**JAGS_to_monitor**

*Create list of monitored parameters for 'JAGS' model*

---

**Description**

Creates a vector of parameter names to be monitored in a 'JAGS' model.

**Usage**

```
JAGS_to_monitor(prior_list)
```

**Arguments**

- `prior_list`: named list of prior distribution (names correspond to the parameter names)

**Value**

`JAGS_to_monitor` returns a character vector of parameter names.
kitchen_rolls  

*Kitchen Rolls data from Wagenmakers et al. (2015) replication study.*

**Description**

The data set contains mean NEO PI-R scores for two groups of students. Each of them filled a personality questionnaire while rotating a kitchen roll either clock or counter-clock wise. See Wagenmakers et al. (2015) for more details about the replication study and the https://osf.io/uszvx/ for the original data.

**Usage**

`kitchen_rolls`

**Format**

A data.frame with 2 columns and 102 observations.

**Value**

a data.frame.

**References**


---

lines.prior  

*Add prior object to a plot*

**Description**

Add prior object to a plot

**Usage**

```r
## S3 method for class 'prior'
lines(
  x,
  xlim = NULL,
  x_seq = NULL,
  x_range_quant = NULL,
  n_points = 1000,
  n_samples = 10000,
)```

force_samples = FALSE,
transformation = NULL,
transformation_arguments = NULL,
transformation_settings = FALSE,
show_parameter = if (individual) 1 else NULL,
individual = FALSE,
rescale_x = FALSE,
scale_y2 = 1,
...)

Arguments

x  a prior
xlim  plotting range of the prior
x_seq  sequence of x coordinates
x_range_quant  quantile used for automatically obtaining x_range if both x_range and x_seq are unspecified. Defaults to 0.005 for all but Cauchy, Student-t, Gamma, and Inverse-gamme distributions that use 0.010.
n_points  number of equally spaced points in the x_range if x_seq is unspecified
n_samples  number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with force_samples = TRUE)
force_samples  should prior be sampled instead of obtaining analytic solution whenever possible
transformation  transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:
  lin  linear transformation in form of a + b*x
tanh  also known as Fisher’s z transformation
exp  exponential transformation
  , or a list containing the transformation function fun, inverse transformation function inv, and the Jacobian of the transformation jac. See examples for details.
transformation_arguments  a list with named arguments for the transformation
transformation_settings  boolean indicating whether the settings the x_seq or x_range was specified on the transformed support
show_parameter  which parameter should be returned in case of multiple parameters per prior. Useful when priors for the omega parameter are plotted and individual = TRUE.
individual  should individual densities be returned (e.g., in case of weightfunction)
rescale_x  allows to rescale x-axis in case a weightfunction is plotted.
scale_y2  scaling factor for a secondary axis
...  additional arguments
lines_prior_list

Add list of prior objects to a plot

Description

Add list of prior objects to a plot

Usage

lines_prior_list(
  prior_list,
  xlim = NULL,
  x_seq = NULL,
  x_range_quant = NULL,
  n_points = 500,
  n_samples = 10000,
  force_samples = FALSE,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  rescale_x = FALSE,
  scale_y2 = NULL,
  prior_list_mu = NULL,
  ...
)

Arguments

prior_list list of prior distributions
xlim x plotting range
x_seq sequence of x coordinates
x_range_quant quantile used for automatically obtaining x_range if both x_range and x_seq are unspecified. Defaults to 0.005 for all but Cauchy, Student-t, Gamma, and Inverse-gamma distributions that use 0.010.
n_points number of equally spaced points in the x_range if x_seq is unspecified
n_samples number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with force_samples = TRUE)
force_samples should prior be sampled instead of obtaining analytic solution whenever possible

Value

lines.prior returns NULL.

See Also

plot.prior() geom_prior()
transformation transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:
  - lin linear transformation in form of a + b*x
  - tanh also known as Fisher’s z transformation
  - exp exponential transformation

  or a list containing the transformation function fun, inverse transformation function inv, and the Jacobian of the transformation jac. See examples for details.
transformation_arguments a list with named arguments for the transformation
transformation_settings boolean indicating whether the settings the x_seq or x_range was specified on the transformed support
rescale_x allows to rescale x-axis in case a weightfunction is plotted.
scale_y2 scaling factor for a secondary axis
prior_list_mu list of priors for the mu parameter required when plotting PET-PEESE
... additional arguments

Value

lines_prior_list returns NULL.

See Also

plot_prior_list() geom_prior_list()
mix_posteriors

Value
a mean of an object of class 'prior'.

See Also
prior()

Examples

# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# compute mean of the prior distribution
mean(p1)

mix_posteriors Model-average posterior distributions

Description
Model-averages posterior distributions based on a list of models, vector of parameters, and a list of indicators the models represent the null or alternative hypothesis for each parameter.

Usage

mix_posteriors(
  model_list,
  parameters,
  is_null_list,
  conditional = FALSE,
  seed = NULL,
  n_samples = 10000
)

Arguments

model_list list of models, each of which contains marginal likelihood estimated with bridge sampling marglik and prior model odds prior_weights
parameters vector of parameters names for which inference should be drawn
is_null_list list with entries for each parameter carrying either logical vector of indicators specifying whether the model corresponds to the null or alternative hypothesis (or an integer vector indexing models corresponding to the null hypothesis)
conditional whether prior and posterior model probabilities should be returned only for the conditional model. Defaults to FALSE
seed integer specifying seed for sampling posteriors for model averaging. Defaults to 1.
n_samples number of samples to be drawn for the model-averaged posterior distribution
Value

mix_posteriors returns a named list of mixed posterior distributions (either a vector of matrix).

See Also

ensemble_inference BayesTools_ensemble_tables

Description

Removes additional formatting from parameter names outputted from JAGS.

Usage

format_parameter_names(
  parameters,
  formula_parameters = NULL,
  formula_prefix = TRUE
)

JAGS_parameter_names(parameters, formula_parameter = NULL)

Arguments

parameters a vector of parameter names
formula_parameters a vector of formula parameter prefix names
formula_prefix whether the formula_parameters names should be kept. Defaults to TRUE.
formula_parameter a formula parameter prefix name

Value

A character vector with reformatted parameter names.

Examples

format_parameter_names(c("mu_x_cont", "mu_x_fac3t", "mu_x_fac3t__xXx__x_cont"),
  formula_parameters = "mu")
plot.prior

Plots a prior object

Description

Plots a prior object

Usage

## S3 method for class 'prior'
plot(
    x,
    plot_type = "base",
    x_seq = NULL,
    xlim = NULL,
    x_range_quant = NULL,
    n_points = 1000,
    n_samples = 10000,
    force_samples = FALSE,
    transformation = NULL,
    transformation_arguments = NULL,
    transformation_settings = FALSE,
    showFigures = if (individual) ~1 else NULL,
    individual = FALSE,
    rescale_x = FALSE,
    par_name = NULL,
    ...
)

Arguments

x        a prior
plot_type whether to use a base plot "base" or ggplot2 "ggplot" for plotting.
x_seq         sequence of x coordinates
xlim         x plotting range
x_range_quant quantile used for automatically obtaining x_range if both x_range and x_seq are unspecified. Defaults to 0.005 for all but Cauchy, Student-t, Gamma, and Inverse-gamma distributions that use 0.010.
n_points     number of equally spaced points in the x_range if x_seq is unspecified
n_samples    number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with force_samples = TRUE)
force_samples should prior be sampled instead of obtaining analytic solution whenever possible
transformation transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:
lin linear transformation in form of $a + b \times x$

tanh also known as Fisher’s z transformation

exp exponential transformation,
, or a list containing the transformation function fun, inverse transformation function inv, and the Jacobian of the transformation jac. See examples for details.

transformation_arguments
a list with named arguments for the transformation

transformation_settings
boolean indicating whether the settings the x_seq or x_range was specified on the transformed support

show_figures which figures should be returned in case of multiple plots are generated. Useful when priors for the omega parameter are plotted and individual = TRUE.

individual should individual densities be returned (e.g., in case of weightfunction)

rescale_x allows to rescale x-axis in case a weightfunction is plotted.

par_name a type of parameter for which the prior is specified. Only relevant if the prior corresponds to a mu parameter that needs to be transformed.

... additional arguments

Value

plot.prior returns either NULL or an object of class ’ggplot’ if plot_type is plot_type = ”ggplot”.

See Also

prior() lines.prior() geom_prior()

Examples

# create some prior distributions
p0 <- prior(distribution = "point", parameters = list(location = 0))

p1 <- prior(distribution = "normal", parameters = list(mean = 0, sd = 1))

p2 <- prior(distribution = "normal", parameters = list(mean = 0, sd = 1), truncation = list(0, Inf))

# a default plot
plot(p0)

# manipulate line thickness and color, change the parameter name
plot(p1, lwd = 2, col = "blue", par_name = bquote(mu))

# use ggplot
plot(p2, plot_type = "ggplot")

# utilize the ggplot prior geom
plot(p2, plot_type = "ggplot", xlim = c(-2, 2)) + geom_prior(p1, col = "red", lty = 2)

# apply transformation
plot(p1, transformation = "exp")
plot_models

Plot estimates from models

Description
Plot estimates from models

Usage

plot_models(
  model_list,  
  samples,  
  inference,  
  parameter,  
  plot_type = "base",  
  prior = FALSE,  
  conditional = FALSE,  
  order = NULL,  
  transformation = NULL,  
  transformation_arguments = NULL,  
  transformation_settings = FALSE,  
  par_name = NULL,  
  formula_prefix = TRUE,
  ...
)

Arguments

model_list     list of models, each of which contains marginal likelihood estimated with bridge sampling marglik and prior model odds prior_weights
samples        samples from a posterior distribution for a parameter generated by mix_posteriors.
inference      object created by ensemble_inference function
parameter      parameter name to be plotted. Does not support PET-PEESE and weightfunction.
plot_type      whether to use a base plot "base" or ggplot2 "ggplot" for plotting.
prior          whether prior distribution should be added to the figure
conditional    whether conditional models should be displayed
order          list specifying ordering of the models. The first element describes whether the ordering should be "increasing" or "decreasing" and the second element describes whether the ordering should be based "model" order, "estimate" size, posterior "probability", or the inclusion "BF".
transformation transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:

  lin  linear transformation in form of a + b*x
tanh also known as Fisher’s z transformation
exp exponential transformation

, or a list containing the transformation function fun, inverse transformation function inv, and the Jacobian of the transformation jac. See examples for details.

transformation_arguments
  a list with named arguments for the transformation

transformation_settings
  boolean indicating whether the settings the x_seq or x_range was specified on the transformed support

par_name
  a type of parameter for which the prior is specified. Only relevant if the prior corresponds to a mu parameter that needs to be transformed.

formula_prefix
  whether the formula_parameters names should be kept. Defaults to TRUE.

... additional arguments. E.g.:
  "show_updating" whether Bayes factors and change from prior to posterior odds should be shown on the secondary y-axis
  "show_estimates" whether posterior estimates and 95% CI should be shown on the secondary y-axis
  "y_axis2" whether the secondary y-axis should be shown

Details

Plots prior and posterior estimates of the same parameter across multiple models (prior distributions with orthonormal contrast) are always plotted as differences from the grand mean).

Value

plot_models returns either NULL or an object of class ‘ggplot’ if plot_type is plot_type = "ggplot".

See Also

prior() lines_prio_list() geom_prio_list()
Usage

plot_posterior(
  samples,  
  parameter,  
  plot_type = "base",  
  prior = FALSE,  
  n_points = 1000,  
  n_samples = 10000,  
  force_samples = FALSE,  
  transformation = NULL,  
  transformation_arguments = NULL,  
  transformation_settings = FALSE,  
  rescale_x = FALSE,  
  par_name = NULL,  
  dots_prior = list(),  
  ...  
)

Arguments

samples  
samples from a posterior distribution for a parameter generated by `mix_posteriors`.

parameter  
parameter name to be plotted. Use "PETPEESE" for PET-PEESE plot with parameters "PET" and "PEESE", and "weightfunction" for plotting a weight-function with parameters "omega".

plot_type  
whether to use a base plot "base" or ggplot2 "ggplot" for plotting.

prior  
whether prior distribution should be added to the figure

n_points  
number of equally spaced points in the x_range if x_seq is unspecified

n_samples  
number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with force_samples = TRUE)

force_samples  
should prior be sampled instead of obtaining analytic solution whenever possible

transformation  
transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:

  - **lin** linear transformation in form of a + b*x
  - **tanh** also known as Fisher’s z transformation
  - **exp** exponential transformation

  , or a list containing the transformation function `fun`, inverse transformation function `inv`, and the Jacobian of the transformation `jac`. See examples for details.

transformation_arguments  
a list with named arguments for the transformation

transformation_settings  
boolean indicating whether the settings the x_seq or x_range was specified on the transformed support

rescale_x  
allows to rescale x-axis in case a weightfunction is plotted.
plot_prior_list

par_name               a type of parameter for which the prior is specified. Only relevant if the prior corresponds to a mu parameter that needs to be transformed.
dots_prior            additional arguments for the prior distribution plot
...                   additional arguments

Value

plot_posterior returns either NULL or an object of class 'ggplot' if plot_type is plot_type = "ggplot".

See Also

prior() lines_prior_list() geom_prior_list()

plot_prior_list  Plot a list of prior distributions

Description

Plot a list of prior distributions

Usage

plot_prior_list(
  prior_list,
  plot_type = "base",
  x_seq = NULL,
  xlim = NULL,
  x_range_quant = NULL,
  n_points = 500,
  n_samples = 10000,
  force_samples = FALSE,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  rescale_x = FALSE,
  par_name = NULL,
  prior_list_mu = NULL,
  ...
)

Arguments

prior_list            list of prior distributions
plot_type             whether to use a base plot "base" or ggplot2 "ggplot" for plotting.
x_seq                 sequence of x coordinates
plot_prior_list

`xlim` x plotting range

`x_range_quant` quantile used for automatically obtaining `x_range` if both `x_range` and `x_seq` are unspecified. Defaults to 0.005 for all but Cauchy, Student-t, Gamma, and Inverse-gamma distributions that use 0.010.

`n_points` number of equally spaced points in the `x_range` if `x_seq` is unspecified

`n_samples` number of samples from the prior distribution if the density cannot be obtained analytically (or if samples are forced with `force_samples = TRUE`)

`force_samples` should prior be sampled instead of obtaining analytic solution whenever possible

`transformation` transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:

- `lin` linear transformation in form of `a + b * x`
- `tanh` also known as Fisher’s z transformation
- `exp` exponential transformation

, or a list containing the transformation function `fun`, inverse transformation function `inv`, and the Jacobian of the transformation `jac`. See examples for details.

`transformation_arguments` a list with named arguments for the transformation

`transformation_settings` boolean indicating whether the settings the `x_seq` or `x_range` was specified on the transformed support

`rescale_x` allows to rescale x-axis in case a weightfunction is plotted.

`par_name` a type of parameter for which the prior is specified. Only relevant if the prior corresponds to a mu parameter that needs to be transformed.

`prior_list_mu` list of priors for the mu parameter required when plotting PET-PEESE

`...` additional arguments

**Value**

`plot_prior_list` returns either NULL or an object of class ’ggplot’ if plot_type is `plot_type = "ggplot"`.

**See Also**

`prior()` `lines_prior_list()` `geom_prior_list()`
Description

Density, distribution function, quantile function and random generation for point distribution.

Usage

dpoint(x, location, log = FALSE)
rpoint(n, location)
ppoint(q, location, lower.tail = TRUE, log.p = FALSE)
qpoint(p, location, lower.tail = TRUE, log.p = FALSE)

Arguments

- `x, q`: vector or matrix of quantiles.
- `location`: vector of locations.
- `log, log.p`: logical; if TRUE, probabilities p are given as log(p).
- `n`: number of observations.
- `lower.tail`: logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X \geq x] \).
- `p`: vector of probabilities.

Value

dpoint gives the density, ppoint gives the distribution function, qpoint gives the quantile function, and rpoint generates random deviates.

Examples

# draw samples from a point distribution
rpoint(10, location = 1)
print.BayesTools_table

*Print a BayesTools table*

**Description**

Print a BayesTools table

**Usage**

```r
## S3 method for class 'BayesTools_table'
print(x, ...)
```

**Arguments**

- `x`: a `BayesTools_values_tables`
- `...`: additional arguments.

**Value**

`print.BayesTools_table` returns `NULL`.

print.prior

*Prints a prior object*

**Description**

Prints a prior object

**Usage**

```r
## S3 method for class 'prior'
print(
  x,
  short_name = FALSE,
  parameter_names = FALSE,
  plot = FALSE,
  digits_estimates = 2,
  silent = FALSE,
  ...
)
```

prior

Arguments

- `x`: a prior
- `short_name`: whether prior distribution names should be shorted
- `parameter_names`: whether parameter names should be printed
- `plot`: to return `bquote` formatted prior name for plotting.
- `digits_estimates`: number of decimals to be displayed for printed parameters.
- `silent`: to silently return the print message.
- `...`: additional arguments

Value

`print.prior` invisibly returns the print statement.

See Also

- `prior()`

Examples

```r
# create some prior distributions
p0 <- prior(distribution = "point", parameters = list(location = 0))
p1 <- prior(distribution = "normal", parameters = list(mean = 0, sd = 1))

# print them
p0
p1

# use short names
print(p1, short_name = TRUE)

# print parameter names
print(p1, parameter_names = TRUE)

# generate bquote plotting syntax
plot(0, main = print(p1, plot = TRUE))
```

prior

Creates a prior distribution

Description

`prior` creates a prior distribution. The prior can be visualized by the `plot` function.
prior

Usage

prior(
  distribution,  # name of the prior distribution. The possible options are
  parameters,   # list of appropriate parameters for a given distribution.
  truncation = list(lower = -Inf, upper = Inf),  # list with two elements, lower and upper, that define the lower and upper truncation of the distribution. Defaults to list(lower = -Inf, upper = Inf). The truncation is automatically set to the bounds of the support.
  prior_weights = 1  # prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.
)

prior_none(prior_weights = 1)

Arguments

distribution  # name of the prior distribution. The possible options are
  "point" for a point density characterized by a location parameter.
  "normal" for a normal distribution characterized by a mean and sd parameters.
  "lognormal" for a lognormal distribution characterized by a meanlog and sdlog parameters.
  "cauchy" for a Cauchy distribution characterized by a location and scale parameters. Internally converted into a generalized t-distribution with df = 1.
  "t" for a generalized t-distribution characterized by a location, scale, and df parameters.
  "gamma" for a gamma distribution characterized by either shape and rate, or shape and scale parameters. The later is internally converted to the shape and rate parametrization
  "invgamma" for an inverse-gamma distribution characterized by a shape and scale parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter.
  "beta" for a beta distribution characterized by an alpha and beta parameters.
  "exp" for an exponential distribution characterized by either rate or scale parameter. The later is internally converted to rate.
  "uniform" for a uniform distribution defined on a range from a to b

parameters  # list of appropriate parameters for a given distribution.

truncation  # list with two elements, lower and upper, that define the lower and upper truncation of the distribution. Defaults to list(lower = -Inf, upper = Inf). The truncation is automatically set to the bounds of the support.

prior_weights  # prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

prior and prior_none return an object of class 'prior'. A named list containing the distribution name, parameters, and prior weights.
prior_factor

See Also

plot.prior(), Normal, Lognormal, Cauchy, Beta, Exponential, LocationScaleT, InvGamma.

Examples

# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# create a half-normal standard normal prior distribution
p2 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1),
            truncation = list(lower = 0, upper = Inf))

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)

prior_factor

Creates a prior distribution for factors

Description

prior_factor creates a prior distribution for fitting models with factor predictors. (Note that results
across different operating systems might vary due to differences in JAGS numerical precision.)

Usage

prior_factor(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1,
  contrast = "orthonormal"
)

Arguments

distribution  name of the prior distribution. The possible options are
              "point" for a point density characterized by a location parameter.
              "normal" for a normal distribution characterized by a mean and sd parameters.
              "lognormal" for a lognormal distribution characterized by a meanlog and sdlog
                          parameters. Internally converted into a generalized t-distribution with df = 1.
              "cauchy" for a Cauchy distribution characterized by a location and scale
                        parameters.
              "t" for a generalized t-distribution characterized by a location, scale, and df
                          parameters.
"gamma" for a gamma distribution characterized by either shape and rate, or shape and scale parameters. The later is internally converted to the shape and rate parametrization.

"invgamma" for an inverse-gamma distribution characterized by a shape and scale parameters. The JAGS part uses a 1/gamma distribution with a shape and rate parameter.

"beta" for a beta distribution characterized by an alpha and beta parameters.

"exp" for an exponential distribution characterized by either rate or scale parameter. The later is internally converted to rate.

"uniform" for a uniform distribution defined on a range from a to b.

- **parameters**: list of appropriate parameters for a given distribution.
- **truncation**: list with two elements, lower and upper, that define the lower and upper truncation of the distribution. Defaults to list(lower = -Inf, upper = Inf). The truncation is automatically set to the bounds of the support.
- **prior_weights**: prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.
- **contrast**: type of contrast for the prior distribution. The possible options are
  - "orthonormal" for contrast centered around the grand mean with equal marginal distributions, making the prior distribution exchangeable across factor levels. Only supports distribution = "mnormal" and distribution = "mt" which generates the corresponding multivariate normal/t distributions.
  - "treatment" for contrasts using the first level as a comparison group and setting equal prior distribution on differences between the individual factor levels and the comparison level.

**Value**

return an object of class 'prior'.

**See Also**

prior()

**Examples**

```r
# create an orthonormal prior distribution
p1 <- prior_factor(distribution = "mnormal", contrast = "orthonormal",
                    parameters = list(mean = 0, sd = 1))
```
**Description**

Density (pdf / lpdf), distribution function (cdf / ccdf), quantile function (quant), random generation (rng), mean, standard deviation (sd), and marginal variants of the functions (mpdf, mlpf, mcdf, mccdf, mquant) for prior distributions.

**Usage**

```r
## S3 method for class 'prior'
rng(x, n, ...)

## S3 method for class 'prior'
cdf(x, q, ...)

## S3 method for class 'prior'
ccdf(x, q, ...)

## S3 method for class 'prior'
lpdf(x, y, ...)

## S3 method for class 'prior'
pdf(x, y, ...)

## S3 method for class 'prior'
quant(x, p, ...)

## S3 method for class 'prior'
mcdf(x, q, ...)

## S3 method for class 'prior'
mccdf(x, q, ...)

## S3 method for class 'prior'
mlpdf(x, y, ...)

## S3 method for class 'prior'
mpdf(x, y, ...)
```

**Arguments**

- `x` prior distribution
n  number of observations
...
q  vector or matrix of quantiles
y  vector of observations
p  vector of probabilities

Value

dpdf (mpdf) and lpdf (mlpdf) give the (marginal) density and the log of (marginal) density, cdf (mcdf) and ccdf (mccdf) give the (marginal) distribution and the complement of (marginal) distribution function, quant (mquant) give the (marginal) quantile function, and rng generates random deviates for an object of class 'prior'.

Examples

# create a standard normal prior distribution
p1 <- prior(distribution = 'normal', parameters = list(mean = 1, sd = 1))

# generate a random sample from the prior
rng(p1, 10)

# compute cumulative density function
cdf(p1, 0)

# obtain quantile
quant(p1, .5)

# compute probability density
pdf(p1, c(0, 1, 2))

prior_functions_methods

Creates generics for common statistical functions

Description

Density (pdf / lpdf), distribution function (cdf / ccdf), quantile function (quant), random generation (rng), mean, standard deviation (sd), and marginal variants of the functions (mpdf, mlpf, mcdf, mccdf, mquant).

Usage

rng(x, ...)  
cdf(x, ...)
prior_informed creates an informed prior distribution based on research.

Description

prior_informed creates an informed prior distribution based on past research. The prior can be visualized by the plot function.

Usage

prior_informed(name, parameter = NULL, type = "smd")
Arguments

name  name of the prior distribution. There are many options based on prior psychological or medical research. For psychology, the possible options are

"van Erp" for an informed prior distribution for the heterogeneity parameter tau of meta-analytic effect size estimates based on standardized mean differences (van Erp et al. 2017),

"Oosterwijk" for an informed prior distribution for the effect sizes expected in social psychology based on prior elicitation with dr. Oosterwijk (Gronau et al. 2017).

For medicine, the possible options are based on Bartoš et al. (2021) who developed empirical prior distributions for the effect size and heterogeneity parameters of the continuous standardized outcomes based on the Cochrane database of systematic reviews. Use "Cochrane" for a prior distribution based on the whole database or call print(prior_informed_medicine_names) to inspect the names of all 46 subfields and set the appropriate parameter and type.

parameter  parameter name describing what prior distribution is supposed to be produced in cases where the name corresponds to multiple prior distributions. Relevant only for the empirical medical prior distributions.

type  prior type describing what prior distribution is supposed to be produced in cases where the name and parameter correspond to multiple prior distributions. Relevant only for the empirical medical prior distributions.

Value

prior_informed returns an object of class ’prior’.

References


See Also

prior(), prior_informed_medicine_names

Examples

# prior distribution representing expected effect sizes in social psychology
# based on prior elicitation with dr. Oosterwijk
prior_informed_medicine_names

Names of medical subfields from the Cochrane database of systematic reviews

Description

Contain names identifying the individual subfields from the Cochrane database of systematic reviews. The individual elements correspond to valid name arguments for the prior_informed() function.

Usage

prior_informed_medicine_names

Format

An object of class character of length 47.

Value

returns a character vector with names of medical subfields from Cochrane database of systematic reviews.

See Also

prior_informed()

Examples

print(prior_informed_medicine_names)
prior_PP

Creates a prior distribution for PET or PEESE models

Description

prior creates a prior distribution for fitting a PET or PEESE style models in RoBMA. The prior
distribution can be visualized by the plot function.

Usage

prior_PET(
  distribution,
  parameters,
  truncation = list(lower = 0, upper = Inf),
  prior_weights = 1
)

prior_PEESE(
  distribution,
  parameters,
  truncation = list(lower = 0, upper = Inf),
  prior_weights = 1
)

Arguments

distribution name of the prior distribution. The possible options are
  "point" for a point density characterized by a location parameter.
  "normal" for a normal distribution characterized by a mean and sd parameters.
  "lognormal" for a lognormal distribution characterized by a meanlog and sdlog
  parameters.
  "cauchy" for a Cauchy distribution characterized by a location and scale
  parameters. Internally converted into a generalized t-distribution with df = 1.
  "t" for a generalized t-distribution characterized by a location, scale, and df
  parameters.
  "gamma" for a gamma distribution characterized by either shape and rate, or
  shape and scale parameters. The later is internally converted to the shape
  and rate parametrization
  "invgamma" for an inverse-gamma distribution characterized by a shape and
  scale parameters. The JAGS part uses a 1/gamma distribution with a shape
  and rate parameter.
  "beta" for a beta distribution characterized by an alpha and beta parameters.
  "exp" for an exponential distribution characterized by either rate or scale
  parameter. The later is internally converted to rate.
prior_spike_and_slab

"uniform" for a uniform distribution defined on a range from a to b

parameters  list of appropriate parameters for a given distribution.

truncation  list with two elements, lower and upper, that define the lower and upper truncation of the distribution. Defaults to list(lower = -Inf, upper = Inf). The truncation is automatically set to the bounds of the support.

prior_weights  prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

prior_PET and prior_PEESE return an object of class 'prior'.

See Also

plot.prior(), prior()

Examples

# create a half-Cauchy prior distribution
# (PET and PEESE specific functions automatically set lower truncation at 0)
p1 <- prior_PET(distribution = "Cauchy", parameters = list(location = 0, scale = 1))

plot(p1)

prior_spike_and_slab  Creates a spike and slab prior distribution

Description

prior_spike_and_slab creates a spike and slab prior distribution corresponding to the specification in Kuo and Mallick (1998) (see O'Hara and Sillanpäälä (2009) for further details). I.e., a prior distribution is multiplied by an independent indicator with values either zero or one.

Usage

prior_spike_and_slab(
  prior_parameter,
  prior_inclusion = prior(distribution = "spike", parameters = list(location = 0.5)),
  prior_weights = 1
)
prior_weightfunction

Arguments

prior_parameter

- a prior distribution for the parameter

prior_inclusion

- a prior distribution for the inclusion probability. The inclusion probability must be bounded within 0 and 1 range. Defaults to prior("spike", parameters = list(location = 0.5)) which corresponds to 1/2 prior probability of including the slab prior distribution (but other prior distributions, like beta etc can be also specified).

prior_weights

- prior odds associated with a given distribution. The value is passed into the model fitting function, which creates models corresponding to all combinations of prior distributions for each of the model parameters and sets the model priors odds to the product of its prior distributions.

Value

- return an object of class 'prior'.

See Also

- prior()

Examples

```r
# create a spike and slab prior distribution
p1 <- prior_spike_and_slab(
  prior(distribution = "normal", parameters = list(mean = 0, sd = 1)),
  prior_inclusion = prior(distribution = "beta", parameters = list(alpha = 1, beta = 1))
)
```

prior_weightfunction

Create a prior distribution for a weight function

Description

prior_weightfunction creates a prior distribution for fitting a RoBMA selection model. The prior can be visualized by the plot function.

Usage

```r
prior_weightfunction(distribution, parameters, prior_weights = 1)
```
Arguments

distribution name of the prior distribution. The possible options are
   "two.sided" for a two-sided weight function characterized by a vector steps
   and vector alpha parameters. The alpha parameter determines an alpha
   parameter of Dirichlet distribution which cumulative sum is used for the
   weights omega.
   "one.sided" for a one-sided weight function characterized by either a vector
   steps and vector alpha parameter, leading to a monotonic one-sided func-
   tion, or by a vector steps, vector alpha1, and vector alpha2 parameters
   leading non-monotonic one-sided weight function. The alpha/alpha1 and
   alpha2 parameters determine an alpha parameter of Dirichlet distribution
   which cumulative sum is used for the weights omega.

parameters list of appropriate parameters for a given
distribution.
prior_weights prior odds associated with a given distribution. The model fitting function usu-
ally creates models corresponding to all combinations of prior distributions for
each of the model parameters, and sets the model priors odds to the product of
its prior distributions.

Value

prior_weightfunction returns an object of class 'prior'.

See Also

plot.prior()

Examples

p1 <- prior_weightfunction("one-sided", parameters = list(steps = c(.05, .10), alpha = c(1, 1, 1)))

# the prior distribution can be visualized using the plot function
# (see ?plot.prior for all options)
plot(p1)

---

range.prior Prior range

Description

Computes range of a prior distribution (if the prior distribution is unbounded range from quantiles
to 1 -quantiles) is returned.

Usage

## S3 method for class 'prior'
range(x, quantiles = NULL, ..., na.rm = FALSE)
Arguments

x a prior
quantiles quantile to be returned in case of unbounded distribution.
... additional arguments
na.rm unused

Value

range.prior returns a numeric vector of length with a plotting range of a prior distribution.

See Also

prior()

sd

Creates generic for sd function

Description

Creates generic for sd function

Usage

sd(x, ...)

Arguments

x main argument
... additional arguments

Value

sd returns a standard deviation of the supplied object (if it is either a numeric vector or an object of class 'prior').

See Also

sd
Description

Computes standard deviation of a prior distribution.

Usage

```r
## S3 method for class 'prior'
sd(x, ...)
```

Arguments

- `x` : a prior
- `...` : unused arguments

Value

A standard deviation of an object of class 'prior'.

See Also

`prior()`

Examples

```r
# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# compute sd of the prior distribution
sd(p1)
```

Description

Transforms posterior samples from model-averaged posterior distributions based on orthonormal prior distributions into differences from the mean.

Usage

```r
transform_orthonormal_samples(samples)
```
Arguments
samples (a list) of mixed posterior distributions created with mix_posteriors function

Value
transform_orthonormal_samples returns a named list of mixed posterior distributions (either a vector of matrix).

See Also
mix_posteriors contr.orthonormal

---

var

Creates generic for var function

Description
Creates generic for var function

Usage
var(x, ...)

Arguments
x main argument
... additional arguments

Value
var returns a variance of the supplied object (if it is either a numeric vector or an object of class 'prior').

See Also
cor
**Description**

Computes variance of a prior distribution.

**Usage**

```r
## S3 method for class 'prior'
var(x, ...)
```

**Arguments**

- `x` a prior
- `...` unused arguments

**Value**

a variance of an object of class 'prior'.

**See Also**

`prior()`

**Examples**

```r
# create a standard normal prior distribution
p1 <- prior(distribution = "normal", parameters = list(mean = 1, sd = 1))

# compute variance of the prior distribution
var(p1)
```

---

**weightfunctions**

**Description**

Marginal density, marginal distribution function, marginal quantile function and random generation for weight functions.
Usage

mdone.sided(x, alpha = NULL, omega1 = NULL, omega2 = NULL, log = FALSE)
mdtwo.sided(x, omega, log = FALSE)
mdone.sided_fixed(x, omega, log = FALSE)
mdtwo.sided_fixed(x, omega, log = FALSE)
rone.sided(n, alpha = NULL, omega1 = NULL, omega2 = NULL)
rtwo.sided(n, omega)
rone.sided_fixed(n, omega)
rtwo.sided_fixed(n, omega)
mpone.sided(
  q,
  alpha = NULL,
  omega1 = NULL,
  omega2 = NULL,
  lower.tail = TRUE,
  log.p = FALSE
)
mptwo.sided(q, alpha, lower.tail = TRUE, log.p = FALSE)
mpone.sided_fixed(q, omega, lower.tail = TRUE, log.p = FALSE)
mptwo.sided_fixed(q, omega, lower.tail = TRUE, log.p = FALSE)
mqone.sided(
  p,
  alpha = NULL,
  omega1 = NULL,
  omega2 = NULL,
  lower.tail = TRUE,
  log.p = FALSE
)
mqtwo.sided(p, alpha, lower.tail = TRUE, log.p = FALSE)
mqone.sided_fixed(p, omega, lower.tail = TRUE, log.p = FALSE)
mqtwo.sided_fixed(p, omega, lower.tail = TRUE, log.p = FALSE)
weightfunctions_mapping

Create coefficient mapping between multiple weightfunctions

Description

Creates coefficients mapping between multiple weightfunctions.

Usage

weightfunctions_mapping(prior_list, cuts_only = FALSE)
Arguments

prior_list  list of prior distributions
cuts_only  whether only p-value cuts should be returned

Value

weightfunctions_mapping returns a list of indices mapping the publication weights omega from the individual weightfunctions into a joint weightfunction.
Index

* datasets
  - kitchen_rolls, 36
  - prior_informed_medicine_names, 59

* package
  - BayesTools, 4
  - _PACKAGE (BayesTools), 4

add_column, 3
BayesTools, 4
BayesTools-package (BayesTools), 4
BayesTools_ensemble_tables, 4, 9, 14, 19, 41
BayesTools_model_tables, 6, 6, 19
Beta, 53
bquote, 51
bridge_sampler, 21, 22

Cauchy, 53
ccdf (prior_functions_methods), 56
ccdf.prior (prior_functions), 55
cdf (prior_functions_methods), 56
cdf.prior (prior_functions), 55
check_bool (check_input), 9
check_char (check_input), 9
check_input, 9
check_int (check_input), 9
check_list (check_input), 9
check_real (check_input), 9
compute_inference (ensemble_inference), 13
contr.orthonormal, 11, 66
cor, 66
density.prior, 11
difftime, 25, 31
dpoint (point), 49

ensemble_diagnostics_table
  - (BayesTools_ensemble_tables), 4
ensemble_estimates_table
  - (BayesTools_ensemble_tables), 4
ensemble_inference, 4–6, 13, 18, 19, 41, 44
ensemble_inference_table
  - (BayesTools_ensemble_tables), 4
ensemble_summary_table
  - (BayesTools_ensemble_tables), 4
Exponential, 53

format_BF, 14
format_parameter_names
  - (parameter_names), 41
geom_prior, 14
geom_prior(), 38, 43
geom_prior_list, 16
geom_prior_list(), 39, 45, 47, 48
inclusion_BF, 18
interpret, 18
InvGamma, 53
is.prior, 19
JAGS_add_priors, 20
JAGS_bridgesampling, 21
JAGS_bridgesampling_posterior, 23
JAGS_check_and_list, 23
JAGS_check_and_list_autofit_settings
  - (JAGS_check_and_list), 23
JAGS_check_and_list_fit_settings
  - (JAGS_check_and_list), 23
JAGS_check_convergence, 25
JAGS_check_convergence(), 28, 31
JAGS_diagnostics, 26
JAGS_diagnostics_autocorrelation
  - (JAGS_diagnostics), 26
JAGS_diagnostics_density
  - (JAGS_diagnostics), 26
JAGS_diagnostics_trace
  - (JAGS_diagnostics), 26
INDEX

JAGS_estimates_table (BayesTools_model_tables), 6
JAGS_evaluate_formula, 29
JAGS_fit, 23, 29
JAGS_fit(), 26, 28, 29, 32
JAGS_formula, 32
JAGS_formula(), 29
JAGS_get_inits, 33
JAGS_inference_table (BayesTools_model_tables), 6
JAGS_marglik_parameters, 34
JAGS_marglik_parameters_formula (JAGS_marglik_parameters), 34
JAGS_marglik_priors, 35
JAGS_marglik_priors_formula (JAGS_marglik_priors), 35
JAGS_parameter_names (parameter_names), 41
JAGS_summary_table (BayesTools_model_tables), 6
JAGS_to_monitor, 35

kitchen_rolls, 36

lines.prior, 36
lines.prior(), 16, 43
lines.prior_list, 38
lines.prior_list(), 17, 45, 47, 48
LocationScaleT, 53
Lognormal, 53
lpdf (prior_functions_methods), 56
lpdf.prior (prior_functions), 55
mccdf (prior_functions_methods), 56
mccdf.prior (prior_functions), 55
mcdf (prior_functions_methods), 56
mcdf.prior (prior_functions), 55
mdone.sided (weightfunctions), 67
mdone.sided_fixed (weightfunctions), 67
mdtwo.sided (weightfunctions), 67
mdtwo.sided_fixed (weightfunctions), 67
mean.prior, 39
mix_posteriors, 4–6, 14, 18, 19, 40, 44, 46, 66
mlpdf (prior_functions_methods), 56
mlpdf.prior (prior_functions), 55
model_summary_table (BayesTools_model_tables), 6
models_inference (ensemble_inference), 13
mpdf (prior_functions_methods), 56
mpdf.prior (prior_functions), 55
mpone.sided (weightfunctions), 67
mpone.sided_fixed (weightfunctions), 67
mptwo.sided (weightfunctions), 67
mptwo.sided_fixed (weightfunctions), 67
mqone.sided (weightfunctions), 67
mqtwo.sided (weightfunctions), 67
mqtwo.sided_fixed (weightfunctions), 67
mquant (prior_functions_methods), 56
mquant.prior (prior_functions), 55

Normal, 53

parameter_names, 41
pdf (prior_functions_methods), 56
pdf.prior (prior_functions), 55
plot.prior, 42
plot.prior(), 16, 38, 53, 61, 63
plot_models, 44
plot_posterior, 45
plot_prior_list, 47
plot_prior_list(), 17, 39
point, 49
ppoint (point), 49
print.BayesTools_table, 50
print.prior, 50
prior, 51
prior(), 13, 40, 43, 45, 47, 48, 51, 54, 58, 61, 62, 64, 65, 67
prior_factor, 53
prior_functions, 55
prior_functions_methods, 56
prior_informed, 57
prior_informed(), 59
prior_informed_medicine_names, 58, 59
prior_none (prior), 51
prior_PEESE (prior_PP), 60
prior_PET (prior_PP), 60
prior_PP, 60
prior_spike_and_slab, 61
prior_weightfunction, 62
qpoint (point), 49
quant (prior_functions_methods), 56
quant.prior (prior_functions), 55
range.prior, 63
rjags-package, 21, 29
rng (prior_functions_methods), 56
rng.prior (prior_functions), 55
rone.sided (weightfunctions), 67
rone.sided_fixed (weightfunctions), 67
rpoint (point), 49
rtwo.sided (weightfunctions), 67
rtwo.sided_fixed (weightfunctions), 67
run.jags, 29
runjags, 21, 29
runjags-package, 25
runjags_estimates_empty_table
  (BayesTools_model_tables), 6
runjags_estimates_table, 6
runjags_estimates_table
  (BayesTools_model_tables), 6
runjags_inference_empty_table
  (BayesTools_model_tables), 6
runjags_inference_table
  (BayesTools_model_tables), 6
sd, 64, 64
sd.prior, 65
stan_estimates_table
  (BayesTools_model_tables), 6
transform_orthonormal_samples, 65
var, 66
var.prior, 67
weightfunctions, 67
weightfunctions_mapping, 69