Package 'BayesTools'

October 4, 2021

Title Tools for Bayesian Analyses

Version 0.1.2

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

RoxygenNote 7.1.1

Depends stats

Imports graphics, extraDistr, runjags, coda, bridgesampling, parallel, ggplot2

Suggests scales, testthat, vdiffr, covr, rjags, rstan

NeedsCompilation no

Author František Bartoš [aut, cre] (<https://orcid.org/0000-0002-0018-5573>)

Repository CRAN

Date/Publication 2021-10-04 09:20:02 UTC

R topics documented:

`BayesTools` *BayesTools*

Description

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/
- Report bugs at https://github.com/FBartos/BayesTools/issues

---

`BayesTools_ensemble_tables`

Create BayesTools ensemble summary tables

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

```
ensemble_estimates_table(
  samples,
  parameters,
  probs = c(0.025, 0.95),
  title = NULL,
  footnotes = NULL,
  warnings = NULL
)

ensemble_inference_table(
  inference,
```

```
  parameters,
  logBF = FALSE,
  BF01 = FALSE,
  title = NULL,
  footnotes = NULL,
  warnings = NULL
)

ensemble_summary_table(
  models,
  parameters,
  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
`inference`	model inference created by ensemble_inference
`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
)

runjags_estimates_table(
  fit,
  prior_list,
  transformations = NULL,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  remove_spike_0 = TRUE
)

runjags_estimates_empty_table(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.
fit	runjags model fit
prior_list	list of prior distributions
transformations	named list of transformations to be applied to specific 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

---

check_input *Check input*

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 = ""
)

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)
```

---

density.prior *Prior density*

Description

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

Usage

```
## 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-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
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
truncate_end	whether the density should be set to zero in for the endpoints of truncated distributions
...	additional arguments

Value

density.prior returns an object of class 'density'.

See Also

prior()

`ensemble_inference` *Compute posterior probabilities and inclusion Bayes factors*

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)
```

Arguments

BF	Bayes factor(s)
logBF	log(BF)
BF01	1/BF

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-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

Value

geom_prior_list returns an object of class 'ggplot'.

See Also

plot.prior() lines.prior()

`geom_prior_list` *Add list of prior objects to a plot*

Description

Add list of prior objects to a plot

Usage

```
geom_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-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
`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

`geom_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 and posterior model probabilities and indicator whether the models represent the null or alternative hypothesis

Usage

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

Arguments

`prior_probs`	vector of prior model probabilities
`post_probs`	vector of posterior model probabilities
`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)

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

```
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.PET(x)

is.prior.PEESE(x)

is.prior.weightfunction(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

Adds 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.

JAGS_bridgesampling *Compute marginal likelihood of a 'JAGS' model*

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

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

Arguments

fit	model fitted with either runjags posterior samples obtained with rjags-package
data	data that were used to fit the model
prior_list	named list of prior distribution (names correspond to the parameter names)
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
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

```
# 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, chains = 1, sample = 1000, seed = 0)

# 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, data, priors_list, log_posterior)
```

JAGS_bridgesampling_posterior

Prepare 'JAGS' posterior for 'bridgesampling'

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)
`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.

`JAGS_check_and_list` *Check and list 'JAGS' fitting settings*

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

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.
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
check_mins	named list of minimal values for which should some input be checked. Defaults to: **chains** 1 **adapt** 50 **burnin** 50 **sample** 100 **thin** 1
call	string to be placed as a prefix to the error call.
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.
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_set
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

```
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

```
# 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, seed = 0)
JAGS_check_convergence(fit, priors_list)
```

JAGS_fit *Fits a 'JAGS' model*

Description

A wrapper around run.jags that simplifies fitting 'JAGS' models with usage with pre-specified model part of the 'JAGS' syntax, data and list of prior distributions.

Usage

```
JAGS_fit(
  model_syntax,
  data,
  prior_list,
  chains = 4,
  adapt = 500,
  burnin = 1000,
  sample = 4000,
  thin = 1,
  autofit = FALSE,
 autofit_control = list(max_Rhat = 1.05, min_ESS = 500, max_error = 0.01, max_SD_error
    = 0.05, max_time = list(time = 60, unit = "mins"), sample_extend = 1000),
  parallel = FALSE,
```

```
  cores = chains,
  silent = TRUE,
  seed = NULL,
  add_parameters = NULL,
  required_packages = NULL
)
```

Arguments

model_syntax	jags syntax for the model part
data	data fit the model
prior_list	named list of prior distribution (names correspond to the parameter names)
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

```
# 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, seed = 0)
```

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)
```

Arguments

samples	samples provided by bridgesampling function
prior_list	named list of prior distribution (names correspond to the parameter names)

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)
```

Arguments

samples	samples provided by bridgesampling function
prior_list	named list of prior distribution (names correspond to the parameter names)

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.

lines.prior *Add prior object to a plot*

Description

Add prior object to a plot

Usage

```
## 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

Value

lines.prior returns NULL.

See Also

plot.prior() geom_prior()

`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-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
`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()`

`mean.prior` *Prior mean*

Description

Computes mean of a prior distribution.

Usage

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

Arguments

`x`	a prior
`...`	unused

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

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,
  show_figures = 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-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_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,
  ...
)
```

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.

`...`
: 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

Value

`plot_models` 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_posterior *Plot samples from the mixed posterior distributions*

Description

Plot samples from the mixed posterior distributions

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 weightfunction 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.
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
`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-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

`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()`

`point` *Point mass distribution*

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

```
## 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

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

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

```
# 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.

Usage

```
prior(
  distribution,
  parameters,
  truncation = list(lower = -Inf, upper = Inf),
  prior_weights = 1
)

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.

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_functions *Elementary prior related 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) for prior distributions.

Usage

```
## 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, ...)

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

Arguments

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

Value

pdf (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) distri-

bution 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, ...)

ccdf(x, ...)

quant(x, ...)

lpdf(x, ...)

pdf(x, ...)

mcdf(x, ...)

mccdf(x, ...)
```

```
mquant(x, ...)

mlpdf(x, ...)

mpdf(x, ...)
```

Arguments

x	main argument
...	unused arguments

Value

pdf (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'.

The pdf function proceeds to PDF graphics device if x is a character.

---

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. "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_weightfunction` *Creates 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

```
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 function, 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 usually 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

`sd.prior` *Prior sd*

Description

Computes standard deviation of a prior distribution.

Usage

```
## 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

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

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

`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

`var.prior` *Prior var*

Description

Computes variance of a prior distribution.

Usage

```
## 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

```
# 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 *Weight functions*

Description

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

Usage

```
mdone.sided(x, alpha = NULL, alpha1 = NULL, alpha2 = NULL, log = FALSE)

mdtwo.sided(x, alpha, log = FALSE)

mdone.sided_fixed(x, omega, log = FALSE)

mdtwo.sided_fixed(x, omega, log = FALSE)

rone.sided(n, alpha = NULL, alpha1 = NULL, alpha2 = NULL)

rtwo.sided(n, alpha)

rone.sided_fixed(n, omega)

rtwo.sided_fixed(n, omega)

mpone.sided(
  q,
  alpha = NULL,
  alpha1 = NULL,
  alpha2 = 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,
  alpha1 = NULL,
  alpha2 = 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)
```

Arguments

x, q	vector or matrix of quantiles.
alpha	vector or matrix with concentration parameters for the Dirichlet distribution for a monotonic one.sided or a two.sided weight function.
alpha1	vector or matrix with concentration parameters for the Dirichlet distribution for the expected direction of non-monotonic one.sided of weight function.
alpha2	vector or matrix with concentration parameters for the Dirichlet distribution for the unexpected direction of non-monotonic one.sided of weight function.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
omega	vector or matrix of fixed probabilities for a one.sided or a two.sided weight function.
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

mdone.sided, mdtwo.sided, mdone.sided_fixed, and mdtwo.sided_fixed give the marginal density, mpone.sided, mptwo.sided, mpone.sided_fixed, and mptwo.sided_fixed give the marginal distribution function, mqone.sided, mqtwo.sided, mqone.sided_fixed, and mqtwo.sided_fixed give the marginal quantile function, and rone.sided, rtwo.sided, rone.sided_fixed, and rtwo.sided_fixed generate random deviates.

Examples

```
# draw samples from a two-sided weight function
rtwo.sided(10, alpha = c(1, 1))

# draw samples from a monotone one-sided weight function
rone.sided(10, alpha = c(1, 1, 1))

# draw samples from a non-monotone one-sided weight function
rone.sided(10, alpha1 = c(1, 1), alpha2 = c(1, 1))
```

`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