# Package 'BayesTools'

May 6, 2026

**Title** Tools for Bayesian Analyses

**Version** 0.3.0

**Description** Provides tools for conducting Bayesian analyses and Bayesian model averaging (Kass and Raftery, 1995, <doi:10.1080/01621459.1995.10476572>, Hoeting et al., 1999, <doi:10.1214/ss/1009212519>). 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** 7.3.3

**SystemRequirements** JAGS >= 4.3.0 (https://mcmc-jags.sourceforge.io/)

**Depends** R (>= 4.1.0), stats

**Imports** graphics, grid, extraDistr, mvtnorm, coda, bridgesampling, parallel, ggplot2, Rdpack, rlang

**Suggests** scales, testthat (>= 3.3.0), vdiffr, covr, knitr, rstan, rjags, runjags, lme4, BayesFactor, RoBMA, rmarkdown

**RdMacros** Rdpack

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2026-05-06 05:20:02 UTC

# Contents

---

`add_column` *Adds column to BayesTools table*

---

## Description

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

### Usage

```
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 table (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.

---

## as_marginal_inference *Model-average marginal posterior distributions and marginal Bayes factors based on BayesTools JAGS model via* marginal_inference

---

### Description

Creates marginal model-averaged and conditional posterior distributions based on a BayesTools JAGS model, vector of parameters, formula, and a list of conditional specifications for each parameter. Computes inclusion Bayes factors for each marginal estimate via a Savage-Dickey density approximation.

### Usage

```
as_marginal_inference(
  model,
  marginal_parameters,
  parameters,
  conditional_list,
  conditional_rule,
  formula,
  null_hypothesis = 0,
```

```
  normal_approximation = FALSE,
  n_samples = 10000,
  silent = FALSE,
  force_plots = FALSE
)
```

## Arguments

| | |
|---|---|
| `model` | model fit via the JAGS_fit function |
| `marginal_parameters` | parameters for which the the marginal summary should be created |
| `parameters` | all parameters included in the model_list that are relevant for the formula (all of which need to have specification of `is_null_list`) |
| `conditional_list` | list of conditional parameters for each marginal parameter |
| `conditional_rule` | a character string specifying the rule for conditioning. Either "AND" or "OR". Defaults to "AND". |
| `formula` | model formula (needs to be specified if `parameter` was part of a formula) |
| `null_hypothesis` | point null hypothesis to test. Defaults to `0` |
| `normal_approximation` | whether the height of prior and posterior density should be approximated via a normal distribution (rather than kernel density). Defaults to `FALSE`. |
| `n_samples` | controls the numerical grid used for model-averaged prior densities |
| `silent` | whether warnings should be returned silently. Defaults to `FALSE` |
| `force_plots` | temporal argument allowing to generate conditional posterior samples suitable for prior and posterior plots. Only available when conditioning on a single parameter. |

## Value

`as_marginal_inference` returns an object of class 'marginal_inference'.

## See Also

marginal_inference as_mixed_posteriors

`as_mixed_posteriors` *Export BayesTools JAGS model posterior distribution as model-average posterior distributions via* `mix_posteriors`

## Description

Creates a model-averages posterior distributions on a single model that allows mimicking the mix_posteriors functionality. This function is useful when the model-averaged ensemble is based on prior_spike_and_slab or prior_mixture priors - the model-averaging is done within the model.

## Usage

```
as_mixed_posteriors(
  model,
  parameters,
  conditional = NULL,
  conditional_rule = "AND",
  force_plots = FALSE,
  transform_scaled = FALSE,
  n_prior_samples = 10000
)
```

## Arguments

| | |
|---|---|
| `model` | model fit via the JAGS_fit function |
| `parameters` | vector of parameters names for which inference should be drawn |
| `conditional` | a character vector of parameters to be conditioned on |
| `conditional_rule` | a character string specifying the rule for conditioning. Either "AND" or "OR". Defaults to "AND". |
| `force_plots` | temporal argument allowing to generate conditional posterior samples suitable for prior and posterior plots. Only available when conditioning on a single parameter. |
| `transform_scaled` | whether to transform samples from standardized (scaled) to original (unscaled) scale. When `TRUE`, posterior samples are transformed, and the result can be directly passed to plot_posterior which will automatically detect the transformation and use transformed deterministic prior densities. Requires a model fitted with `formula_scale_list`. Defaults to `FALSE`. |
| `n_prior_samples` | controls the numerical grid used for transformed prior densities when `transform_scaled = TRUE`. Defaults to 10000. |

## Value

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

### See Also

mix_posteriors

---

`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 (or marginal_inference in case of marginal_estimates_table).

### Usage

```
ensemble_estimates_table(
  samples,
  parameters,
  probs = c(0.025, 0.975),
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  transform_factors = FALSE,
```

```
  transform_orthonormal = FALSE,
  formula_prefix = TRUE,
  transform_scaled = FALSE,
  formula_scale = NULL
)

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
)

ensemble_estimates_empty_table(
  probs = c(0.025, 0.975),
  title = NULL,
  footnotes = NULL,
  warnings = NULL
)

ensemble_inference_empty_table(title = NULL, footnotes = NULL, warnings = NULL)

ensemble_summary_empty_table(title = NULL, footnotes = NULL, warnings = NULL)
```

```
ensemble_diagnostics_empty_table(
  title = NULL,
  footnotes = NULL,
  warnings = NULL
)

marginal_estimates_table(
  samples,
  inference,
  parameters,
  probs = c(0.025, 0.5, 0.975),
  logBF = FALSE,
  BF01 = FALSE,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  formula_prefix = TRUE,
  transform_scaled = FALSE,
  formula_scale = NULL
)
```

## 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_factors | whether factors with orthonormal/meandif prior distribution should be transformed to differences from the grand mean |
| transform_orthonormal | (to be depreciated) 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. |
| transform_scaled | whether coefficients from standardized continuous predictors should be transformed back to the original scale. Defaults to FALSE. |
| formula_scale | named list containing standardization information (mean and sd) for each standardized predictor. Required when transform_scaled = TRUE for ensemble/marginal tables. For runjags_estimates_table, this is automatically extracted from the fit object's formula_scale attribute. |

| | |
|---|---|
| `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,
  formula_prefix = TRUE,
  remove_parameters = NULL
)
```

```
runjags_estimates_table(
  fit,
  transformations = NULL,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  conditional = FALSE,
  probs = c(0.025, 0.5, 0.975),
  remove_spike_0 = TRUE,
  transform_factors = FALSE,
  transform_orthonormal = FALSE,
  formula_prefix = TRUE,
  remove_inclusion = FALSE,
  remove_parameters = NULL,
  remove_formulas = NULL,
  keep_parameters = NULL,
  keep_formulas = NULL,
  return_samples = FALSE,
  transform_scaled = FALSE,
  remove_diagnostics = FALSE,
 diagnostic_columns = getOption("BayesTools.JAGS_estimates_diagnostic_columns", if
    (remove_diagnostics) "none" else "all")
)

runjags_inference_table(
  fit,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  formula_prefix = TRUE,
  logBF = FALSE,
  BF01 = FALSE,
  BF_diagnostics = FALSE,
  BF_diagnostic_columns = getOption("BayesTools.JAGS_BF_diagnostic_columns", if
    (BF_diagnostics) "all" else "none")
)

JAGS_estimates_table(
  fit,
  transformations = NULL,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  conditional = FALSE,
  probs = c(0.025, 0.5, 0.975),
  remove_spike_0 = TRUE,
  transform_factors = FALSE,
```

```
  transform_orthonormal = FALSE,
  formula_prefix = TRUE,
  remove_inclusion = FALSE,
  remove_parameters = NULL,
  remove_formulas = NULL,
  keep_parameters = NULL,
  keep_formulas = NULL,
  return_samples = FALSE,
  transform_scaled = FALSE,
  remove_diagnostics = FALSE,
 diagnostic_columns = getOption("BayesTools.JAGS_estimates_diagnostic_columns", if
    (remove_diagnostics) "none" else "all")
)

JAGS_inference_table(
  fit,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  formula_prefix = TRUE,
  logBF = FALSE,
  BF01 = FALSE,
  BF_diagnostics = FALSE,
  BF_diagnostic_columns = getOption("BayesTools.JAGS_BF_diagnostic_columns", if
    (BF_diagnostics) "all" else "none")
)

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

model_summary_empty_table(
  model_description = NULL,
  title = NULL,
  footnotes = NULL,
  warnings = NULL
)

runjags_estimates_empty_table(
  probs = c(0.025, 0.5, 0.975),
```

```
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  remove_diagnostics = FALSE,
 diagnostic_columns = getOption("BayesTools.JAGS_estimates_diagnostic_columns", if
    (remove_diagnostics) "none" else "all")
)

runjags_inference_empty_table(
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  logBF = FALSE,
  BF01 = FALSE,
  BF_diagnostics = FALSE,
  BF_diagnostic_columns = getOption("BayesTools.JAGS_BF_diagnostic_columns", if
    (BF_diagnostics) "all" else "none")
)

JAGS_estimates_empty_table(
  probs = c(0.025, 0.5, 0.975),
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  remove_diagnostics = FALSE,
 diagnostic_columns = getOption("BayesTools.JAGS_estimates_diagnostic_columns", if
    (remove_diagnostics) "none" else "all")
)

JAGS_inference_empty_table(
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  logBF = FALSE,
  BF01 = FALSE,
  BF_diagnostics = FALSE,
  BF_diagnostic_columns = getOption("BayesTools.JAGS_BF_diagnostic_columns", if
    (BF_diagnostics) "all" else "none")
)

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. |
| remove_parameters | parameters to be removed from the summary. Can be NULL (default, no removal), a character vector of parameter names to remove, or TRUE to remove all parameters that are not part of any formula. |
| 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. |
| probs | quantiles for parameter estimates |
| transform_factors | whether factors with orthonormal/meandif prior distribution should be transformed to differences from the grand mean |
| transform_orthonormal | (to be depreciated) 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_formulas | character vector of formula names whose parameters should be removed from the summary. Defaults to NULL. |
| keep_parameters | character vector of parameter names to keep. All other parameters will be removed unless they belong to formulas specified in keep_formulas. Defaults to NULL. |
| keep_formulas | character vector of formula names whose parameters should be kept. All other parameters will be removed unless they are specified in keep_parameters. Defaults to NULL. |

`return_samples` whether to return the transoformed and formated samples instead of the table. Defaults to `FALSE`.

`transform_scaled` whether coefficients from standardized continuous predictors should be transformed back to the original scale. Defaults to `FALSE`.

`remove_diagnostics` whether to exclude MCMC diagnostics (MCMC error, ESS, R-hat) from the output table. Defaults to `FALSE`. Setting to `TRUE` will exclude diagnostics columns regardless of the `conditional` setting.

`diagnostic_columns` MCMC diagnostic columns to display in JAGS estimates tables. Can be `"all"`, `"none"`, `TRUE`, `FALSE`, or a character vector containing any subset of `"MCMC_error"`, `"MCMC_SD_error"`, `"ESS"`, and `"R_hat"`. Defaults to the `BayesTools.JAGS_estimates_diagnostic_co` option, or all diagnostics unless `remove_diagnostics = TRUE`.

`logBF` whether the Bayes factor should be on log scale

`BF01` whether the Bayes factor should be inverted

`BF_diagnostics` whether to add MCMC diagnostics for Bayes factors computed from model indicator frequencies. The Bayes factor error is reported as a relative Monte Carlo standard error percentage. Defaults to `FALSE`.

`BF_diagnostic_columns` MCMC diagnostic columns to display in JAGS inclusion Bayes factor tables. Can be `"all"`, `"none"`, `TRUE`, `FALSE`, or a character vector containing any subset of `"ESS"`, `"MCMC_error"`, and `"BF_error_percent"`. Defaults to the `BayesTools.JAGS_BF_diagnostic` option, or all diagnostics when `BF_diagnostics = TRUE` and none otherwise.

## Details

For product-space JAGS inclusion Bayes factors, posterior inclusion probabilities of exactly 0 or 1 cannot produce a finite point estimate of the model odds ratio. In that case, inclusion BFs marked with `"<"` or `">"` are finite-sample bounds: posterior inclusion probabilities of 0 or 1 were replaced by `1/S` or `(S - 1)/S`, where `S` is the number of posterior samples. This is a reporting convention, not an unbiased finite Bayes-factor estimate. If the prior inclusion probability is exactly 0 or 1, the inclusion Bayes factor is undefined and reported as `NA`, because the corresponding inclusion/exclusion comparison was not tested.

## 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

## bridgesampling_object *Create a 'bridgesampling' object*

### Description

prepares a 'bridgesampling' object with a given log marginal likelihood.

### Usage

```
bridgesampling_object(logml = -Inf)
```

### Arguments

| | |
|---|---|
| logml | log marginal likelihood. Defaults to -Inf. |

### Value

JAGS_bridgesampling returns an object of class 'bridge'.

## 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 = deparse(substitute(x)),
  check_length = 1,
  allow_NULL = FALSE,
  allow_NA = TRUE,
  call = ""
)

check_char(
  x,
  name = deparse(substitute(x)),
  check_length = 1,
  allow_values = NULL,
  allow_NULL = FALSE,
  allow_NA = TRUE,
  call = ""
)
```

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

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

check_list(
  x,
  name = deparse(substitute(x)),
  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. |
| allow_NA | whether the object can contain NA or NaN values. |
| 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.BayesTools *BayesTools Contrast Matrices*

---

## Description

BayesTools provides several contrast matrix functions for Bayesian factor analysis. These functions create different types of contrast matrices that can be used with factor variables in Bayesian models.

## Usage

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

contr.meandif(n, contrasts = TRUE)

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

## Arguments

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

## Details

The package includes the following contrast functions:

`contr.orthonormal` Return a matrix of orthonormal contrasts. Code is based on `stanova::contr.bayes` and corresponding to description by Rouder et al. (2012). Returns a matrix with n rows and k columns, with k = n - 1 if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

`contr.meandif` Return a matrix of mean difference contrasts. This is an adjustment to the `contr.orthonormal` that ascertains that the prior distributions on difference between the gran mean and factor level are identical independent of the number of factor levels (which does not hold for the orthonormal contrast). Furthermore, the contrast is re-scaled so the specified prior distribution exactly corresponds to the prior distribution on difference between each factor level and the grand mean – this is approximately twice the scale of `contr.orthonormal`. Returns a matrix with n rows and k columns, with k = n - 1 if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

`contr.independent` Return a matrix of independent contrasts – a level for each term. Returns a matrix with n rows and k columns, with k = n if `contrasts` = `TRUE` and k = n if `contrasts` = `FALSE`.

## References

Rouder JN, Morey RD, Speckman PL, Province JM (2012). "Default Bayes factors for ANOVA designs." *Journal of Mathematical Psychology*, **56**(5), 356–374. doi:10.1016/j.jmp.2012.08.001.

## Examples

```
# Orthonormal contrasts
contr.orthonormal(c(1, 2))
contr.orthonormal(c(1, 2, 3))

# Mean difference contrasts
contr.meandif(c(1, 2))
contr.meandif(c(1, 2, 3))

# Independent contrasts
contr.independent(c(1, 2))
contr.independent(c(1, 2, 3))
```

---

# `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:<br>**lin** linear transformation in form of a + b*x<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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, 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.

---

# formula_add_intercept *Add an Intercept to a Formula*

---

## Description

Converts a no-intercept formula to the corresponding formula with an intercept while preserving the formula environment. Top-level no-intercept encodings such as - 1, + 0, and 0 + are removed without editing transformed calls such as I(x - 1) or offset(x - 1).

## Usage

```
formula_add_intercept(formula)
```

### Arguments

| | |
|---|---|
| formula | a formula object. |

### Value

A formula object with an intercept.

---

`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** hyperbolic tangent transformation

**exp** exponential transformation

, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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,
  individual = FALSE,
  show_figures = if (individual) 1 else NULL,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  rescale_x = FALSE,
  scale_y2 = NULL,
  prior_list_mu = NULL,
  effect_direction = "positive",
  ...
)
```

## 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 |
| individual | should individual densities be returned (e.g., in case of weightfunction) |
| 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. |
| transformation | transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:<br>**lin** linear transformation in form of a + b*x<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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 |
| effect_direction | direction of the effect for PET-PEESE regression. Use "positive" (default) for mu + PET*se + PEESE*se^2 or "negative" for mu - PET*se - PEESE*se^2. |
| ... | 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 model probabilities, posterior model probabilities (or marginal likelihoods), and indicator whether the models represent the null or alternative hypothesis.

## Usage

```
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. If the prior probability of either the null or alternative hypothesis is zero, the Bayes factor is undefined and NA is returned.

## 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)

interpret2(specification, method = NULL)
```

## 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"`. Finite-sample BF bounds can be supplied either as a `bound_operator` attribute on the numeric BF or as `inference_BF_bound_operator` / `BF_bound_operator` with value `"<"` or `">"`. (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

`interpret_records` *Normalize structured interpretation sources*

## Description

`interpret_records` normalizes interpretation inputs from one or more structured sources into traceable evidence, estimate, header, note, prior, and footnote records. It is intended for downstream packages that already own the domain-specific ordering and wording of an interpretation, but want BayesTools to consistently extract table rows, preserve Bayes-factor metadata, and carry traceability information.

## Usage

```
interpret_records(
  sources,
  plan,
  output = c("records", "text"),
  missing = c("error", "skip", "warn"),
  method = NULL,
  digits = 3
)

interpret_tables(sources, spec, ...)
```

## Arguments

| | |
|---|---|
| `sources` | named list of interpretation sources. Each source can be a `data.frame`/`BayesTools_table`, or a list with entries `data`, optional `type`, and optional per-source `schema`. Supported source types are `"table"`, `"record"`, and `"records"`. Table schemas can define columns such as `row`, `BF`, `central`, `lower`, `upper`, and metadata such as `BF_orientation`, `BF_scale`, `BF_name`, `BF_bound_operator`, `lower_prob`, `upper_prob`, `interval_level`, `units`, and `conditioning`. |
| `plan` | ordered list specifying which records to create. Plan items can have `kind = "evidence"`, `"estimate"`, `"pair"`, `"test_estimate"`, `"for_each"`, `"header"`, `"note"`, `"prior"`, or `"footnote"`. Pair items contain `evidence` and/or `estimate` references. Dynamic `"for_each"` items generate records from source row order. |
| `output` | whether to return normalized `"records"` or a simple generic `"text"` rendering. Downstream packages should usually request `"records"` and render domain-specific text themselves. |
| `missing` | how missing optional sources or rows should be handled. |
| `method` | optional method name used only by the generic text renderer. |
| `digits` | number of digits used only by the generic text renderer. |
| `spec` | alias for `plan` used by `interpret_tables`. |
| `...` | additional arguments passed from `interpret_tables` to `interpret_records`. |

## Value

interpret_records returns a data frame with class "BayesTools_interpret_records" when output = "records", or a character vector when output = "text". interpret_tables is a convenience wrapper around interpret_records.

---

# 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.treatment(x)

is.prior.independent(x)

is.prior.spike_and_slab(x)

is.prior.meandif(x)

is.prior.mixture(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)
```

---

## is_prior_bias *Reports whether x is a composed publication-bias prior*

---

### Description

Reports whether x is a composed publication-bias prior

### Usage

```
is_prior_bias(x)
```

### Arguments

| | |
|---|---|
| x | object to test. |

### Value

A logical value.

## is_prior_phacking *Reports whether x is a p-hacking prior*

### Description

Reports whether x is a p-hacking prior

### Usage

```
is_prior_phacking(x)
```

### Arguments

| | |
|---|---|
| x | object to test. |

### Value

A logical value.

## 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,
  log_posterior,
  data = NULL,
  prior_list = NULL,
  formula_list = NULL,
  formula_data_list = NULL,
  formula_prior_list = NULL,
  formula_scale_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) |
| `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`

`formula_scale_list`
: named list of named lists for standardizing continuous predictors (names of the lists correspond to the parameter name created by each of the formula). Each entry should be a named list where continuous predictors with `TRUE` values will be standardized. Defaults to `NULL` (no standardization).

`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)
```

---

`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) 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.

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

**max_extend** number of times after which the automatic fitting function is stopped.

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

**restarts** number of times new initial values should be generated in case the model fails to initialize. Defaults to 10.

**check_indicators** whether model indicator variables should be included in convergence checks. Defaults to FALSE.

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,
  add_parameters = NULL,
  fail_fast = FALSE,
  check_indicators = FALSE
)
```

## 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 0.01. |
| max_SD_error | maximum MCMC error as the proportion of standard deviation of the parameters. Defaults to 0.05. |
| add_parameters | vector of additional parameter names that should be used (only allows removing last, fixed, omega element if omega is tracked manually). |
| fail_fast | whether the function should stop after the first failed convergence check. |
| check_indicators | whether model indicator variables should be included in convergence checks. Defaults to FALSE. |

## 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_factors = FALSE,
  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_factors = FALSE,
  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_factors = FALSE,
  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_factors = FALSE,
  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_factors | whether factors with orthonormal/meandif prior distribution should be transformed to differences from the grand mean |
| transform_orthonormal | (to be depreciated) 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()

---

# JAGS_evaluate_formula *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). If the formula has a "log(intercept)" attribute set to TRUE, the intercept values will be log-transformed before computing the linear predictor. |
| 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()

---

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 = NULL,
  prior_list = NULL,
  formula_list = NULL,
  formula_data_list = NULL,
  formula_prior_list = NULL,
  formula_scale_list = NULL,
  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, restarts = 10,
    max_extend = 10, check_indicators = FALSE),
  parallel = FALSE,
  cores = chains,
  silent = TRUE,
  seed = NULL,
  add_parameters = NULL,
  required_packages = NULL,
  ...
)

JAGS_extend(
  fit,
 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, restarts = 10,
```

```
    max_extend = 10, check_indicators = FALSE),
  parallel = FALSE,
  cores = NULL,
  silent = TRUE,
  seed = 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 |
| formula_scale_list | named list of named lists for standardizing continuous predictors (names of the lists correspond to the parameter name created by each of the formula). Each entry should be a named list where continuous predictors with TRUE values will be standardized. Defaults to NULL (no standardization). |
| 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:<br>**max_Rhat** maximum R-hat error for the autofit function. Defaults to 1.05.<br>**min_ESS** minimum effective sample size. Defaults to 500.<br>**max_error** maximum MCMC error. Defaults to 0.01.<br>**max_SD_error** maximum MCMC error as the proportion of standard deviation of the parameters. Defaults to 0.05.<br>**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. |

**max_extend** number of times after which the automatic fitting function is stopped.

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

**restarts** number of times new initial values should be generated in case the model fails to initialize. Defaults to `10`.

**check_indicators** whether model indicator variables should be included in convergence checks. 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

`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`.

`...` additional hidden arguments

`fit` a 'BayesTools_fit' object (created by `JAGS_fit()` function) to be extended

## Value

`JAGS_fit` returns an object of class 'runjags' and 'BayesTools_fit'.

## 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

```
JAGS_formula(formula, parameter, data, prior_list, formula_scale = NULL)
```

## Arguments

| | |
|---|---|
| formula | formula specifying the right hand side of the assignment (the left hand side is ignored). If the formula contains -1, it will be automatically converted to include an intercept with a spike(0) prior. The formula can also have a "log(intercept)" attribute set to TRUE to generate syntax of the form log(intercept) + sum(beta_i * x_i), which is useful for parameters that must be positive (e.g., standard deviation). |
| 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. When using -1 in the formula, an "intercept" prior can be explicitly specified; otherwise, prior("spike", list(0)) is automatically added. The list can also include two special entries: |
| formula_scale | named list specifying whether to standardize continuous predictors. If NULL (default), no standardization is applied. If a named list is provided, continuous predictors with TRUE values will be standardized (mean-centered and scaled by standard deviation). The intercept is never standardized.<br>"__default_continuous" A prior to use for any continuous predictors (including the intercept) that are not explicitly specified in the prior list.<br>"__default_factor" A prior to use for any factor predictors (including interactions involving factors) that are not explicitly specified in the prior list.<br>These default priors allow for more concise specification when many predictors share the same prior distribution. |

## Details

When a formula with `-1` (no intercept) is specified, the function automatically removes the `-1`, adds an intercept back to the formula, and includes a spike(0) prior for the intercept to ensure equivalent model behavior while maintaining consistent formula parsing.

When using default priors (`"__default_continuous"` or `"__default_factor"`), explicitly specified priors for individual terms take precedence over the defaults. The defaults are only applied to terms that are not already in the prior list.

## Value

`JAGS_formula` returns a list containing the formula JAGS syntax, JAGS data object, modified prior_list, and (if standardization was applied) a `formula_scale` list with standardization information for back-transformation.

## See Also

`JAGS_fit()`

## Examples

```
# 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 with intercept
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_obj <- JAGS_formula(
  formula = ~ x_cont + x_fac3 * x_fac4,
  parameter = "mu", data = df, prior_list = prior_list)

# using -1 notation (automatically adds spike(0) intercept)
prior_list_no_intercept <- list(
  "x_fac3" = prior_factor("normal", list(0, 1), contrast = "treatment")
)
formula_no_intercept <- JAGS_formula(
  formula = ~ x_fac3 - 1,
  parameter = "mu", data = df, prior_list = prior_list_no_intercept)
```

```
# Equivalent to specifying intercept = prior("spike", list(0))

# using default priors for simpler specification
prior_list_defaults <- list(
  "__default_continuous" = prior("normal", list(0, 1)),
  "__default_factor"     = prior_factor("normal", list(0, 0.5), contrast = "treatment")
)
formula_defaults <- JAGS_formula(
  formula = ~ x_cont + x_fac3,
  parameter = "mu", data = df, prior_list = prior_list_defaults)
# intercept and x_cont get the default continuous prior
# x_fac3 gets the default factor prior
```

---

`JAGS_formula_design` *Extract Fitted JAGS Formula Design Metadata*

---

## Description

Returns the fitted formula design metadata stored by `JAGS_fit()`. The design contains the processed formula, fitted model frame, exact model matrix used for JAGS data construction, JAGS-safe coefficient names, contrast and factor-level metadata, rank diagnostics, prior metadata, and formula-scale information.

## Usage

```
JAGS_formula_design(fit, parameter = NULL)
```

## Arguments

| | |
|---|---|
| `fit` | a fitted object returned by `JAGS_fit()`. |
| `parameter` | optional formula parameter name. If `NULL`, all stored formula designs are returned. |

## Value

A named list of formula designs, or one formula design when `parameter` is supplied. Returns `NULL` when no formula design metadata is stored on `fit`.

## See Also

`JAGS_fit()` `JAGS_formula()`

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_list,
  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_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 |
| 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

Wagenmakers E, Beek TF, Rotteveel M, Gierholz A, Matzke D, Steingroever H, Ly A, Verhagen J, Selker R, Sasiadek A, others (2015). "Turning the hands of time again: a purely confirmatory replication study and a Bayesian analysis." *Frontiers in Psychology*, **6**, 1–6. doi:10.3389/fpsyg.2015.00494.

---

`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** hyperbolic tangent transformation

**exp** exponential transformation

, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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,
  individual = FALSE,
  show_figures = if (individual) 1 else NULL,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  rescale_x = FALSE,
  scale_y2 = NULL,
  prior_list_mu = NULL,
  effect_direction = "positive",
  ...
)
```

## 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 |
| individual | should individual densities be returned (e.g., in case of weightfunction) |
| 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. |
| transformation | transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:<br>**lin** linear transformation in form of a + b*x<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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

effect_direction
direction of the effect for PET-PEESE regression. Use "positive" (default) for mu + PET*se + PEESE*se^2 or "negative" for mu - PET*se - PEESE*se^2.

... additional arguments

## Value

lines_prior_list returns NULL.

## See Also

plot_prior_list() geom_prior_list()

---

# marginal_inference *Model-average marginal posterior distributions and marginal Bayes factors*

---

## Description

Creates marginal model-averaged and conditional posterior distributions based on a list of models, vector of parameters, formula, and a list of indicators of the null or alternative hypothesis models for each parameter. Computes inclusion Bayes factors for each marginal estimate via a Savage-Dickey density approximation.

## Usage

```
marginal_inference(
  model_list,
  marginal_parameters,
  parameters,
  is_null_list,
  formula,
  null_hypothesis = 0,
  normal_approximation = FALSE,
  n_samples = 10000,
  seed = NULL,
  silent = FALSE
)
```

## Arguments

model_list list of models, each of which contains marginal likelihood estimated with bridge sampling marglik and prior model odds prior_weights

marginal_parameters
parameters for which the the marginal summary should be created

| | |
|---|---|
| parameters | all parameters included in the model_list that are relevant for the formula (all of which need to have specification of is_null_list) |
| 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) |
| formula | model formula (needs to be specified if parameter was part of a formula) |
| null_hypothesis | point null hypothesis to test. Defaults to 0 |
| normal_approximation | whether the height of prior and posterior density should be approximated via a normal distribution (rather than kernel density). Defaults to FALSE. |
| n_samples | controls the numerical grid used for model-averaged prior densities |
| seed | seed for random number generation |
| silent | whether warnings should be returned silently. Defaults to FALSE |

## Value

marginal_inference returns an object of class 'marginal_inference'.

## See Also

ensemble_inference mix_posteriors BayesTools_ensemble_tables

---

# marginal_posterior *Model-average marginal posterior distributions*

---

## Description

Creates marginal model-averages posterior distributions for a given parameter based on model-averaged posterior samples and parameter name (and formula with at specification).

## Usage

```
marginal_posterior(
  samples,
  parameter,
  formula = NULL,
  at = NULL,
  prior_samples = FALSE,
  use_formula = TRUE,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  n_samples = 10000,
  ...
)
```

## Arguments

| | |
|---|---|
| samples | model-averaged posterior samples created by mix_posteriors() |
| parameter | parameter of interest |
| formula | model formula (needs to be specified if parameter was part of a formula) |
| at | named list with predictor levels of the formula for which marginalization should be performed. If a predictor level is missing, 0 is used for continuous predictors, the baseline factor level is used for factors with contrast = "treatment" prior distributions, and the parameter is completely omitted for for factors with contrast = "meandif", |
| prior_samples | whether marginal prior distributions should be generated contrast = "orthonormal", and contrast = "independent" levels |
| use_formula | whether the parameter should be evaluated as a part of supplied formula |
| transformation | transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:<br>**lin** linear transformation in form of a + b*x<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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 |
| n_samples | controls the numerical grid used for model-averaged prior densities |
| ... | additional arguments |

## Value

marginal_posterior returns a named list of mixed marginal posterior distributions (either a vector of matrix). #'

---

# mean.prior *Prior mean*

---

## Description

Computes mean of a prior distribution. (In case of orthonormal prior distributions for factors, the mean of for the deviations from intercept is returned.)

## 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 of the null or alternative hypothesis models 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 NULL. |
| 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 as_mixed_posteriors

---

## mpoint *Multivariate point mass distribution*

---

### Description

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

### Usage

```
dmpoint(x, location, log = FALSE)

rmpoint(n, location)

pmpoint(q, location, lower.tail = TRUE, log.p = FALSE)

qmpoint(p, location, lower.tail = TRUE, log.p = FALSE)
```

### Arguments

| | |
|---|---|
| x, q | vector or matrix of quantiles. |
| location | vector of locations corresponding to the location of individual points. Alternatively, a matrix with rows corresponding to the location of individual samples and columns correspond to the location of individual points. |

| | |
|---|---|
| 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, $P[X > x]$. |
| 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 multivariate point distribution
rmpoint(10, location = c(0, 1))
```

---

# parameter_names *Clean parameter names from JAGS*

---

## Description

Removes additional formatting from parameter names outputted from JAGS.

## Usage

```
format_parameter_names(
  parameters,
  formula_parameters = NULL,
  formula_random = NULL,
  formula_prefix = TRUE,
  formula_scale = NULL
)

JAGS_parameter_names(parameters, formula_parameter = NULL)
```

## Arguments

| | |
|---|---|
| parameters | a vector of parameter names |
| formula_parameters | a vector of formula parameter prefix names |
| formula_random | a vector of random effects grouping factors |
| formula_prefix | whether the formula_parameters names should be kept. Defaults to TRUE. |
| formula_scale | optional nested list containing scaling info. When provided, intercepts from parameters with log_intercept = TRUE attribute will be renamed to exp(intercept). |
| 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")
```

---

`phack_pi_null` *P-hacking calibration helpers*

---

## Description

`phack_pi_null()` converts the sampled severity `alpha` to the amount of null probability mass depleted from the source interval. `phack_alpha_from_pi_null()` applies the inverse calibration. `phack_backend_constants()` returns the z-scale constants used by the backend.

## Usage

```
phack_pi_null(alpha, form, source, destination, target = 0.025)

phack_alpha_from_pi_null(pi_null, form, source, destination, target = 0.025)

phack_backend_constants(form, source, destination, target = 0.025)
```

## Arguments

| | |
|---|---|
| `alpha` | p-hacking severity parameter. |
| `form` | power depletion form, either `"linear"` or `"quadratic"`. |
| `source` | source p-value cut point. |
| `destination` | destination p-value cut point. |
| `target` | target p-value cut point. |
| `pi_null` | null probability mass depleted from the source interval. |

## Value

Numeric vector for the calibration helpers and a named list for `phack_backend_constants()`.

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** hyperbolic tangent transformation

**exp** exponential transformation

, or a list containing the transformation function `fun`, inverse transformation function `inv`, and derivative of the transformation `jac`, evaluated on the original support. 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_marginal *Plot samples from the marginal posterior distributions*

## Description

Plot samples from the marginal posterior distributions

## Usage

```
plot_marginal(
  samples,
  parameter,
  plot_type = "base",
  prior = FALSE,
  n_points = 1000,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  rescale_x = FALSE,
  par_name = NULL,
  dots_prior = list(),
  legend = TRUE,
  legend_title = NULL,
  legend_labels = NULL,
  legend_position = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| samples | samples from a posterior distribution for a parameter generated by marginal_inference. |
| parameter | parameter name to be plotted. |
| plot_type | whether to use a base plot "base" or ggplot2 "ggplot" for plotting. |
| prior | whether prior distribution should be added to the figure. When samples were prepared with as_mixed_posteriors(..., transform_scaled = TRUE), the transformed prior samples are automatically used. |
| n_points | number of equally spaced points in the x_range if x_seq is unspecified |
| transformation | transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:<br>**lin** linear transformation in form of a + b*x<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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

legend
: whether factor legends should be drawn.

legend_title
: optional title for factor legends.

legend_labels
: optional labels for factor legend levels.

legend_position
: optional legend position for factor legends.

...
: additional arguments

## Value

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

## See Also

prior() marginal_inference() plot_posterior()

---

# 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 or as_mixed_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. When samples were prepared with `as_mixed_posteriors(..., transform_scaled = TRUE)`, the transformed prior samples are automatically used. |
| 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:<br>**lin** linear transformation in form of `a + b*x`<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function `fun`, inverse transformation function `inv`, and derivative of the transformation `jac`, evaluated on the original support. 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.:<br>`"show_updating"` whether Bayes factors and change from prior to posterior odds should be shown on the secondary y-axis<br>`"show_estimates"` whether posterior estimates and 95% CI should be shown on the secondary y-axis<br>`"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/meandif 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_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,
  individual = FALSE,
  show_figures = NULL,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  rescale_x = FALSE,
  par_name = NULL,
  effect_direction = "positive",
  dots_prior = list(),
  data = NULL,
  show_data = FALSE,
  dots_data = list(),
  legend = TRUE,
  legend_title = NULL,
  legend_labels = NULL,
  legend_position = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| samples | samples from a posterior distribution for a parameter generated by mix_posteriors or as_mixed_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. When samples were prepared with as_mixed_posteriors(..., transform_scaled = TRUE), the transformed prior samples are automatically used. |
| 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) |
| 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. |
| transformation | transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:<br>**lin** linear transformation in form of a + b*x<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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. |
| effect_direction | direction of the effect for PET-PEESE regression. Use "positive" (default) for mu + PET*se + PEESE*se^2 or "negative" for mu - PET*se - PEESE*se^2. |
| dots_prior | additional arguments for the prior distribution plot |
| data | optional numeric vector of observed p-values in [0, 1], or a data frame with a p column. Used only for weightfunction plots. |
| show_data | whether observed p-values should be shown as rug marks on the weightfunction x-axis. |

| | |
|---|---|
| dots_data | additional styling arguments for observed p-value rug marks. Supports col/color, alpha, lwd/linewidth, side/rug_side, and height/rug_height. |
| legend | whether factor legends should be drawn. |
| legend_title | optional title for factor legends. |
| legend_labels | optional labels for factor legend levels. |
| legend_position | optional legend position for factor legends. |
| ... | additional arguments |

## Details

When using scaled predictors (via formula_scale_list in JAGS_fit), you can plot posteriors on the original (unscaled) scale by preparing samples with as_mixed_posteriors(..., transform_scaled = TRUE). The function automatically detects this and uses the pre-computed transformed prior samples when prior = TRUE.

## 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,
  individual = FALSE,
  show_figures = if (individual) 1 else NULL,
  transformation = NULL,
```

```
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  rescale_x = FALSE,
  par_name = NULL,
  prior_list_mu = NULL,
  effect_direction = "positive",
  legend = TRUE,
  legend_title = NULL,
  legend_labels = NULL,
  legend_position = 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 |
| individual | should individual densities be returned (e.g., in case of weightfunction) |
| 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. |
| transformation | transformation to be applied to the prior distribution. Either a character specifying one of the prepared transformations:<br>**lin** linear transformation in form of a + b*x<br>**tanh** hyperbolic tangent transformation<br>**exp** exponential transformation<br>, or a list containing the transformation function fun, inverse transformation function inv, and derivative of the transformation jac, evaluated on the original support. 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 |
| effect_direction | direction of the effect for PET-PEESE regression. Use "positive" (default) for mu + PET*se + PEESE*se^2 or "negative" for mu - PET*se - PEESE*se^2. |
| legend | whether factor legends should be drawn. |
| legend_title | optional title for factor legends. |
| legend_labels | optional labels for factor legend levels. |
| legend_position | optional legend position for factor legends. |
| ... | 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()

---

# plot_transformed_prior

*Plot Transformed Prior Densities*

---

## Description

Plots a prior density after applying the same deterministic coefficient transformation used for formula-scale back-transforms. The helper supports continuous densities and point-mass priors in the same plot.

## Usage

```
plot_transformed_prior(
  prior_list,
  column_names,
  formula_scale = NULL,
  parameter,
  n_points = 1000,
  x_range = NULL,
  transformation = NULL,
  transformation_arguments = NULL,
  transformation_settings = FALSE,
  plot_type = c("base", "ggplot"),
  par_name = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| `prior_list` | named list of prior distributions. |
| `column_names` | character vector of coefficient column names defining the fitted parameter space. |
| `formula_scale` | optional nested formula-scale metadata, in the same shape as `attr(fit, "formula_scale")`. |
| `parameter` | coefficient name to plot. |
| `n_points` | number of plotting points. |
| `x_range` | optional plotting range on the untransformed scale. |
| `transformation` | optional output transformation passed to the prior plotting machinery. |
| `transformation_arguments` | optional list of transformation arguments. |
| `transformation_settings` | whether `x_range` is supplied on the transformed scale. |
| `plot_type` | either `"base"` or `"ggplot"`. |
| `par_name` | optional plotting label. |
| `...` | additional plotting arguments. |

### Value

For `plot_type = "base"`, invisibly returns base-plot metadata. For `plot_type = "ggplot"`, returns a `ggplot` object. Returns `NULL` when the requested parameter is an identity transform and no output transformation was requested.

### See Also

`plot_prior_list()` `transform_scale_samples()`

---

## `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, $P[X > x]$. |
| 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<br>`"point"` for a point density characterized by a `location` parameter.<br>`"normal"` for a normal distribution characterized by a `mean` and `sd` parameters.<br>`"lognormal"` for a lognormal distribution characterized by a `meanlog` and `sdlog` parameters.<br>`"cauchy"` for a Cauchy distribution characterized by a `location` and `scale` parameters. Internally converted into a generalized t-distribution with `df = 1`.<br>`"t"` for a generalized t-distribution characterized by a `location`, `scale`, and `df` parameters.<br>`"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_bias *Creates a composed publication-bias prior*

---

## Description

prior_bias() composes a step-selection prior_weightfunction() and/or a prior_phacking() object. It does not introduce new math; the backend compiler treats the composition as the product of both kernels.

### Usage

```
prior_bias(selection = NULL, phacking = NULL, prior_weights = 1)
```

### Arguments

| | |
|---|---|
| selection | optional prior_weightfunction() object. |
| phacking | optional prior_phacking() object. |
| prior_weights | prior odds associated with a given distribution. |

### Value

prior_bias() returns an object of class "prior".

---

## 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 = "meandif"
)
```

### Arguments

| | |
|---|---|
| distribution | name of the prior distribution. The possible options are<br>"point" for a point density characterized by a location parameter.<br>"normal" for a normal distribution characterized by a mean and sd parameters.<br>"lognormal" for a lognormal distribution characterized by a meanlog and sdlog parameters.<br>"cauchy" for a Cauchy distribution characterized by a location and scale parameters. Internally converted into a generalized t-distribution with df = 1.<br>"t" for a generalized t-distribution characterized by a location, scale, and df parameters.<br>"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

"meandif" for contrast centered around the grand mean with equal marginal distributions, making the prior distribution exchangeable across factor levels. In contrast to "orthonormal", the marginal distributions are identical regardless of the number of factor levels and the specified prior distribution corresponds to the difference from grand mean for each factor level. Only supports distribution = "mnormal" and distribution = "mt" which generates the corresponding multivariate normal/t distributions.

"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.

"independent" for contrasts specifying dependent prior distribution for each factor level (note that this leads to an overparameterized model if the intercept is included).

## Value

return an object of class 'prior'.

## See Also

prior()

## Examples

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

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) 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, ...)
```

```
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_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<br>`"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),<br>`"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).<br>For medicine, the possible options are based on Bartoš et al. (2021) and Bartoš et al. (2023) who developed empirical prior distributions for the effect size and heterogeneity parameters of the continuous outcomes (standardized mean differences), dichotomous outcomes (logOR, logRR, and risk differences), and time to event outcomes (logHR) 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 available subfields and set the appropriate `parameter` and `type`; availability of subfield-specific priors depends on the selected `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 with the following options<br>`"smd"` for standardized mean differences<br>`"logOR"` for log odds ratios<br>`"logRR"` for log risk ratios<br>`"RD"` for risk differences<br>`"logHR"` for log hazard ratios |

## Value

`prior_informed` returns an object of class 'prior'.

## References

Bartoš F, Gronau QF, Timmers B, Otte WM, Ly A, Wagenmakers E (2021). "Bayesian model-averaged meta-analysis in medicine." *Statistics in Medicine*, **40**(30), 6743–6761. doi:10.1002/sim.9170.

Bartoš F, Otte WM, Gronau QF, Timmers B, Ly A, Wagenmakers E (2023). "Empirical prior distributions for Bayesian meta-analyses of binary and time-to-event outcomes." doi:10.48550/arXiv.2306.11468. preprint at https://doi.org/10.48550/arXiv.2306.11468.

Gronau QF, Van Erp S, Heck DW, Cesario J, Jonas KJ, Wagenmakers E (2017). "A Bayesian model-averaged meta-analysis of the power pose effect with informed and default priors: The

case of felt power." *Comprehensive Results in Social Psychology*, **2**(1), 123–138. doi:10.1080/23743603.2017.1326760.

van Erp S, Verhagen J, Grasman RP, Wagenmakers E (2017). "Estimates of between-study heterogeneity for 705 meta-analyses reported in Psychological Bulletin from 1990–2013." *Journal of Open Psychology Data*, **5**(1). doi:10.5334/jopd.33.

## See Also

prior(), prior_informed_medicine_names

## Examples

```
# prior distribution representing expected effect sizes in social psychology
# based on prior elicitation with dr. Oosterwijk
p1 <- prior_informed("Oosterwijk")

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


# empirical prior distribution for the standardized mean differences from the oral health
# medical subfield based on meta-analytic effect size estimates from the
# Cochrane database of systematic reviews
p2 <- prior_informed("Oral Health", parameter ="effect", type ="smd")
print(p2)
```

---

# prior_informed_medicine_names

*Names of medical subfields from the Cochrane database of systematic reviews*

---

## Description

Contains the union of names identifying the individual subfields from the Cochrane database of systematic reviews. Type-specific availability depends on the type argument passed to prior_informed().

## Usage

```
prior_informed_medicine_names
```

## Format

An object of class character of length 57.

## 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_mixture *Creates a mixture of prior distributions*

---

## Description

prior_mixture creates a mixture of prior distributions. This is a more generic version of the prior_spike_and_slab function.

## Usage

```
prior_mixture(
  prior_list,
  is_null = rep(FALSE, length(prior_list)),
  components = NULL
)
```

## Arguments

| | |
|---|---|
| prior_list | a list of prior distributions to be mixed. |
| is_null | a logical vector indicating which of the prior distributions should be considered as a null distribution. Defaults to rep(FALSE, length(prior_list)). |
| components | a character vector indicating which of the prior distributions belong to the same mixture component (this is an alternative specification to the is_null argument). Defaults to NULL (i.e., is_null is used. |

## See Also

prior()

prior_phacking *Creates a prior distribution for a p-hacking selection kernel*

## Description

`prior_phacking()` creates a prior distribution for a mass-preserving p-hacking redistribution kernel. The initial backend supports linear and quadratic power depletion between `source` and `target`, with depleted probability mass redistributed between `target` and `destination`.

## Usage

```
prior_phacking(
  side = "one-sided",
  target = 0.025,
  source = 0.25,
  destination = 0.005,
  form = c("linear", "quadratic"),
  alpha = prior("beta", list(1, 1)),
  report_scale = "pi_null",
  prior_weights = 1
)
```

## Arguments

| | |
|---|---|
| `side` | side geometry. Currently only `"one-sided"` is supported. |
| `target` | target p-value cut point. |
| `source` | source p-value cut point. Must be larger than `target`. |
| `destination` | destination p-value cut point. Must be smaller than `target`. |
| `form` | power depletion form, either `"linear"` or `"quadratic"`. |
| `alpha` | prior distribution for the p-hacking severity parameter. |
| `report_scale` | reporting scale for deterministic summaries. |
| `prior_weights` | prior odds associated with a given distribution. |

## Value

`prior_phacking()` returns an object of class `"prior"`.

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_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 Sillanpää (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
)
```

### 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

```
# 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 *Creates a prior distribution for a weight function*

---

### Description

`prior_weightfunction` creates a prior distribution for fitting a RoBMA selection model. The `side` and `steps` arguments define the p-value bins, and the `weights` argument defines the prior on the publication weights in those bins.

### Usage

```
prior_weightfunction(
  side = "one-sided",
  steps = c(0.025, 0.05),
  weights = wf_cumulative(),
  reference = "most_significant",
```

```
  prior_weights = 1
)

wf_cumulative(alpha = NULL)

wf_fixed(omega)

wf_independent(prior, scale = "omega")
```

## Arguments

| | |
|---|---|
| side | side geometry. Either "one-sided" or "two-sided". |
| steps | increasing p-value cut points between 0 and 1. |
| weights | a weight-prior object created by wf_cumulative(), wf_fixed(), or wf_independent(). |
| reference | reference bin. Currently only "most_significant" is supported and fixes the most significant bin to omega = 1. |
| prior_weights | prior odds associated with a given distribution. |
| alpha | positive cumulative-Dirichlet concentration parameters. If omitted, a flat Dirichlet prior is used with one concentration parameter per bin. |
| omega | fixed non-negative relative publication weights, one per bin. The reference-bin weight must be exactly 1. |
| prior | prior distribution for each non-reference weight. |
| scale | latent scale for independent weights. "omega" places the prior directly on the non-negative publication weight. "log_omega" places the prior on log(omega) and transforms with omega = exp(log_omega), allowing weights above one whenever the log prior assigns mass above zero. |

## Value

prior_weightfunction returns an object of class 'prior'.

## See Also

plot.prior()

## Examples

```
p1 <- prior_weightfunction(
  side = "one-sided",
  steps = c(.05, .10),
  weights = wf_cumulative(alpha = c(1, 1, 1))
)

p2 <- prior_weightfunction(
  side = "one-sided",
  steps = c(.05),
  weights = wf_independent(prior("beta", list(1, 1)))
)
```

## 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()`

## remove_column *Removes column to BayesTools table*

### Description

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

### Usage

```
remove_column(table, column_position = NULL)
```

### Arguments

| | |
|---|---|
| `table` | BayesTools table |
| `column_position` | position of the to be removed column (defaults to `NULL` which removes the last column) |

### Value

returns an object of 'BayesTools_table' class.

---

`Savage_Dickey_BF` *Compute Savage-Dickey inclusion Bayes factors*

---

### Description

Computes Savage-Dickey (density ratio) inclusion Bayes factors based the change of height from prior to posterior distribution at the test value.

### Usage

```
Savage_Dickey_BF(
  posterior,
  null_hypothesis = 0,
  normal_approximation = FALSE,
  silent = FALSE
)
```

### Arguments

| | |
|---|---|
| `posterior` | marginal posterior distribution generated via the `marginal_posterior` function |
| `null_hypothesis` | point null hypothesis to test. Defaults to `0` |
| `normal_approximation` | whether the height of prior and posterior density should be approximated via a normal distribution (rather than kernel density). Defaults to `FALSE`. |
| `silent` | whether warnings should be returned silently. Defaults to `FALSE` |

### Value

`Savage_Dickey_BF` returns a Bayes factor.

---

`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)
```

## selection_backend_spec

*Compile selection priors for backend consumers*

### Description

`selection_backend_spec()` compiles step-selection and p-hacking prior objects into active backend parameters. The returned object contains stable p/z geometry, JAGS prior/transform code, monitor names, initial values, and data constants.

### Usage

```
selection_backend_spec(
  priors,
  backend = "jags",
  names = list(omega = "omega", alpha = "alpha"),
  global_breaks = NULL
)
```

### Arguments

| | |
|---|---|
| `priors` | a selection prior, p-hacking prior, composed bias prior, `prior_none()`, `prior_mixture()`, or a list of those priors. |
| `backend` | backend target. Currently only `"jags"` is supported. |
| `names` | list of backend parameter names. |
| `global_breaks` | optional global p-value break grid. |

### Value

A list describing the compiled backend specification.

## transform_factor_samples

*Transform factor posterior samples into differences from the mean*

### Description

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

### Usage

```
transform_factor_samples(samples)
```

### Arguments

| | |
|---|---|
| samples | (a list) of mixed posterior distributions created with mix_posteriors function |

### Value

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

### See Also

mix_posteriors transform_meandif_samples transform_meandif_samples transform_orthonormal_samples

---

## transform_meandif_samples

*Transform meandif posterior samples into differences from the mean*

---

### Description

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

### Usage

```
transform_meandif_samples(samples)
```

### Arguments

| | |
|---|---|
| samples | (a list) of mixed posterior distributions created with mix_posteriors function |

### Value

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

### See Also

mix_posteriors contr.meandif

`transform_orthonormal_samples` *Transform orthonomal posterior samples into differences from the mean*

## Description

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

## Usage

```
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

`transform_prior_samples` *Transform prior samples to original scale*

## Description

Generate prior samples and transform them using the same matrix transformation as posterior samples. This is the correct approach for visualizing priors on the original (unscaled) scale, especially for the intercept which depends on contributions from multiple coefficient priors.

## Usage

```
transform_prior_samples(
  fit,
  n_samples = 10000,
  seed = NULL,
  formula_scale = NULL
)
```

### Arguments

| | |
|---|---|
| `fit` | a fitted model object with `prior_list` and optionally `formula_scale` attributes |
| `n_samples` | number of samples to generate (default: 10000) |
| `seed` | random seed for reproducibility (optional) |
| `formula_scale` | optional nested list containing standardization information. If not provided, extracted from `fit` attribute. |

### Details

When models use auto-scaling (standardizing predictors), the posterior samples are on the standardized scale. To correctly visualize priors on the original scale, we cannot simply apply a linear transformation to individual priors because the intercept on the original scale is a weighted sum of multiple priors:

$$\beta_0^{orig} = \beta_0^* - \sum_i \frac{\mu_i}{\sigma_i} \beta_i^*$$

This function generates samples from ALL priors simultaneously and applies the same matrix transformation used for posterior samples, which correctly handles the intercept and all other parameters.

### Value

A matrix of prior samples on the original (unscaled) scale, with columns matching the structure of posterior samples.

### See Also

`transform_scale_samples()` `plot_posterior()`

### Examples

```
# With a fitted model that used formula_scale:
# prior_samples <- transform_prior_samples(fit, n_samples = 10000)
# This can then be used with density() or for custom plotting
```

---

## `transform_scale_samples` *Transform standardized posterior samples back to original scale*

---

### Description

Transforms posterior samples from standardized continuous predictors back to the original scale. This function is used when predictors were standardized during model fitting via the `formula_scale` parameter.

## Usage

```
transform_scale_samples(fit, formula_scale = NULL)
```

## Arguments

| | |
|---|---|
| `fit` | a fitted model object with `formula_scale` attribute, or a matrix of posterior samples |
| `formula_scale` | nested list containing standardization information keyed by parameter name. Each parameter entry contains scaling info (mean and sd) for each standardized predictor, e.g., `list(mu = list(mu_x1 = list(mean = 0, sd = 1)))`. If `fit` is provided and has a `formula_scale` attribute, this will be used automatically. |

## Details

The function transforms regression coefficients and intercepts to account for predictor standardization using a combinatorial approach that correctly handles interactions of any order.

For a k-way interaction between standardized predictors, the expansion of $\prod_i (x_i - \mu_i)/\sigma_i$ contributes to all lower-order terms. The contribution to a target term T from a source term S (where T is a subset of S's scaled components) is:

$$(-1)^{|extra|} \cdot \prod_{i \in extra} \mu_i / \prod_{i \in S_{scaled}} \sigma_i$$

where $extra = S_{scaled} \setminus T_{scaled}$.

## Value

`transform_scale_samples` returns posterior samples transformed back to the original predictor scale.

## See Also

`JAGS_formula()` `JAGS_fit()`

---

`update.BayesTools_table`

*Updates BayesTools table*

---

## Description

Updates BayesTools table while not breaking formatting, attributes, etc...

### Usage

```
## S3 method for class 'BayesTools_table'
update(
  object,
  title = NULL,
  footnotes = NULL,
  warnings = NULL,
  remove_parameters = NULL,
  logBF = FALSE,
  BF01 = FALSE,
  ...
)
```

### Arguments

| | |
|---|---|
| object | a BayesTools table |
| title | title of the table |
| footnotes | add footnotes to the table |
| warnings | add warnings of the table |
| remove_parameters | remove parameters from the table |
| logBF | whether to format Bayes factors as log(BF) |
| BF01 | whether to format Bayes factors as 1/BF |
| ... | additional arguments. |

### Value

returns an object of 'BayesTools_table' class.

---

## 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 non-negative relative weights for a one.sided or a two.sided weight function. The first/reference weight must be exactly 1. |
| n | number of observations. |
| lower.tail | logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$. |
| 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, one_sided = FALSE)
```

## Arguments

| | |
|---|---|
| `prior_list` | list of prior distributions |
| `cuts_only` | whether only p-value cuts should be returned |
| `one_sided` | force one-sided output |

## Value

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

# Index