

Package: bayesian (via r-universe)

October 1, 2024

Type Package

Version 1.0.1

Title Bindings for Bayesian TidyModels

Description Fit Bayesian models using 'brms'/'Stan' with 'parsnip'/'tidymodels' via 'bayesian' <[doi:10.5281/zenodo.4426836](https://doi.org/10.5281/zenodo.4426836)>. 'tidymodels' is a collection of packages for machine learning; see Kuhn and Wickham (2020) <<https://www.tidymodels.org>>. The technical details of 'brms' and 'Stan' are described in Bürkner (2017) <[doi:10.18637/jss.v080.i01](https://doi.org/10.18637/jss.v080.i01)>, Bürkner (2018) <[doi:10.32614/RJ-2018-017](https://doi.org/10.32614/RJ-2018-017)>, and Carpenter et al. (2017) <[doi:10.18637/jss.v076.i01](https://doi.org/10.18637/jss.v076.i01)>.

License MIT + file LICENSE

URL <https://hsbadr.github.io/bayesian/>,
<https://github.com/hsbadr/bayesian>

BugReports <https://github.com/hsbadr/bayesian/issues>

Depends brms (>= 2.21.0), parsnip (>= 1.2.1), R (>= 4.1.0)

Imports dplyr, purrr, rlang, stats, tibble, utils

Suggests covr, devtools, future, knitr, recipes, rmarkdown, roxygen2, rstan, spelling, testthat, workflows

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Collate 'bayesian_init.R' 'bayesian_load.R' 'bayesian_make.R' 'bayesian.R'

LazyLoad yes

Language en-US

Repository <https://hsbadr.r-universe.dev>

RemoteUrl <https://github.com/hsbadr/bayesian>

RemoteRef HEAD

RemoteSha 8178e590363f547e777f8f0f1336ade1fe16dabe

Contents

bayesian	2
Index	9

bayesian	<i>General Interface for Bayesian TidyModels</i>
----------	--

Description

bayesian() is a way to generate a *specification* of a model before fitting and allows the model to be created using **Stan** via **brms** package in **R**.

Usage

```
bayesian(
  mode = "regression",
  engine = "brms",
  formula.override = NULL,
  family = NULL,
  prior = NULL,
  sample_prior = NULL,
  knots = NULL,
  stanvars = NULL,
  fit = NULL,
  init = NULL,
  chains = NULL,
  iter = NULL,
  warmup = NULL,
  thin = NULL,
  cores = NULL,
  threads = NULL,
  algorithm = NULL,
  backend = NULL,
  stan_args = NULL,
  control = NULL,
  save_pars = NULL,
  save_model = NULL,
  file = NULL,
  file_refit = NULL,
  normalize = NULL,
  future = NULL,
```

```
    seed = NULL,
    silent = NULL
)

## S3 method for class 'bayesian'
update(
  object,
  parameters = NULL,
  formula.override = NULL,
  family = NULL,
  prior = NULL,
  sample_prior = NULL,
  knots = NULL,
  stanvars = NULL,
  fit = NULL,
  init = NULL,
  chains = NULL,
  iter = NULL,
  warmup = NULL,
  thin = NULL,
  cores = NULL,
  threads = NULL,
  algorithm = NULL,
  backend = NULL,
  stan_args = NULL,
  control = NULL,
  save_pars = NULL,
  save_model = NULL,
  file = NULL,
  file_refit = NULL,
  normalize = NULL,
  future = NULL,
  seed = NULL,
  silent = NULL,
  fresh = FALSE,
  ...
)

bayesian_fit(formula, data, ...)

bayesian_formula(formula, ...)

bayesian_terms(formula, ...)

bayesian_family(family, ...)

bayesian_predict(object, ...)
```

```
bayesian_write(object, file)
```

```
bayesian_read(file)
```

Arguments

mode	A single character string for the prediction outcome mode. Possible values for this model are "unknown", "regression", or "classification".
engine	A single character string specifying what computational engine to use for fitting. Possible engines are listed below. The default for this model is "brms".
formula.override	Overrides the formula; for details see brmsformula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a <code>link</code> argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details of supported families see brmsfamily . By default, a linear gaussian model is applied. In multivariate models, family might also be a list of families.
prior	One or more <code>brmsprior</code> objects created by set_prior or related functions and combined using the <code>c</code> method or the <code>+</code> operator. See also default_prior for more help.
sample_prior	Indicate if draws from priors should be drawn additionally to the posterior draws. Options are "no" (the default), "yes", and "only". Among others, these draws can be used to calculate Bayes factors for point hypotheses via hypothesis . Please note that improper priors are not sampled, including the default improper priors used by <code>brm</code> . See set_prior on how to set (proper) priors. Please also note that prior draws for the overall intercept are not obtained by default for technical reasons. See brmsformula how to obtain prior draws for the intercept. If <code>sample_prior</code> is set to "only", draws are drawn solely from the priors ignoring the likelihood, which allows among others to generate draws from the prior predictive distribution. In this case, all parameters must have proper priors.
knots	Optional list containing user specified knot values to be used for basis construction of smoothing terms. See gamm for more details.
stanvars	An optional <code>stanvars</code> object generated by function stanvar to define additional variables for use in Stan 's program blocks.
fit	An instance of S3 class <code>brmsfit</code> derived from a previous fit; defaults to <code>NA</code> . If <code>fit</code> is of class <code>brmsfit</code> , the compiled model associated with the fitted result is re-used and all arguments modifying the model code or data are ignored. It is not recommended to use this argument directly, but to call the update method, instead.
init	Initial values for the sampler. If <code>NULL</code> (the default) or "random", Stan will randomly generate initial values for parameters in a reasonable range. If <code>0</code> , all parameters are initialized to zero on the unconstrained space. This option is sometimes useful for certain families, as it happens that default random initial values cause draws to be essentially constant. Generally, setting <code>init = 0</code> is

worth a try, if chains do not initialize or behave well. Alternatively, `init` can be a list of lists containing the initial values, or a function (or function name) generating initial values. The latter options are mainly implemented for internal testing but are available to users if necessary. If specifying initial values using a list or a function then currently the parameter names must correspond to the names used in the generated Stan code (not the names used in R). For more details on specifying initial values you can consult the documentation of the selected backend.

<code>chains</code>	Number of Markov chains (defaults to 4).
<code>iter</code>	Number of total iterations per chain (including warmup; defaults to 2000).
<code>warmup</code>	A positive integer specifying number of warmup (aka burnin) iterations. This also specifies the number of iterations used for stepsize adaptation, so warmup draws should not be used for inference. The number of warmup should not be larger than <code>iter</code> and the default is <code>iter/2</code> .
<code>thin</code>	Thinning rate. Must be a positive integer. Set <code>thin > 1</code> to save memory and computation time if <code>iter</code> is large.
<code>cores</code>	Number of cores to use when executing the chains in parallel, which defaults to 1 but we recommend setting the <code>mc.cores</code> option to be as many processors as the hardware and RAM allow (up to the number of chains). For non-Windows OS in non-interactive R sessions, forking is used instead of PSOCK clusters.
<code>threads</code>	Number of threads to use in within-chain parallelization. For more control over the threading process, <code>threads</code> may also be a <code>brmsthreads</code> object created by threading . Within-chain parallelization is experimental! We recommend its use only if you are experienced with Stan's <code>reduce_sum</code> function and have a slow running model that cannot be sped up by any other means. Can be set globally for the current R session via the " <code>brms.threads</code> " option (see options).
<code>algorithm</code>	Character string naming the estimation approach to use. Options are " <code>sampling</code> " for MCMC (the default), " <code>meanfield</code> " for variational inference with independent normal distributions, " <code>fullrank</code> " for variational inference with a multivariate normal distribution, " <code>pathfinder</code> " for the pathfinder algorithm, " <code>laplace</code> " for the laplace approximation, or " <code>fixed_param</code> " for sampling from fixed parameter values. Can be set globally for the current R session via the " <code>brms.algorithm</code> " option (see options).
<code>backend</code>	Character string naming the package to use as the backend for fitting the Stan model. Options are " <code>rstan</code> " (the default) or " <code>cmdstanr</code> ". Can be set globally for the current R session via the " <code>brms.backend</code> " option (see options). Details on the rstan and cmdstanr packages are available at https://mc-stan.org/rstan/ and https://mc-stan.org/cmdstanr/ , respectively. Additionally a "mock" backend is available to make testing brms and packages that depend on it easier. The "mock" backend does not actually do any fitting, it only checks the generated Stan code for correctness and then returns whatever is passed in an additional <code>mock_fit</code> argument as the result of the fit.
<code>stan_args</code>	A list of extra arguments to Stan .
<code>control</code>	A named list of parameters to control the sampler's behavior. It defaults to NULL so all the default values are used. The most important control parameters

are discussed in the 'Details' section below. For a comprehensive overview see [stan](#).

<code>save_pars</code>	An object generated by <code>save_pars</code> controlling which parameters should be saved in the model. The argument has no impact on the model fitting itself.
<code>save_model</code>	Either NULL or a character string. In the latter case, the model's Stan code is saved via <code>cat</code> in a text file named after the string supplied in <code>save_model</code> .
<code>file</code>	A character string of the file path to <code>brmsfit</code> object saved via <code>saveRDS</code> .
<code>file_refit</code>	Modifies when the fit stored via the <code>file</code> argument is re-used. Can be set globally for the current R session via the <code>"brms.file_refit"</code> option (see options). For "never" (default) the fit is always loaded if it exists and fitting is skipped. For "always" the model is always refitted. If set to "on_change", <code>brms</code> will refit the model if model, data or algorithm as passed to Stan differ from what is stored in the file. This also covers changes in priors, <code>sample_prior</code> , <code>stanvars</code> , covariance structure, etc. If you believe there was a false positive, you can use <code>brmsfit_needs_refit</code> to see why refit is deemed necessary. Refit will not be triggered for changes in additional parameters of the fit (e.g., initial values, number of iterations, control arguments, ...). A known limitation is that a refit will be triggered if within-chain parallelization is switched on/off.
<code>normalize</code>	Logical. Indicates whether normalization constants should be included in the Stan code (defaults to TRUE). Setting it to FALSE requires Stan version ≥ 2.25 to work. If FALSE, sampling efficiency may be increased but some post processing functions such as <code>bridge_sampler</code> will not be available. Can be controlled globally for the current R session via the 'brms.normalize' option.
<code>future</code>	Logical; If TRUE, the <code>future</code> package is used for parallel execution of the chains and argument cores will be ignored. Can be set globally for the current R session via the "future" option. The execution type is controlled via <code>plan</code> (see the examples section below).
<code>seed</code>	The seed for random number generation to make results reproducible. If NA (the default), <code>Stan</code> will set the seed randomly.
<code>silent</code>	Verbosity level between 0 and 2. If 1 (the default), most of the informational messages of compiler and sampler are suppressed. If 2, even more messages are suppressed. The actual sampling progress is still printed. Set <code>refresh = 0</code> to turn this off as well. If using <code>backend = "rstan"</code> you can also set <code>open_progress = FALSE</code> to prevent opening additional progress bars.
<code>object</code>	A Bayesian model specification.
<code>parameters</code>	A 1-row tibble or named list with <i>main</i> parameters to update. If the individual arguments are used, these will supersede the values in <code>parameters</code> . Also, using engine arguments in this object will result in an error.
<code>fresh</code>	A logical for whether the arguments should be modified in-place of or replaced wholesale.
<code>...</code>	Other arguments passed to internal functions.
<code>formula</code>	An object of class <code>formula</code> , <code>brmsformula</code> , or <code>mvbrmsformula</code> (or one that can be coerced to that classes): A symbolic description of the model to be fitted. The details of model specification are explained in <code>brmsformula</code> .
<code>data</code>	An object of class <code>data.frame</code> (or one that can be coerced to that class) containing data of all variables used in the model.

Details

The arguments are converted to their specific names at the time that the model is fit. Other options and argument can be set using `set_engine()`. If left to their defaults here (NULL), the values are taken from the underlying model functions. If parameters need to be modified, `update()` can be used in lieu of recreating the object from scratch.

The data given to the function are not saved and are only used to determine the *mode* of the model. For `bayesian()`, the possible modes are "regression" and "classification".

The model can be created by the `fit()` function using the following *engines*:

- **brms**: "brms"

Value

An updated model specification.

Engine Details

Engines may have pre-set default arguments when executing the model fit call. For this type of model, the template of the fit calls are:

```
bayesian() |>
  set_engine("brms") |>
  translate()

## Bayesian Model Specification (regression)
##
## Computational engine: brms
##
## Model fit template:
## bayesian::bayesian_fit(formula = missing_arg(), data = missing_arg(),
##   weights = missing_arg())
```

See Also

[brm](#), [brmsfit](#), [update.brmsfit](#), [predict.brmsfit](#), [posterior_epred.brmsfit](#), [posterior_predict.brmsfit](#), [brmsformula](#), [brmsformula-helpers](#), [brmsterms](#), [brmsfamily](#), [customfamily](#), [family](#), [formula](#), [update.formula](#).

Examples

```
bayesian()

show_model_info("bayesian")

bayesian(mode = "classification")
bayesian(mode = "regression")
## Not run:
bayesian_mod <-
  bayesian() |>
```

```
set_engine("brms") |>
fit(
  rating ~ treat + period + carry + (1 | subject),
  data = inhaler
)

summary(bayesian_mod$fit)

## End(Not run)

model <- bayesian(init = "random")
model
update(model, init = "0")
update(model, init = "0", fresh = TRUE)
```


Index

bayesian, 2
bayesian_family (bayesian), 2
bayesian_fit (bayesian), 2
bayesian_formula (bayesian), 2
bayesian_predict (bayesian), 2
bayesian_read (bayesian), 2
bayesian_terms (bayesian), 2
bayesian_write (bayesian), 2
bridge_sampler, 6
brm, 7
brmsfamily, 4, 7
brmsfit, 7
brmsfit_needs_refit, 6
brmsformula, 4, 6, 7
brmsterms, 7

cat, 6
customfamily, 7

default_prior, 4

family, 7
formula, 6, 7
future, 6

gamm, 4

hypothesis, 4

mvbrmsformula, 6

options, 5, 6

plan, 6
posterior_epred.brmsfit, 7
posterior_predict.brmsfit, 7
predict.brmsfit, 7

save_pars, 6
saveRDS, 6
set_prior, 4

stan, 6
stanvar, 4

threading, 5

update, 4
update.bayesian (bayesian), 2
update.brmsfit, 7
update.formula, 7