Package 'bayesian'

January 20, 2022

Type Package

Version 0.0.8

Title Bindings for Bayesian TidyModels

Description Fit Bayesian models using 'brms'/'Stan' with 'parsnip'/'tidymodels' via 'bayesian' <doi:10.5281/zenodo.5884879>. '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>, Bürkner (2018) <doi:10.32614/RJ-2018-017>, and Carpenter et al. (2017) <doi: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 (>= 2.16.3), parsnip (>= 0.1.7), R (>= 3.5.0)

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 7.1.2

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

LazyLoad yes

Language en-US

NeedsCompilation no

Author Hamada S. Badr [aut, cre] (<https://orcid.org/0000-0002-9808-2344>), Paul-Christian Bürkner [aut]

Maintainer Hamada S. Badr <badr@jhu.edu>

Repository CRAN

Date/Publication 2022-01-20 19:32:42 UTC

R topics documented:

`bayesian` *General Interface for Bayesian TidyModels*

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,
  inits = 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,
  inits = 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 link 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 brmsprior objects created by set_prior or related functions and combined using the c method or the + operator. See also get_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 brm. 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 sample_prior 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 stanvars object generated by function stanvar to define additional variables for use in **Stan**'s program blocks.
fit	An instance of S3 class brmsfit derived from a previous fit; defaults to NA. If fit is of class brmsfit, 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.
inits	Either "random" or "0". If inits is "random" (the default), Stan will randomly generate initial values for parameters. If it is "0", all parameters are initialized to zero. This option is sometimes useful for certain families, as it happens that default ("random") inits cause draws to be essentially constant. Generally, setting inits = "0" is worth a try, if chains do not behave well. Alternatively,

inits 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.

chains — Number of Markov chains (defaults to 4).

iter — Number of total iterations per chain (including warmup; defaults to 2000).

warmup — 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 iter and the default is iter/2.

thin — Thinning rate. Must be a positive integer. Set thin > 1 to save memory and computation time if iter is large.

cores — Number of cores to use when executing the chains in parallel, which defaults to 1 but we recommend setting the mc.cores 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.

threads — Number of threads to use in within-chain parallelization. For more control over the threading process, threads may also be a brmsthreads object created by threading. Within-chain parallelization is experimental! We recommend its use only if you are experienced with Stan's reduce_sum function and have a slow running model that cannot be sped up by any other means.

algorithm — Character string naming the estimation approach to use. Options are "sampling" for MCMC (the default), "meanfield" for variational inference with independent normal distributions, "fullrank" for variational inference with a multivariate normal distribution, or "fixed_param" for sampling from fixed parameter values. Can be set globally for the current R session via the "brms.algorithm" option (see options).

backend — Character string naming the package to use as the backend for fitting the Stan model. Options are "rstan" (the default) or "cmdstanr". Can be set globally for the current R session via the "brms.backend" 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 mock_fit argument as the result of the fit.

stan_args — A list of extra arguments to **Stan**.

control — 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.

save_pars — An object generated by save_pars controlling which parameters should be saved in the model. The argument has no impact on the model fitting itself.

save_model — Either NULL or a character string. In the latter case, the model's Stan code is saved via cat in a text file named after the string supplied in save_model.

file — A character string of the file path to brmsfit object saved via saveRDS.

file_refit — Modifies when the fit stored via the file parameter is re-used. Can be set globally for the current R session via the "brms.file_refit" 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", brms 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, sample_prior, stanvars, covariance structure, etc. If you believe there was a false positive, you can use brmsfit_needs_refit 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.

normalize — 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 bridge_sampler will not be available. Can be controlled globally for the current R session via the 'brms.normalize' option.

future — Logical; If TRUE, the **future** 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 plan (see the examples section below).

seed — The seed for random number generation to make results reproducible. If NA (the default), **Stan** will set the seed randomly.

silent — 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 refresh = 0 to turn this off as well. If using backend = "rstan" you can also set open_progress = FALSE to prevent opening additional progress bars.

object — A Bayesian model specification.

parameters — A 1-row tibble or named list with *main* parameters to update. If the individual arguments are used, these will supersede the values in parameters. Also, using engine arguments in this object will result in an error.

fresh — A logical for whether the arguments should be modified in-place of or replaced wholesale.

... — Other arguments passed to internal functions.

formula — An object of class formula, brmsformula, or mvbrmsformula (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 brmsformula.

data — An object of class data.frame (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())
```

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(inits = "random")
model
update(model, inits = "0")
update(model, inits = "0", fresh = TRUE)
```

Index