Package: stan4bart (via r-universe)

October 21, 2026

Version 0.0-13

Date 2026-08-21

Title Bayesian Additive Regression Trees with Stan-Sampled Parametric Extensions

Depends R (>= 3.5-0), methods, dbarts (>= 0.9-34)

Imports stats, Matrix, parallel, RcppParallel (>= 5.1.1)

LinkingTo BH (>= 1.72.0.3), Rcpp (>= 1.0.5), RcppEigen (>= 0.3.3.7.0), RcppParallel (>= 5.1.1), dbarts (>= 0.9-20)

Suggests testthat (>= 2.0-0), lme4, reformulas

Description Fits semiparametric linear and multilevel models with non-parametric additive Bayesian additive regression tree (BART; Chipman, George, and McCulloch (2010) <doi:10.1214/09-AOAS285>) components and Stan (Stan Development Team (2021) <https://mc-stan.org/>) sampled parametric ones. Multilevel models can be expressed using 'lme4' syntax (Bates, Maechler, Bolker, and Walker (2015) <doi:10.18637/jss.v067.i01>).

License GPL (>= 3)

NeedsCompilation yes

Biarch true

UseLTO true

URL https://github.com/vdorie/stan4bart

BugReports https://github.com/vdorie/stan4bart/issues

Author Vincent Dorie [aut, cre] (ORCID: <https://orcid.org/0000-0002-9576-3064>), Ben Goodrich [ctb] (rstanarm_functions.R, StanHeaders), Jonah Gabry [ctb] (rstanarm_functions.R, StanHeaders), Imad Ali [ctb] (rstanarm_functions.R), Sam Brilleman [ctb] (rstanarm_functions.R), Paul-Christian Burkner [ctb] (rstanarm_functions.R, ORCID: <https://orcid.org/0000-0001-5765-8995>), Joshua Pritikin [ctb]

(StanHeaders, ORCID: <https://orcid.org/0000-0002-9862-5484>), Andrew Gelman [ctb] (StanHeaders, ORCID: <https://orcid.org/0000-0002-6975-2601>), Bob Carpenter [ctb] (StanHeaders), Matt Hoffman [ctb] (StanHeaders), Daniel Lee [ctb] (StanHeaders), Michael Betancourt [ctb] (StanHeaders, ORCID: <https://orcid.org/0000-0002-2900-0931>), Marcus Brubaker [ctb] (StanHeaders, ORCID: <https://orcid.org/0000-0002-7892-9026>), Jiqiang Guo [ctb] (StanHeaders), Peter Li [ctb] (StanHeaders), Allen Riddell [ctb] (StanHeaders), Marco Inacio [ctb] (StanHeaders, ORCID: <https://orcid.org/0000-0002-6865-5404>), Mitzi Morris [ctb] (StanHeaders), Jeffrey Arnold [ctb] (StanHeaders, ORCID: <https://orcid.org/0000-0001-9953-3904>), Rob Goedman [ctb] (StanHeaders), Brian Lau [ctb] (StanHeaders), Rob Trangucci [ctb] (StanHeaders), Alp Kucukelbir [ctb] (StanHeaders), Robert Grant [ctb] (StanHeaders), Dustin Tran [ctb] (StanHeaders), Michael Malecki [ctb] (StanHeaders), Yuanjun Gao [ctb] (StanHeaders), Trustees of Columbia University [cph] (rstanarm_functions.R, StanHeaders), Lawrence Livermore National Security [cph] (CVODES), The Regents of the University of California [cph] (CVODES), Southern Methodist University [cph] (CVODES), Douglas Bates [ctb] (lme4_functions.R, ORCID: <https://orcid.org/0000-0001-8316-9503>), Martin Maechler [ctb] (lme4_functions.R, ORCID: <https://orcid.org/0000-0002-8685-9910>), Ben Bolker [ctb] (lme4_functions.R, ORCID: <https://orcid.org/0000-0002-2127-0443>), Steve Walker [ctb] (lme4_functions.R, ORCID: <https://orcid.org/0000-0002-4394-9078>), Armon Dadgar [ctb] (adaptive radix tree), Bothner Per [ctb] (config.guess), Elliston Ben [ctb] (config.guess), Free Software Foundation [cph] (config.sub), Guido U Draheim [ctb] (ax_check_compile_flag.m4), Maarten Bosmans [ctb] (ax_check_compile_flag.m4), Christophe Tournayre [ctb] (ax_ext.m4), Michael Petch [ctb] (ax_ext.m4, ax_gcc_x86_avx_xgetbv.m4, ax_gcc_x86_cpuid.m4), Rafael de Lucena Valle [ctb] (ax_ext.m4), Steven G. Johnson [ctb] (ax_gcc_x86_cpuid.m4, ORCID: <https://orcid.org/0000-0001-7327-4967>), Matteo Frigo [ctb] (ax_gcc_x86_cpuid.m4), Scott Pakin [ctb] (ax_func_posix_memalign.m4, ORCID: <https://orcid.org/0000-0002-5220-1985>)

Maintainer Vincent Dorie <vdorie@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-08-21 23:20:02 UTC

RemoteUrl https://github.com/cran/stan4bart

RemoteRef HEAD

RemoteSha bf0093d3eefe32acda1c2d3ba8ae36467fbd0037

Contents

stan4bart 3
stan4bart-generics 9

Index **12**

`stan4bart` *Semiparametric Models Using Stan and BART*

Description

This function fits semi-parametric linear and probit models that have a non-parametric, BART component and one or more of a parametric fixed effect (unmodeled coefficients), or a parametric random effect (modeled coefficients). If $f(x)$ is a BART "sum-of-trees" model, fits:

- For continuous response variables:

$$Y \mid b \sim \mathrm{N}\left(f(X^b) + X^f\beta + Zb, \sigma^2\right) b \sim \mathrm{N}(0, \Sigma_b)$$

- For binary response variables:

$$P(Y = 1 \mid b) = \Phi\left(f(X^b) + X^f\beta + Zb\right) b \sim \mathrm{N}(0, \Sigma_b)$$

Usage

```
stan4bart(formula,
          data = NULL,
          subset,
          weights,
          na.action = getOption("na.action", "na.omit"),
          offset,
          contrasts = NULL,
          test = NULL,
          treatment = NULL,
          offset_test = NULL,
          verbose = FALSE,
          iter = 2000L,
          warmup = iter %/% 2L,
          skip = 1L,
          chains = 4L,
          cores = getOption("mc.cores", 1L),
          refresh = max(iter %/% 10L, 1L),
          offset_type = c("default", "fixef", "ranef", "bart", "parametric"),
          seed = NA_integer_,
          keep_fits = TRUE,
```

```
            callback = NULL,
            stan_args = NULL,
            bart_args = NULL)
```

Arguments

formula	a formula object, or one that can be coerced to that type. Terms on the right-hand-side of the formula that are encased in a symbolic call to bart() will be used to create the non-parametric component. Terms that use the lmer-style grouping syntax will be added as parametric, hierarchical varying intercepts and slopes. All other terms will be added as fixed effects.
data	an optional data frame containing the variables in the formula. Its use is strongly encouraged.
subset, weights, na.action, offset, contrasts	optional components adjusting the constructed model matrices and otherwise changing the linear predictor. na.action cannot be "na.pass". See lm and model.matrix.default.
test	an optional data frame to be used as test data. If present, the test predictions will be stored as the sampler runs and can be extracted later.
treatment	an optional symbol, that when present and refers to a binary variable, will be used to create a test data frame with the treatment variable set to its counterfactual. Only one of test and treatment can be supplied.
offset_test	optional vector which will be added to the test predictions.
verbose	a logical or integer. If FALSE or non-positive, runs quietly. Additional levels of information may be displayed for increasingly positive numbers, however a large number of diagnostics are suppressed when running multi-threaded. If negative, all diagnostic information is ignored.
iter	positive integer indicating the number of posterior samples to draw and return. Includes warmup samples.
warmup	non-negative integer indicating number of posterior samples to draw and throw away. Also controls the number of iterations for which adaptation is run.
skip	one or two positive integers. Every skip sample will be kept, while every other sample will be discarded. If argument is length two, an attempt will be made to use he named element "bart" for BART and "stan" for Stan. If not named, BART is the first skip element and Stan is the second. This argument does not impact the number of iters returned, unlike a conventional "thinning" parameter.
chains	positive integer giving the number of Markov Chains to sample.
cores	positive integer giving the number of units of parallelization. Computation for each chain will be divide among the cores available. When greater than one, verbose output within chains will not be available.
refresh	positive integer giving the frequency with which diagnostic information should be printed as the sampler runs. Only applies with cores (or chains) equal to 1.
offset_type	character; an experimental/testing feature that controls how offset is to be interpreted. When one of "fixef", "ranef", or "bart", the offset is used to

	replace that part of the model. When `"parametric"`, it replaces both of the fixed and random parametric components. Sampling is still done for these components and their draws are stored, however whenever they were present in the fit the supplied value is used instead.
`seed`	Optional integer specifying the desired pRNG seed. It should not be needed when running single-threaded - calling `set.seed` will suffice. The primary use of `seed` is to obtain reproducible results when multi-threaded. See Reproducibility section below.
`keep_fits`	Logical that, when false, prevents the sampler from storing each draw. Intended to be used with `callback`.
`callback`	A function that will be called at each iteration, accepting three arguments: `yhat.train`, `yhat.test`, `stan_pars`. See details for more information.
`stan_args`	optional list, specifying further arguments to Stan. See details below.
`bart_args`	optional list, specifying further arguments to BART. See details below.

Details

Fits a Bayesian "mixed effect" model with a non-parametric Bayesian Additive Regression Trees (BART) component. For continuous responses:

- $Y_i \mid b \sim \mathrm{N}\left(f(X_i^b) + X_i^f\beta + Z_i b_{g[i]}, \sigma^2\right)$
- $b_j \sim \mathrm{N}(0, \Sigma_b)$

where b_j are the "random effects" - random intercepts and slopes - that correspond to group j, $g[i]$ is a mapping from individual i to its group index, f - a BART sum-of-trees model, X^b are predictors used in the BART model, X^f are predictors in a parametric, linear "fixed effect" component, Z is the design matrix for the random intercept and slopes, and $sigma$ and $Sigma_b$ are variance components.

Binary outcome models are obtained by assuming a latent variable that has the above distribution, and that the observed response is 1 when that variable is positive and 0 otherwise. The response variable marginally has the distribution:

- $$P(Y_i = 1 \mid b) = \Phi\left(f(X_i^b) + X_i^f\beta + Z_i b_{g[i]}\right)$$

where Φ is the cumulative distribution function of the standard normal distribution.

Terminology: As `stan4bart` fits a Bayesian model, essentially all components are "modeled". Furthermore, as it has two first-level, non-hierarchical components, "fixed" effects are ambiguous. Thus we adopt:

- "fixed" - refers only to the parametric, linear, individual level mean component, $X^f\beta$; these are "unmodeled coefficients" in other contexts
- "random" - refers only to the parametric, linear, hierarchical mean component, Zb; these are "modeled coefficients" in other contexts
- "bart" - refers only to the nonparametric, individual level mean component, $f(X^b)$

Model Specification: Model specification occurs in the `formula` object, according to the following rules:

- variables or terms specified inside a pseudo-call to `bart` are used for the "bart" component, e.g. `y ~ bart(x_1 + x_2)`
- variables or terms specified according to `lmer` syntax are used for the "random" effect component, e.g. `y ~ (1 | g_1) + (1 + x_3 | g_1)`
- remaining variables not inside a `bart` or "bars" construct are used for the "fixed" effect component; e.g. `y ~ x_4`

All three components can be present in a single model, however are `bart` part must present. If you wish to fit a model without one, use `stan_glmer` in the `rstanarm` package instead.

Additional Arguments: The `stan_args` and `bart_args` arguments to `stan4bart` can be used to pass further arguments to `stan` and `bart` respectively. These are similar to the functions `stan` in the `rstan` package and `bart`, but not identical as `stan4bart` constructs its own model internally.

Stan arguments include:

- `prior_covariance`
- `prior`, `prior_intercept`, `prior_aux`, `QR`
- `init_r`, `adapt_gamma`, `adapt_delta`, `adapt_kappa` - see the help page for `stan` in the `rstan` package.

For reference on the first two sets of options, see the help page for `stan_glmer` in the `rstanarm` package; for reference on the third set, see the help page for `stan` in the `rstan` package.

BART arguments include:

- further arguments to `dbartsControl` that are not specified by `stan4bart`, such as `keepTrees` or `n.trees`; keeping trees can be costly in terms of memory, but is required to use `predict`

Reproducibility: Behavior differs when running multi- and single-threaded, as the pseudo random number generators (pRNG) used by R are not thread safe. When single-threaded, R's built-in generator is used; if set at the start, `.Random.seed` will be used and its value updated as samples are drawn. When multi-threaded, the default behavior is draw new random seeds for each thread using the clock and use thread-specific pRNGs.

This behavior can be modified by setting `seed`. For the single-threaded case, that seed will be installed and the existing seed replaced at the end, if applicable. For multi-threaded runs, the seeds for threads are drawn sequentially using the supplied seed, and will not change the state of R's built-in generator.

Consequently, the `seed` argument should not be needed when running single-threaded - `set.seed` will suffice. When multi-threaded, `seed` can be used to obtain reproducible results.

Callbacks: Callbacks can be used to avoid expensive memory allocation, aggregating results as the sampler proceeds instead of waiting until the end. A callback funtion must accept the arguments:

- `yhat.train` - BART predictions of the expected value of the training data, conditioned on the Stan parameters
- `yhat.test` - when applicable, the same values as above but for the test data; `NULL` otherwise
- `stan_pars` - named vector of Stan samples, including fixed and random effects and variance parameters

It is expected that the callback will return a vector of the same length each time. If not, invalid memory writes will likely result. If the result of the callback has names, they will be added to the result.

Serialization: At present, `stan4bart` models cannot be safely `saved` and `loaded` in a way that the sampler can be restarted. This feature may be added in the future.

Value

Returns a list assigned class `stan4bartFit`. Has components below, some of which will be `NULL` if not applicable.

Input values:

`y`	response vector
`weights`	weights vector or null
`offset`	offset vector or null
`frame`	joint model frame for all components
`formula`	formula used to specify the model
`na.action`	supplied na.action
`call`	original call

Stored data:

`bartData`	`data` object used for BART component
`X`	fixed effect design matrix or NULL
`X_means`	column means of fixed effect design matrix when appropriate
`reTrms`	random effect "terms" object when applicable, as used by `lmer`
`test`	named list when applicable, having components `X` and `reTrms`; test data for BART is added to the `bartData` result
`treatment`	treatment vector, when applicable

Results, better accessed using `extract`:

`bart_train`	samples of individual posterior predictions for BART component
`bart_test`	predicted test values for BART component, when applicable
`bart_varcount`	BART variable counts
`sigma`	samples of residual standard error; not present for binary outcomes
`k`	samples of the end-node sensitivity parameter; only present when it is modeled
`ranef`	samples of random effects, or modeled coefficients; will be a named list, with effects for each grouping factor
`Sigma`	samples of covariance of random effects; also a named list with one element for each grouping factor
`fixef`	samples of the fixed effects, or unmodeled coefficients
`callback`	samples returned by an optional callback function

Other items:

warmup	a list of warmup samples, containing the same objects in the results subsection
diagnostics	Stan sampler produced diagnostic information, include tree depth and divergent transitions
sampler.bart	external points to BART samplers; used only for predict when keepTrees is TRUE
range.bart	internal scale used by BART samplers, used by predict when keepTrees is TRUE

Author(s)

Vincent Dorie: <vdorie@gmail.com>.

See Also

bart, lmer, and stan_glmer in the rstanarm package

Examples

```
# simulate data (extension of Friedman MARS paper)
# x consists of 10 variables, only first 5 matter
# x_4 is linear
f <- function(x)
  10 * sin(pi * x[,1] * x[,2]) + 20 * (x[,3] - 0.5)^2 +
      10 * x[,4] + 5 * x[,5]

set.seed(99)
sigma <- 1.0

n <- 100
n.g.1 <- 5L
n.g.2 <- 8L

# sample observation level covariates and calculate marginal mean
x <- matrix(runif(n * 10), n, 10)
mu.bart  <- f(x) - 10 * x[,4]
mu.fixef <- 10 * x[,4]

# varying intercepts and slopes for first grouping factor
g.1 <- sample(n.g.1, n, replace = TRUE)
Sigma.b.1 <- matrix(c(1.5^2, .2, .2, 1^2), 2)
b.1 <- matrix(rnorm(2 * n.g.1), n.g.1) %*% chol(Sigma.b.1)

# varying intercepts for second grouping factor
g.2 <- sample(n.g.2, n, replace = TRUE)
Sigma.b.2 <- as.matrix(1.2)
b.2 <- rnorm(n.g.2, 0, sqrt(Sigma.b.2))

mu.ranef <- b.1[g.1,1] + x[,4] * b.1[g.1,2] + b.2[g.2]
```

```
y <- mu.bart + mu.fixef + mu.ranef + rnorm(n, 0, sigma)

df <- data.frame(y, x, g.1, g.2)

fit <- stan4bart(
    formula = y ~
        X4 +                          # linear component ("fixef")
        (1 + X4 | g.1) + (1 | g.2) + # multilevel ("ranef")
        bart(. - g.1 - g.2 - X4),    # use bart for other variables
    verbose = -1, # suppress ALL output
    # low numbers for illustration
    data = df,
    chains = 1, iter = 10, bart_args = list(n.trees = 5))

# posterior means of individual expected values
y.hat <- fitted(fit)

# posterior means of the random effects
ranef.hat <- fitted(fit, type = "ranef")
```

---

stan4bart-generics *Generic Functions for stan4bart Model Fits*

Description

Commonly expected utility functions to derive useful quantities from fitted models.

Usage

```
## S3 method for class 'stan4bartFit'
extract(
  object,
  type = c("ev", "ppd", "fixef", "indiv.fixef", "ranef", "indiv.ranef",
           "indiv.bart", "sigma", "Sigma", "k", "varcount", "stan",
           "trees", "callback"),
  sample = c("train", "test"),
  combine_chains = TRUE,
  sample_new_levels = TRUE,
  include_warmup = FALSE,
  ...)

## S3 method for class 'stan4bartFit'
fitted(
  object,
  type = c("ev", "ppd", "fixef", "indiv.fixef", "ranef", "indiv.ranef",
           "indiv.bart", "sigma", "Sigma", "k", "varcount", "stan",
           "callback"),
  sample = c("train", "test"),
```

```
  sample_new_levels = TRUE,
  ...)

## S3 method for class 'stan4bartFit'
predict(
  object, newdata, offset,
  type = c("ev", "ppd", "indiv.fixef", "indiv.ranef", "indiv.bart"),
  combine_chains = TRUE,
  sample_new_levels = TRUE,
  ...)
```

Arguments

`object`	a fitted model resulting from a call to `stan4bart`.
`type`	a character vector; one of the options listed below.
`sample`	one of `"train"` or `"test"`, indicating if the training or test data frames should be used.
`combine_chains`	logical controlling if chain information should be discarded and the result returned as a matrix instead of an array.
`sample_new_levels`	logical; if `TRUE`, levels out of the training sample will have random effects drawn from their posterior predictive distribution. If `FALSE`, their random effects will be fixed to 0.
`include_warmup`	logical or `"only"`; when `TRUE`/`FALSE`, warmup samples will or will not be included in the result respectively. When `"only"`, only the warmup samples will be returned.
`newdata`	data frame for making out of sample predictions.
`offset`	optional vector which will be added to test predictors.
`...`	not currently in use, but provided to match signatures of other generics.

Details

`extract` is used to obtain raw samples using the training or test data, `fitted` averages those samples, and `predict` operates on data not available at the time of fitting. Note: `predict` requires that the model be fit with `args_bart = list(keepTrees = TRUE)`.

Return type: The type argument accepts:

- `"ev"` - the individual level expected value, that is draws from $E[Y \mid X^b, X^f, Z] \mid Y = f(X^b) + X^f\beta + Zb \mid Y$ where the expectation is with respect to the posterior distribution of the parameters given the data
- `"ppd"` - draws from the individual level posterior predictive distribution, generally speaking adding noise to the result for `"ev"` or simulating new Bernoulli trials.
- `"fixef"` - draws from the posterior of the fixed effects (also known as the "unmodeled" coefficients), $\beta \mid Y$
- `"indiv.fixef"` - draws from the posterior distribution of the individual level mean component deriving from the fixed effects, $X^f\beta$

- "ranef" - the random effects, varying intercepts and slopes, or "modeled" coefficients, b; b has substantial structure that is represented as the returned value, where coefficients are reported within their grouping factors
- "indiv.ranef" - individual level mean component deriving from the random effects, Zb
- "indiv.bart" - individual level mean component deriving from the BART model, $f(X^b)$
- "sigma" - for continuous responses, the residual standard error
- "Sigma" - when applicable, the covariance matrices of the random effects
- "stan" - raw matrix or array of Stan sampled transformed parameters.
- "trees" - a data frame of flatted trees; see the subsection on extracted trees in bart and note that stan4bart variable names can be found in the bartData@x element of a fitted stan4bart model
- "callback" - if a callback function was provided while fitting, the results of that for each sample

Value

extract and predict return either arrays of dimensions equal to n.observations x n.samples x n.chains when combine_chains is FALSE, or matrices of dimensions equal to n.observations x (n.samples * n.chains) when combine_chains is TRUE.

fitted returns a vector of the appropriate length by averaging the result of a call to extract.

Author(s)

Vincent Dorie: <vdorie@gmail.com>.

Index

.Random.seed, *6*

bart, *6*, *8*, *11*

data, *7*
dbartsControl, *6*

extract, *7*
extract *(*stan4bart-generics*)*, 9

fitted.stan4bartFit
        *(*stan4bart-generics*)*, 9
formula, *4*

lm, *4*
lmer, *4*, *6–8*
load, *7*

model, *3*
model.matrix.default, *4*

predict, *6*, *8*
predict.stan4bartFit
        *(*stan4bart-generics*)*, 9

save, *7*
seed, *5*
set.seed, *5*, *6*
stan4bart, 3, *10*
stan4bart-generics, 9