Package 'BATTS'

November 13, 2026

Title Boosting and Bayesian Additive Trees for Two-Sample Comparison

Version 0.4.0

Description Implements boosting and Bayesian additive tree methods for two-sample comparison through density ratio estimation, following Awaya, Xu, and Ma (2025) <doi:10.48550/arXiv.2508.03059>. The package supports both point estimation and uncertainty quantification via Bayesian additive tree ensembles.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

LinkingTo Rcpp, RcppArmadillo

Imports graphics, Rcpp, stats

Suggests ggplot2, knitr, mvtnorm, rmarkdown, scales

SystemRequirements C++17

VignetteBuilder knitr

URL https://github.com/nawaya040/BATTS/tree/cran/dev, https://arxiv.org/abs/2508.03059

BugReports https://github.com/nawaya040/BATTS/issues

NeedsCompilation yes

Author Naoki Awaya [aut, cre, cph], Yuliang Xu [aut, cph], Li Ma [aut, cph]

Maintainer Naoki Awaya <nawaya@waseda.jp>

Repository CRAN

Date/Publication 2026-11-13 22:50:20 UTC

Contents

BATTS-package . . . 2
bart_control . . . 3
batts-methods . . . 4
batts-plot . . . 5
batts-predict . . . 6
batts_bart . . . 8
batts_boost . . . 9
boost_control . . . 11
density_ratio . . . 12

Index **13**

`BATTS-package` *Boosting and Bayesian Additive Trees for Two-Sample Comparison*

Description

Estimate the density ratio of two samples using boosting or generalized Bayesian additive trees.

Details

- `batts_boost()` fits by boosting, with settings in `boost_control()`.
- `batts_bart()` draws from the generalized posterior, with settings in `bart_control()`.
- `predict()`, `fitted()` and `density_ratio()` evaluate a fit; see batts-predict. `print()`, `summary()` and plot() describe it.

Author(s)

Maintainer: Naoki Awaya <nawaya@waseda.jp> [copyright holder]

Authors:

- Yuliang Xu [copyright holder]
- Li Ma [copyright holder]

References

Awaya, N., Xu, Y., and Ma, L. (2025). Two-sample comparison through additive tree models for density ratios. arXiv:2508.03059. doi:10.48550/arXiv.2508.03059.

See Also

Useful links:

- https://github.com/nawaya040/BATTS/tree/cran/dev
- https://arxiv.org/abs/2508.03059
- Report bugs at https://github.com/nawaya040/BATTS/issues

`bart_control` *Settings for Bayesian additive tree fits*

Description

Collect the settings of `batts_bart()`. The defaults are the settings used in the simulations of Awaya, Xu, and Ma (2025): 200 trees, 2000 burn-in iterations, 1000 saved draws, `lambda_0 = 5` and margin 0.1. Each fit starts from unsplit trees.

Usage

```
bart_control(
  num_trees = 200,
  burnin = 2000,
  draws = 1000,
  thin = 1,
  prob_moves = c(1/3, 1/3, 1/3),
  lambda_0 = 5,
  omega_prior = c(1, 1),
  tree_prior = c(0.95, 2),
  margin_scale = 0.1,
  keep_draws = FALSE
)
```

Arguments

`num_trees`	Number of trees in the ensemble.
`burnin`	Number of discarded backfitting iterations. `NULL` uses `floor(draws / 2)`.
`draws`	Number of saved posterior draws.
`thin`	Number of backfitting sweeps per iteration, including burn-in iterations.
`prob_moves`	Probabilities of the GROW, PRUNE and CHANGE moves: three positive values summing to one.
`lambda_0`	Leaf-prior scale per tree; the ensemble scale `num_trees * lambda_0` stays fixed during sampling.
`omega_prior`	Shape and rate of the Gamma prior on the loss temperature.
`tree_prior`	Two tree-depth prior parameters: the split probability at depth d is `tree_prior[1] / (1 + d)^tree_prior[2]`.
`margin_scale`	Fraction of each training range added at both ends of the fitted domain.
`keep_draws`	Logical; save the posterior forests so that the fit can be evaluated at new points with `predict()`. Saving them does not change the draws or the use of random numbers, but can make the fitted object large (over 100 MB with the default numbers of trees and draws).

Value

An object of class `bart_control`.

See Also

`batts_bart()`

Examples

```
bart_control(num_trees = 100, burnin = 300, draws = 301, keep_draws = TRUE)
```

---

batts-methods *Methods for BATTS fits*

Description

Display and summary methods for fits from `batts_boost()` and `batts_bart()`.

Usage

```
## S3 method for class 'batts_boost'
print(x, ...)

## S3 method for class 'batts_bart'
print(x, ...)

## S3 method for class 'batts_fit'
summary(object, ...)

## S3 method for class 'summary.batts_fit'
print(x, ...)

## S3 method for class 'batts_fit'
nobs(object, ...)
```

Arguments

`x, object`	A `batts_boost` or `batts_bart` fit, or a `summary.batts_fit` object.
`...`	Not used.

Value

`print` returns its argument invisibly. `summary` returns an object of class `summary.batts_fit`. `nobs` returns the number of training observations.

Examples

```
set.seed(1)
fit <- batts_boost(rnorm(50), y = rnorm(50, 0.5),
                   control = boost_control(num_trees_max = 50, depth = 2),
                   quiet = TRUE)
fit
summary(fit)
nobs(fit)
```

---

batts-plot *Plot a BATTS fit*

Description

Diagnostic and summary plots drawn with base graphics.

Usage

```
## S3 method for class 'batts_boost'
plot(x, which = c("cv", "fit"), ...)

## S3 method for class 'batts_bart'
plot(x, which = c("trace", "fit"), level = 0.95, ...)
```

Arguments

x	A batts_boost or batts_bart fit.
which	The plot to draw. For a boosting fit the default is "cv" when cross-validation was used and "fit" otherwise; for a Bayesian additive trees fit it is "trace".
...	Graphical parameters passed to plot.default(), such as main, xlab, ylab or col; they replace the defaults.
level	Credible level of the band for which = "fit".

Details

For a boosting fit, which = "cv" plots the cross-validation loss, averaged over folds, against the number of trees and marks the selected number; which = "fit" plots the fitted log-density ratio log(p/q) at the training points. For a Bayesian additive trees fit, which = "trace" plots the saved draws of the loss temperature, and which = "fit" plots the posterior mean of log(p/q) at the training points with a pointwise credible band.

which = "fit" needs a fit with one variable. For more variables, evaluate the fit with predict() and plot the result.

Plotting draws no random numbers.

Value

x, invisibly.

See Also

batts_boost(), batts_bart(), batts-predict

Examples

```
set.seed(1)
fit <- batts_boost(rnorm(100), y = rnorm(100, 0.5),
                   control = boost_control(num_trees_max = 100, depth = 2),
                   quiet = TRUE)
plot(fit)
plot(fit, which = "fit")
```

---

batts-predict *Predict balancing weights and density ratios*

Description

Predict balancing weights and density ratios

Usage

```
## S3 method for class 'batts_boost'
predict(
  object,
  newdata,
  type = c("log_ratio", "ratio", "weight"),
  outside = c("error", "na", "clamp"),
  ...
)

## S3 method for class 'batts_bart'
predict(
  object,
  newdata,
  type = c("log_ratio", "ratio", "weight"),
  summary = c("mean", "median", "none"),
  level = 0.95,
  interval = FALSE,
  outside = c("error", "na", "clamp"),
  ...
)

## S3 method for class 'batts_boost'
fitted(object, type = c("log_ratio", "ratio", "weight"), ...)

## S3 method for class 'batts_bart'
fitted(
```

```
  object,
  type = c("log_ratio", "ratio", "weight"),
  summary = c("mean", "median", "none"),
  ...
)
```

Arguments

object	A batts_boost or batts_bart fit.
newdata	Points at which to evaluate the fit: a numeric matrix or data frame with the training variables, or a numeric vector (several points for a fit with one variable, otherwise one point). When the fit recorded column names and newdata has names, columns are matched by name; otherwise by position. Values must be finite. If omitted, the training observations are used.
type	"log_ratio", the default, returns the log-density ratio 2 * log(w) = log(p/q). "ratio" returns w^2 = p/q. "weight" returns the balancing weight w = sqrt(p/q) itself. The first group is in the numerator throughout.
outside	What to do with points outside the fitted domain (the training range widened by margin_scale): "error" (the default) stops and names the offending values, "na" returns NA for those points, and "clamp" moves each coordinate to the nearest boundary of the domain before evaluating.
...	Not used. A batts_boost fit has no posterior draws, so passing summary, level or interval to it is an error rather than silently ignored.
summary	For a batts_bart fit: "mean" or "median" collapses the posterior draws and "none" returns the points-by-draws matrix. The transformation implied by type is applied to each draw before summarizing, so the result summarizes the posterior of the transformed quantity.
level	Credible level used when interval = TRUE.
interval	For a batts_bart fit, return a matrix with columns fit, lower and upper instead of a vector.

Details

Prediction draws no random numbers.

Value

A numeric vector, or a matrix when summary = "none" or interval = TRUE.

Examples

```
set.seed(1)
x <- rnorm(50)
y <- rnorm(50, 0.5)
fit_boost <- batts_boost(x, y = y,
                         control = boost_control(num_trees_max = 50, depth = 2),
                         quiet = TRUE)
predict(fit_boost, c(-1, 0, 1))
```

```
predict(fit_boost, c(-1, 0, 1), type = "ratio")
head(fitted(fit_boost))
predict(fit_boost, c(-10, 0), outside = "na")

# Small demonstration; these settings are not a convergence recommendation.
fit_bart <- batts_bart(x, y = y,
                       control = bart_control(num_trees = 10, burnin = 20,
                                              draws = 21, keep_draws = TRUE),
                       quiet = TRUE)
predict(fit_bart, c(-1, 0, 1), interval = TRUE)
dim(predict(fit_bart, c(-1, 0, 1), summary = "none"))
```

`batts_bart` *Fit a two-sample density ratio by Bayesian additive trees*

Description

Draw the balancing weight `w = sqrt(p/q)` from its generalized posterior, where `p` is the density of the first group and `q` the density of the second. The log-density ratio is `2 * log(w)`.

Usage

```
batts_bart(x, group = NULL, y = NULL, control = bart_control(), quiet = FALSE)
```

Arguments

`x`	Observations in rows and variables in columns: a numeric matrix, a data frame with numeric columns, or a numeric vector for one variable. Values must be finite.
`group`	Two-valued labels, one per row of `x`: numeric, logical, character or factor. The first group (the first factor level, otherwise the smaller value in sort order; 0 for 0/1 labels) has density `p`, the numerator.
`y`	Instead of `group`, a second sample with the same columns as `x`. Then `x` has density `p` and `y` has density `q`.
`control`	Settings created by `bart_control()`.
`quiet`	Logical; suppress progress messages.

Details

Supply exactly one of `group` and `y`. Set the seed before fitting.

Use several chains and convergence diagnostics for substantive applications; a short successful run does not establish convergence or frequentist coverage. Sparse overlap and complete separation can produce prior-sensitive estimates and slow mixing. The intervals are pointwise; that an interval contains zero does not establish equality of the distributions.

Value

An object of class `batts_bart`, inheriting `batts_fit`. Besides the posterior draws at the training points, it records the call, the settings, the group labels (`group_levels`, numerator first) and the column names (`variable_names`). Use `predict()`, `fitted()` and `density_ratio()` to evaluate it; evaluating it at new points requires `bart_control(keep_draws = TRUE)`. Use `print()`, `summary()` and plot() to describe it.

See Also

`bart_control()`, `batts_boost()`, batts-predict

Examples

```
set.seed(2)
# Small demonstration; these settings are not a convergence recommendation.
fit <- batts_bart(rnorm(30), y = rnorm(30, 0.5),
                  control = bart_control(num_trees = 10, burnin = 20,
                                         draws = 21, keep_draws = TRUE),
                  quiet = TRUE)
fit
predict(fit, c(-1, 0, 1), interval = TRUE)
```

`batts_boost` *Fit a two-sample density ratio by boosting*

Description

Fit the balancing weight `w = sqrt(p/q)` by tree boosting, where `p` is the density of the first group and `q` the density of the second. The log-density ratio is `2 * log(w)`.

Usage

```
batts_boost(
  x,
  group = NULL,
  y = NULL,
  control = boost_control(),
  quiet = FALSE
)
```

Arguments

`x`	Observations in rows and variables in columns: a numeric matrix, a data frame with numeric columns, or a numeric vector for one variable. Values must be finite.
`group`	Two-valued labels, one per row of `x`: numeric, logical, character or factor. The first group (the first factor level, otherwise the smaller value in sort order; 0 for 0/1 labels) has density `p`, the numerator.

y	Instead of group, a second sample with the same columns as x. Then x has density p and y has density q.
control	Settings created by boost_control().
quiet	Logical; suppress progress messages.

Details

Supply exactly one of group and y. Set the seed before fitting; cross-validation folds and subsamples use the R random number generator.

With subsampling, the splits, gradients and leaf values of each tree use its subsample, while the balancing weights of all observations are updated. Cross-validation losses are evaluated on the held-out folds.

Value

An object of class batts_boost, inheriting batts_fit. Besides the fitted trees, it records the call, the settings, the group labels (group_levels, numerator first) and the column names (variable_names). Use predict(), fitted() and density_ratio() to evaluate it, and print(), summary() and plot() to describe it.

See Also

boost_control(), batts_bart(), batts-predict

Examples

```
set.seed(1)
p_sample <- rnorm(50)
q_sample <- rnorm(50, 0.5)
fit <- batts_boost(p_sample, y = q_sample,
                   control = boost_control(num_trees_max = 50, depth = 2),
                   quiet = TRUE)
fit
predict(fit, c(-1, 0, 1))

# The same data as one sample with group labels; the first level is p
x <- c(p_sample, q_sample)
g <- factor(rep(c("p", "q"), each = 50))
ctl <- boost_control(num_trees_max = 50, depth = 2)
set.seed(2); a <- batts_boost(x, g, control = ctl, quiet = TRUE)
set.seed(2); b <- batts_boost(p_sample, y = q_sample, control = ctl, quiet = TRUE)
identical(predict(a, 0), predict(b, 0))
```

`boost_control` *Settings for boosting fits*

Description

Collect the settings of `batts_boost()`. The defaults are the settings used in the simulations of Awaya, Xu, and Ma (2025): gradient boosting with at most 1000 trees selected by 5-fold cross-validation, depth 4, learning rate 0.01, 32 bins, margin 0.1 and per-tree subsampling of one half.

Usage

```
boost_control(
  num_trees_max = 1000,
  K_CV = 5,
  depth = 4,
  learn_rate = 0.01,
  n_bins = 32,
  margin_scale = 0.1,
  subsample_fraction = 0.5,
  min_obs_per_node = 1,
  use_gradient = TRUE
)
```

Arguments

`num_trees_max`	Maximum number of boosting trees.
`K_CV`	Number of group-stratified cross-validation folds used to select the number of trees, or zero to fit `num_trees_max` trees without cross-validation. A positive value must be an integer between two and the smaller group size; the number of trees with the smallest mean held-out loss is selected.
`depth`	Maximum depth of each tree.
`learn_rate`	Shrinkage factor of each boosting step.
`n_bins`	Number of bins used to construct the uniformly spaced candidate cut points, relative to the current node.
`margin_scale`	Fraction of each training range added at both ends of the fitted domain. A negative value uses the unit box and requires the data to lie in it.
`subsample_fraction`	Fraction of the training observations used to construct each tree, in (0, 1]. Each tree uses `ceiling(subsample_fraction * m)` of the `m` training observations (the cross-validation training folds, or all data for the final fit): one from each group and the rest uniformly without replacement. `1` uses all of them and draws no random numbers.
`min_obs_per_node`	Minimum count from each group in a child node. Splits that would leave fewer are excluded; under complete separation this can prevent informative splits.
`use_gradient`	Logical; gradient boosting when `TRUE`, forward-stagewise boosting when `FALSE`.

Value

An object of class `boost_control`.

See Also

`batts_boost()`

Examples

```
boost_control(depth = 2)
```

`density_ratio` *Density ratio of a fitted BATTS model*

Description

The density ratio `p/q` with group 0 in the numerator, that is `w^2` for the fitted balancing weight `w`. Equivalent to `predict(object, newdata, type = "ratio")`.

Usage

```
density_ratio(object, newdata, ...)

## S3 method for class 'batts_fit'
density_ratio(object, newdata, ...)
```

Arguments

`object`	A `batts_boost` or `batts_bart` fit.
`newdata`	Evaluation points; see batts-predict. If omitted, the training observations are used.
`...`	Passed to the `predict` method.

Value

A numeric vector, or a matrix for a `batts_bart` fit with `summary = "none"` or `interval = TRUE`.

See Also

batts-predict

Examples

```
set.seed(1)
fit <- batts_boost(rnorm(50), y = rnorm(50, 0.5),
                   control = boost_control(num_trees_max = 50, depth = 2),
                   quiet = TRUE)
density_ratio(fit, c(-1, 0, 1))
```

Index

bart_control, 3
bart_control(), *2*, *8, 9*
BATTS *(*BATTS-package*)*, 2
batts-methods, 4
BATTS-package, 2
batts-plot, 5
batts-predict, *2*, *6*, 6, *9, 10*, *12*
batts_bart, 8
batts_bart(), *2–4*, *6*, *10*
batts_boost, 9
batts_boost(), *2*, *4*, *6*, *9*, *11, 12*
boost_control, 11
boost_control(), *2*, *10*

density_ratio, 12
density_ratio(), *2*, *9, 10*

fitted(), *2*, *9, 10*
fitted.batts_bart *(*batts-predict*)*, 6
fitted.batts_boost *(*batts-predict*)*, 6

nobs.batts_fit *(*batts-methods*)*, 4

plot(), *2*, *9, 10*
plot.batts_bart *(*batts-plot*)*, 5
plot.batts_boost *(*batts-plot*)*, 5
plot.default(), *5*
predict(), *2, 3*, *5*, *9, 10*
predict.batts_bart *(*batts-predict*)*, 6
predict.batts_boost *(*batts-predict*)*, 6
print(), *2*, *9, 10*
print.batts_bart *(*batts-methods*)*, 4
print.batts_boost *(*batts-methods*)*, 4
print.summary.batts_fit
*(*batts-methods*)*, 4

summary(), *2*, *9, 10*
summary.batts_fit *(*batts-methods*)*, 4