Package 'bcf'

May 7, 2026

Type Package

Title Causal Inference using Bayesian Causal Forests

Version 2.0.2

Date 2024-02-23

Description Causal inference for a binary treatment and continuous outcome using Bayesian Causal Forests. See Hahn, Murray and Carvalho (2020) <doi:10.1214/19-BA1195> for additional information. This implementation relies on code originally accompanying Pratola et. al. (2013) <doi:10.48550/arXiv.1309.1906>.

License GPL-3

LinkingTo Rcpp, RcppArmadillo, RcppParallel

NeedsCompilation yes

Repository CRAN

Imports Rcpp, RcppParallel, coda (>= 0.19.3), Hmisc, parallel, doParallel, foreach, matrixStats

Suggests testthat, spelling, knitr, rmarkdown, latex2exp, ggplot2, rpart, rpart.plot, partykit

SystemRequirements GNU make

Language en-US

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Author Jared S. Murray [aut, cre], P. Richard Hahn [aut], Carlos Carvalho [aut], Peter Mariani [ctb], Constance Delannoy [ctb], Mariel Finucane [ctb], Lauren V. Forrow [ctb], Drew Herren [ctb]

Maintainer Jared S. Murray <jared.murray@mccombs.utexas.edu>

Date/Publication 2024-02-27 17:30:05 UTC

Contents

---

bcf *Fit Bayesian Causal Forests*

Description

Fit Bayesian Causal Forests

Usage

```
bcf(
  y,
  z,
  x_control,
  x_moderate = x_control,
  pihat,
  w = NULL,
  random_seed = sample.int(.Machine$integer.max, 1),
  n_chains = 4,
  n_threads = max((RcppParallel::defaultNumThreads() - 2), 1),
  nburn,
  nsim,
  nthin = 1,
  update_interval = 100,
  ntree_control = 200,
  sd_control = 2 * sd(y),
  base_control = 0.95,
  power_control = 2,
  ntree_moderate = 50,
  sd_moderate = sd(y),
  base_moderate = 0.25,
  power_moderate = 3,
  no_output = FALSE,
  save_tree_directory = ".",
  log_file = file.path(".", sprintf("bcf_log_%s.txt", format(Sys.time(),
    "%Y%m%d_%H%M%S"))),
  nu = 3,
  lambda = NULL,
  sigq = 0.9,
  sighat = NULL,
  include_pi = "control",
```

```
  use_muscale = TRUE,
  use_tauscale = TRUE,
  verbose = TRUE
)
```

Arguments

y	Response variable
z	Treatment variable
x_control	Design matrix for the prognostic function mu(x)
x_moderate	Design matrix for the covariate-dependent treatment effects tau(x)
pihat	Length n estimates of propensity score
w	An optional vector of weights. When present, BCF fits a model $y\|x\ N(f(x), \sigma^2/w)$, where $f(x)$ is the unknown function.
random_seed	A random seed passed to R's set.seed
n_chains	An optional integer of the number of MCMC chains to run
n_threads	An optional integer of the number of threads to parallelize within chain bcf operations on
nburn	Number of burn-in MCMC iterations
nsim	Number of MCMC iterations to save after burn-in. The chain will run for nsim*nthin iterations after burn-in
nthin	Save every nthin'th MCMC iterate. The total number of MCMC iterations will be nsim*nthin + nburn.
update_interval	Print status every update_interval MCMC iterations
ntree_control	Number of trees in mu(x)
sd_control	SD(mu(x)) marginally at any covariate value (or its prior median if use_muscale=TRUE)
base_control	Base for tree prior on mu(x) trees (see details)
power_control	Power for the tree prior on mu(x) trees
ntree_moderate	Number of trees in tau(x)
sd_moderate	SD(tau(x)) marginally at any covariate value (or its prior median if use_tauscale=TRUE)
base_moderate	Base for tree prior on tau(x) trees (see details)
power_moderate	Power for the tree prior on tau(x) trees (see details)
no_output	logical, whether to suppress writing trees and training log to text files, defaults to FALSE.
save_tree_directory	Specify where trees should be saved. Keep track of this for predict(). Defaults to working directory. Setting to NULL skips writing of trees.
log_file	file where BCF should save its logs when running multiple chains in parallel. This file is not written too when only running one chain.
nu	Degrees of freedom in the chisq prior on $sigma^2$
lambda	Scale parameter in the chisq prior on $sigma^2$

sigq	Calibration quantile for the chisq prior on $sigma^2$
sighat	Calibration estimate for the chisq prior on $sigma^2$
include_pi	Takes values "control", "moderate", "both" or "none". Whether to include pihat in mu(x) ("control"), tau(x) ("moderate"), both or none. Values of "control" or "both" are HIGHLY recommended with observational data.
use_muscale	Use a half-Cauchy hyperprior on the scale of mu.
use_tauscale	Use a half-Normal prior on the scale of tau.
verbose	logical, whether to print log of MCMC iterations, defaults to TRUE.

Details

Fits the Bayesian Causal Forest model (Hahn et. al. 2020): For a response variable y, binary treatment z, and covariates x, we return estimates of mu, tau, and sigma in the model

$$y_i = \mu(x_i, \pi_i) + \tau(x_i, \pi_i) z_i + \epsilon_i$$

where π_i is an (optional) estimate of the propensity score $\Pr(Z_i = 1 | X_i = x_i)$ and $\epsilon_i \sim N(0, \sigma^2)$

Some notes:

- By default, bcf writes each sample (including the trees in the ensemble) for each chain to a text file, which is used for prediction by the predict.bcf function. These text files may be large if bcf is run for many samples, so we also provide an option to suppress this output by setting no_output = TRUE. If bcf is run with no_output = TRUE, it will not be possible to predict from the model after the fact.
- x_control and x_moderate must be numeric matrices. See e.g. the makeModelMatrix function in the dbarts package for appropriately constructing a design matrix from a data.frame
- sd_control and sd_moderate are the prior SD(mu(x)) and SD(tau(x)) at a given value of x (respectively). If use_muscale = FALSE, then this is the parameter σ_μ from the original BART paper, where the leaf parameters have prior distribution $N(0, \sigma_\mu/m)$, where m is the number of trees. If use_muscale=TRUE then sd_control is the prior median of a half Cauchy prior for SD(mu(x)). If use_tauscale = TRUE, then sd_moderate is the prior median of a half Normal prior for SD(tau(x)).
- By default the prior on σ^2 is calibrated as in Chipman, George and McCulloch (2010).

Value

A fitted bcf object that is a list with elements

tau	nsim by n matrix of posterior samples of individual-level treatment effect estimates
mu	nsim by n matrix of posterior samples of prognostic function E(Y|Z=0, x=x) estimates
sigma	Length nsim vector of posterior samples of sigma

References

Hahn, Murray, and Carvalho (2020). Bayesian regression tree models for causal inference: regularization, confounding, and heterogeneous effects. https://projecteuclid.org/journals/bayesian-analysis/volume-15/issue-3/Bayesian-Regression-Tree-Models-for-Causal-Inference–Regularization-Confounding/10.1214/19-BA1195.full. (Call citation("bcf") from the command line for citation information in Bibtex format.)

Examples

```
## Not run:

# data generating process
p = 3 #two control variables and one moderator
n = 250

set.seed(1)

x = matrix(rnorm(n*p), nrow=n)

# create targeted selection
q = -1*(x[,1]>(x[,2])) + 1*(x[,1]<(x[,2]))

# generate treatment variable
pi = pnorm(q)
z = rbinom(n,1,pi)

# tau is the true (homogeneous) treatment effect
tau = (0.5*(x[,3] > -3/4) + 0.25*(x[,3] > 0) + 0.25*(x[,3]>3/4))

# generate the response using q, tau and z
mu = (q + tau*z)

# set the noise level relative to the expected mean function of Y
sigma = diff(range(q + tau*pi))/8

# draw the response variable with additive error
y = mu + sigma*rnorm(n)

# If you didn't know pi, you would estimate it here
pihat = pnorm(q)

bcf_fit = bcf(y, z, x, x, pihat, nburn=2000, nsim=2000)

# Get posterior of treatment effects
tau_post = bcf_fit$tau
tauhat = colMeans(tau_post)
plot(tau, tauhat); abline(0,1)


## End(Not run)
## Not run:
```

```
# data generating process
p = 3 #two control variables and one moderator
n = 250
#
set.seed(1)

x = matrix(rnorm(n*p), nrow=n)

# create targeted selection
q = -1*(x[,1]>(x[,2])) + 1*(x[,1]<(x[,2]))

# generate treatment variable
pi = pnorm(q)
z = rbinom(n,1,pi)

# tau is the true (homogeneous) treatment effect
tau = (0.5*(x[,3] > -3/4) + 0.25*(x[,3] > 0) + 0.25*(x[,3]>3/4))

# generate the response using q, tau and z
mu = (q + tau*z)

# set the noise level relative to the expected mean function of Y
sigma = diff(range(q + tau*pi))/8

# draw the response variable with additive error
y = mu + sigma*rnorm(n)

pihat = pnorm(q)

# nburn and nsim should be much larger, at least a few thousand each
# The low values below are for CRAN.
bcf_fit = bcf(y, z, x, x, pihat, nburn=100, nsim=10)

# Get posterior of treatment effects
tau_post = bcf_fit$tau
tauhat = colMeans(tau_post)
plot(tau, tauhat); abline(0,1)

## End(Not run)
```

`predict.bcf` *Takes a fitted bcf object produced by bcf() along with serialized tree samples and produces predictions for a new set of covariate values*

Description

This function takes in an existing BCF model fit and uses it to predict estimates for new data. It is important to note that this function requires that you indicate where the trees from the model fit are saved. You can do so using the save_tree_directory argument in bcf(). Otherwise, they will be saved in the working directory.

Usage

```
## S3 method for class 'bcf'
predict(
  object,
  x_predict_control,
  x_predict_moderate,
  pi_pred,
  z_pred,
  save_tree_directory,
  log_file = file.path(".", sprintf("bcf_log_%s.txt", format(Sys.time(),
    "%Y%m%d_%H%M%S"))),
  n_cores = 2,
  verbose = TRUE,
  ...
)
```

Arguments

object	output from a BCF predict run
x_predict_control	matrix of covariates for the "prognostic" function mu(x) for predictions (optional)
x_predict_moderate	matrix of covariates for the covariate-dependent treatment effects tau(x) for predictions (optional)
pi_pred	propensity score for prediction
z_pred	Treatment variable for predictions (optional except if x_pre is not empty)
save_tree_directory	directory where the trees have been saved
log_file	File to log progress
n_cores	An optional integer of the number of cores to run your MCMC chains on
verbose	Logical; set to FALSE to suppress extra output
...	additional arguments affecting the predictions produced.

Value

A list with elements: tau (samples of treatment effects), mu (samples of predicted control outcomes), yhat (samples of predicted values), and coda_chains (coda objects for scalar summaries)

Examples

```
## Not run:

# data generating process
p = 3 #two control variables and one moderator
n = 250
```

```
x = matrix(rnorm(n*p), nrow=n)

# create targeted selection
q = -1*(x[,1]>(x[,2])) + 1*(x[,1]<(x[,2]))

# generate treatment variable
pi = pnorm(q)
z = rbinom(n,1,pi)

# tau is the true (homogeneous) treatment effect
tau = (0.5*(x[,3] > -3/4) + 0.25*(x[,3] > 0) + 0.25*(x[,3]>3/4))

# generate the response using q, tau and z
mu = (q + tau*z)

# set the noise level relative to the expected mean function of Y
sigma = diff(range(q + tau*pi))/8

# draw the response variable with additive error
y = mu + sigma*rnorm(n)

# If you didn't know pi, you would estimate it here
pihat = pnorm(q)

n_burn = 5000
n_sim = 5000

bcf_fit = bcf(y              = y,
              z              = z,
              x_control      = x,
              x_moderate     = x,
              pihat          = pihat,
              nburn          = n_burn,
              nsim           = n_sim,
              n_chains       = 2,
              update_interval = 100,
              save_tree_directory = './trees')

# Predict using new data

x_pred = matrix(rnorm(n*p), nrow=n)

pred_out = predict(bcf_out=bcf_fit,
                   x_predict_control=x_pred,
                   x_predict_moderate=x_pred,
                   pi_pred=pihat,
                   z_pred=z,
                   save_tree_directory = './trees')


## End(Not run)
```

summary.bcf *Takes a fitted bcf object produced by bcf() and produces summary stats and MCMC diagnostics. This function is built using the coda package and meant to mimic output from rstan::print.stanfit(). It includes, for key parameters, posterior summary stats, effective sample sizes, and Gelman and Rubin's convergence diagnostics. By default, those parameters are: sigma (the error standard deviation when the weights are all equal), tau_bar (the estimated sample average treatment effect), mu_bar (the average outcome under control/z=0 across all observations in the sample), and yhat_bat (the average outcome under the realized treatment assignment across all observations in the sample).*

Description

We strongly suggest updating the coda package to our Github version, which uses the Stan effective size computation. We found the native coda effective size computation to be overly optimistic in some situations and are in discussions with the coda package authors to change it on CRAN.

Usage

```
## S3 method for class 'bcf'
summary(
  object,
  ...,
  params_2_summarise = c("sigma", "tau_bar", "mu_bar", "yhat_bar")
)
```

Arguments

object	output from a BCF predict run.
...	additional arguments affecting the summary produced.
params_2_summarise	parameters to summarise.

Value

No return value, called for side effects

Examples

```
## Not run:

# data generating process
p = 3 #two control variables and one moderator
n = 250
```

```
set.seed(1)

x = matrix(rnorm(n*p), nrow=n)

# create targeted selection
q = -1*(x[,1]>(x[,2])) + 1*(x[,1]<(x[,2]))

# generate treatment variable
pi = pnorm(q)
z = rbinom(n,1,pi)

# tau is the true (homogeneous) treatment effect
tau = (0.5*(x[,3] > -3/4) + 0.25*(x[,3] > 0) + 0.25*(x[,3]>3/4))

# generate the response using q, tau and z
mu = (q + tau*z)

# set the noise level relative to the expected mean function of Y
sigma = diff(range(q + tau*pi))/8

# draw the response variable with additive error
y = mu + sigma*rnorm(n)

# If you didn't know pi, you would estimate it here
pihat = pnorm(q)

bcf_fit = bcf(y, z, x, x, pihat, nburn=2000, nsim=2000)

# Get model fit diagnostics
summary(bcf_fit)


## End(Not run)
```

Index