Package: flexBCF (via r-universe)

October 21, 2026

Type Package

Title Fast & Flexible Implementation of Bayesian Causal Forests

Version 1.0.2

Date 2025-11-21

Description A faster implementation of Bayesian Causal Forests (BCF; Hahn et al. (2020) <doi:10.1214/19-BA1195>), which uses regression tree ensembles to estimate the conditional average treatment effect of a binary treatment on a scalar output as a function of many covariates. This implementation avoids many redundant computations and memory allocations present in the original BCF implementation, allowing the model to be fit to larger datasets. The implementation was originally developed for the 2022 American Causal Inference Conference's Data Challenge. See Kokandakar et al. (2023) <doi:10.1353/obs.2023.0024> for more details.

License GPL (>= 3)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp

URL https://github.com/skdeshpande91/flexBCF

NeedsCompilation yes

Author Sameer K. Deshpande [aut, cre] (ORCID: <https://orcid.org/0000-0003-4116-5533>), Ajinkya H. Kokandakar [aut] (ORCID: <https://orcid.org/0000-0001-6628-2272>)

Maintainer Sameer K. Deshpande <sameer.deshpande@wisc.edu>

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

Date/Publication 2025-11-25 21:32:39 UTC

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

RemoteRef HEAD

RemoteSha 4fb98b8dcf33b274613295f6aac6561cedd6dce8

Contents

average_tree_fits 2
flexBCF 3
get_tree_fits 8

Index **10**

`average_tree_fits` *Summarize posterior distribution of the average fit of tree ensembles.*

Description

Computes posterior summaries of average prognostic or treatment effects based on the fitted BCF model.

Usage

```
average_tree_fits(fit,
                  type = c("mu","tau"),
                  X_cont = matrix(0, nrow = 1, ncol = 1),
                  X_cat = matrix(0, nrow = 1, ncol = 1),
                  quantiles = c(0.05, 0.95),
                  weights = NULL,
                  verbose = TRUE,
                  print_every = floor(max(c(nrow(X_cont), nrow(X_cat)))/10))
```

Arguments

`fit`	Object returned by `flexBCF`.
`type`	Character which is equal to either "mu" or "tau". Determines which tree ensemble is used to make predictions before averaging.
`X_cont`	Matrix of continuous covariates for either the prognostic function μ (`type == "mu"`) or treatment effect function τ (`type == "tau"`) for the individuals in the group over which you wish to average. Note, predictors must be re-scaled to lie in the interval [-1,1]. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no continuous covariates
`X_cat`	Integer matrix of categorical covariates for either the prognostic function μ if (`type == "mu"`) or treatment effect function τ (`type == "tau"`) for the individuals in the group over which you wish to average. Note categorical levels should be 0-indexed. That is, if a categorical predictor has 10 levels, the values should run from 0 to 9. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no categorical covariates
`quantiles`	Vector of quantiles which you wish to compute. Default is `c(0.05, 0.95)` so that the function will return 90% posterior credible intervals for the average effect.

weights	Vector of non-negative weights for each individual. Default is a NULL, which internally gets converted to a vector of all 1's.
verbose	Logical, inciating whether to print progress to R console. Default is TRUE.
print_every	As the MCMC runs, a message is printed every print_every iterations. Default is floor(max(c(nrow(X_cont), nrow(X_cat)))/10) so that only 10 messages are printed.

Details

Returns posterior mean and quantiles for the conditional average treatment or prognostic effects averaged over the covariate values contained in X_cont and X_cat. Use the argument weights to pass observation weights, which may be useful to extrapolate findings to a different population than the one observed at training time.

Value

A list containing

mean	Posterior mean estimate of the desired average treatment (or prognostic) effect
quantiles	Posterior quantiles of the desired average treatment (or prognostic) effect.

---

flexBCF *A faster and more flexible Bayesian Causal Forests*

Description

Implements a version of Hahn et al. (2020)'s Bayesian Causal Forest (BCF) model that is faster and handles categorical outcomes more flexibly than existing BCF implementations. This implementation was created for the 2022 American Causal Inference Conference's Data Challenge. See Kokandakar et al. (2023) for details about the implementation and results from the competition.

Usage

```
flexBCF(Y_train,
        treated,
        X_cont_mu = matrix(0, nrow = 1, ncol = 1),
        X_cat_mu = matrix(0, nrow = 1, ncol = 1),
        X_cont_tau = matrix(0, nrow = 1, ncol = 1),
        X_cat_tau = matrix(0, nrow = 1, ncol = 1),
        unif_cuts_mu = rep(TRUE, times = ncol(X_cont_mu)),
        unif_cuts_tau = rep(TRUE, times = ncol(X_cont_tau)),
        cutpoints_list_mu = NULL,
        cutpoints_list_tau = NULL,
        cat_levels_list_mu = NULL,
        cat_levels_list_tau = NULL,
        sparse = TRUE,
        M_mu = 50, M_tau = 50,
        nd = 1000, burn = 1000, thin = 1,
        verbose = TRUE, print_every = floor( (nd*thin + burn))/10)
```

Arguments

Y_train	Vector of observed outcomes
treated	Vector of treatment assignments. Assumes that all control subjects are listed first (treated ==0) and then all treated subjects are listed (treated == 1).
X_cont_mu	Matrix of continuous covariates for prognostic function mu. Note, predictors must be re-scaled to lie in the interval [-1,1]. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no continuous covariates for mu.
X_cat_mu	Integer matrix of categorical covariates for prognostic function mu. Note categorical levels should be 0-indexed. That is, if a categorical predictor has 10 levels, the values should run from 0 to 9. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no categorical covariates for mu.
X_cont_tau	Matrix of continuous covariates for treatment effect function mu. Note, predictors must be re-scaled to lie in the interval [-1,1]. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no continuous covariates for tau.
X_cat_tau	Integer matrix of categorical covariates for treatment effect function tau. Note categorical levels should be 0-indexed. That is, if a categorical predictor has 10 levels, the values should run from 0 to 9. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no categorical covariates for tau.
unif_cuts_mu	Vector of logical values indicating whether cutpoints for each continuous covariate for mu should be drawn from a continuous uniform distribution (TRUE) or a discrete set (FALSE) specified in cutpoints_list_mu. Default is TRUE for each variable in X_cont_mu
unif_cuts_tau	Vector of logical values indicating whether cutpoints for each continuous covariate for tau should be drawn from a continuous uniform distribution (TRUE) or a discrete set (FALSE) specified in cutpoints_list_tau. Default is TRUE for each variable in X_cont_mu
cutpoints_list_mu	List of length ncol(X_cont_mu) containing a vector of cutpoints for each continuous predictor. By default, this is set to NULL so that cutpoints are drawn uniformly from a continuous distribution.
cutpoints_list_tau	List of length ncol(X_cont_tau) containing a vector of cutpoints for each continuous predictor. By default, this is set to NULL so that cutpoints are drawn uniformly from a continuous distribution.
cat_levels_list_mu	List of length ncol(X_cat_train) containing a vector of levels for each categorical predictor. If the j-th categorical predictor contains L levels, cat_levels_list[[j]] should be the vector 0:(L-1). Default is NULL, which corresponds to the case that no categorical predictors are available.
cat_levels_list_tau	List of length ncol(X_cat_train) containing a vector of levels for each categorical predictor. If the j-th categorical predictor contains L levels, cat_levels_list[[j]]

should be the vector `0:(L-1)`. Default is `NULL`, which corresponds to the case that no categorical predictors are available.

`sparse` Logical, indicating whether or not to perform variable selection based on a sparse Dirichlet prior rather than uniform prior; see Linero 2018. Default is `FALSE`.

`M_mu` Number of trees in the ensemble for prognostic function mu. Default is 50.

`M_tau` Number of trees in the ensemble for treatment effect function tau. Default is 50.

`nd` Number of posterior draws to return. Default is 1000.

`burn` Number of MCMC iterations to be treated as "warmup" or "burn-in". Default is 1000.

`thin` Number of post-warmup MCMC iteration by which to thin. Default is 1.

`verbose` Logical, inciating whether to print progress to R console. Default is `TRUE`.

`print_every` As the MCMC runs, a message is printed every `print_every` iterations. Default is `floor( (nd*thin + burn)/10)` so that only 10 messages are printed.

Details

The Bayesian Causal Forest (BCF) model of Hahn et al. (2022) expresses the conditional expectation of a response Y given covariates X and binary treament Z as $E[Y|X,Z] = \mu(X) + \tau(X) * Z$ and approximates both $\mu(X)$ and $\tau(X)$ with ensembles of binary regression trees. Under standard identifying assumptions, μ is the prognostic function $\mu(X) = E[Y(0)|X = x]$ and τ is the conditional average treatment effect function $\tau(X) = E[Y(1) - Y(0)|X]$, where $Y(0)$ and $Y(1)$ are the potential outcomes under control and treatment.

Like the original BCF implementation, `flex_BCF()` simulates a Markov chain to compute approximate posterior draws of $\mu(X)$ and $\tau(X)$. By avoiding several redundant internal computations and memory allocations, `flex_BCF()` is much faster and scales more gracefully to large datasets than the original BCF implementation. It is also based on a new regression tree prior (described in Deshpande (2025)) that permits more flexible "borrowing of strength" across levels of categorical predictors. Under the new prior conditional on splitting on a categorical predictor at a particular node in the tree, levels of the predictor are sent to the left and right child uniformly at random.

To reduce its memory footprint, `flex_BCF()` only returns posterior samples of the trees used to approximate $\mu(X)$ and $\tau(X)$. It does **not** return matrices containing posterior samples of these functions evaluated at each training observation. Use the functions `get_tree_fits` and `average_tree_fits`, respectively, to obtain posterior samples of these function evaluations and to compute posterior summaries of weighted averages of these evaluations.

Value

A list containing

`sigma` Vector containing ALL samples of the residual standard deviation, including burn-in.

`mu_trees` A list (of length `nd`) of character vectors (of lenght `M`) containing textual representations of the regression trees in the ensemble for the prognostic function mu. These strings are parsed by `get_tree_fits` and `average_tree_fits` to reconstruct the C++ representations of the sampled trees.

`tau_trees` A list (of length `nd`) of character vectors (of lenght `M`) containing textual representations of the regression trees in the ensemble for the prognostic function mu. These strings are parsed by `get_tree_fits` and `average_tree_fits` to reconstruct the C++ representations of the sampled trees.

`y_mean` Mean of the observed responses. Used by `get_tree_fits` and `average_tree_fits` to transform predictions to original scale.

`y_sd` Standard deviation of the observed responses. Used by `get_tree_fits` and `average_tree_fits` to transform predictions to original scale.

`varcounts_mu` Matrix that counts the number of times a covariate was used in a decision rule in the ensemble used to approximate treatment effect function tau in each MCMC iteration. Rows correspond to MCMC iterations and columns corresponding to covariates

`varcounts_tau` Matrix that counts the number of times a covariate was used in a decision rule in the ensemble used to approximate treatment effect function tau in each MCMC iteration. Rows correspond to MCMC iterations and columns corresponding to covariates

`cat_levels_list` List of containing `cat_levels_list_mu` and `cat_levels_list_tau`. Used by `get_tree_fits` and `average_tree_fits`.

References

Hahn, P.R., Murray, J.S., and Carvalho, C.M. (2020). Bayesian regression tree models for causal inference: Regularization, confounding, and heterogeneous effects (with Discussion). *Bayesian Analysis*. **15**(3):965–1056. doi:10.1214/19BA1195.

Kokandakar, A.H., Kang, H., and Deshpande, S.K. (2023). Bayesian Causal Forests & the 2022 ACIC Data Challenge: Scalability and Sensitivity. *Observational Studies*. **9**(3):29–41. doi:10.1353/obs.2023.0024.

Deshpande, S.K. (2025). flexBART: Flexible Bayesian regression trees with categorical predictors. *Journal of Computational and Graphical Statistics*. **34**(3):1117–1126. doi:10.1080/10618600.2024.2431072.

See Also

`get_tree_fits` to get posterior samples of evaluations of μ and τ and `average_tree_fits` to compute posterior means and quantiles of weighted averages of such evaluations.

Examples

```
p_cont_mu <- 5
p_cat_mu <- 5
p_mu <- p_cont_mu + p_cat_mu

p_cont_tau <- 5
p_cat_tau <- 5
p_tau <- p_cont_tau + p_cat_tau

mu_func <- function(X_cont_mu, X_cat_mu){
  scaled_X_cont <- (X_cont_mu + 1)/2 # moves it to [0,1]
```

```
  tmp1 <- sin(pi * scaled_X_cont[,1] * scaled_X_cont[,2])
  tmp2 <- (scaled_X_cont[,3] - 0.5)^2
  tmp3 <- scaled_X_cont[,4]
  tmp4 <- scaled_X_cont[,5]
  return(1 * tmp1 +
           2 * (X_cat_mu[,1] %in% c(0, 2, 4,6,8)) * tmp2 +
           1 * (X_cat_mu[,2] %in% c(1,2,3,6,7,8)) * tmp3 +
           0.5 * (X_cat_mu[,1] %in% c(1,2,3,4,5)) * tmp4)
}

tau_func <- function(X_cont_tau, X_cat_tau){
  scaled_X_cont <- (X_cont_tau + 1)/2 # moves it to [0,1]
  z1 <- scaled_X_cont[,1]
  z2 <- 1*(X_cat_tau[,1] %in% c(0,1,2,8,9))
  return(3 * z1 + (2 - 5*z2)*sin(pi * z1) - 2 * (z2))
}

n_control <- 20000
n_treat <- 10000
n <- n_control+ n_treat

treated <- c(rep(0, times = n_control), rep(1, times = n_treat))

# controls first and then treated
set.seed(129)
X_cont_mu <- matrix(runif(n * p_cont_mu, -1, 1), nrow = n, ncol = p_cont_mu)
X_cat_mu <- matrix(NA, nrow = n, ncol = p_cat_mu)
cat_levels_list_mu <- list()
for(j in 1:p_cat_mu){
  X_cat_mu[,j] <- as.integer(sample(0:9), size = n, replace = TRUE)
  cat_levels_list_mu[[j]] <- 0:9
}

# only contains stuff for treated subjects
X_cont_tau <- matrix(runif(n_treat * p_cont_tau, -1, 1), nrow = n_treat, ncol = p_cont_tau)
X_cat_tau <- matrix(NA, nrow = n_treat, ncol = p_cat_tau)
cat_levels_list_tau <- list()
for(j in 1:p_cat_mu){
  X_cat_tau[,j] <- as.integer(sample(0:9), size = n_treat, replace = TRUE)
  cat_levels_list_tau[[j]] <- 0:9
}

mu_true <- mu_func(X_cont_mu, X_cat_mu)
tau_true <- tau_func(X_cont_tau, X_cat_tau)

mean_response <- mu_true + c(rep(0, times = n_control), tau_true)

set.seed(328)
sigma <- 0.1
Y <- mean_response + sigma * rnorm(n = n, mean = 0, sd = 1)
```

```
fit <-
  flexBCF(Y_train = Y,
          treated = treated,
          X_cont_mu = X_cont_mu,
          X_cat_mu = X_cat_mu,
          X_cont_tau = X_cont_tau,
          X_cat_tau = X_cat_tau,
          cat_levels_list_mu = cat_levels_list_mu,
          cat_levels_list_tau = cat_levels_list_tau,
          sparse = TRUE, M_mu = 50, M_tau = 50)

# Get posterior evaluations of mu

mu_samples <-
  get_tree_fits(fit = fit, type = c("mu"),
                X_cont = X_cont_mu,
                X_cat = X_cat_mu)

plot(mu_true, colMeans(mu_samples), pch = 16, cex = 0.2)

tau_samples <-
  get_tree_fits(fit = fit, type = c("tau"),
                X_cont = X_cont_tau,
                X_cat = X_cat_tau)
plot(tau_true, colMeans(tau_samples), pch = 16, cex = 0.2)
```

`get_tree_fits` *Get fits of regression tree ensembles*

Description

Computes posterior samples of evaluations of regression tree ensembles

Usage

```
get_tree_fits(fit,
              type = c("mu","tau"),
              X_cont = matrix(0, nrow = 1, ncol = 1),
              X_cat = matrix(0, nrow = 1, ncol = 1),
              verbose = TRUE,
              print_every = NULL)
```

Arguments

`fit`	Object returned by `flexBCF`.

type	Character which is equal to either "mu" or "tau". Determines which tree ensemble is used to make predictions before averaging. For evaluations of the prognostic function (resp. conditional average treatment effect function), set type == "mu" (resp. type == "tau").
X_cont	Matrix of continuous covariates for either the prognostic function μ if (type == "mu") or CATE function τ (type == "tau"). Note, predictors must be re-scaled to lie in the interval [-1,1]. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no continuous covariates
X_cat	Integer matrix of categorical covariates for either the prognostic function μ if (type == "mu") or CATE function τ (type == "tau"). Note categorical levels should be 0-indexed. That is, if a categorical predictor has 10 levels, the values should run from 0 to 9. Assumes that subjects are ordered so that all control subjects are listed before all treated subjects. Default is a 1x1 matrix, which signals that there are no categorical covariates
verbose	Logical, inciating whether to print progress to R console. Default is TRUE.
print_every	A status message is printed every print_every MCMC iterations. Whens print_every is NULL, get_tree_fits sets the value internally to be 1/10 the number of MCMC iterations.

Details

To reduce its memory footprint, flexBCF only returns posterior samples of the trees used to approximate $\mu(X)$ and $\tau(X)$. It does **not** return matrices containing posterior samples of these functions evaluated at each training observation. The function get_tree_fits() takes an object returned by flexBCF and evaluates the trees at the covariate values supplied by X_cont and X_cat.

Value

A matrix whose rows correspond to MCMC iterations and who columns correspond to observations (i.e., rows of X_cont or X_cat).

Index

average_tree_fits, 2, *5, 6*

flexBCF, *2*, 3, *8, 9*

get_tree_fits, *5, 6*, 8