Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

bartcs: Bayesian Additive Regression Trees for Confounder Selection in R

Yeonghoon Yoo
SungKyunKwan University

Chanmin Kim
SungKyunKwan University

Abstract

This article presents an overview of the **bartcs** R package, which employs a Bayesian additive regression trees-based method for selecting confounders. It uses a Dirichlet distribution as a common variable selection probability prior, updating both the exposure and outcome models simultaneously while fitting tree priors for each. This data-driven method determines which variables (i.e., confounders) affect both models by assigning more posterior weight to them. It supports continuous and binary exposure variables, as well as continuous outcome variables, and is written in C++ for improved computational speed. Additionally, it can take advantage of multiple threads for parallel computing if OpenMP is available on the platform.

Keywords: Bayesian nonparametric, causal inference, high-dimensional confounders, continuous outcome.

1. Introduction

In observational studies, drawing causality always relies on the ignorability assumption (Rosenbaum and Rubin 1983b) that all confounders are included in the adjustment procedure. In many recent applications, the number of potential confounders is often enormous, making it difficult to select the optimal set of true confounders among them. In this context, the optimal set is a confounder set with an appropriate level of uncertainty that reduces bias in estimating the final causal effect.

The main distinction between confounder selection and the traditional variable selection method is that variables that meet the unconfoundedness assumption should be chosen. Several criteria need to be met by the selected confounders in order to reduce the bias of estimated causal effects. Among them, "disjunctive cause criterion"(VanderWeele 2019) requires that the chosen variables be related to exposure and/or outcome. A better condition than this

is "disjunctive cause criterion without instruments"(VanderWeele 2019), which removes the variables related to exposure but not directly associated with outcome. Manually identifying a set of confounders that meet these criteria among a large number of potential confounders is challenging.

Methods based on data and statistical models for performing such confounder selection have recently been proposed. One such method is the Bayesian adjustment for confounding (BAC) method proposed by Wang, Parmigiani, and Dominici (2012); Lefebvre, Delaney, and McClelland (2014), which connects exposure and outcome models through common variable inclusion indicator variables to identify confounders. Wang, Dominici, Parmigiani, and Zigler (2015) later modified the BAC method to work with generalized linear outcome models. Wilson and Reich (2014) suggested a method based on decision theory with a similar goal, which performs well for a variety of sample sizes. In terms of selecting relevant covariates for use in propensity score, Shortreed and Ertefaie (2017) proposed the outcome-adaptive LASSO method. In addition, Häggström (2018) proposed a method for identifying the causal structure and estimating the causal effect using a probability graphical model.

Despite the advantages of the previously mentioned methods, they each have limitations as outlined in Table 1. To address these shortcomings, Kim, Tec, and Zigler (2023) proposed a novel Bayesian non-parametric model that aims to overcome these limitations. They suggested a new method that employs Bayesian additive regression trees (BART; Chipman, George, McCulloch *et al.* (2010)) with a shared prior for the selection probabilities, which links the exposure and outcome models. This approach allows for the flexibility and precision of a Bayesian nonparametric model, while also identifying and integrating covariates that are related to both the exposure and outcome into the final estimator. This paper introduces **bartcs**, a new R (R Core Team 2021) package developed by Yoo (2023) that implements the Bayesian additive regression trees method for confounder selection proposed by Kim *et al.* (2023). The package, which is written in C++ and integrated into R via Rcpp for fast computation and easy use, can be downloaded from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/package=bartcs.

In this paper, we provide an overview of the package, including installation instructions, usage examples, and a demonstration of its performance on simulated data. We also include a comparison with other existing confounder selection methods. Our aim is to provide researchers with a useful tool for identifying relevant confounders in their causal inference studies and to enable them to make more accurate causal inferences.

2. Overview of model

We first express causal estimation within a potential outcome framework (Rubin 1974). For each unit $i = 1, \cdots, N$, the potential outcome for the i-th unit is defined as $Y_i(a)$, representing the potential value of the outcome Y_i that could be observed under the exposure $A = a$. The target causal estimand is

$$\Delta(a, a') = E[Y(a) - Y(a')],$$

which represents the average difference between two potential outcomes under two different exposure levels a and a'. However, unlike randomized trials, the exposure assignment is not randomized in observational studies, making it impossible to directly identify either $E[Y(a)]$ or $E[Y(a')]$ from observed data. With a proper set of confounders $\mathbf{X}_i$, the following strong

Package	Prog. Lang.	Description
bacr (Wang 2016)	R	Assume (generalized-) linear models (i.e., *parametric* models) for exposure and outcome. Supports binomial, Poisson, Gaussian exposure and outcome.
BayesPen (Wilson, Bondell, and Reich 2014)	R	Assume linear models (i.e., *parametric* models) for exposure and outcome. Support continuous outcome.
CovSelHigh (Häggström 2017)	R	Confounder selection performed via either Markov/Bayesian networks (Model-free selection of confounders).
BART† (McCulloch, Sparapani, Spanbauer, Gramacy, Pratola, Plummer, Best, Cowles, and Vines 2023)	C++	Incorporate the Dirichlet sparse prior of Linero (2018) for variable selection in the BART outcome model. Support continuous outcome.
bartcs (Yoo 2023)	C++	Use BART outcome and exposure models with the common Dirichlet prior for confounder selection. Support binary and continuous exposure, and continuous outcome.

Table 1: Summary of different confounder selection methods. †Note that this model (DBART) does not primarily focus on confounder selection, but rather variable selection, and this variable selection functionality is enabled by setting `sparse=TRUE` in `wbart` function from the **BART** package.

ignorable treatment assignment assumption (Rosenbaum and Rubin 1983a) holds

$$Y_i(a) \perp A_i | \mathbf{X}_i,$$

and $0 < Pr(A_i = 1|\mathbf{X}_i = \mathbf{x}) < 1$ for all $\mathbf{x}$; $i = 1, \cdots, N$. With this assumption in place, we can represent the causal effect by the following equation of the observable quantities:

$$\Delta(a, a'; \mathbf{x}) = E[Y|A = a, \mathbf{X} = \mathbf{x}] - E[Y|A = a', \mathbf{X} = \mathbf{x}],$$

and finally identify and estimate the target estimand $\Delta(a, a')$ by averaging over confounders $\mathbf{X}$. Thus, the two key tasks in estimating causal effects are identifying the appropriate confounders among a potentially large set of covariates, and determining the outcome model (i.e., $E[Y|A = a, \mathbf{X} = \mathbf{x}]$) with flexibility and precision. The **bartcs** R package was developed to address these challenges by utilizing Bayesian additive regression trees (BART) models for confounder selection and causal effect estimation.

2.1. Overview of BART

The BART model (Chipman *et al.* 2010) is an ensemble of decision trees that can be repre-

sented by the following equation:

$$y_i = f(\mathbf{X}_i) + \epsilon_i \approx \sum_{t=1}^{T} g(\mathbf{X}_i; \mathcal{T}_t, \mathcal{M}_t) + \epsilon_i,$$

where ϵ_i follows a normal distribution with mean 0 and variance σ^2, and $g(\mathbf{X}_i; \mathcal{T}_t, \mathcal{M}_t)$ is a function that maps the tree structure and parameters to the response, for all $i = 1, \cdots, N$. For each of T distinct trees, $\mathcal{T}_t$ represents the structure of the t-th tree and $\mathcal{M}_t = \{\mu_{t,1}, \mu_{t,2}, \cdots, \mu_{t,n_t}\}$ represents its mean parameters at the terminal nodes. Each tree has internal nodes that are split based on a "splitting variable" X_j and "splitting value" c (Figure 1).

In the Markov Chain Monte Carlo (MCMC) update, Bayesian backfitting(Hastie, Tibshirani *et al.* 2000) is utilized within a Metropolis-within-Gibbs sampler. This involves fitting each tree in the ensemble sequentially, using the residual responses: $\mathbf{R}_{-t} := \mathbf{y} - \sum_{j \neq t} g(\mathbf{X}; \mathcal{T}_j, \mathcal{M}_j)$ where $\mathbf{R}_{-t}$ denotes unexplained outcome residuals for the t-th tree. In each iteration of the MCMC update, a new tree structure is proposed by randomly selecting one of three possible tree alterations:

> GROW: Choose a terminal node at random, and create two new terminal nodes. This process involves randomly selecting a predictor, X_j, and its associated "splitting value," c, to create the two new terminal nodes.
>
> PRUNE: Pick an internal node at random where both children are terminal nodes (known as a "singly internal node" (Kapelner and Bleich 2016)) and remove both of its children (thus making it a terminal node).
>
> CHANGE: Select an internal node at random and modify its splitting variable and value according to the priors.

Specifically, when using the grow and change alterations, a new covariate is randomly selected from a set of P available covariates as the splitting variable, according to the assumed prior. The original BART model used a uniform prior of $\{1/P, 1/P, \cdots, 1/P\}$ on the selection probabilities $\boldsymbol{s} = (s_1, s_2, \cdots, s_P)$. However, to promote sparsity, Linero (2018) proposed using a Dirichlet prior $(s_1, s_2, \cdots, s_P) \sim \mathcal{D}(\alpha/P, \cdots, \alpha/P)$. For a detailed explanation of the parameter setting and the steps involved in computing the posterior, refer to Kim *et al.* (2023).

2.2. BART confounder selection

The **bartcs** package in R is designed for selecting confounding variables, particularly when a large number of potential confounding variables are present, and for estimating the average treatment effect (ATE) given the chosen set of confounding variables. To accomplish this, the package uses the Bayesian additive regression trees (BART) model to specify the exposure and outcome models as follows:

$$P(A_i = 1) = \Phi\left(\sum_{t=1}^{T} g_a(\mathbf{X}_i; \mathcal{T}_t, \mathcal{M}_t)\right) \quad (1)$$

$$Y_i = \sum_{t=1}^{T} g_y(A_i, \mathbf{X}_i; \mathcal{T}'_t, \mathcal{M}'_t) + \epsilon_i, \quad (2)$$

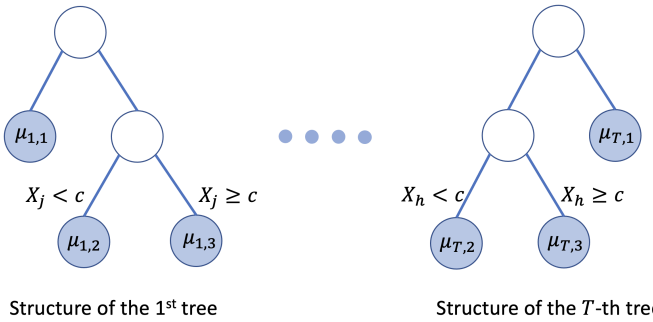


Figure 1: The tree structures consist of T trees, each with nodes represented by circles. Terminal nodes, shown in blue, have μ values. The outcome estimate $\hat{Y}$ of each observation is calculated by adding up the μ values of the terminal nodes where the observation falls within each tree. The method used to split each internal node into two different children nodes is the "splitting rule," which consists of a "splitting variable" (i.e., X_j) and a "splitting value" (i.e., c).

where $\epsilon_i \sim N(0, \sigma^2)$ for $i = 1, \cdots, N$. In Equation 1, $\Phi(\cdot)$ is the standard normal cumulative distribution function. Note that it is required to replace Equation 1 with $A_i = \sum_{t=1}^{T} g_a(\mathbf{X}_i; \mathcal{T}_t^\star, \mathcal{M}_t^\star) + \epsilon_i^\star, \epsilon_j^\star \sim N(0, \tau^2)$ when considering a continuous exposure. We incorporate a common sparsity-inducing Dirichlet prior $(s_1, s_2, \cdots, s_P) \sim \mathcal{D}(\alpha/P, \cdots, \alpha/P)$ on Equation 1 and Equation 2 resulting in a conjugate update $(s_1, s_2, \cdots, s_P) \sim \mathcal{D}(\alpha/P + n_1^a + n_1^y, \cdots, \alpha/P + n_P^a + n_P^y)$ where n_j^a and n_j^y are the numbers of splits on potential confounder X_j in Equation 1 and Equation 2, respectively (Figure 2).

If a particular covariate, X_j, is frequently used as a splitting variable in either the model for A or the model for Y, the model will assign more weight to the selection probability s_j through larger values of n_j^a or n_j^y. This means that the selection probabilities will tend to favor covariates that have a relationship with A, Y, or both A and Y. The final confounders chosen for effect estimation in the model for Y will be those that were proposed for splitting through this prior and were accepted during the updating step of the model for Y, which will further prioritize variables that have a relationship with Y. This characteristic satisfies the "disjunctive cause criterion without instruments" in confounder selection. Please see Kim *et al.* (2023) for further discussion on this prioritization.

Separate outcome models

For a binary exposure, we separate the outcome model in Equation 2 into two distinct models, in order to align the dimensions of the covariates in both the exposure and outcome models (note that Equation 2 includes exposure A as an additional covariate). For each $A = a \in \{0, 1\}$,

$$Y_i = \sum_{t=1}^{T} g_y^a(\mathbf{X}_i; \mathcal{T}_t^a, \mathcal{M}_t^a) + \epsilon_i^a, \quad \epsilon_i^a \sim N(0, \sigma_a^2), \tag{3}$$

for $i \in N_a$ where N_a denotes a set of units under each exposure arm $a \in \{0, 1\}$. A sparsity-inducing prior is applied to $(s_1, s_2, \cdots, s_P)$, which is shared among three models: one for exposure and two for outcomes. The resulting update based on this prior is $(s_1, s_2, \cdots, s_P) \sim \mathcal{D}(\alpha/P + n_1^a + n_1^{y_1} + n_1^{y_0}, \cdots, \alpha/P + n_P^a + n_P^{y_1} + n_P^{y_0})$, where $n_j^{y_1}$ and $n_j^{y_0}$ represent the numbers

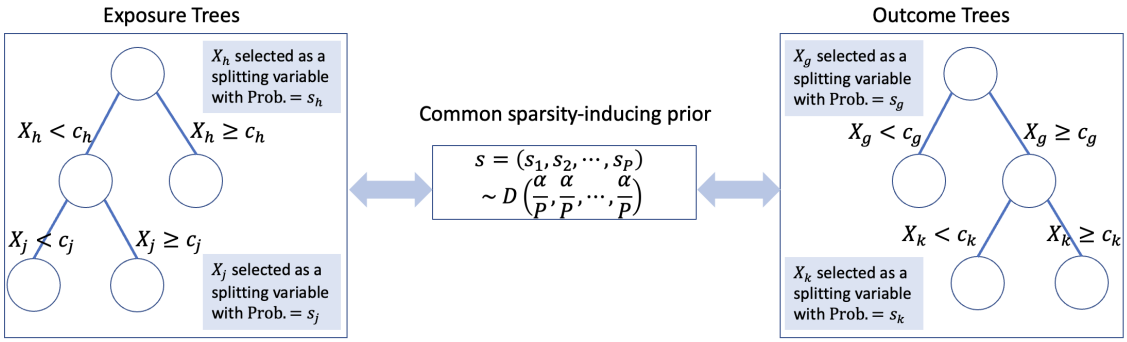


Figure 2: A shared sparsity-inducing prior for the selection probability vector connects the exposure model and outcome model, enabling the selection of the splitting variables in both models. The selection probability vector is updated based on the number of splitting variables used to describe each tree.

of splits on the confounder X_j in two separate outcome models.

Single outcome model

Using two separate outcome models for two exposure levels, as outlined in Hill (2011) and Hahn, Murray, and Carvalho (2020), can result in biased estimates if there is a lack of common support in confounders. While a single outcome model can be a viable alternative, it can be challenging to apply a shared sparsity-inducing prior to $(s_1, s_2, \cdots, s_P)$ due to differences in covariate dimensions between the exposure and outcome models. Let $\boldsymbol{s} = (s_0, s_1, s_2, \cdots, s_P)$ represent the selection probabilities, with s_0 denoting the probability of exposure A used in the outcome model. To apply this vector to the exposure model, s is transformed to $\boldsymbol{s}' = (s_1/(1-s_0), s_2/(1-s_0), \cdots, s_P/(1-s_0))$. Then, updating $\boldsymbol{s}$ is based on the following equation (likelihood $\times$ prior):

$$Q = \left(\frac{1}{1-s_0}\right)^{\sum_{j=1}^{P} n_j^a} s_0^{n_0^y+\alpha/P-1} s_1^{n_1^y+n_1^a+\alpha/P-1} \cdots s_P^{n_P^y+n_P^a+\alpha/P-1},$$

using the Metropolis-Hastings algorithm. The proposal distribution for $\boldsymbol{s}$ is designed to follow the full conditional in the separate outcome model, $\mathcal{D}(n_0^y + c + \alpha/P, n_1^a + n_1^y + \alpha/P, n_2^a + n_2^y + \alpha/P, \cdots, n_P^a + n_P^y + \alpha/P)$, and a positive value c is added to prevent proposals for infrequent exposure. For a detailed explanation of the posterior computation step, refer to the appendix and Kim *et al.* (2023).

Given the M set of posterior samples for BART parameters, the causal effect estimand $\Delta(a, a')$ can be estimated using either the separate model or the single model. For the separate outcome model, the estimate is obtained by

$$\hat{\Delta}(1,0) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{M}\sum_{m=1}^{M}\left\{\sum_{t=1}^{T} g_y^{1,(m)}(\mathbf{X}_i; \mathcal{T}_t^1, \mathcal{M}_t^1) - \sum_{t=1}^{T} g_y^{0,(m)}(\mathbf{X}_i; \mathcal{T}_t^0, \mathcal{M}_t^0)\right\}\right],$$

where $g_y^{a,(m)}$ is the m-th posterior samples for $A = a \in \{0, 1\}$. For the single outcome model,

the estimate is obtained by

$$\hat{\Delta}(1,0) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{M}\sum_{m=1}^{M}\left\{\sum_{t=1}^{T} g_y^{(m)}(1, \mathbf{X}_i; \mathcal{T}_t, \mathcal{M}_t) - \sum_{t=1}^{T} g_y^{(m)}(0, \mathbf{X}_i; \mathcal{T}_t, \mathcal{M}_t)\right\}\right],$$

where $g_y^{(m)}$ is the m-th posterior samples.

3. Preliminaries

The **bartcs** R package makes it easy to implement the confounder selection process described in the previous section. It includes two main functions, `separate_bart()` for the separate outcome model and `single_bart()` for the single outcome model. The package not only offers a summary of the estimated causal effects but also includes visualizations of posterior inclusion probabilities and convergence.

bartcs offers multi-threading support through Open Multi-Processing (OpenMP), an API for shared memory parallel programming that manages thread creation, management, and synchronization for efficient data and computation division among different threads. This allows **bartcs** to specify intensive computations as parallel regions, leading to improved computational efficiency through parallel computing.

The package **bartcs** is available under the general public license (GPL $\geq$ 3) from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/package=bartcs and can be installed and loaded into the current R session as follows:

```
R> install.packages("bartcs")
R> library("bartcs")
```

In the following sections, we will showcase the practical usage of the features in the **barcs** package using simulated examples and the IHDP data.

4. Simulated example

As a simple example of the **bartcs** package, we use a simulated dataset from Scenario 1 in Kim *et al.* (2023) to illustrate its features. The data-generating model incorporates both the nonlinear propensity score and outcome models, and serves to evaluate the ability to detect 5 true confounding variables out of a huge set of possible covariates, along with the precision of the model's estimation. The dataset consists of 300 observations with 100 potential confounders ($X_1 - X_{100}$), each generated from a normal distribution with mean 0 and variance 1. Of the 100 possible confounders, $X_1 - X_5$ are true confounders. The outcome model includes the five true confounders and two additional predictors, X_6 and X_7 as follows:

$$\begin{aligned} P(A_i = 1) &= \Phi(0.5 + 0.5h_1(X_{i,1}) + 0.5h_2(X_{i,2}) - 0.5|X_{i,3} - 1| + 1.5X_{i,4}X_{i,5}) \\ Y_i &\sim N(\mu(\boldsymbol{X}_i),\, 0.3^2) \\ \mu(\boldsymbol{X}_i) &= h_1(X_{i,1}) + 1.5h_2(X_{i,2}) - A_i + 2|X_{i,3} + 1| + 2X_{i,4} + \exp(0.5X_{i,5}) \\ &\quad -0.5A_i|X_{i,6}| - A_i|X_{i,7} + 1| \end{aligned}$$

where $h_1(x) = (-1)^{I(x<0)}$ and $h_2(x) = (-1)^{I(x\geq 0)}$ for $i = 1, \cdots, 300$. The data was generated with the following code:

```
R> set.seed(42)
R> N <- 300
R> P <- 100
R> cov <- list()
R> for (i in 1:P) {
+    cov[[i]] <- rnorm(N, 0, 1)
+  }
R> X <- do.call(cbind, cov)
R> h1 <- ifelse(X[, 1] < 0, 1, -1)
R> h2 <- ifelse(X[, 2] < 0, -1, 1)
R> prob <- pnorm(0.5 + h1 + h2 - 0.5 * abs(X[, 3] - 1) +
+                 1.5 * X[, 4] * X[, 5])
R> Trt <- rbinom(N, 1, prob)
R> mu1 <- 1 * h1 + 1.5 * h2 - 1 + 2 * abs(X[, 3] + 1) +
+    2 * X[, 4] + exp(0.5 * X[, 5]) -
+    0.5 * 1 * abs(X[, 6]) - 1 * 1 * abs(X[, 7] + 1)
R> mu0 <- 1 * h1 + 1.5 * h2 - 0 + 2 * abs(X[, 3] + 1) +
+    2 * X[, 4] + exp(0.5 * X[, 5]) -
+    0.5 * 0 * abs(X[, 6]) - 1 * 0 * abs(X[, 7] + 1)
R> Y1 <- rnorm(N, mu1, 0.3)
R> Y0 <- rnorm(N, mu0, 0.3)
R> Y <- Trt * Y1 + (1 - Trt) * Y0
```

With a generated data set, we fit the BART confounder selection model (the separate outcome model) using `separate_bart()`.

```
R> library("bartcs")
R> separate_fit <- separate_bart(
+    Y = Y, trt = Trt, X = X, num_tree = 200, num_chain = 4,
+    num_burn_in = 10000, num_thin = 5, num_post_sample = 2000
+  )
```

The following are the main arguments used in the `separate_bart()` function call:

- `Y` represents a vector of observed outcome values.
- `trt` denotes a vector of exposure(treatment) values, which can be either binary or continuous depending on the function. Binary treatment values need to be either 0 or 1.
- `X` is a data frame of potential confounders.

The following are the remaining settings for the fit: 4 MCMC chains (`num_chain`) with 200 trees (`num_tree`) are used. Each MCMC chain runs 20000 iterations, with 10000 burn-in iterations (`num_burn_in`) and a thinning factor of 5 (`num_thin`). There are other optional arguments available for hyper-parameter settings with the following default values:

- $\alpha = 0.95$ (`alpha`) and $\beta = 2$ (`beta`): these govern the probability that a node at depth d is nonterminal as follows
$$\alpha(1 + d)^{-\beta}.$$

- $\nu = 3$ (`nu`) and $q = 0.95$ (`q`): to set a conjugate prior for the variance σ^2 with $\sigma^2 \sim \nu\lambda/\chi^2_\nu$, we use the following equation to determine the values $P(\sigma < \hat{\sigma}) = q$, where $\hat{\sigma}$ represents the residual standard deviation obtained from a linear regression of Y on X.
- $P_{\text{GROW}} = 0.28, P_{\text{PRUNE}} = 0.28, P_{\text{CHANGE}} = 0.44$ (`step_prob = c(0.28, 0.28, 0.44)`): probabilities of three tree alteration steps.
- `dir_alpha` $= 5$: this is an initial value for hyperparameter α in the sparsity inducing Dirichlet prior $\mathcal{D}(\alpha/P, \alpha/P, \cdots, \alpha/P)$.

```
R> separate_fit

`bartcs` fit by `separate_bart()`

          mean       2.5%      97.5%
ATE -2.2851546 -2.6022894 -1.9692134
Y1   0.7195622  0.4663024  0.9833689
Y0   3.0047169  2.8116436  3.1946016
```

The `separate_bart()` returns a S3 `bartcs` object. A `bartcs` object includes the posterior means and 95% credible intervals for the sample average treatment effect (ATE), and the potential outcomes $Y(1)$ and $Y(0)$. It is important to note that the true values for the ATE, $E[Y(1)]$, and $E[Y(0)]$ are -2.55, 0.64, and 3.19 respectively, and the 95% credible intervals produced by the `separate_bart()` function include these values.

For a more in-depth understanding of the output, the `summary()` function can be used. It provides details regarding the treatment values, tree structure, MCMC chain, and outcomes for each of the chains.

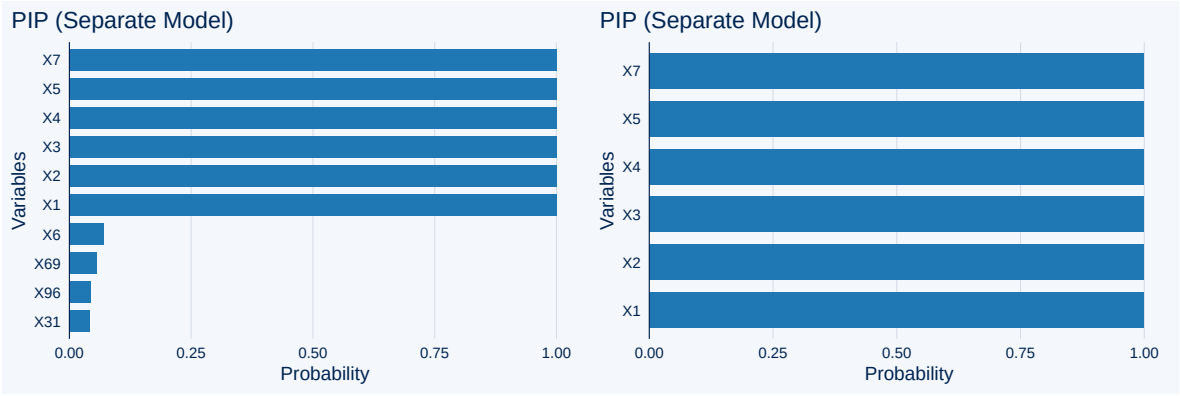
```
R> summary(separate_fit)

`bartcs` fit by `separate_bart()`

Treatment Value
  Treated group    :      1
  Control group    :      0

Tree Parameters
  Number of Tree   :    200              Value  of alpha    :   0.95
  Prob.  of Grow   :   0.28              Value  of beta     :      2
  Prob.  of Prune  :   0.28              Value  of nu       :      3
  Prob.  of Change :   0.44              Value  of q        :   0.95

Chain Parameters
  Number of Chains :      4              Number of burn-in  :  10000
  Number of Iter   :  20000              Number of thinning :      5
  Number of Sample :   2000
```



(a) Plotting PIP with `top_n` argument (b) Plotting PIP with `threshold` argument

Figure 3: Posterior inclusion probability (PIP) plots

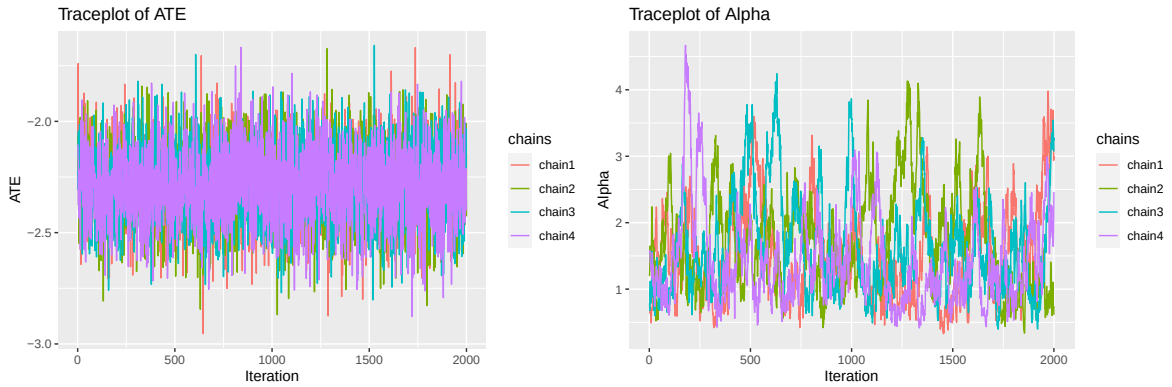
```
Outcome
 estimand chain       2.5%         1Q       mean     median         3Q      97.5%
      ATE     1 -2.6070044 -2.3892357 -2.2830389 -2.2800555 -2.1765359 -1.9757766
      ATE     2 -2.6013548 -2.4017854 -2.2877997 -2.2877071 -2.1798863 -1.9611401
      ATE     3 -2.5961329 -2.3952700 -2.2794876 -2.2793208 -2.1609143 -1.9644475
      ATE     4 -2.6090523 -2.4001084 -2.2902924 -2.2923171 -2.1812900 -1.9761443
      ATE   agg -2.6022894 -2.3965077 -2.2851546 -2.2842764 -2.1748201 -1.9692134
       Y1     1  0.4705203  0.6322748  0.7174467  0.7174147  0.8027479  0.9668359
       Y1     2  0.4707973  0.6305094  0.7223111  0.7213076  0.8153911  0.9851455
       Y1     3  0.4653391  0.6277828  0.7190511  0.7194586  0.8080547  0.9804701
       Y1     4  0.4614500  0.6273396  0.7194400  0.7175295  0.8087480  0.9920899
       Y1   agg  0.4663024  0.6292846  0.7195622  0.7185828  0.8087121  0.9833689
       Y0     1  2.8082437  2.9361088  3.0004857  2.9998629  3.0664869  3.1897135
       Y0     2  2.8189069  2.9442181  3.0101107  3.0107896  3.0778268  3.2013420
       Y0     3  2.8002284  2.9362280  2.9985387  2.9972708  3.0646989  3.1920314
       Y0     4  2.8210957  2.9427012  3.0097324  3.0133450  3.0772579  3.1960383
       Y0   agg  2.8116436  2.9404458  3.0047169  3.0053406  3.0713420  3.1946016
```

For each estimand category, there are five results (rows) that represent the output from each of the 4 MCMC chains and an aggregated output.

For visualization purposes, there are two options available as S3 methods for the `bartcs` object. The first option is the posterior inclusion probability (PIP) plot. PIP is the probability that a variable is used as a splitting variable, and can be interpreted as the importance of a variable. The `inclusion_plot()` function is a wrapper for the `bar_chart()` function from the **ggcharts** package, allowing the use of its arguments to customize the plot. The recommended arguments to use are `top_n` and `threshold`.

```
R> plot(separate_fit, method = "pip", top_n = 10)
R> plot(separate_fit, method = "pip", threshold = 0.5)
```

In Figure 3, the argument `top_n` allows us to select variables with the top `top_n` highest PIPs. The argument `threshold` displays variables with PIP greater than `threshold`. From a



(a) Traceplot of ATE

(b) Traceplot of `dir_alpha` parameter

Figure 4: Traceplots for multiple MCMC chains

decision-theoretical perspective (Barbieri and Berger 2004; Linero 2018), variables with PIPs larger than 0.5 can be considered chosen confounders. It is worth noting that the five true confounders $X_1 - X_5$ are all correctly selected as true confounders with PIPs of 1, along with one extra predictor X_7 in the outcome model.

The second option for visualization is the traceplot, which is mainly used to check MCMC convergence. The function provides a traceplot of the average treatment effect (ATE) for each MCMC chain. Traceplots of other parameters such as `dir_alpha` (the hyperparameter α in the sparsity-inducing Dirichlet prior $\mathcal{D}(\alpha/P, \cdots, \alpha/P)$) and `sigma2_out` (the variance parameter in the outcome model) are also available by using the argument `parameter`.

```
R> plot(separate_fit, method = 'trace')
R> plot(separate_fit, method = 'trace', parameter = 'dir_alpha')
```

In Figure 4, the traceplots of the ATE and `dir_alpha` parameters are shown for four different MCMC chains. Regarding the `dir_alpha` parameter (α), the actual value used as the hyper-parameter for the Dirichlet prior is obtained by dividing the total number of potential confounders, denoted as P (i.e., α/P). Considering the simulation data setting where $P = 100$ is used, the hyper-parameter to be estimated is a significantly small value, which is $\alpha/100$. Therefore, compared to the variation observed in the traceplot, the variation of the actual α/P can be interpreted as considerably smaller. While visual inspection using traceplots is convenient, it is advised to utilize the gelman-rubin diagnostics offered by the `gelman.diag()` function in the **coda** R package (Plummer, Best, Cowles, Vines, Sarkar, Bates, Almond, and Magnusson 2020) for a more thorough evaluation of convergence, as demonstrated in the following section.

We evaluated the performance of **bartcs** in comparison to other models, including those generated by the **bacr** R package (Wang 2016) that inspired our model development. The **bacr** package is easily installed via CRAN and loaded into the current R session as follows:

```
R> install.packages("bacr")
R> library("bacr")
```

To fit the model of this package, we used the `bac()` function where the input data needs to be provided in the form of a data frame. To fit the exposure and outcome models in this case, a generalized linear model is used, and it is necessary to specify the family of the model based on the data type (e.g.. `familyX="binomial"` and `familyY="gaussian"`). The MCMC algorithm was run for 10000 iterations after discarding the first 10000 iterations as burn-ins. Additionally, no interaction between the exposure and each confounder was assumed.

```
R> Z <- as.data.frame(cbind(Y,Trt,X))
R> fit.bac <- bac(
+    data = Z, exposure = "Trt", outcome = "Y",
+    confounders = paste("V", 3:(P + 2), sep = ""),
+    interactors = NULL, familyX = "binomial", familyY = "gaussian",
+    omega = Inf, num_its = 20000, burnM = 10000, burnB = 10000, thin = 5
+  )
```

The result can be checked through the `summary()` function as follows:

```
R> summary(fit.bac)

BAC objects:

Exposure effect estimate:
        posterior mean     95% posterior interval
                  -1.6               (-2.1, -1.3)


Covariates with posterior inclusion probability > 0.5:
    posterior inclusion probability
V3                          1.00000
V4                          1.00000
V5                          1.00000
V6                          1.00000
V7                          1.00000
V99                         0.92100
V14                         0.70305
V54                         0.67480
V90                         0.62345
```

The posterior mean of the ATE was estimated to be -1.6, which was significantly different from the true ATE value of -2.55. Moreover, the 95% credible interval $(-2.1, -1.3)$ did not include the true value. When considering the importance of selected confounders based on the posterior inclusion probability, **bacr** included all important confounders $X_1 - X_5$ (that is, $V3 - V7$ in the summary), but also added X_{12}, X_{52}, X_{88}, and X_{97} (that is, $V14, V54, V90, V99$ in the summary) with high PIPs, which were not true confounders. Notably, X_6 and X_7, which are additional predictors of the outcome model, were not included. This result may be attributed to the fact that **bacr** relies on a parametric model and therefore may struggle to account for the non-linear and complex data structure.

4.1. Connection to coda package

To summarize the results, generic functions such as `summary()` and `plot()` were adapted to work on the `bartcs` objects. Additionally, `mcmc.list` objects were included as components in the `bartcs` object to allow for the use of functions from the **coda** R package (Plummer *et al.* 2020). The `mcmc_list` component of the **bartcs** object can produce summary statistics for each of $E[Y(1)], E[Y(0)], ATE$ using the summary function and generate trace plots and posterior densities for parameters using the plot function. Figure 5 displays plot of `mcmc_list` based on **coda** package.

```
R> summary(separate_fit$mcmc_list)

Iterations = 10005:20000
Thinning interval = 5
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                 Mean        SD  Naive SE Time-series SE
ATE         -2.285155 0.1639173 1.833e-03      2.489e-03
Y1           0.719562 0.1328857 1.486e-03      2.122e-03
Y0           3.004717 0.0977850 1.093e-03      1.790e-03
dir_alpha    1.576421 0.7317213 8.181e-03      6.836e-02
sigma2_out1  0.001731 0.0003359 3.756e-06      5.533e-06
sigma2_out0  0.001310 0.0002334 2.609e-06      3.784e-06

2. Quantiles for each variable:

                2.5%       25%       50%       75%     97.5%
ATE         -2.6022894 -2.396508 -2.284276 -2.174820 -1.969213
Y1           0.4663024  0.629285  0.718583  0.808712  0.983369
Y0           2.8116436  2.940446  3.005341  3.071342  3.194602
dir_alpha    0.5772667  1.024000  1.423480  2.003651  3.390765
sigma2_out1  0.0011718  0.001490  0.001698  0.001935  0.002483
sigma2_out0  0.0009263  0.001146  0.001289  0.001446  0.001845

R> plot(separate_fit$mcmc_list)
```

The convergence of the MCMC object can be assessed by utilizing the convergence diagnostics offered by the **coda** package. To examine the convergence of six parameters. we can employ the `gelman.diag()` function on the `mcmc.list` object, specifically on `separate_fit$mcmc_list`.

```
R> library("coda")
R> gelman.diag(separate_fit$mcmc_list)
```

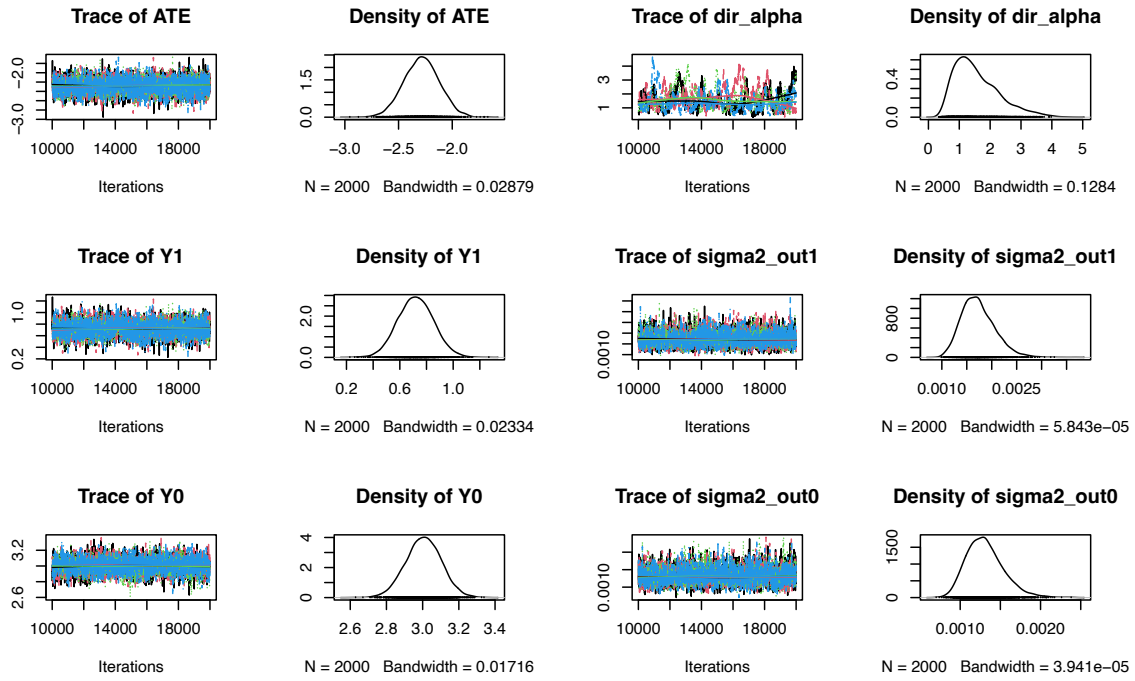


Figure 5: Plot of `mcmc_list` using the **coda** R package

```
Potential scale reduction factors:

            Point est. Upper C.I.
ATE               1.00       1.00
Y1                1.00       1.00
Y0                1.00       1.01
dir_alpha         1.02       1.07
sigma2_out1       1.00       1.00
sigma2_out0       1.00       1.00

Multivariate psrf

1.02
```

Based on the convergence diagnostics, it can be concluded that there are no issues with the convergence of the MCMC chain, similar to the visual inspection.

5. Real data example

In the previous section, the `separate_bart()` function was used to demonstrate a separate outcome model scheme. In this section, a single outcome model is tested using the `single_bart()` function, based on the Infant Health and Development Program (IHDP) dataset as an example. This dataset was collected from a longitudinal study that tracked the

	`treatment`=1 (n=139)		`treatment`=0 (n=608)	
Category	Mean	IQR	Mean	IQR
Y	6.43	(5.84, 7.34)	2.41	(1.45, 3.08)
$X_1^\star$	0.21	(-0.39, 0.95)	-0.05	(-0.75, 0.79)
$X_2^\star$	0.18	(-0.20, 0.59)	-0.04	(-0.60, 0.59)
$X_3^\star$	-0.04	(-0.73, 0.38)	0.01	(-0.73, 0.76)
$X_4^\star$	-0.22	(-0.88, 0.16)	0.05	(-0.88, 0.16)
$X_5^\star$	-0.14	(-0.69, 0.56)	0.03	(-0.50, 0.68)
$X_6^\star$	0.21	(-0.53, 0.96)	-0.05	(-0.86, 0.63)
X_7	0.52	(0, 1)	0.51	(0, 1)
X_8	0.09	(0, 0)	0.09	(0, 1)
X_9	0.68	(0, 1)	0.48	(0, 1)
X_{10}	0.29	(0, 1)	0.38	(0, 1)
X_{11}	0.25	(0, 1)	0.27	(0, 1)
X_{12}	0.22	(0, 0)	0.22	(0, 0)
X_{13}	0.38	(0, 1)	0.35	(0, 1)
X_{14}	1.58	(1, 2)	1.44	(1, 2)
X_{15}	0.14	(0, 0)	0.14	(0, 0)
X_{16}	0.94	(1, 1)	0.97	(1, 1)
X_{17}	0.69	(0, 1)	0.57	(0, 1)
X_{18}	0.99	(1, 1)	0.96	(1, 1)
X_{19}	0.15	(0, 0)	0.13	(0, 0)
X_{20}	0.06	(0, 0)	0.15	(0, 0)
X_{21}	0.17	(0, 0)	0. 15	(0, 0)
X_{22}	0.04	(0, 0)	0.09	(0, 0)
X_{23}	0.01	(0, 0)	0.09	(0, 0)
X_{24}	0.06	(0, 0)	0.14	(0, 0)
X_{25}	0.27	(0, 1)	0.13	(0, 0)

Table 2: Summary statistics for the IHDP data set. $\star$ denotes a continuous potential confounder.

development of low-birth-weight premature infants. The study participants in the treatment group received intensive care and home visits from trained providers and their cognitive test scores were evaluated at the end of the intervention period. The dataset includes a variety of pretreatment variables, including 6 continuous and 19 binary covariates. Although the original IHDP dataset was used in causal research by Hill (2011), we will use a synthetic version of the IHDP dataset created by Louizos, Shalit, Mooij, Sontag, Zemel, and Welling (2017) which provides true counterfactual values for comparison purposes. This data can be loaded by

```
R> data("ihdp", package = "bartcs")
```

and Table 2 displays the summary statistics of the variables. In the dataset, `y_factual` is the observed outcome Y (i.e., $Y(A)$) and `y_cfactual` is the counterfactual outcome Y (i.e., $Y(1-A)$). We fit the single outcome model using the `single_bart()` function.

```
R> single_fit <- single_bart(
```

```
  Y               = ihdp$y_factual,
  trt             = ihdp$treatment,
  X               = ihdp[, 6:30],
  num_tree        = 50,
  num_chain       = 4,
  num_post_sample = 2000,
  num_thin        = 5,
  num_burn_in     = 10000
)
R> single_fit

`bartcs` fit by `single_bart()`

        mean     2.5%    97.5%
ATE 3.964842 3.747028 4.180764
Y1  6.382810 6.188199 6.581852
Y0  2.417969 2.338264 2.496962
```

The function `single_bart()` returns a **bartcs** object, which displays the posterior means and 95% credible intervals for the average treatment effect (ATE), and the potential outcomes $Y(1)$ and $Y(0)$. The `summary()` and `plot()` functions can also be used with this **bartcs** object generated by `single_bart()`.

```
R> summary(single_fit)

`bartcs` fit by `single_bart()`

Treatment Value
  Treated group    :      1
  Control group    :      0

Tree Parameters
  Number of Tree   :     50              Value  of alpha    :   0.95
  Prob.  of Grow   :   0.28              Value  of beta     :      2
  Prob.  of Prune  :   0.28              Value  of nu       :      3
  Prob.  of Change :   0.44              Value  of q        :   0.95

Chain Parameters
  Number of Chains :      4              Number of burn-in  :  10000
  Number of Iter   :  20000              Number of thinning :      5
  Number of Sample :   2000

Outcome
 estimand chain     2.5%       1Q     mean   median       3Q    97.5%
      ATE     1 3.758373 3.894465 3.969119 3.968867 4.042380 4.183131
      ATE     2 3.744731 3.886575 3.957434 3.956101 4.026455 4.165961
      ATE     3 3.760480 3.905973 3.980315 3.980162 4.054086 4.206488
```

```
 ATE     4 3.730287 3.879606 3.952498 3.953050 4.028315 4.158430
 ATE   agg 3.747028 3.891543 3.964842 3.965384 4.038288 4.180764
  Y1     1 6.196530 6.318675 6.387760 6.387443 6.453303 6.589611
  Y1     2 6.181788 6.310026 6.376027 6.376233 6.439727 6.573960
  Y1     3 6.196317 6.329945 6.396885 6.397153 6.464297 6.601299
  Y1     4 6.169429 6.303404 6.370570 6.371514 6.435679 6.562172
  Y1   agg 6.188199 6.314489 6.382810 6.382215 6.449542 6.581852
  Y0     1 2.339020 2.391137 2.418640 2.418824 2.446414 2.498677
  Y0     2 2.336131 2.392407 2.418593 2.418124 2.446167 2.495229
  Y0     3 2.337997 2.388738 2.416570 2.416414 2.444457 2.495583
  Y0     4 2.340288 2.389536 2.418073 2.418218 2.446018 2.497264
  Y0   agg 2.338264 2.390199 2.417969 2.418042 2.445718 2.496962
```

We also fitted a separate outcome model to the `ihdp` data and compared the results from the single outcome model.

```
R> separate_fit <- separate_bart(
+    Y               = ihdp$y_factual,
+    trt             = ihdp$treatment,
+    X               = ihdp[, 6:30],
+    num_tree        = 50,
+    num_chain       = 4,
+    num_post_sample = 2000,
+    num_thin        = 5,
+    num_burn_in     = 10000
+  )
R> separate_fit

`bartcs` fit by `separate_bart()`

        mean     2.5%    97.5%
ATE 3.924013 3.702316 4.148937
Y1  6.342504 6.134043 6.550242
Y0  2.418491 2.340920 2.497081
```

Similar to the separate outcome model, in `single_bart()`, the `plot()` function for the **bartcs** object can also be employed to check the convergence of the MCMC chain. The traceplots for the ATE is presented in Figure 6 with the following line.

```
R> plot(single_fit, method = 'trace')
```

As this is a simulated version of the IHDP data, the true values are known and are 4.02 for the average treatment effect (ATE), 6.45 for $E[Y(1)]$, and 2.43 for $E[Y(0)]$. The outputs from the two models accurately reflect these true values within their 95% credible intervals. Additionally, the PIP plots (Figure 7) depict chosen confounders with PIP values larger than 0.5.

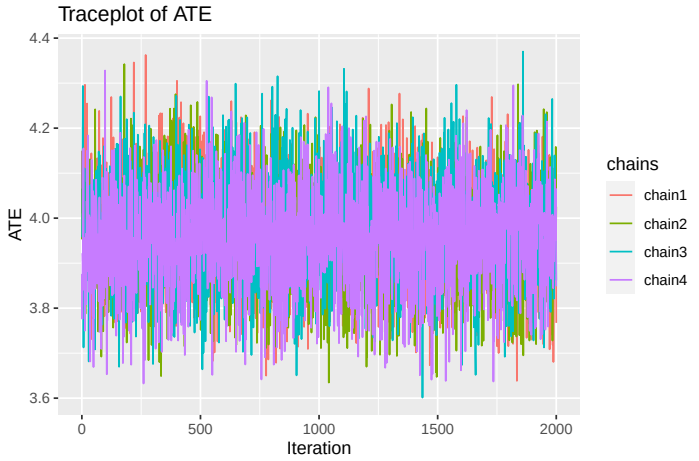
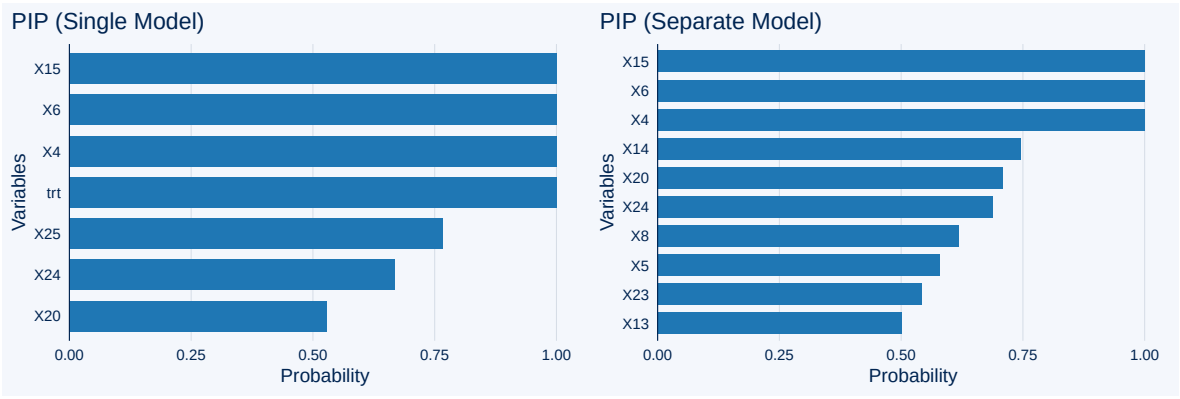


Figure 6: Traceplot of ATE for IHDP dataset



(a) PIP plot of single model

(b) PIP plot of separate model

Figure 7: PIP plot for IHDP dataset

The important aspect here is that in the case of the single outcome model, the exposure variable (`trt`) is also incorporated into the selection process. As indicated in Equation 2, because the exposure variable is included as one of the covariates in the outcome model, it is subject to variable selection. This means that in the computation of PIP, it is treated similarly to other confounders, producing the following plot (a) in Figure 7. In Figure 7, plot (a) displays the potential confounders for the single outcome model, which have a posterior inclusion probability of 0.5 or more, while plot (b) illustrates the confounders with a posterior inclusion probability of 0.5 or more when the separate outcome model is used. It is noteworthy that X_4, X_6, and X_{15} were consistently chosen as confounders with posterior inclusion probability 1.

6. Computation speed

In Figure 8, the computational speed of two models, the separate and single models, is depicted for two different settings of the number of trees (100 vs. 200). The speed was assessed using

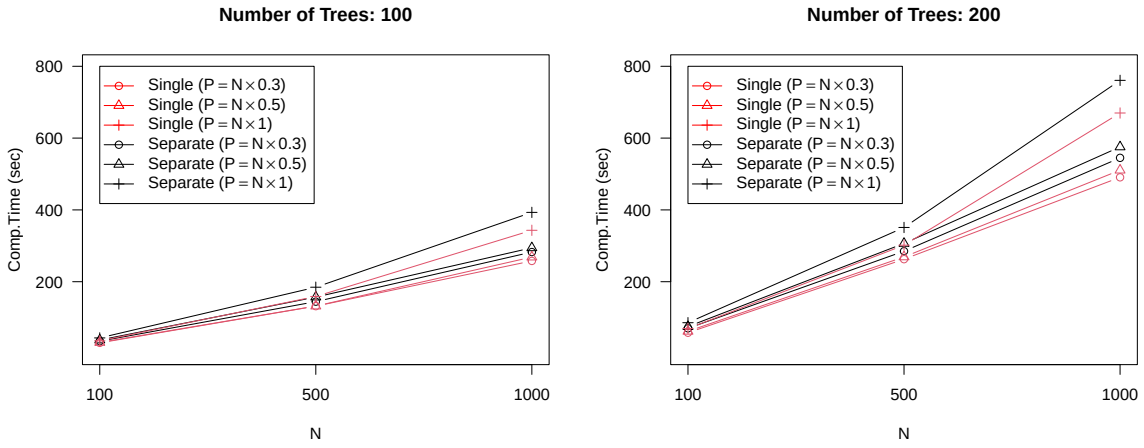


Figure 8: The computation times for both the single outcome model (red) and separate outcome model (black) based on the number of observations (N) under two different numbers of trees. A cross symbol ($+$) represents the scenario where the number of potential confounders (P) is equal to the number of observations (N), a triangle ($\triangle$) represents the scenario where $P = N \times 0.5$ and a circle ($\bigcirc$) represents the scenario where $P = N \times 0.3$. These results are obtained from 20000 MCMC iterations based on the scenario in Section 4.

20000 MCMC iterations across various combinations of N and P. We considered three values of N (100, 500, and 1000) and three values of P (circle for $N \times 0.3$, triangle for $N \times 0.5$, and cross for $N \times 1$).

For 100 BART trees, the separate model required 34 to 393 seconds (70 to 761 seconds for 200 BART trees) for computation, while the single model took 30 to 343 seconds (58 to 670 seconds for 200 BART trees), depending on the (N, P) combination. Both models exhibited similar computational speeds overall, considering the MCMC iterations. However, the single model, which fits two BART models (exposure and one outcome model), was found to be more efficient with slightly smaller biases and mean square errors (MSEs) across various scenarios (Kim *et al.* 2023). Therefore, it is recommended to utilize the single model (`single_bart()` function), especially when N is large, due to its faster computational speed.

Additionally, depending on the number of trees used, a significant improvement in computation speed can be observed. It is generally suggested to start with 50 trees as a "good starting value," (Kapelner and Bleich 2016) so using a smaller number of trees is also advised to gain computational advantages in terms of speed.

7. Continuous exposure example

When it comes to a continuous exposure variable, the formula in Equation 1 is changed as follows:

$$A_i = \sum_{t=1}^{T} g_a(\mathbf{X}_i; \mathcal{T}_t, \mathcal{M}_t) + \epsilon_i, \epsilon_j \sim N(0, \tau^2).$$

This altered formula is used in conjunction with the single outcome model to perform confounder selection. However, the separate outcome model, which fits two distinct outcome

models based on the two exposure levels, is not suitable for the continuous exposure variable. The **bartcs** has an advantage in handling continuous exposure through its `single_bart()` function. This function has the versatility to handle both binary and continuous treatments, and automatically identifies the binary treatment when there are only two unique values. To demonstrate this, we generate a data set similar to the previous example.

```
R> set.seed(42)
R> N <- 300
R> P <- 100
R> cov <- list()
R> for (i in 1:P) {
+    cov[[i]] <- rnorm(N, 0, 1)
+  }
R> X <- do.call(cbind, cov)
R> h1 <- ifelse(X[, 1] < 0, 1, -1)
R> h2 <- ifelse(X[, 2] < 0, -1, 1)
R> mu_trt <- 0.5 + h1 + h2 - 0.5 * abs(X[, 3] - 1) + 0.5 * X[, 4] * X[, 5]
R> Trt <- rnorm(N, mu_trt, 0.3)
R> mu_y <- 1 * h1 + 1 * h2 - Trt + 1 * abs(X[, 3] + 1) +
+    1 * X[, 4] + exp(0.5 * X[, 5]) -
+    0.5 * Trt * abs(X[, 6]) - 0.5 * Trt * abs(X[, 7] + 1)
R> Y <- rnorm(N, mu_y, 0.3)
R> treatment <- quantile(Trt, 0.75)
R> control <- quantile(Trt, 0.25)
```

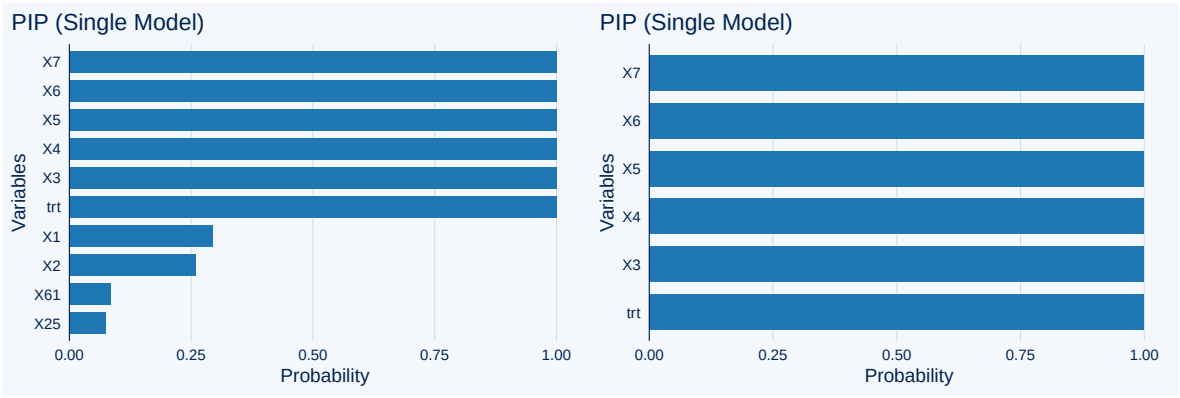
We use the function `single_bart()` to fit the generated data. The first and third quantile values of `Trt` will serve as the basis for comparing two different exposure levels. As arguments in `single_bart()`, we need to provide these two pre-specified exposure levels (a =`trt_treated` and a' =`trt_control`). In the this case, the causal estimand is $\Delta(a, a') = E[(Y(a) - Y(a')]$.

```
R> single_fit <- single_bart(
+    Y = Y, trt = Trt, X = X,
+    trt_treated = treatment, trt_control = control,
+    num_tree = 200, num_chain = 4,
+    num_burn_in = 10000, num_thin = 5, num_post_sample = 2000
+  )
R> single_fit

`bartcs` fit by `single_bart()`

          mean       2.5%     97.5%
ATE -2.8097339 -4.2581469 -1.732448
Y1   0.9982417  0.2753606  1.677726
Y0   3.8079756  3.0967180  4.740133
```

Similar to other **bartcs** objects, the `summary()` and `plot()` functions can be applied to the continuous exposure scenario. Figure 9 displays a PIP plot, which demonstrates that out



(a) Plotting PIP with `top_n` argument (b) Plotting PIP with `threshold = 0.5` argument

Figure 9: PIP plot for continuous exposure

of 100 possible confounders, all of the true confounders except X_1, X_2, and two additional predictors were captured effectively, with high PIP values.

8. Summary and discussion

In conclusion, the **bartcs** R package is a powerful tool for causal inference using BART. It allows users to adjust for confounders and estimate treatment effects using a flexible non-parametric method. The package's ability to handle high-dimensional and non-linear confounding, binary treatments, and continuous treatments makes it a versatile tool for a wide range of applications. Additionally, the package's support for parallel computing and visualization of results make it a user-friendly and easy-to-interpret tool. The **bartcs** package is a valuable resource for researchers in various fields.

Computational details

The results in this paper were obtained using R 4.3.0 on a Mac Studio with a M1 chip and 128 GB of memory. **bartcs** 1.2.0 and **bacr** 1.0.1 were used for the analysis. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

Acknowledgments

This work is supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (NRF-2022R1F1A1062904).

References

Barbieri MM, Berger JO (2004). "Optimal Predictive Model Selection." *The Annals of Statistics*, **32**(3), 870–897.

Chipman HA, George EI, McCulloch RE, *et al.* (2010). "BART: Bayesian Additive Regression Trees." *The Annals of Applied Statistics*, **4**(1), 266–298.

Häggström J (2017). ***CovSelHigh**: Model-Free Covariate Selection in High Dimensions*. R package version 1.1.1, URL https://CRAN.R-project.org/package=CovSelHigh.

Häggström J (2018). "Data-Driven Confounder Selection via Markov and Bayesian networks." *Biometrics*, **74**(2), 389–398.

Hahn PR, Murray JS, Carvalho CM (2020). "Bayesian Regression Tree Models for Causal Inference: Regularization, Confounding, and Heterogeneous Effects (with Discussion)." *Bayesian Analysis*, **15**(3), 965–1056.

Hastie T, Tibshirani R, *et al.* (2000). "Bayesian Backfitting (with Comments and a Rejoinder by the Authors)." *Statistical Science*, **15**(3), 196–223.

Hill JL (2011). "Bayesian Nonparametric Modeling for Causal Inference." *Journal of Computational and Graphical Statistics*, **20**(1), 217–240.

Kapelner A, Bleich J (2016). "bartMachine: Machine Learning with Bayesian Additive Regression Trees." *Journal of Statistical Software*, **70**(4), 1–40. doi:10.18637/jss.v070.i04.

Kim C, Tec M, Zigler CM (2023). "Bayesian Nonparametric Adjustment of Confounding." *Biometrics*, **in press**. doi:10.1111/biom.13833.

Lefebvre G, Delaney JA, McClelland RL (2014). "Extending the Bayesian Adjustment for Confounding Algorithm to Binary Treatment Covariates to Estimate the Effect of Smoking on Carotid Intima-Media Thickness: the Multi-ethnic Study of Atherosclerosis." *Statistics in Medicine*, **33**(16), 2797–2813.

Linero AR (2018). "Bayesian Regression Trees for High-Dimensional Prediction and Variable Selection." *Journal of the American Statistical Association*, **113**(522), 626–636.

Louizos C, Shalit U, Mooij JM, Sontag D, Zemel R, Welling M (2017). "Causal Effect Inference with Deep Latent-Variable Models." *Advances in Neural Information Processing Systems*, **30**. doi:10.48550/arXiv.1705.08821. URL https://github.com/AMLab-Amsterdam/CEVAE.

McCulloch R, Sparapani R, Spanbauer C, Gramacy R, Pratola M, Plummer M, Best N, Cowles K, Vines K (2023). ***BART**: Bayesian Additive Regression Trees*. R package version 2.9.4, URL https://CRAN.R-project.org/package=BART.

Plummer M, Best N, Cowles K, Vines K, Sarkar D, Bates D, Almond R, Magnusson A (2020). ***coda**: Output Analysis and Diagnostics for MCMC*. R package version 0.19-4, URL https://CRAN.R-project.org/package=coda.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rosenbaum PR, Rubin DB (1983a). "Assessing Sensitivity to an Unobserved Binary Covariate in an Observational Study with Binary Outcome." *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 212–218. URL http://www.jstor.org/stable/2345524.

Rosenbaum PR, Rubin DB (1983b). "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika*, **70**(1), 41–55. ISSN 00063444.

Rubin DB (1974). "Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies." *Journal of Educational Psychology*, **66**(5), 688.

Shortreed SM, Ertefaie A (2017). "Outcome-Adaptive Lasso: Variable Selection for Causal Inference." *Biometrics*, **73**(4), 1111–1122.

VanderWeele TJ (2019). "Principles of Confounder Selection." *European Journal of Epidemiology*, **34**(3), 211–219.

Wang C (2016). ***bacr**: Bayesian Adjustment for Confounding*. R package version 1.0.1, URL https://CRAN.R-project.org/package=bacr.

Wang C, Dominici F, Parmigiani G, Zigler CM (2015). "Accounting for Uncertainty in Confounder and Effect Modifier Selection when Estimating Average Causal Effects in Generalized Linear Models." *Biometrics*, **71**(3), 654–665.

Wang C, Parmigiani G, Dominici F (2012). "Bayesian Effect Estimation Accounting for Adjustment Uncertainty." *Biometrics*, **68**(3), 661–671.

Wilson A, Bondell HD, Reich BJ (2014). ***BayesPen**: Bayesian Penalized Credible Regions*. R package version 1.0, URL https://CRAN.R-project.org/package=BayesPen.

Wilson A, Reich BJ (2014). "Confounder Selection via Penalized Credible Regions." *Biometrics*, **70**(4), 852–861.

Yoo Y (2023). ***bartcs**: Bayesian Additive Regression Trees for Confounder Selection*. R package version 1.2.0, URL https://CRAN.R-project.org/package=bartcs.

A. Posterior computation

We use "Bayesian backfitting"(Hastie *et al.* 2000) to obtain posterior samples from

$$P(\mathcal{T}_1', \cdots, \mathcal{T}_T', \mathcal{M}_1', \cdots, \mathcal{M}_T', \sigma^2 | \boldsymbol{D})$$

for the outcome model (2) (or (3)). This involves a Metropolis-within-Gibbs sampler, where we fit each tree $\mathcal{T}_t'$ iteratively using residual responses:

$$R_{i,-t} = y_i - \sum_{j \neq t} g_y(\mathbf{X}_i; \mathcal{T}_j', \mathcal{M}_j')$$

for $i = 1, \cdots, N$. For each tree t, we propose a new tree structure $\mathcal{T}_t'$ from the full conditional $[\mathcal{T}_t' | R_{1,-t}, \cdots, R_{n,-t}, \sigma_2]$ (i.e., grow, prune or change alterations), and update the parameter within the tree through the full conditional $[\mathcal{M}_t' | \mathcal{T}_t', R_{1,-t}, \cdots, R_{n,-t}, \sigma_2]$.

To draw samples from $P(\mathcal{M}_t' | \mathcal{T}_t')$, we assume a prior $\mu \sim N(\mu_\mu / T, \sigma_\mu^2)$ on each of the leaf parameters $\mathcal{M}_t' = \{\mu_1, \mu_2, \cdots, \mu_{t_b}\}$, where t_b is the number of terminal nodes in tree $\mathcal{T}_t'$. The range center of the outcome is set as the mean, μ_μ, and σ_μ^2 is empirically determined to satisfy $T\mu_\mu - 2\sqrt{T}\sigma_\mu = y_{\min}$ and $T\mu_\mu + 2\sqrt{T}\sigma_\mu = y_{\max}$ (Kapelner and Bleich 2016).

We generate a sample μ_η from the posterior distribution for the η-th terminal node in tree $\mathcal{T}_t'$ by using the following equation:

$$\mu_\eta \sim N\left(\frac{1}{1/\sigma_\mu^2 + n_\eta/\sigma^2} \left(\frac{\mu_\mu / T}{\sigma_\mu^2} + \frac{\sum_{i \in I_\eta} R_{i,-t}}{\sigma^2} \right), \left(\frac{1}{\sigma_\mu^2} + \frac{n_\eta}{\sigma^2} \right)^{-1} \right),$$

where I_η and n_η correspond to the observation indices and the number of observations, respectively, for the η-th terminal node.

To use the separate model scheme, we perform a backfitting step to draw samples from $P(\mathcal{T}_1^a, \cdots, \mathcal{T}_T^a, \mathcal{M}_1^a, \cdots, \mathcal{M}_T^a, \sigma_a^2 | \boldsymbol{D})$ for each $A = a \in \{0, 1\}$ by computing the residual responses iteratively as follows:

$$R_{i,-t} = y_i - \sum_{j \neq t} g_y^a(\mathbf{X}_i; \mathcal{T}_j^a, \mathcal{M}_j^a) \quad \text{for } i \in I_a,$$

where I_a represents the set of observations corresponding to $A = a \in \{0, 1\}$.

To obtain posterior samples from $P(\mathcal{T}_1, \cdots, \mathcal{T}_T, \mathcal{M}_1, \cdots, \mathcal{M}_T | \boldsymbol{D})$ for the binary exposure model (1), we introduce a latent variable Z and apply the general Bayesian additive regression tree (BART) model for continuous data. Specifically, we define Z_i for $i = 1, \cdots, n$ as:

$$Z_i \sim \begin{cases} N\left(\sum_{t=1}^T g_a(\boldsymbol{X}_i; \mathcal{T}_t, \mathcal{M}_t), 1 \right) I_{(Z_i > 0)} & \text{for } A_i = 1; \\ N\left(\sum_{t=1}^T g_a(\boldsymbol{X}_i; \mathcal{T}_t, \mathcal{M}_t), 1 \right) I_{(Z_i \leq 0)} & \text{for } A_i = 0 \end{cases}$$

where $g_a(\boldsymbol{X}_i; \mathcal{T}_t, \mathcal{M}_t)$ denotes the function that maps $\boldsymbol{X}_i$ to a predicted value based on the t-th tree. For continuous A, the updating step follows the Bayesian backfitting method as described for the outcome model.

Once all the tree structures and corresponding parameters have been updated, we proceed to update the variance parameter (σ^2 in the outcome model (2)) using the Gibbs sampler and

the final residuals. This is achieved by sampling from the inverse gamma distribution given by:

$$\sigma^2 \sim \text{Inv.Gamma}\left(a_\sigma + \frac{N}{2}, b_\sigma + \frac{1}{2}\left\{\sum_{i=1}^{N}\left(y_i - \sum_{t=1}^{T} g_y(\boldsymbol{X}_i; \mathcal{T}_t', \mathcal{M}_t')\right)\right\}\right),$$

where $a_\sigma = b_\sigma = 3$ are set as suggested in Chipman *et al.* (2010). In the case of the separate model, two variance parameters (σ_0^2, σ_1^2) need to be updated for the two outcome models (i.e., Model (3) for $A = a \in \{0, 1\}$), which is done using the inverse gamma distribution as follows:

$$\sigma_a^2 \sim \text{Inv.Gamma}\left(a_\sigma + \frac{N_a}{2}, b_\sigma + \frac{1}{2}\left\{\sum_{i \in I_a}\left(y_i - \sum_{t=1}^{T} g_y^a(\boldsymbol{X}_i; \mathcal{T}_t^a, \mathcal{M}_t^a)\right)\right\}\right),$$

where N_a is the number of observations under $A = a$. If A is continuous, the variance parameter for the exposure model, τ^2, is updated in a similar manner:

$$\tau^2 \sim \text{Inv.Gamma}\left(a_\tau + \frac{N}{2}, b_\tau + \frac{1}{2}\left\{\sum_{i=1}^{N}\left(A_i - \sum_{t=1}^{T} g_a(\boldsymbol{X}_i; \mathcal{T}_t^\star, \mathcal{M}_t^\star)\right)\right\}\right),$$

where $a_\tau = b_\tau = 3$.

Next, we update the parameter α in the prior distribution of selection probabilities $\boldsymbol{s} \sim \mathcal{D}(\alpha/P, \cdots, \alpha/P)$ based on a prior of the form $\alpha/(\alpha + P) \sim \text{Beta}(a_0, b_0)$, where $a_0 = 0.5$ and $b_0 = 1$, as suggested in Linero (2018). The Metropolis-Hastings algorithm is then used to update the parameter. For the single model scheme, we update the vector of selection probabilities $\boldsymbol{s}$ using the Metropolis-Hastings algorithm, with the acceptance ratio given by

$$P(\boldsymbol{s} \to \boldsymbol{s}^{\text{new}}) = \min\left\{1, \left[(1 - \sum_{j=1}^{P} s_j)/(1 - \sum_{j=1}^{P} s_j^{\text{new}})\right]^{\sum_{j=1}^{P} n_j^a}\right\}.$$

In this step, the proposal distribution for $\boldsymbol{s}$ is given as $\mathcal{D}(n_0^y + c + \alpha/P, n_1^a + n_1^y + \alpha/P, n_2^a + n_2^y + \alpha/P, \cdots, n_P^a + n_P^y + \alpha/P)$. For the separate model approach, we update $\boldsymbol{s}$ using a conjugate sampling update as follows: $\boldsymbol{s} \sim \mathcal{D}(\alpha/P + n_1^a + n_1^{y_1} + n_1^{y_0}, \cdots, \alpha/P + n_P^a + n_P^{y_1} + n_P^{y_0})$, where $n_j^{y_1}$ and $n_j^{y_0}$ represent the numbers of splits on the confounder X_j in two separate outcome models.

Affiliation:

Chanmin Kim
Department of Statistics
SungKyunKwan University
25-2 SungKyunKwan-ro, Jongno-gu
Seoul 03063, Korea
E-mail: chanmin.kim@skku.edu
URL: https://lit777.github.io/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd