Bayesian Variable Selection: BART Approaches

Antonio R. Linero

June 2021

Learning Objectives

- Know what Bayesian additive regression trees (BART) is
  - How does it differ from other methods we've talked about?
  - Under what situations might you want to use it?

Learning Objectives

- Know what Bayesian additive regression trees (BART) is
  - How does it differ from other methods we've talked about?
  - Under what situations might you want to use it?
- Be able to apply BART to nonparametric regression and classification problems

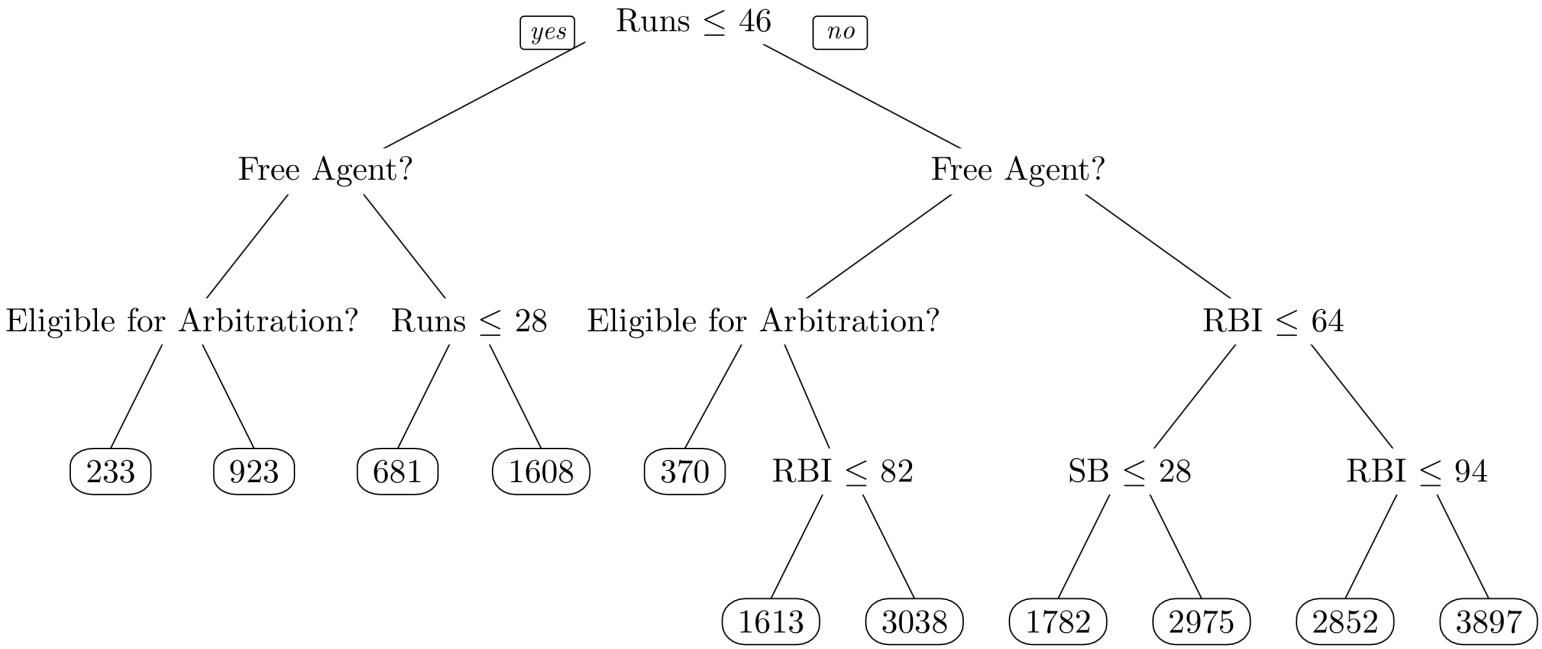
Learning Objectives

- Know what Bayesian additive regression trees (BART) is
  - How does it differ from other methods we've talked about?
  - Under what situations might you want to use it?
- Be able to apply BART to nonparametric regression and classification problems
- Be able to apply BART and its extensions to perform variable selection

Learning Objectives

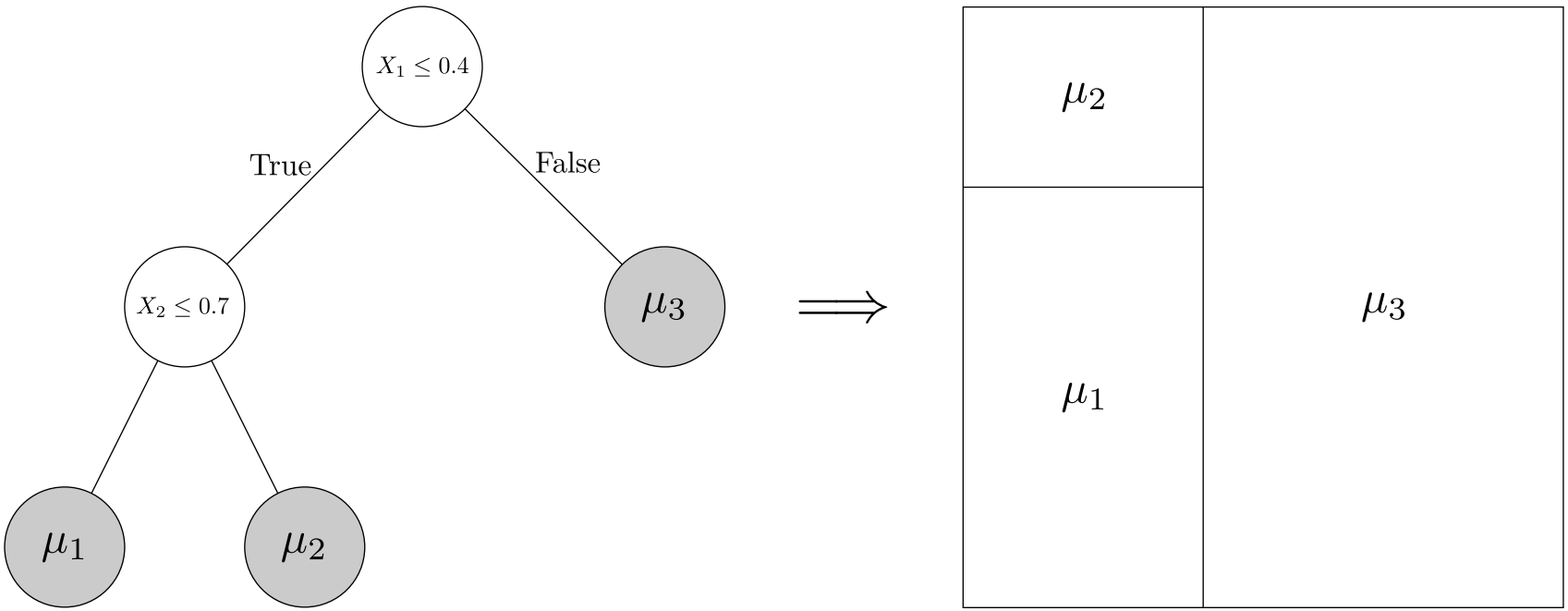
- Know what Bayesian additive regression trees (BART) is
  - How does it differ from other methods we've talked about?
  - Under what situations might you want to use it?
- Be able to apply BART to nonparametric regression and classification problems
- Be able to apply BART and its extensions to perform variable selection
- Be able to perform interaction detection using BART

Decision Trees



Implies variable selection through variables used to build rules.

Decision Trees and Partitions



Decision Tree Ensembling

Sampling Methods:

- Bagging (Breiman, 1996)
- Random forests (Breiman, 2001)
- Bayesian model averaging (Chipman et al., 1998; Denison et al., 1998)

Decision Tree Ensembling

Sampling Methods:

- Bagging (Breiman, 1996)
- Random forests (Breiman, 2001)
- Bayesian model averaging (Chipman et al., 1998; Denison et al., 1998)

Combining Weak Learners

- Boosting (Freund et al., 1999)
- BART (Chipman et al., 2010)

Decision Tree Ensembling

Sampling Methods:

- Bagging (Breiman, 1996)
- Random forests (Breiman, 2001)
- Bayesian model averaging (Chipman et al., 1998; Denison et al., 1998)

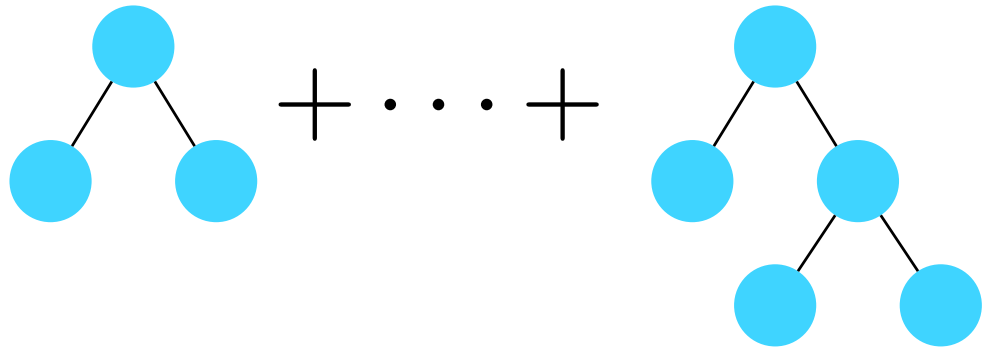
Combining Weak Learners

- Boosting (Freund et al., 1999)
- BART (Chipman et al., 2010)

Gold standard: gradient boosted decision trees (as implemented, e.g., in `xgboost`).

Bayesian Additive Regression Trees

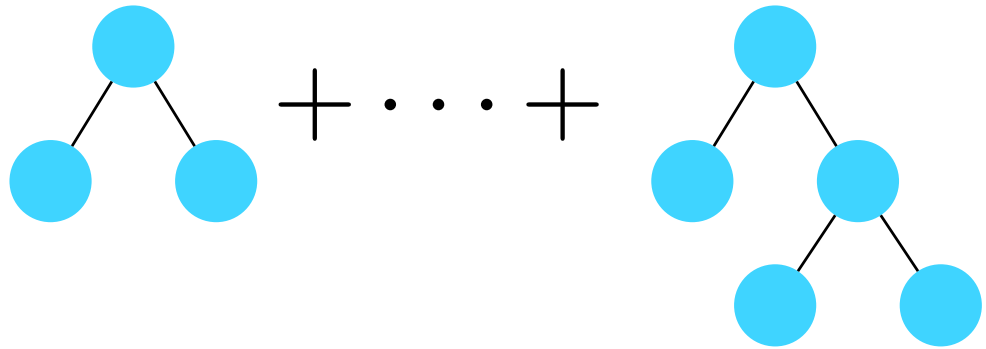
$$r(x) = \quad + \cdots +$$



Each tree $\mathcal{T}_t, t = 1, \ldots, T$ given a prior distribution and collection of leaf parameters $\mathcal{M}_t$ are given priors.

Bayesian Additive Regression Trees

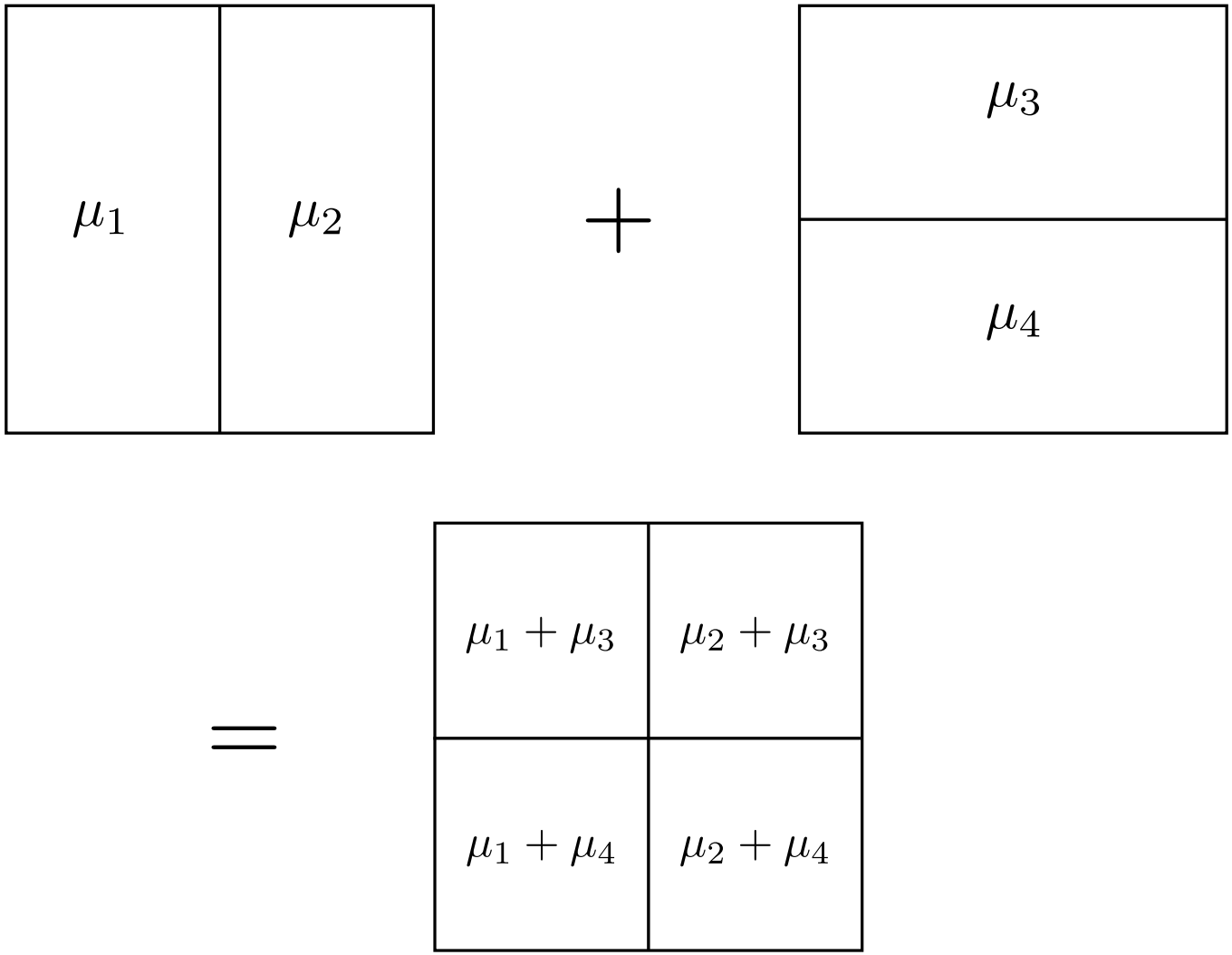
$$r(x) = \quad + \cdots +$$



Each tree $\mathcal{T}_t, t = 1, \ldots, T$ given a prior distribution and collection of leaf parameters $\mathcal{M}_t$ are given priors.

Formally, $r(x) = \sum_{t=1}^{T} g(x; \mathcal{T}_t, \mathcal{M}_t)$ where $g(x; \mathcal{T}_t, \mathcal{M}_t)$ is the associated step function of tree t.

Adding Trees Together Induces Smoothness



Prior on $\mathcal{T}_t$ and $\mathcal{M}_t$

1. Initialize root node to be a leaf of depth $d = 0$.

2. For each node of depth d, convert it to a branch with probability $\gamma/(1+d)^{\beta}$; otherwise, it stays a leaf.

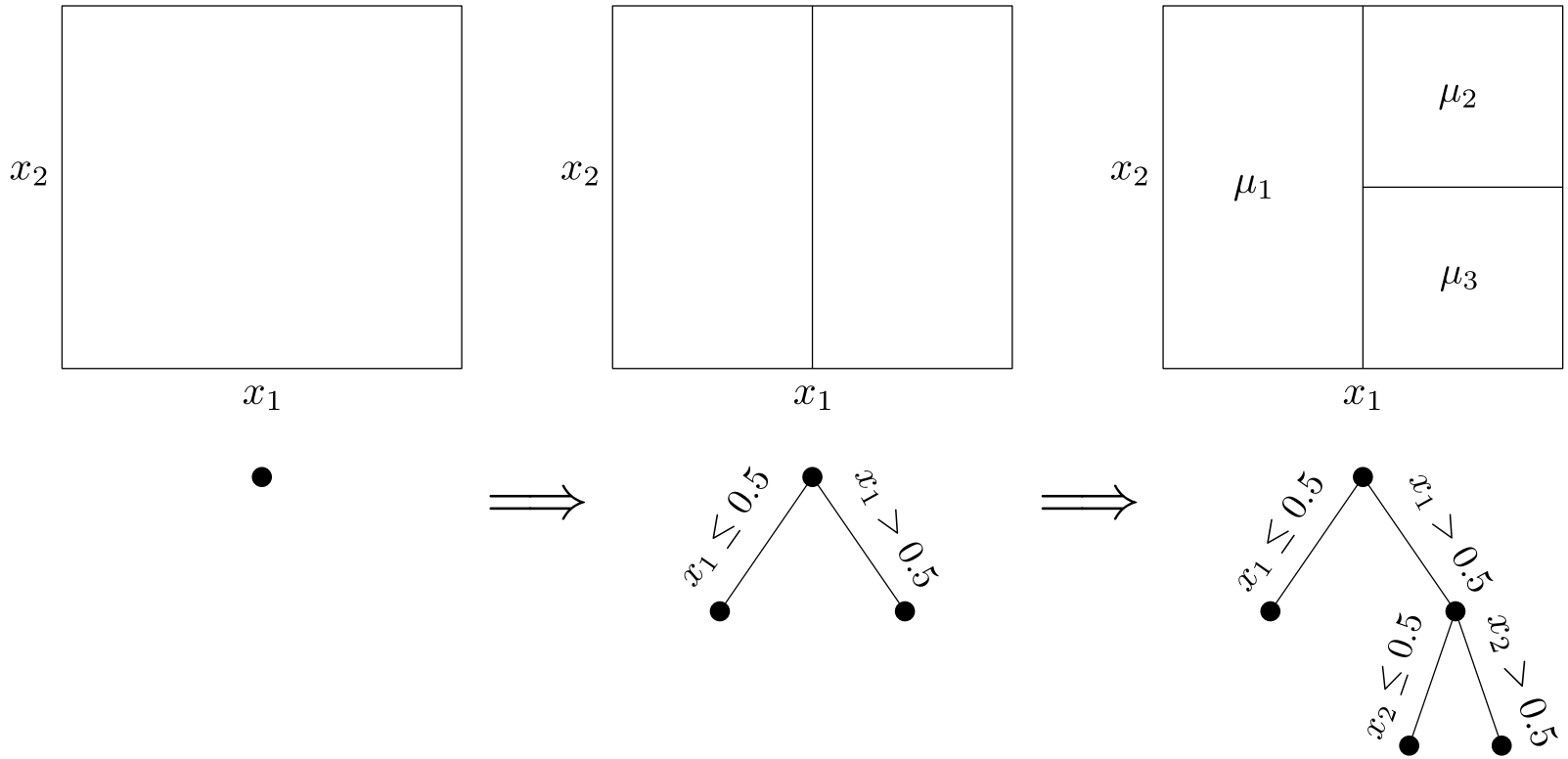
3. If all nodes of depth d are leaves, continue; otherwise, set $d \leftarrow d+1$ and return to Step 2.

4. Assign to each branch a splitting coordinate j_b sampled according to some probability vector s ($s = (P^{-1}, \ldots, P^{-1})$ by default).

5. For each branch b, sample a cutpoint C_b by sampling uniformly from the observed X_{ij_b}'s which fall in that branch.

6. For each leaf ℓ, sample $\mu_{t\ell} \sim \text{Normal}(0, \sigma_\mu^2/T)$.

Sampling from the Prior



Some BART Models Folks Have Used

Semiparametric regression:

$$Y_i = r(X_i) + \epsilon_i, \qquad \epsilon_i \sim \text{Normal}(0, \sigma^2).$$

Some BART Models Folks Have Used

Semiparametric regression:

$$Y_i = r(X_i) + \epsilon_i, \qquad \epsilon_i \sim \text{Normal}(0, \sigma^2).$$

Nonparametric probit regression:

$$Y_i \sim \text{Bernoulli}[\Phi\{r(X_i)\}]$$

Some BART Models Folks Have Used

Semiparametric regression:

$$Y_i = r(X_i) + \epsilon_i, \qquad \epsilon_i \sim \text{Normal}(0, \sigma^2).$$

Nonparametric probit regression:

$$Y_i \sim \text{Bernoulli}[\Phi\{r(X_i)\}]$$

Poisson Loglinear Model:

$$Y_i \sim \text{Poisson}[\exp\{r(X_i)\}]$$

Some BART Models Folks Have Used

Semiparametric regression:

$$Y_i = r(X_i) + \epsilon_i, \qquad \epsilon_i \sim \text{Normal}(0, \sigma^2).$$

Nonparametric probit regression:

$$Y_i \sim \text{Bernoulli}[\Phi\{r(X_i)\}]$$

Poisson Loglinear Model:

$$Y_i \sim \text{Poisson}[\exp\{r(X_i)\}]$$

Many other possibilities! gamma regression, survival analysis, density regression, nonparametric variance estimation.

Magic Defaults

Tree Prior:

$$\gamma = 0.95, \qquad \beta = 2.$$

Leaf Node Prior: after standardizing the response to lie in interval $[-0.5, 0.5]$,

$$\mu_{t\ell} \sim \text{Normal}(0, 1/4T)$$

Variance prior:

$$\sigma^{-2} \sim \text{Gam}(\alpha, \beta)$$

where $\alpha = 1.5$ and β chosen so that $\Pr(\sigma < \widehat{\sigma}) = 0.75$ where $\widehat{\sigma}$ is an empirical estimate of the noise level.

Getting Started with Installing BART

```
## Implements the DART Prior
install.packages("BART")

## dartMachine: requires rJava, install instructions
## available at https://github.com/theodds/dartMachine

## SoftBart: if on Mac, needs gfortran library from
##           MacOSX tools at CRAN.
library(devtools)
install_github("theodds/SoftBART")

## Load Packages
library(BART)
options(java.parameters = "-Xmx4g")
library(dartMachine)
library(SoftBart)
```

Pros and Cons

- `BART:` slowish, but given lots of love by creators.
- `dartMachine:` fast, but I threw it together and its not well-documented.
- `SoftBart:` slower but fancier, better (?) in terms of prediction than others.

Simple Example: Boston Housing

```
?MASS::Boston
set.seed(774837)
folds <- caret::createFolds(1:nrow(MASS::Boston), k = 5)
boston_train <- MASS::Boston[-folds[[1]],]
boston_test  <- MASS::Boston[folds[[1]],]

rmse <- function(x,y) sqrt(mean((x-y)^2))

lm_boston <- lm(medv ~ ., data = boston_train)

fitted_bart <- bartMachine(X = boston_train %>% select(-medv),
                           y = boston_train$medv, num_trees = 200,
                           num_burn_in = 4000,
                           num_iterations_after_burn_in = 4000,
                           seed = 112231)

cor(boston_test$medv, predict(fitted_bart, boston_test %>%
                                  select(-medv)))
cor(boston_test$medv, predict(lm_boston, boston_test))
```

Some Computational Details

Models are fit with *Bayesian backfitting*. For semiparametric regression, algorithmic looks like this:

Algorithm 1 Fitting Nonparametric BART Regression

Input: $\{X_i, Y_i : i = 1, \ldots, N\}, \sigma_\mu^2, \sigma^2, Q, \{\mathcal{T}_t, \mathcal{M}_t : t = 1, \ldots, T\}$

1: **for** $t = 1, \ldots, T$ **do**

2: Compute the residual $R_i = Y_i - \sum_{k \neq t} g(X_i; \mathcal{T}_k, \mathcal{M}_k)$

3: Propose $\mathcal{T}'$ from proposal distribution $Q(\mathcal{T}' \mid \mathcal{T}_t)$

4: Compute the marginal likelihood of $\mathcal{T}_t$ and $\mathcal{T}'$ as

$$\Lambda(\mathcal{T}) = \prod_{\ell \in \mathcal{T}} \int \pi(\mu) \prod_{i \in \ell} \text{Normal}(R_i \mid \mu, \sigma^2) \, d\mu$$

5: Set $\mathcal{T}_t \leftarrow \mathcal{T}'$ with probability

$$\frac{\Lambda(\mathcal{T}')\pi_{\mathcal{T}}(\mathcal{T}')Q(\mathcal{T}_t \mid \mathcal{T}')}{\Lambda(\mathcal{T}_t)\pi_{\mathcal{T}}(\mathcal{T}_t)Q(\mathcal{T}' \mid \mathcal{T}_t)}$$

6: Sample $\mathcal{M}_t$ from its full conditional

7: **end for**

Possible Proposal Mechanisms

- `BIRTH`: take a leaf node, convert to a branch and add two new leaves below.
- `DEATH`: take the parent of a leaf node and delete its children, converting it into a leaf.
- `PRIOR`: sample a new tree from the prior.

Great source for details: (Kapelner and Bleich, 2016)

Measuring Variable Importance with Tree Ensembles

Possibilities:

- A variable is "relevant" if it is included in *at least one branch* of the ensemble.
- A variable is "important" if it is included in *many branches* of the ensemble.

Initial work on BART adopts the *second* viewpoint.

Measuring Variable Importance with Tree Ensembles

Possibilities:

- A variable is "relevant" if it is included in *at least one branch* of the ensemble.
- A variable is "important" if it is included in *many branches* of the ensemble.

Initial work on BART adopts the *second* viewpoint.

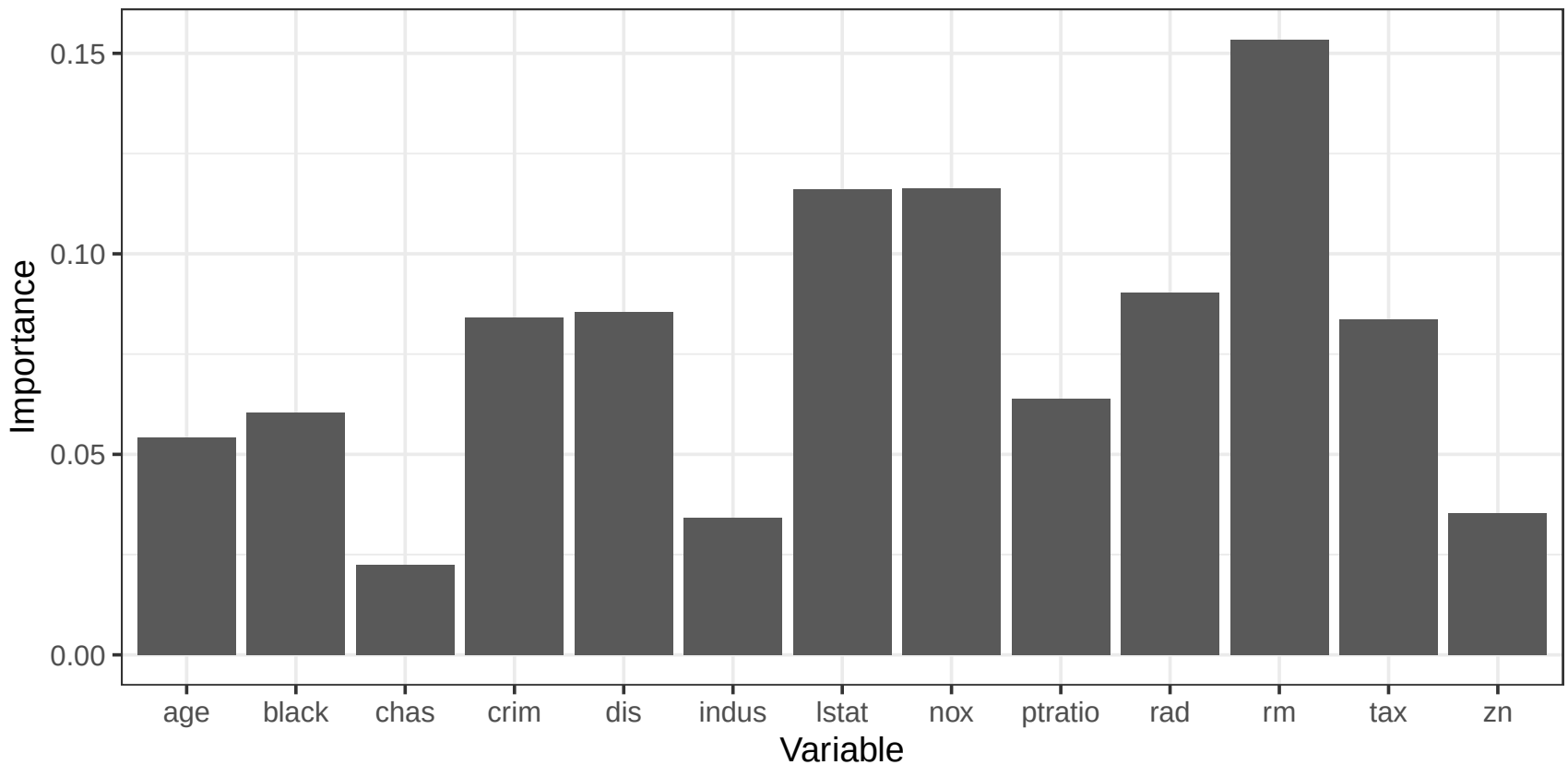
Variable Importance

The *importance* of a variable j is $\mathbb{E}(m_j/B \mid \mathcal{D})$, the average proportion of all branches which split on the variable j.

Boston Housing

```
boston <- MASS::Boston
bart_boston <- bartMachine(X = boston %>% select(-medv), y = boston$medv,
                           num_trees = 40, num_burn_in = 5000,
                           num_iterations_after_burn_in = 5000,
                           serialize = TRUE, seed = 77777)
var_props <- get_var_props_over_chain(bart_boston)
```

```
ggplot() + geom_col(aes(x = names(var_props), y = var_props)) + theme_bw() +
  xlab("Variable") + ylab("Importance") + ylim(0, max(var_props))
```



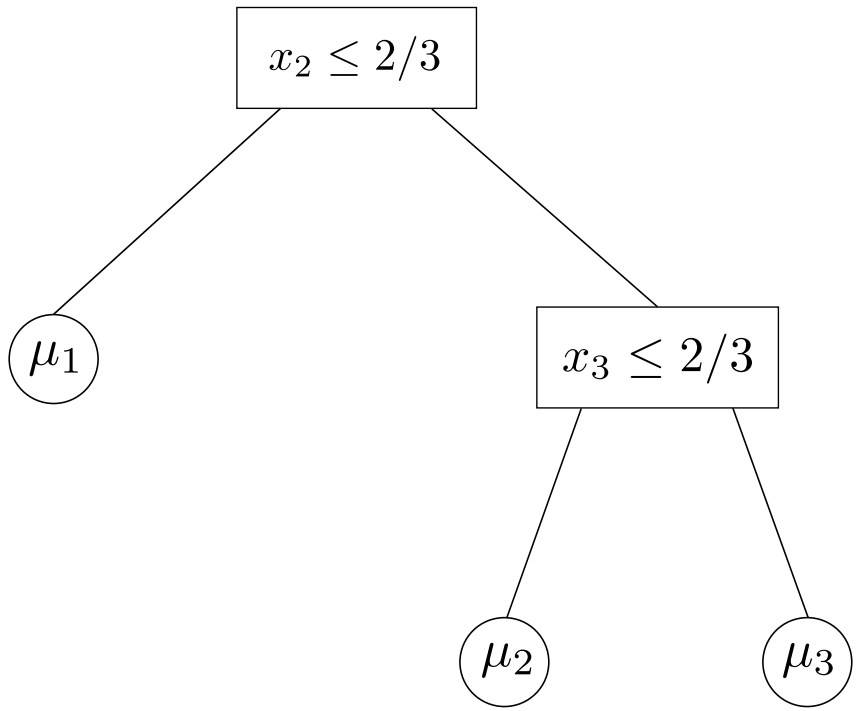
Problems with Raw Variable Importances

1. Continuous variables naturally require more splits than binary.
2. Strong tendency for the "importance" of irrelevant features to be overstated, especially when the number of trees is large.
3. No obvious Bayesian interpretation.

The Splitting Proportions

Let $s \in \mathbb{S}_{P-1}$ be the _prior probabilities that a given decision rule uses coordinate j.

For example, probability of splitting on x_2 and x_3, as in this tree



is $s_2 \cdot s_3$ (given the tree topology). Packages we introduced let you pick s.

Optimizing the Splitting Proportions

To understand the role of the variable importances, consider optimizing the s_j's via the EM-algorithm. Default value of s_j is P^{-1} in most implementations. Update for s_j turns out to be

$$s_j \leftarrow \frac{\mathbb{E}(m_j \mid \mathcal{D})}{\mathbb{E}(B \mid \mathcal{D})}.$$

Very similar to using variable importance; but we can iterate!

Algorithm

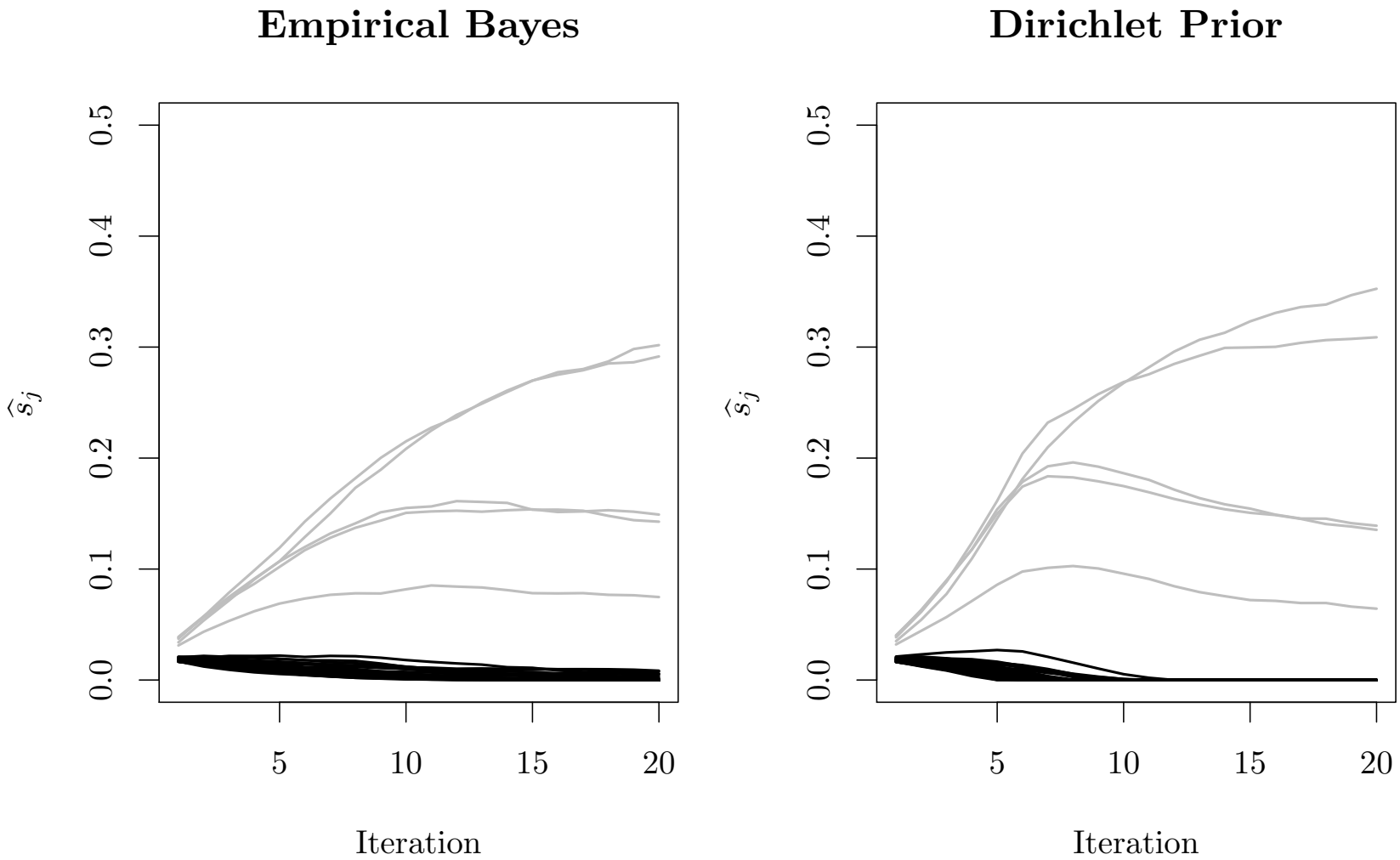
Consider prior $s \sim \text{Dirichlet}(\eta, \ldots, \eta)$. MAP estimator of s can be found with following algorithm (special case $\eta = 1$ is equivalent to EB).

MAP Estimation of Splitting Proportions

1. Initialize $s = (P^{-1}, \ldots, P^{-1})$
2. Fit BART model using s as the splitting proportion
3. Set
$$s_j \leftarrow \frac{\max\{\mathbb{E}(m_j + \eta - 1 \mid \mathcal{D}, s), 0\}}{\sum_k \max\{\mathbb{E}(m_k + \eta - 1 \mid \mathcal{D}, s), 0\}}$$
4. Return to Step 2.

(Linero and Du, 2021+)

EM Algorithm for EB/MAP Variable Selection



But how to pick η?

How Many Predictors do BART Ensembles Use?

Let B denote number branches in ensemble and Q the number of active variables.

How Many Predictors do BART Ensembles Use?

Let B denote number branches in ensemble and Q the number of active variables.

Default setting: $s_j = P^{-1}$; can be shown that

$$\mathbb{E}_{\Pi}(Q \mid B) = B + O(P^{-1})$$

How Many Predictors do BART Ensembles Use?

Let B denote number branches in ensemble and Q the number of active variables.

Default setting: $s_j = P^{-1}$; can be shown that

$$\mathbb{E}_\Pi(Q \mid B) = B + O(P^{-1})$$

Uniform prior: $s \sim \text{Uniform}(\mathbb{S}_{P-1})$; can be shown that

$$\mathbb{E}_\Pi(Q \mid B) = B + O(P^{-1}).$$

How Many Predictors do BART Ensembles Use?

Let B denote number branches in ensemble and Q the number of active variables.

Default setting: $s_j = P^{-1}$; can be shown that

$$\mathbb{E}_\Pi(Q \mid B) = B + O(P^{-1})$$

Uniform prior: $s \sim \text{Uniform}(\mathbb{S}_{P-1})$; can be shown that

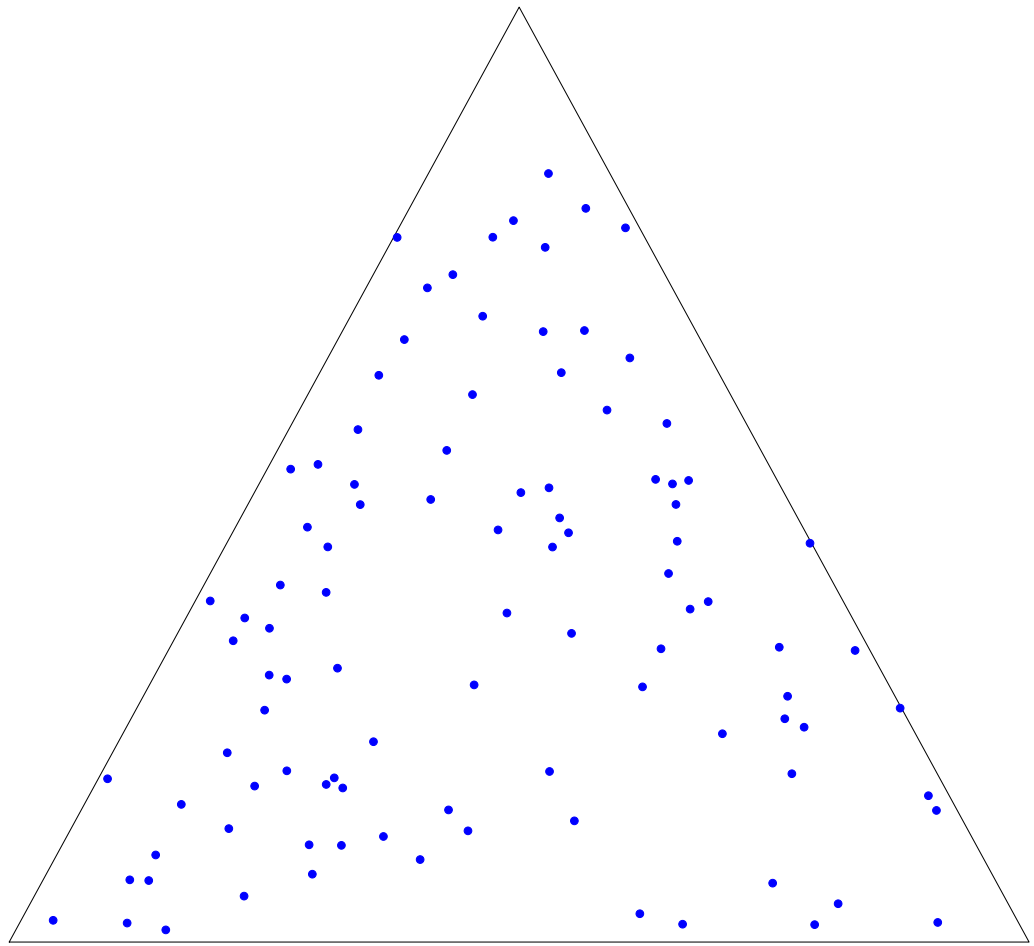
$$\mathbb{E}_\Pi(Q \mid B) = B + O(P^{-1}).$$

Many-weak-predictors regime! Neither is uninformative.

Sparsity Inducing Dirichlet

Idea: set $s \sim \text{Dirichlet}(\alpha/P, \ldots, \alpha/P)$

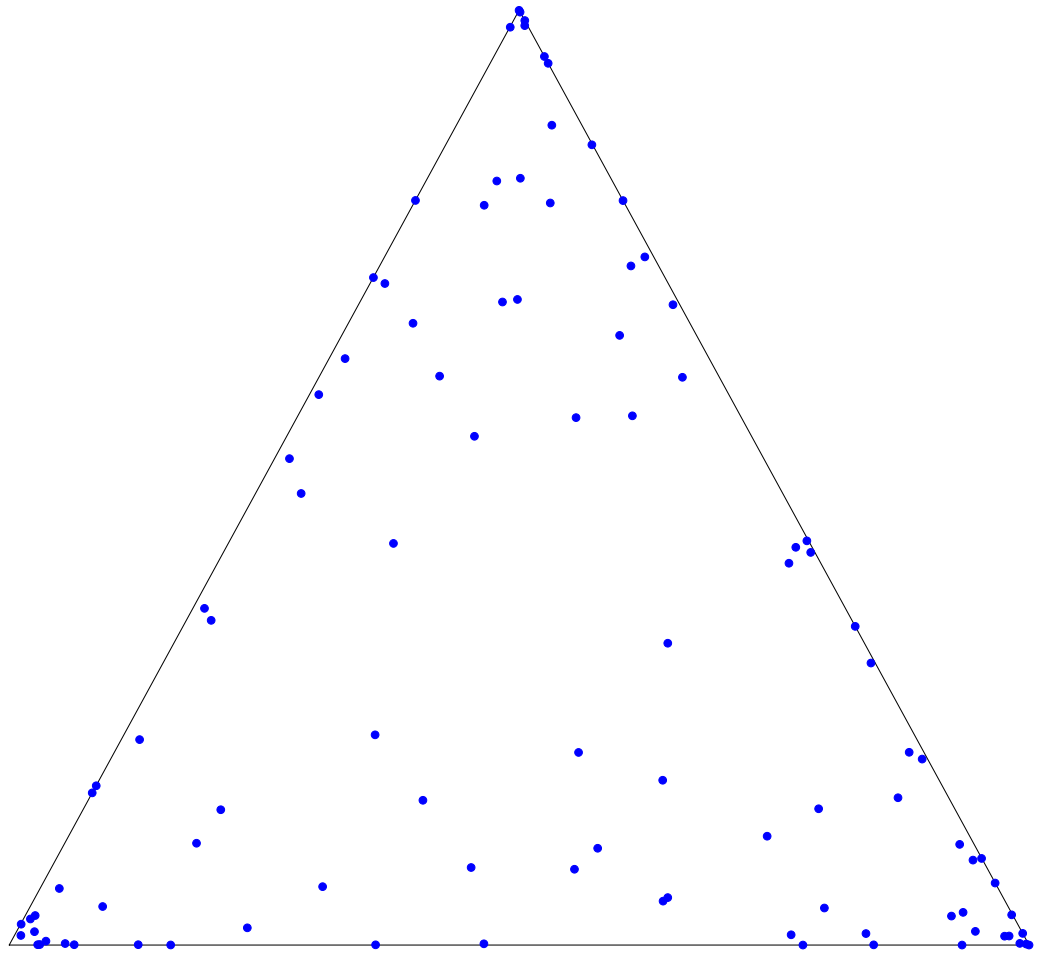
$$\alpha = 3$$



Sparsity Inducing Dirichlet

Idea: set $s \sim \text{Dirichlet}(\alpha/P, \ldots, \alpha/P)$

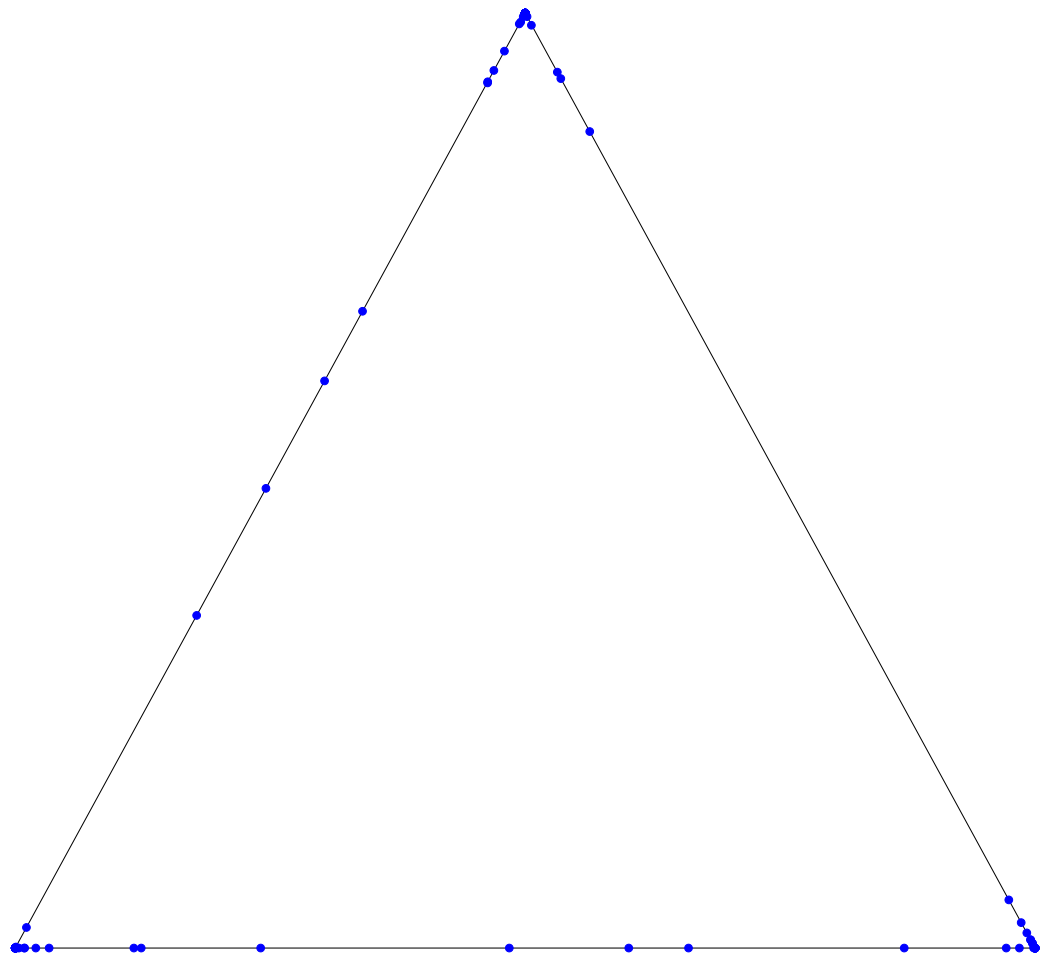
$$\alpha = 1$$



Sparsity Inducing Dirichlet

Idea: set $s \sim \text{Dirichlet}(\alpha/P, \ldots, \alpha/P)$

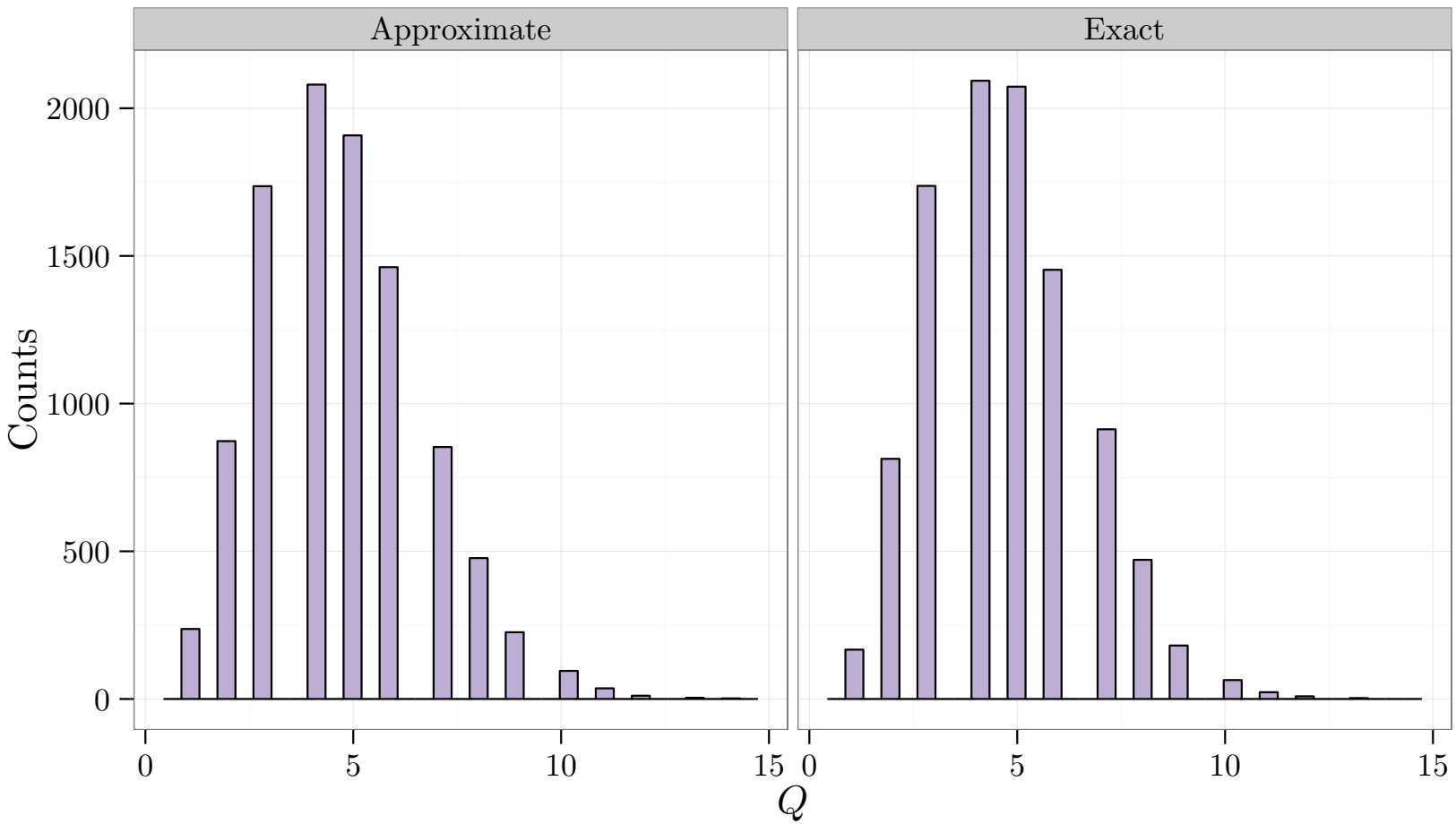
$$\alpha = 0.1$$



Dirichlet Prior on s

Can be shown: $Q - 1 \approx \text{Poisson}(\theta)$ where

$$\theta = \alpha \sum_{i=1}^{B-1} (\alpha + i)^{-1}, \qquad B = \text{number of branches}$$



The Full Conditional

By conditional conjugacy:

$$s \sim \text{Dirichlet}(\alpha/P + m_1, \ldots, \alpha/P + m_P)$$

(Either exactly or *very* good MH proposal)

Simple Illustration

Friedman's Function: $r_0(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$

```
set.seed(1234)
f_fried <- function(x) 10 * sin(pi * x[,1] * x[,2]) +
  20 * (x[,3] - 0.5)^2 +
  10 * x[,4] + 5 * x[,5]

gen_data <- function(n_train, n_test, P, sigma) {
    X <- matrix(runif(n_train * P), nrow = n_train)
    mu <- f_fried(X)
    X_test <- matrix(runif(n_test * P), nrow = n_test)
    mu_test <- f_fried(X_test)
    Y <- mu + sigma * rnorm(n_train)
    Y_test <- mu_test + sigma * rnorm(n_test)

    return(list(X = X, Y = Y, mu = mu, X_test = X_test,
                Y_test = Y_test, mu_test = mu_test))
}

## Simulate dataset
sim_data <- gen_data(250, 100, 1000, 1)

## Metric for evaluating goodness
rmse <- function(x,y) sqrt(mean((x - y)^2))
```

Fitting With BART Package

```
set.seed(digest::digest2int("Using BART package"))
fit_bart <- wbart(x.train = sim_data$X, y.train = sim_data$Y,
                  ndpost = 4000, nskip = 4000)
fit_dart <- wbart(x.train = sim_data$X, y.train = sim_data$Y,
                  sparse = TRUE, ndpost = 4000, nskip = 4000)

predicted_bart <- predict(fit_bart, sim_data$X_test)
predicted_dart <- predict(fit_dart, sim_data$X_test)
```

Fitting With BART Package

```
set.seed(digest::digest2int("Using BART package"))
fit_bart <- wbart(x.train = sim_data$X, y.train = sim_data$Y,
                  ndpost = 4000, nskip = 4000)
fit_dart <- wbart(x.train = sim_data$X, y.train = sim_data$Y,
                  sparse = TRUE, ndpost = 4000, nskip = 4000)

predicted_bart <- predict(fit_bart, sim_data$X_test)
predicted_dart <- predict(fit_dart, sim_data$X_test)
```

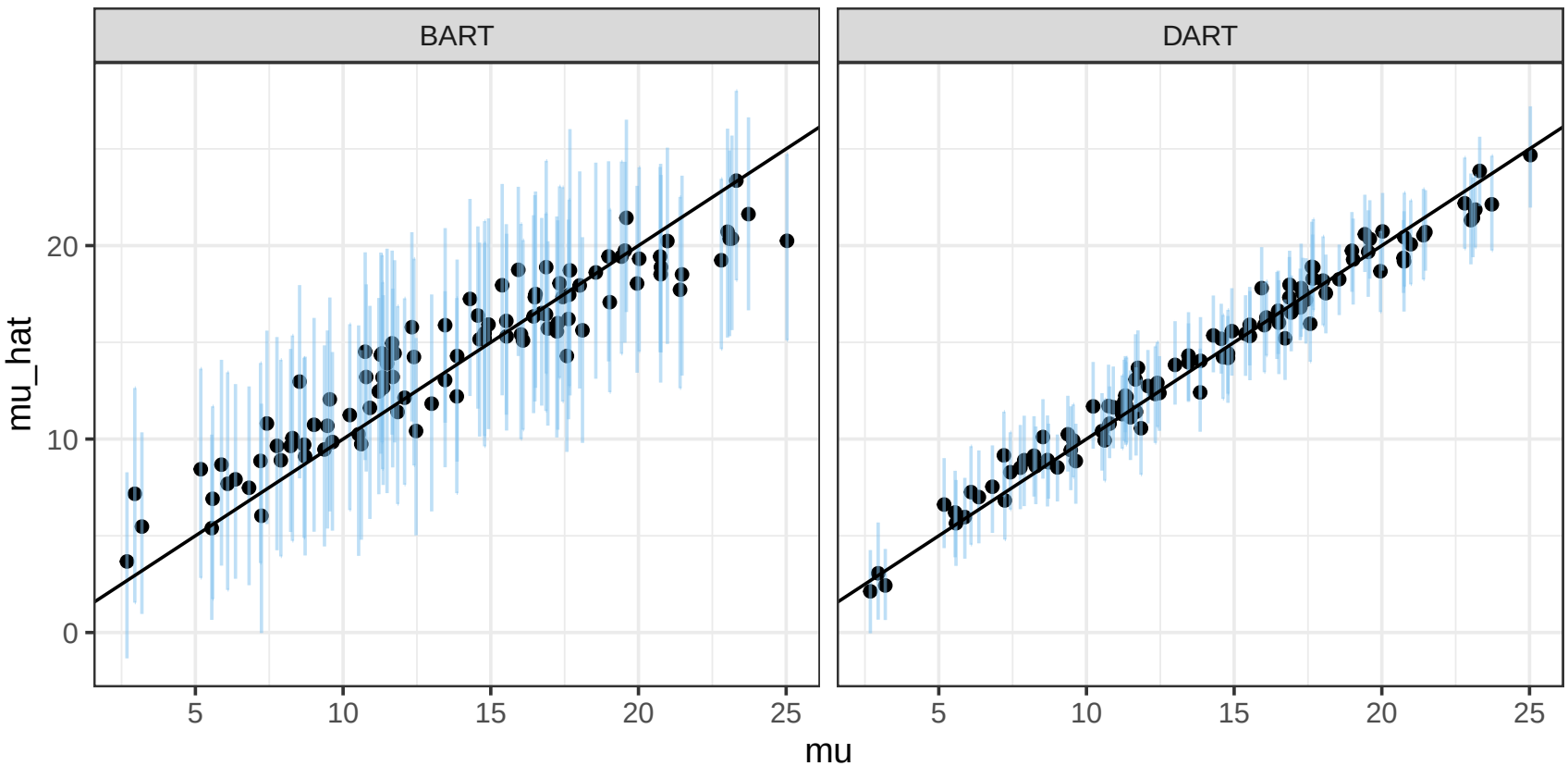
```
mu_hat_bart <- colMeans(predicted_bart)
mu_hat_dart <- colMeans(predicted_dart)

rmse(mu_hat_bart, sim_data$mu_test)
```

```
## [1] 1.947462
```

```
rmse(mu_hat_dart, sim_data$mu_test)
```

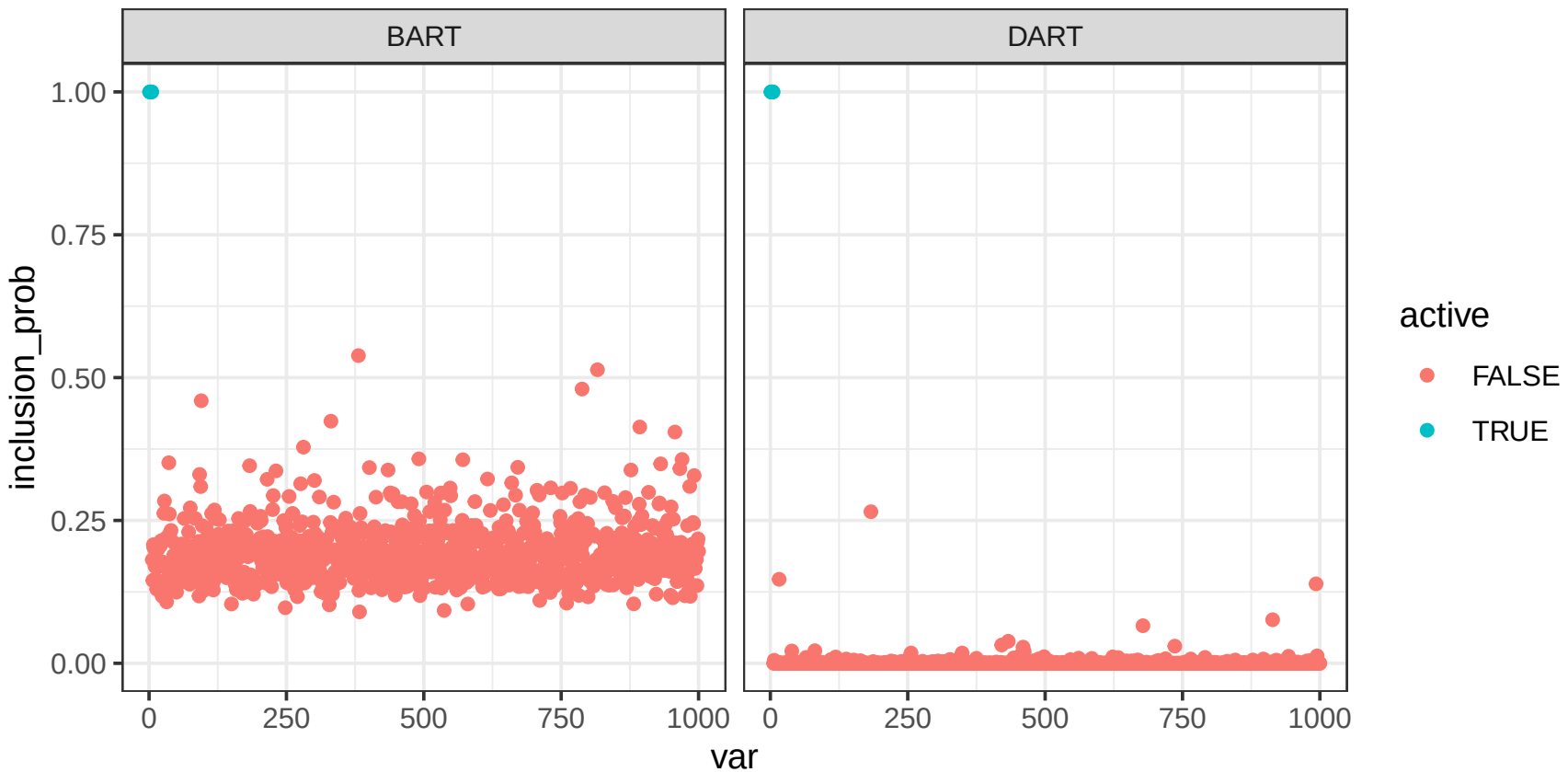
```
## [1] 0.8669519
```

```r
df_for_plot <- data.frame(mu = c(sim_data$mu_test, sim_data$mu_test),
                          mu_hat = c(mu_hat_bart, mu_hat_dart),
                          method = rep(c("BART", "DART"),
                                       each = length(sim_data$mu_test)))
limits_bart <- t(apply(predicted_bart, 2, function(x) quantile(x, c(0.025, 0.975))))
limits_dart <- t(apply(predicted_dart, 2, function(x) quantile(x, c(0.025, 0.975))))

ggplot(df_for_plot, aes(x = mu, y = mu_hat)) + geom_point() +
  geom_errorbar(aes(ymin = c(limits_bart[,1], limits_dart[,1]),
                    ymax = c(limits_bart[,2], limits_dart[,2])),
                color = "skyblue2", alpha = 0.5) +
  facet_wrap(~method) + geom_abline(slope = 1, intercept = 0) + theme_bw()
```



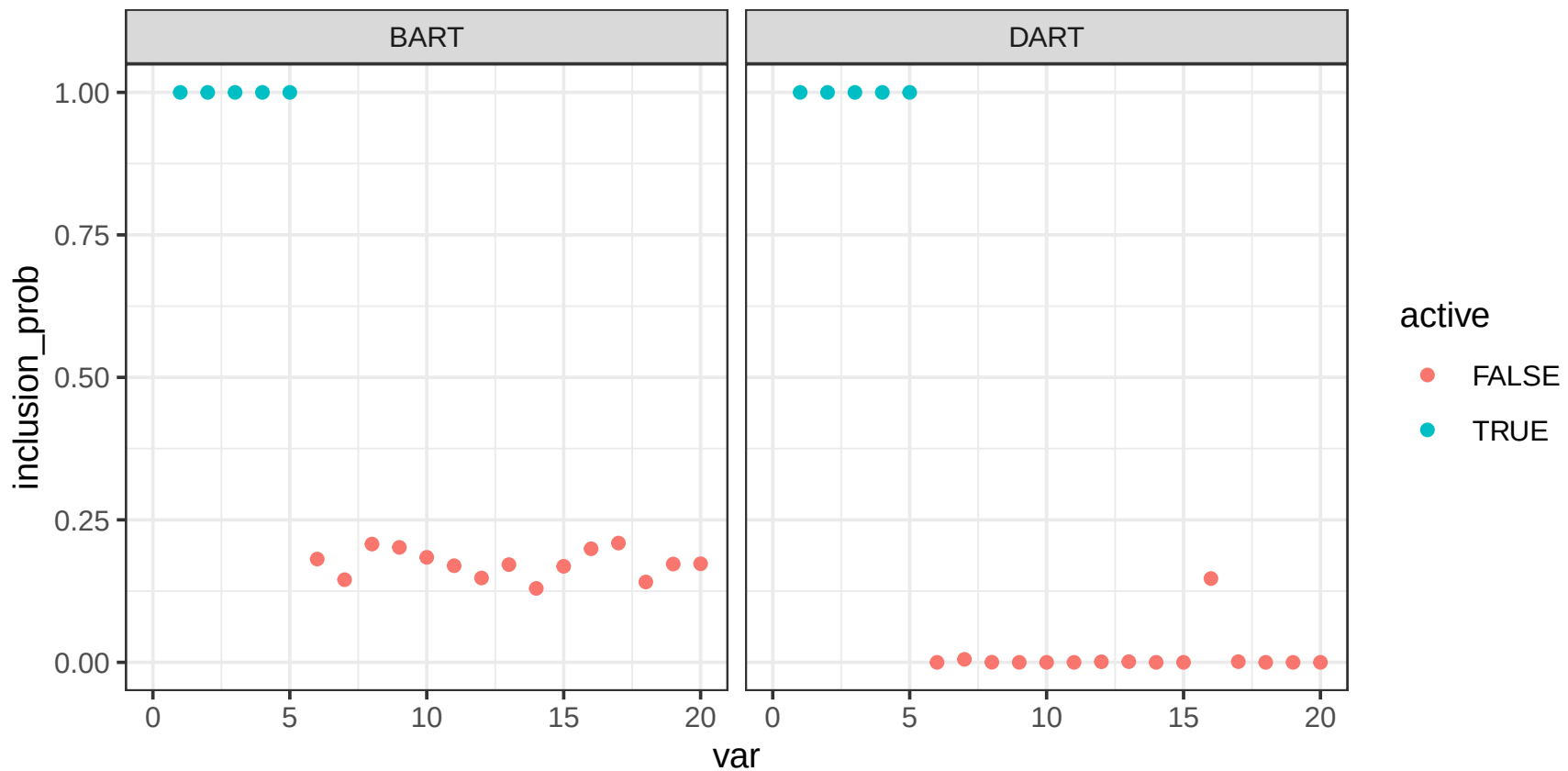
Variable Inclusion

```
var_inc_df <- data.frame(
  inclusion_prob = colMeans(cbind(fit_bart$varcount, fit_dart$varcount) > 0),
  var            = rep(1:1000, 2),
  method         = rep(c("BART", "DART"), each = 1000)
)
var_inc_df$active = var_inc_df$var <= 5

ggplot(var_inc_df, aes(x = var, y = inclusion_prob, color = active)) +
  geom_point(alpha=3) + facet_wrap(~method) + theme_bw()
```



Variable Inclusion

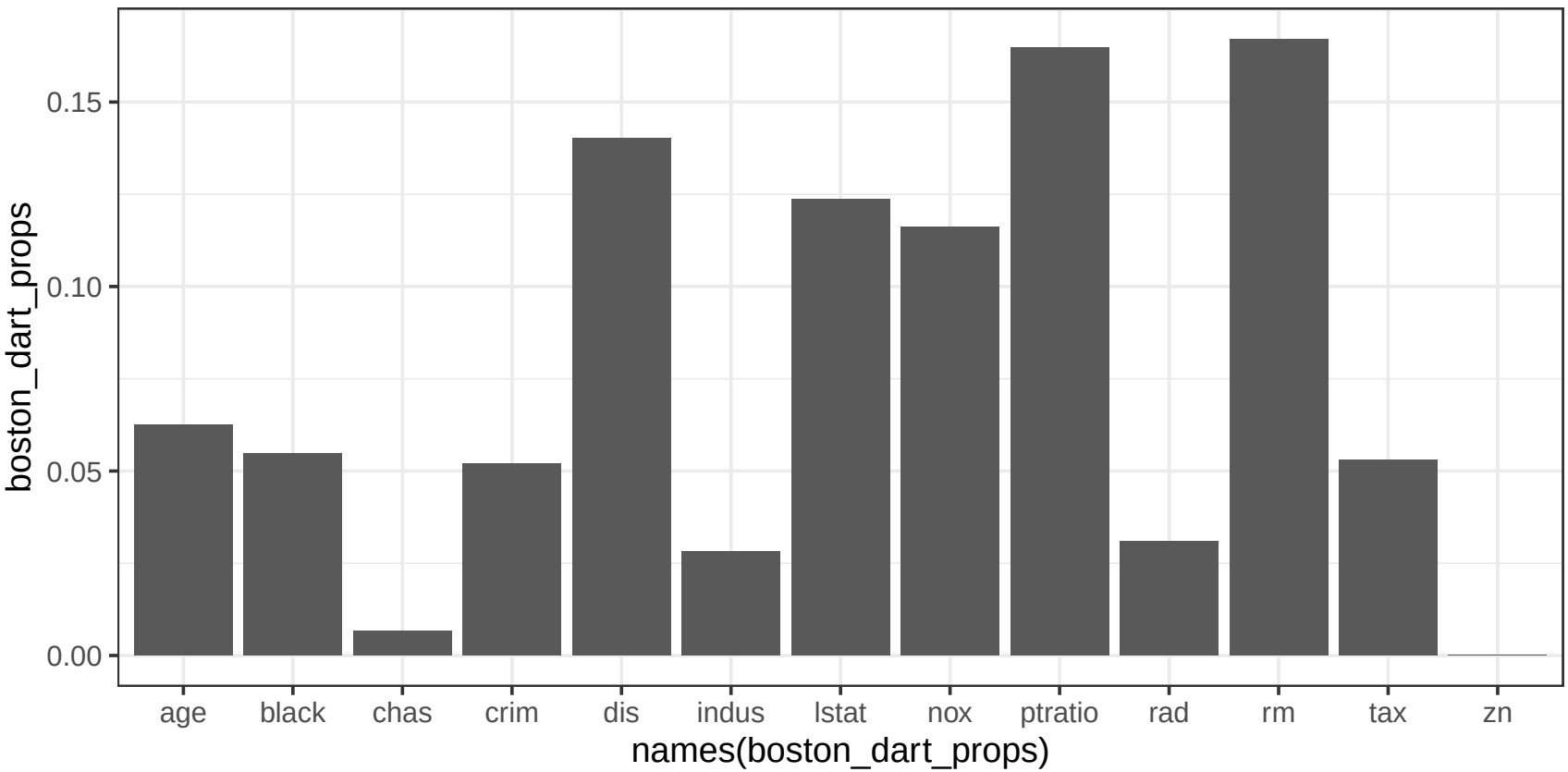
```
var_inc_df <- data.frame(
  inclusion_prob = colMeans(cbind(fit_bart$varcount, fit_dart$varcount) > 0),
  var            = rep(1:1000, 2),
  method         = rep(c("BART", "DART"), each = 1000)
)
var_inc_df$active = var_inc_df$var <= 5

ggplot(var_inc_df, aes(x = var, y = inclusion_prob, color = active)) +
  geom_point(alpha=3) + facet_wrap(~method) + theme_bw() +
  xlim(0,20)
```



Boston

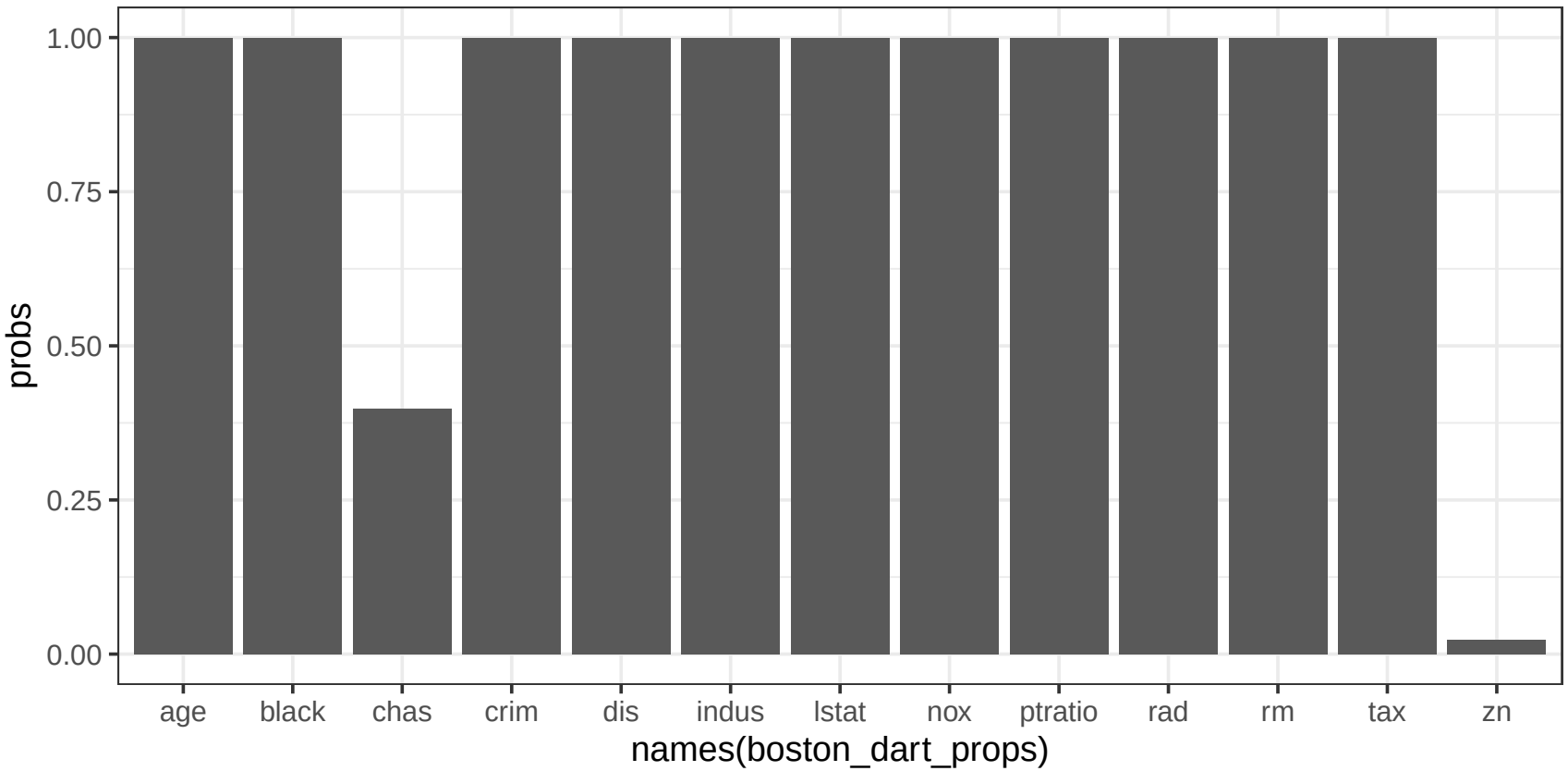
```
boston_dart <- bartMachine(X = boston %>% select(-medv), y = boston$medv,
                           alpha_0 = 1.6, do_ard = TRUE, do_prior = FALSE,
                           serialize = TRUE, seed = 4744777)
```

```
boston_dart_props <- get_var_props_over_chain(boston_dart)
ggplot() + geom_col(aes(x = names(boston_dart_props), y = boston_dart_props)) +
  theme_bw()
```



Boston Inclusion Probs

```
var_counts <- get_var_counts_over_chain(boston_dart)
probs      <- colMeans(var_counts > 0)
ggplot() + geom_col(aes(x = names(boston_dart_props), y = probs)) +
  theme_bw()
```



Predictive Performance

```
set.seed(774837)

folds <- caret::createFolds(1:nrow(boston), k = 5)

boston_train <- boston[-folds[[1]],]
boston_test  <- boston[folds[[1]],]

lm_boston <- lm(medv ~ ., data = boston_train)
cor(boston_test$medv, predict(lm_boston, boston_test))

fitted_dart <- bartMachine(X = boston_train %>% select(-medv),
                           y = boston_train$medv, num_trees = 200,
                           num_burn_in = 4000, num_iterations_after_burn_in = 4000,
                           alpha_0 = 1.6, do_ard = TRUE, do_prior = FALSE,
                           seed = 112231)

fitted_bart <- bartMachine(X = boston_train %>% select(-medv),
                           y = boston_train$medv, num_trees = 200,
                           num_burn_in = 4000, num_iterations_after_burn_in = 4000,
                           seed = 112231)


cor(boston_test$medv, predict(fitted_dart, boston_test %>% select(-medv)))
cor(boston_test$medv, predict(fitted_bart, boston_test %>% select(-medv)))


colMeans(get_var_counts_over_chain(fitted_dart) > 0)
colMeans(get_var_counts_over_chain(fitted_bart) > 0)
```

Eliminating chas and zn Entirely

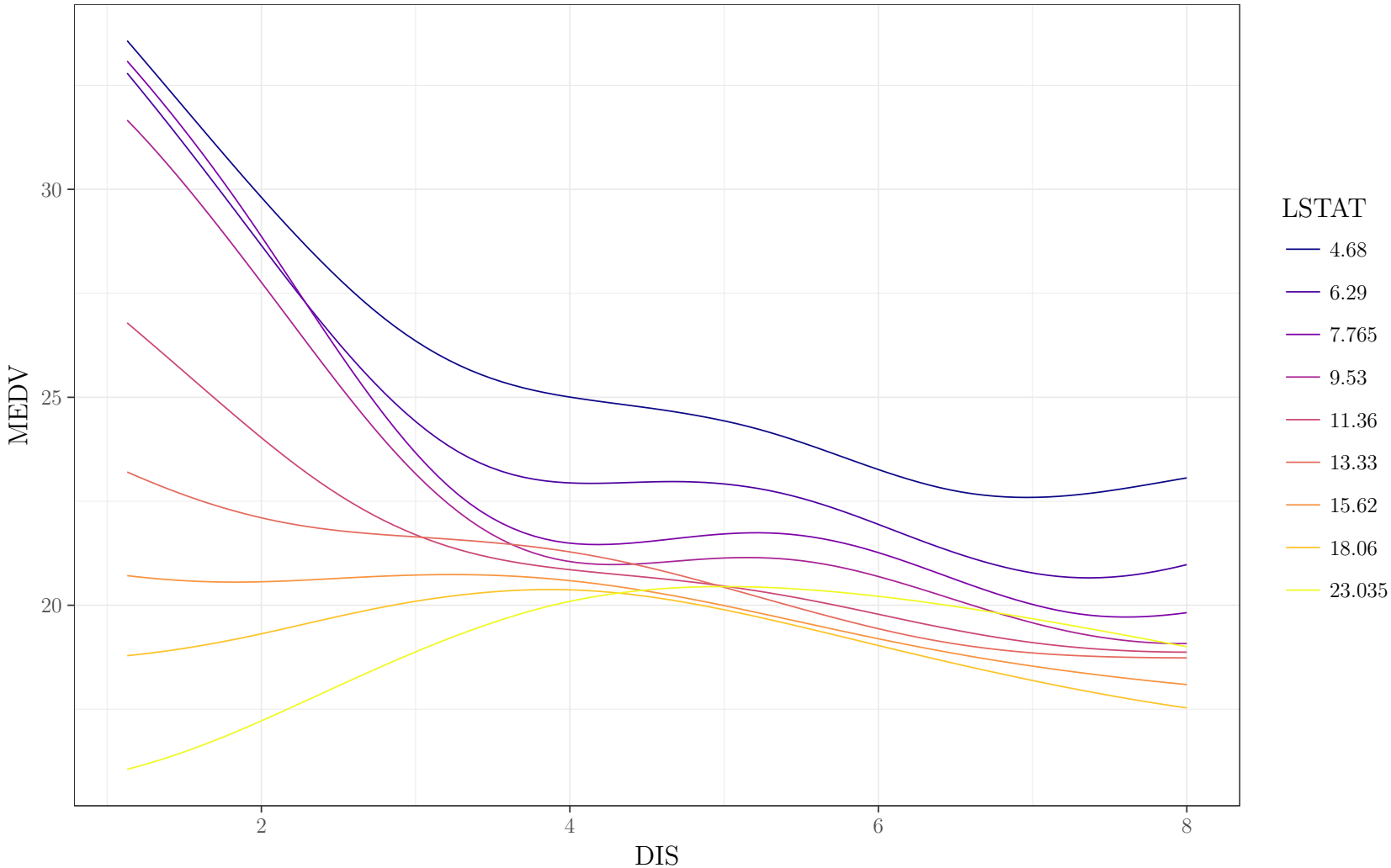
```
fitted_subset <- bartMachine(
  X = boston_train %>% select(-medv, -chas, -zn),
  y = boston_train$medv,
  num_trees = 200,
  num_burn_in = 4000,
  num_iterations_after_burn_in = 4000,
  seed = 112231)

mu_hat_subset <- predict(
  fitted_subset, boston_test %>% select(-medv, -chas, -zn)
)

cor(boston_test$medv, mu_hat_subset)
```

Interaction Detection

We may be interested not only in which variables are important, but which *interactions* between the variables are important.

Two variables *interact* if there exists a leaf node in the ensemble whose path from the root involves decisions on both variables.

Example: Interaction Detection in Boston Housing



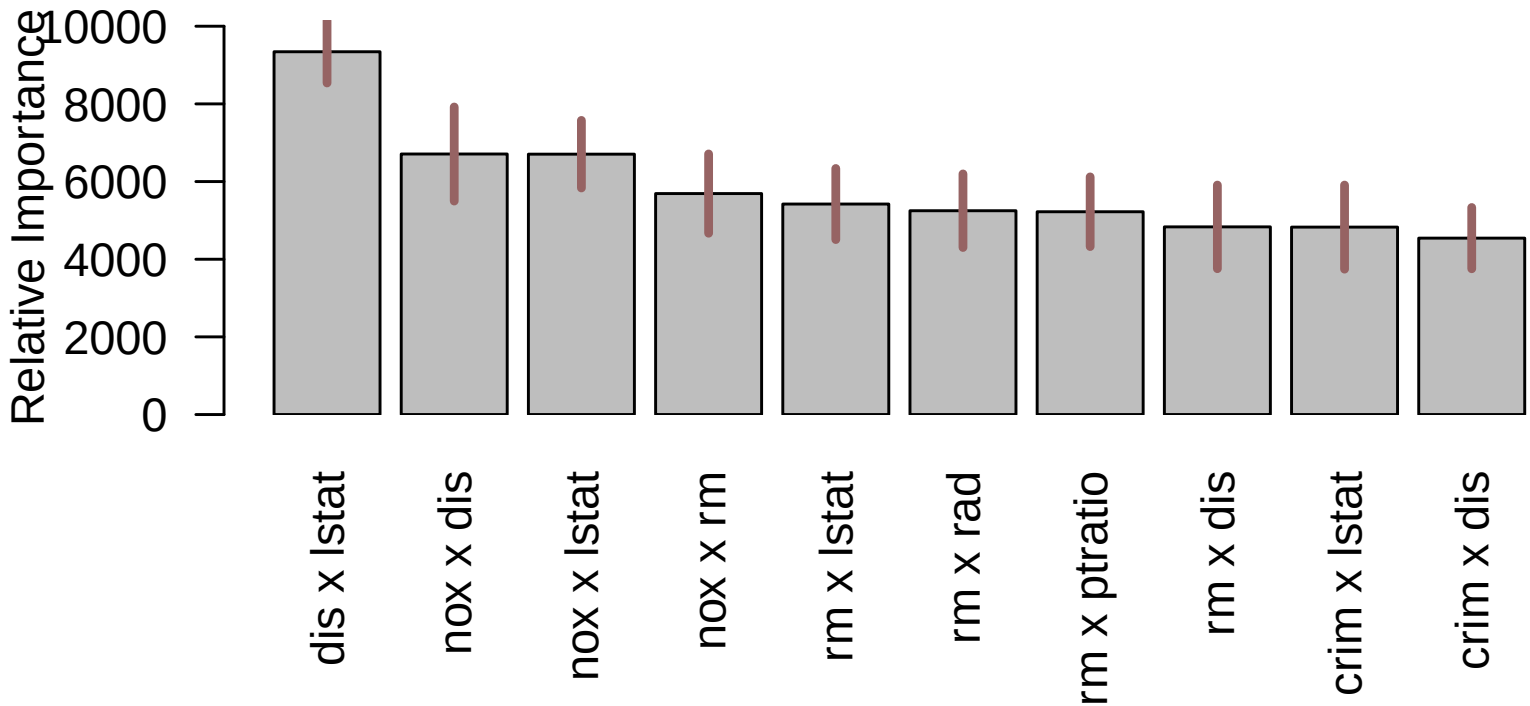
(Du and Linero, 2019)

Example: Interaction Detection in Boston Housing

```
## Packages ----

options(java.parameters = "-Xmx5g")
library(bartMachine)
library(SoftBart)
library(BART)
library(tidyverse)

## Load Data ----

boston <- MASS::Boston

## Fit a BART ----

boston_bart <- bartMachine(X = boston |> select(-medv),
                           y = boston$medv,
                           num_trees = 50,
                           num_burn_in = 4000,
                           num_iterations_after_burn_in = 4000,
                           seed = digest::digest2int("bart interact"))

## Do Interaction Detection ----

boston_interact <- bartMachine::interaction_investigator(
  bart_machine = boston_bart,
  num_var_plot = 10,
  num_replicates_for_avg = 25
)
```

Relative Importance

Bar heights: number of times in the ensemble, across all trees and MCMC iterations, that the interaction appeared as a splitting rule.



Interaction Detection

- To the best of my knowledge, there is no great fully-Bayesian way of doing interaction detection for these models; the models include *lots* of spurious interactions, so it is hard to draw firm conclusions.

Interaction Detection

- To the best of my knowledge, there is no great fully-Bayesian way of doing interaction detection for these models; the models include *lots* of spurious interactions, so it is hard to draw firm conclusions.
- (Du and Linero, 2019) is an *attempt* to do interaction detection using a "Dirichlet Process Forest." Works well enough in my experience, but no public software yet.

Other Variable Selection Approaches: Fit-the-Fit

Idea: given a Bayes estimate $\widehat{r}(x)$ of $r_0(x)$, a variable x_j is "unimportant" if we can approximate $\widehat{r}(x)$ with some $\widetilde{r}(x)$ *which does not depend on* x_j.

Other Variable Selection Approaches: Fit-the-Fit

Idea: given a Bayes estimate $\widehat{r}(x)$ of $r_0(x)$, a variable x_j is "unimportant" if we can approximate $\widehat{r}(x)$ with some $\widetilde{r}(x)$ *which does not depend on* x_j.

Package `nonlinvarsel` uses this idea, taking $\widetilde{r}(x)$ to be a deep decision tree (but could in principle be anything, including another BART). **Measures quality of subset** $S \subseteq \{1, \ldots, P\}$ **via correlation between** $\widetilde{r}(x)$ **and** $\widehat{r}(x)$**.**

More Software

```
install.packages('foreach', dep = T)
url <- 'http://www.rob-mcculloch.org/chm/nonlinvarsel_0.0.1.9001.tar.gz'
download.file(url, destfile = 'temp')
install.packages('temp', repos = NULL, type='source')
```

Example Code

```
options(java.parameters = "-Xmx5g")
library(bartMachine)
library(tidyverse)
library(nonlinvarsel)

boston <- MASS::Boston

fit_bart <- bartMachine(X = boston |> select(-medv),
                        y = boston |> pluck("medv"),
                        num_trees = 200,
                        num_burn_in = 5000,
                        num_iterations_after_burn_in = 5000,
                        seed = 111211211)

var_sel <- nonlinvarsel::vsa(X = boston |> select(-medv),
                             fhat = fit_bart$y_hat_train)

plot(var_sel)
```

Stuff We Did Not Cover

- **Theory for Models Discussed**
  - Linear models: Castillo et al. (2015); Scott and Berger (2010)
  - BART theory: Linero and Yang (2018); Rockova and van der Pas (2017)
  - Additive models: Bai et al. (2020)
- **Other Important Topics**
  - Continuous shrinkage priors: Carvalho et al. (2010); Datta et al. (2013)
  - Hyperparameter selection: Liang et al. (2008)
  - Extensions to GLMs: Albert and Chib (1993); Polson et al. (2013); Murray (2020); Linero et al. (2018)

Summarizing

1. BART is really good for prediction!
   - Works well under sparsity with Dirichlet prior.
2. Lots of neat tools for variable selection:
   - EM algorithm
   - Fully-Bayes with Dirichlet
   - Fit-the-fit approach
3. Not-quite-as-good, but still useful, tools for interaction detection

Albert, J. H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88:669–679.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M. R. (2020). Spike-and-slab group lassos for grouped regression and sparse generalized additive models. *Journal of the American Statistical Association*, pages 1–14.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480.

Castillo, I., Schmidt-Hieber, J., and Van der Vaart, A. (2015). Bayesian linear regression with sparse priors. *The Annals of Statistics*, 43(5):1986–2018.

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298.

Datta, J., Ghosh, J. K., et al. (2013). Asymptotic properties of bayes risk for the horseshoe prior. *Bayesian Analysis*, 8(1):111–132.

Denison, D. G., Mallick, B. K., and Smith, A. F. (1998). A Bayesian CART algorithm. *Biometrika*, 85(2):363–377.

Du, J. and Linero, A. R. (2019). Interaction detection with bayesian decision tree ensembles. In *22nd Proceedings of the International Conference on Artificial Intelligence in Statistics (AISTATS)*.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 4(5):771–780.

Kapelner, A. and Bleich, J. (2016). bartMachine: Machine learning with Bayesian additive regression trees. *Journal of Statistical Software*, 70(4):1–40.

Liang, F., Paulo, R., Molina, G., Clyde, M. A., and Berger, J. O. (2008). Mixtures of g priors for bayesian variable selection. *Journal of the American Statistical Association*, 103(481):410–423.

Linero, A. R., Sinha, D., and Lipsitz, S. R. (2018). Semiparametric Mixed-Scale Models Using Shared Bayesian Forests. *arXiv e-prints arXiv:1809.08521*.

Linero, A. R. and Yang, Y. (2018). Bayesian regression tree ensembles that adapt to smoothness and sparsity. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(5):1087–1110.

Murray, J. S. (2020). Log-linear Bayesian additive regression trees for multinomial logistic and count regression models. *Journal of the American Statistical Association*. To appear.

Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349.

Rockova, V. and van der Pas, S. (2017). Posterior concentration for Bayesian regression trees and their ensembles. *arXiv preprint arXiv:1078.08734*.

Scott, J. G. and Berger, J. O. (2010). Bayes and empirical-bayes multiplicity adjustment in the variable-selection problem. *The Annals of Statistics*, 38(5):2587–2619.