bartMachine: Machine Learning with Bayesian Additive Regression Trees

Adam Kapelner
The Wharton School of the
University of Pennsylvania

Justin Bleich
The Wharton School of the
University of Pennsylvania

Abstract

We present a new package in R implementing Bayesian additive regression trees (BART). The package introduces many new features for data analysis using BART such as variable selection, interaction detection, model diagnostic plots, incorporation of missing data and the ability to save trees for future prediction. It is significantly faster than the current R implementation, parallelized, and capable of handling both large sample sizes and high-dimensional data.

Keywords: Bayesian, machine learning, statistical learning, non-parametric, R, Java.

1. Introduction

Ensemble-of-trees methods have become popular choices for forecasting in both regression and classification problems. Algorithms such as random forests (Breiman 2001) and stochastic gradient boosting (Friedman 2002) are two well-established and widely employed procedures. Recent advances in ensemble methods include dynamic trees (Taddy, Gramacy, and Polson 2011) and Bayesian additive regression trees (BART, Chipman, George, and McCulloch 2010), which depart from predecessors in that they rely on an underlying Bayesian probability model rather than a pure algorithm. BART has demonstrated substantial promise in a wide variety of simulations and real world applications such as predicting avalanches on mountain roads (Blattenberger and Fowles 2014), predicting how transcription factors interact with DNA (Zhou and Liu 2008) and predicting movie box office revenues (Eliashberg 2010). This paper introduces **bartMachine**, a new R (R Core Team 2014) package available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=bartMachine` that significantly expands the capabilities of using BART for data analysis.

Currently, there exists one other implementation of BART on CRAN: **BayesTree** (Chipman and McCulloch 2010), the package developed by the algorithm's original authors. One of the major drawbacks of this implementation is its lack of a `predict` function. Test data must be provided as an argument during the training phase of the model. Hence it is impossible to generate forecasts on future data without re-fitting withe entire model. Since the run time is not trivial, forecasting becomes an arduous exercise. A significantly faster implementation of BART that contains master-slave parallelization exists as Pratola, Chipman, Higdon, McCulloch, and Rust (2013), but this is only available as standalone C++ source code and not integrated with R.

The goal of **bartMachine** is to provide a fast, easy-to-use, visualization-rich machine learning package for R users. Our implementation of BART is in Java and is integrated into R via **rJava** (Urbanek 2013). From a runtime perspective, our algorithm is significantly faster and is parallelized, allowing computation on as many cores as desired. Not only is the model construction itself parallelized, but the additional features such as prediction, variable selection, and many others can be divided across cores as well.

Additionally, we include a variety of expanded and new features. We implement the ability to save trees in memory and provide convenience functions for prediction on test data. We also include plotting functions for both credible and predictive intervals and plots for visually inspecting convergence of BART's Gibbs sampler. We expand variable importance exploration to include permutation tests and interaction detection. We implement recently developed features for BART including a principled approach to variable selection and the ability to incorporate in prior information for covariates (Bleich, Kapelner, Jensen, and George 2013). We also implement the strategy found in Kapelner and Bleich (2013) to incorporate missing data during training and handle missingness during prediction.

In Section 2, we provide an overview of BART with a special emphasis on the features that have been extended. In Section 3 we provide a general introduction to the package, highlighting the novel features. Section 4 provides step-by-step examples of the regression capabilities and Section 5 introduces additional step-by-step examples of features unique to classification problems. We conclude in Section 6. Appendix A discusses the details of our implementation and how it differs from **BayesTree**. Appendix B offers predictive performance comparisons.

2. Overview of BART

BART provides a unique approach to nonparametric function estimation using regression trees. Regression trees rely on recursive binary partitioning of predictor space into a set of hyperrectangles in order to approximate some unknown function f. Such trees have received praise for their ability to flexibly fit interactions and nonlinearities. Models composed of sums of regression trees are able to capture additive effects in f better than single trees.

BART can be considered a sum-of-trees ensemble, with a novel estimation approach relying on a fully Bayesian probability model. Specifically, the BART model can be expressed as:

$$\boldsymbol{Y} = f(\boldsymbol{X}) + \boldsymbol{\mathcal{E}} \approx \text{🌲}_1^{\text{🍃}}(\boldsymbol{X}) + \text{🌲}_2^{\text{🍃}}(\boldsymbol{X}) + \ldots + \text{🌲}_m^{\text{🍃}}(\boldsymbol{X}) + \boldsymbol{\mathcal{E}}, \qquad \boldsymbol{\mathcal{E}} \sim \mathcal{N}_n\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n\right) \tag{1}$$

Here we have m distinct regression trees, each composed of a tree structure, denoted by 🌲, and the parameters at the terminal nodes (also called leaves), denoted by 🍃. The two together, denoted as $\text{🌲}^{\text{🍃}}$ represents an entire tree with both its structure and set of leaf parameters.

The structure of a given tree 🌲_t includes information on how any observation recurses down the tree. For each nonterminal (internal) node of the tree, there is a "splitting rule" taking the form $\boldsymbol{x}_j < c$ consisting of the "splitting variable" $\boldsymbol{x}_j$ and the "splitting value" c. An observation moves to the left daughter node if the condition set by the splitting rule is satisfied and to the right daughter node otherwise. The process continues until a terminal node is reached. Then, the observation receives the leaf value of the terminal node as its predicted value. We denote the set of tree's leaf parameters as $\text{🍃}_t = \{\mu_{t,1}, \mu_{t,2}, \ldots, \mu_{t_{b_t}}\}$ where b_t is the number of terminal nodes for a given tree.

BART can be distinguished from other ensemble-of-trees models due to its underlying probability model. As a Bayesian model, BART consists of a set of priors for the structure and the leaf parameters and a likelihood for data in the terminal nodes. The aim of the priors is to provide regularization, preventing any single regression tree from dominating the total fit.

We provide an overview of the BART priors and likelihood and then discuss how draws from the posterior distribution are made. A more complete exposition can be found in Chipman *et al.* (2010).

2.1. Priors and likelihood

There are three priors for the BART model: a prior on the tree structure itself, a prior on the leaf parameters, and a prior on the error variance σ^2. The prior on σ^2 is independent from the other two and each tree is independent, yielding:

$$\begin{aligned}
\mathbb{P}\left(\text{🌲}^{\text{🍃}}_1, \ldots, \text{🌲}^{\text{🍃}}_m, \sigma^2\right) &= \left[\prod_t \mathbb{P}\left(\text{🌲}^{\text{🍃}}_t\right)\right] \mathbb{P}\left(\sigma^2\right) = \left[\prod_t \mathbb{P}\left(\text{🍃}_t \mid \text{🌲}_t\right) \mathbb{P}\left(\text{🌲}_t\right)\right] \mathbb{P}\left(\sigma^2\right) \\
&= \left[\prod_t \prod_\ell \mathbb{P}\left(\mu_{t,\ell} \mid \text{🌲}_t\right) \mathbb{P}\left(\text{🌲}_t\right)\right] \mathbb{P}\left(\sigma^2\right)
\end{aligned}$$

where the last line follows from an additional assumption of conditional independence of the leaf parameters given the tree's structure.

The first prior is on the locations of nodes within the tree. Nodes at depth d are nonterminal with probability $\alpha(1+d)^{-\beta}$ where $\alpha \in (0, 1)$ and $\beta \in [0, \infty]$. This prior keeps the trees shallow, limiting complexity of any single tree. Default values for these hyperparameters of $\alpha = 0.95$ and $\beta = 2$ are recommended by Chipman *et al.* (2010).

For nonterminal nodes, splitting rules have the following prior. First, a predictor is randomly selected to serve as the splitting variable. In the original formulation, each available predictor is equally likely to be chosen, but this is relaxed in our implementation to allow an arbitrary discrete distribution (see Section 4.11). Then, the splitting value is selected by randomly choosing a value of the selected predictor with equal probability.

The third prior is on the leaf parameters. Given a tree with a set of terminal nodes, each terminal node (or leaf) has a continuous parameter (the leaf parameter) representing the "best guess" of the response in this partition of predictor space. This parameter is the fitted value assigned to any observation that lands in that node. The prior on each of the leaf parameters is given as: $\mu_\ell \overset{iid}{\sim} \mathcal{N}\left(\mu_\mu, \sigma^2_\mu\right)$. The expectation, μ_μ, is picked to be the range center, $(y_{\min} + y_{\max})/2$. The variance is empirically chosen so that the range center plus or minus $k = 2$ variances cover 95% of the provided response values in the training set (by default). The aim of this prior is to provide model regularization by shrinking the leaf parameters towards the center of the distribution of the response.

The final prior is on the error variance and is chosen to be $\sigma^2 \sim \text{InvGamma}\left(\nu/2, \nu\lambda/2\right)$. λ is determined from the data so that there is a $q = 90\%$ a priori chance (by default) that the BART model will improve upon the RMSE from an ordinary least squares regression.

Therefore, the majority of the prior probability mass lies below the RMSE from least squares regression. Additionally, this prior limits the probability mass placed on small values of σ^2 to prevent overfitting.

Note that the adjustable hyperparameters are α, β, k, ν and q. Default values generally provide good performance, but optimal tuning can be achieved via cross-validation, an automatic feature implemented and described in Section 4.2.

Along with a set of priors, BART specifies the likelihood of responses in the terminal nodes. They are assumed a priori Normal with the mean being the "best guess" in the leaf at the current moment (in the Gibbs sampler) and variance being the best guess of the variance at the moment i.e., $\boldsymbol{y}_\ell \sim \mathcal{N}\left(\mu_\ell, \sigma^2/m\right)$. Note that σ^2 is scaled by the number of trees in order that the sum of the variances across the m trees is σ^2.

2.2. Posterior distribution and prediction

A Gibbs sampler (Geman and Geman 1984) is employed to generate draws from the posterior distribution of $\mathbb{P}(\text{🌲}_1^{\text{🍃}}, \ldots, \text{🌲}_m^{\text{🍃}}, \sigma^2 \mid \boldsymbol{y})$. A key feature of the Gibbs sampler for BART is to employ a form of "Bayesian backfitting" (Hastie and Tibshirani 2000) where the jth tree is fit iteratively, holding all other $m-1$ trees constant by exposing only the residual response that remains unfitted:

$$\boldsymbol{R}_j := \boldsymbol{y} - \sum_{t \neq j} \text{🌲}_t^{\text{🍃}}(\boldsymbol{X}). \tag{2}$$

The Gibbs sampler,

$$\begin{aligned}
1: &\quad \text{🌲}_1 \mid \boldsymbol{R}_{-1}, \sigma^2 \\
2: &\quad \text{🍃}_1 \mid \text{🌲}_1, \boldsymbol{R}_{-1}, \sigma^2 \\
3: &\quad \text{🌲}_2 \mid \boldsymbol{R}_{-2}, \sigma^2 \\
4: &\quad \text{🍃}_2 \mid \text{🌲}_2, \boldsymbol{R}_{-2}, \sigma^2 \\
&\;\;\vdots \\
2m-1: &\quad \text{🌲}_m \mid \boldsymbol{R}_{-m}, \sigma^2 \\
2m: &\quad \text{🍃}_m \mid \text{🌲}_m, \boldsymbol{R}_{-m}, \sigma^2 \\
2m+1: &\quad \sigma^2 \mid \text{🌲}_1, \text{🍃}_1, \ldots, \text{🌲}_m, \text{🍃}_m, \boldsymbol{\mathcal{E}},
\end{aligned} \tag{3}$$

proceeds by first proposing a change to the first tree's structure 🌲 which are accepted or rejected via a Metropolis-Hastings step (Hastings 1970). Note that sampling from the posterior of the tree structure does not depend on the leaf parameters, as they can be analytically margined out of the computation (see Appendix A.1). Given the tree structure, samples from the posterior of the b leaf parameters $\text{🍃}_1 := \{\mu_1, \ldots, \mu_b\}$ are then drawn. This procedure proceeeds iteratively for each tree, using the updated set of partial residuals $\boldsymbol{R}_j$. Finally, conditional on the updated set of tree structures and leaf parameters, a draw from the posterior of σ^2 is made based on the full model residuals $\boldsymbol{\mathcal{E}} := \boldsymbol{y} - \sum_{t=1}^m \text{🌲}_t^{\text{🍃}}(\boldsymbol{X})$.

Within a given terminal node, since both the prior and likelihood are normally distributed, the posterior of each of the leaf parameters in 🍃 is conjugate normal with its mean being a weighted combination of the likelihood and prior parameters (lines $2,\ 4, \ldots, 2m$ in Equation set 3). Due to the normal-inverse-gamma conjugacy, the posterior of σ^2 is inverse gamma as well (line $2m+1$ in Equation set 3). The complete expressions for these posteriors can be found in Gelman, Carlin, Stern, and Rubin (2004).

Lines $1,\ 3, \ldots,\ 2m-1$ in Equation set 3 rely on Metropolis-Hastings draws from the posterior of the tree distributions. These involve introducing small perturbations to the tree structure: growing a terminal node by adding two daughter nodes, pruning two daughter nodes (rendering their parent node terminal), or changing a split rule. We denote these possible alterations as: GROW, PRUNE, and CHANGE.[1] The mathematics associated with the Metropolis-Hastings step is simple but is tedious, and we refer the interested reader to Appendix A for the explicit calculations. Once again, over many Gibbs samples, trees can dynamically morph their structure in an effort to capture the fit left currently unexplained.

Pratola *et al.* (2013) argue that a CHANGE step is unnecessary for sufficient mixing of the Gibbs sampler. While we too observed this to be true for estimates of the posterior means, we found that omitting CHANGE can negatively impact the variable inclusion proportions (the feature introduced in Section 4.6). As a result, we implement a modified CHANGE where we only propose new splits for nodes that are singly internal: both children nodes are terminal nodes (details are given in Appendix A.3).

All $2m+1$ steps represent a *single* Gibbs iteration. We have observed that generally no more than 1,000 iterations are needed as "burn-in" (see Section 4.4 for convergence diagnostics). An additional 1,000 iterations is usually sufficient to serve as draws from the posterior for $f(\boldsymbol{x})$. A single predicted value $\hat{f}(\boldsymbol{x})$ can be obtained by taking the average of the posterior values and a quantile estimate can be obtained by computing the appropriate quantile of the posterior values. Additional features of the posterior distribution will be discussed in Section 4.

2.3. BART for classification

BART can easily be modified to handle classification problems for categorical response variables. In Chipman *et al.* (2010), only binary outcomes were explored but recent work has extended BART to the multiclass problem (Kindo, Wang, and Pe 2013). Our implementation handles binary classification and we plan to implement multiclass outcomes in a future release.

For the binary classification problem (coded with outcomes "0" and "1"), we assume a probit model,

$$\mathbb{P}\left(\boldsymbol{Y}=1 \mid \boldsymbol{X}\right)=\Phi\left(\text{🌲}_1^{\text{🍃}}(\boldsymbol{X})+\text{🌲}_2^{\text{🍃}}(\boldsymbol{X})+\ldots+\text{🌲}_m^{\text{🍃}}(\boldsymbol{X})\right),$$

where Φ denotes the cumulative density function of the standard normal distribution. In this formulation, the sum-of-trees model serves as an estimate of the conditional probit at $\boldsymbol{x}$ which can be easily transformed into a conditional probability estimate of $Y=1$.

[1] In the original formulation, Chipman *et al.* (2010) include an additional alteration called SWAP. Due to the complexity of bookkeeping associated with this alteration, we do not implement it.

In the classification setting, the prior on σ^2 is not needed as the model assumes $\sigma^2 = 1$. The prior on the tree structure remains the same as in the regression setting and a few minor modifications are required for the prior on the leaf parameters.

Sampling from the posterior distribution is again obtained via Gibbs sampling with a Metropolis-Hastings step outlined in Section 2.2. Following the data augmentation approach of Albert and Chib (1993), an additional vector of latent variables $\boldsymbol{Z}$ is introduced into the Gibbs sampler. Then, a new step is created in the Gibbs sampler where draws of $\boldsymbol{Z} \,|\, \boldsymbol{y}$ are obtained by conditioning on the sum-of-trees model:

$$
\begin{aligned}
Z_i \mid y_i = 1 &\sim \max\left\{ N\left(\sum_t \mathcal{T}_t^{\mathcal{M}}\left(\boldsymbol{X}\right), 1\right), 0 \right\} \quad \text{and} \\
Z_i \mid y_i = 0 &\sim \min\left\{ N\left(\sum_t \mathcal{T}_t^{\mathcal{M}}\left(\boldsymbol{X}\right), 1\right), 0 \right\}.
\end{aligned}
$$

Next, $\boldsymbol{Z}$ is used as the response vector instead of $\boldsymbol{y}$ in all steps of Equation 3.

Upon obtaining a sufficient number of samples from the posterior, inferences can be made using the the posterior distribution of conditional probabilities and classification can be undertaken by applying a threshold to the to the means (or another summary) of these posterior probabilities. The relevant classification features of **bartMachine** are discussed in Section 5.

3. The bartMachine package

The package **bartMachine** provides a novel implementation of Bayesian additive regression trees in R. The algorithm is substantially faster than the current R package **BayesTree** and our implementation is parallelized at the Gibbs sample level during prediction. Additionally, the interface with **rJava** allows for the entire posterior distribution of tree ensembles to persist throughout the R session, allowing for prediction and other calls to the trees without having to re-run the Gibbs sampler (a limitation in the current implementation). The model object cannot persist across sessions (using R's save command for instance) and we view the addition of this feature as future work. Since our implementation is different from **BayesTree**, we provide a predictive accuracy bakeoff on different datasets in Appendix B which illustrates that the two are about equal.

3.1. Speed improvements and parallelization

We make a number of significant speed improvements over the original implementation.

First, **bartMachine** is fully parallelized (with the number of cores customizable) during model creation, prediction, and many of the other features. During model creation, we chose to parallelize by creating one independent Gibbs chain per core. Thus, if we have 500 burn-in samples and 1,000 post burn-in samples and four cores, each core would sample 750 samples: 500 for a burn-in and 250 post burn-in samples. The final model will aggregate the 250 post burn-in samples for the four cores yielding the desired 1,000 total burn-in samples. This has the drawback of effectively running the burn-in serially, but has the added benefit of reducing auto-correlation of the sum-of-trees samples in the posterior samples since the chains are independent which may provide greater predictive performance. Parallelization at the level of likelihood calculations is left for a future release. Parallelization for prediction and other features scale linearly in the number of cores.

Additionally, we take advantage of a number of additional computational shortcuts:

1. Computing the unfitted responses for each tree (Equation 2) can be accomplished by keeping a running vector and making entry-wise updates as the Gibbs sampler (Equation 3) progresses from step 1 to $2m$. Additionally, during the σ^2 sampling (step $2m+1$), the residuals do not have to be computed by dropping the data down all the trees.

2. Each node caches its acceptable variables for split rules and the acceptable unique split values so they do not need to be calculated at each tree sampling step. This speed enhancement, which we call *memcache* comes at the expense of memory and may cause issues for large data sets. We include a toggle in our implementation defaulted to "on."

3. Careful calculations in Appendix A eliminate many unnecessary computations. For instance, the likelihood ratios are only functions of the squared sum of responses and no longer require computing the sum of the responses squared.

Figure 1 displays model creation speeds for different values of n on a linear model with $p = 20$, normally distributed covariates, $\beta_1, \ldots, \beta_{20} \overset{iid}{\sim} \text{U}\left(-1,\, 1\right)$, and standard normal noise. Note that we do not vary p as it was already shown in Chipman *et al.* (2010) that BART's computation time is largely unaffected by the dimensionality of the problem (relative to the influence of sample size). We include results for BART using **BayesTree**, **bartMachine** with one and four cores, the *memcache* option on and off, as well as four cores, *memcache* off and computation of in-sample statistics off (all with $m = 50$ trees). We also include random forests via the package **randomForest** (Liaw and Wiener 2002) with its default settings.

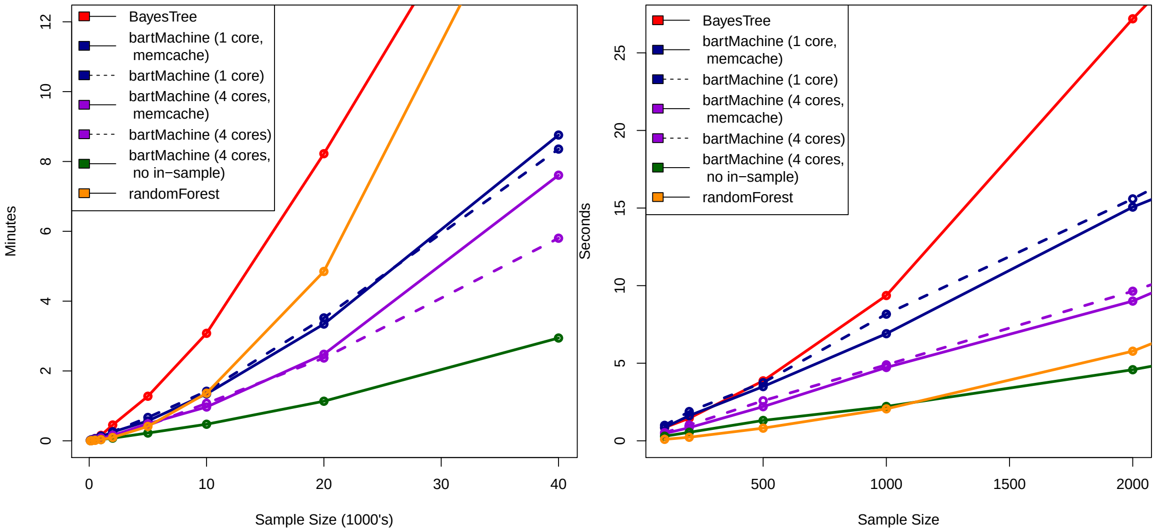
We first note that Figure 1a demonstrates that the **bartMachine** model creation runtime is approximately linear in n. There is about a 30% speed-up when using four cores instead of one. The *memcache* enhancement should be turned off only with sample sizes larger than $n = 20,000$. Noteworthy is the 50% reduction in time of constructing the model when not computing in-sample statistics. In-sample statistics are computed by default because the user generally wishes to see them. Also, for the purposes of this comparison, **BayesTree** models compute the in-sample statistics by necessity since the trees are not saved. The **randomForest** implementation becomes slower just after $n = 1,000$ due to its reliance on a greedy exhaustive search at each node.

Figure 1b displays results for smaller sample sizes ($n \leq 2,000$) that are often encountered in practice. We observe the *memcache* enhancement provides about a 10% speed improvement. Thus, if memory is an issue, it can be turned off with little performance degradation.

3.2. Missing data in bartMachine

bartMachine implements a native method for incorporating missing data into both model creation and future prediction with test data. The details are given in Kapelner and Bleich (2013) but we provide a brief summary here.

There are a number of ways to incorporate missingness into tree-based methods (see Ding and Simonoff 2010 for a review). The method implemented here is known as "Missing Incorporated



(a) Large sample sizes

(b) Small sample sizes

Figure 1: Model creation times as a function of sample size for a number of settings of **bartMachine**, **BayesTree** and `RandomForests`. Simulations were run on a quad-core 3.4GHz Intel i5 desktop with 24GB of RAM running the Windows 7 64bit operating system.

in Attributes" (MIA, Twala, Jones, and Hand 2008, section 2) which natively incorporates missingness by augmenting the nodes' splitting rules to (a) also handle sorting the missing data to the left or right and (b) use missingness *itself* as a variable to be considered in a splitting rule. Algorithm 1 summarizes these new splitting rules as they are implemented within the package.

Implementing MIA into the BART procedure is straightforward. These new splitting rules are sampled uniformly during the GROW or CHANGE steps. For example, a splitting rule might be "$\boldsymbol{x}_j < c$ or $\boldsymbol{x}_j$ is missing." To account for splitting on missingness itself, we create dummy vectors of length n for each of the p attributes, denoted $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_p$, which assume the value 1 when the entry is missing and 0 when the entry is present. The original training matrix is then augmented with these dummies, giving the opportunity to select missingness *itself* when choosing a new splitting rule during the grow or change steps. Note that this can increase the number of predictors by up to a factor of 2. We illustrate building a **bartMachine** model with missing data in Section 4.9. As described in Chipman *et al.* (2010, Section 6), BART's runtime increases negligibly in the number of covariates and this has been our experience using the augmented training matrix.

Algorithm 1 The MIA choices for all attributes $j \in \{1, \ldots, p\}$ and all split points x^*_{ij} where $i \in \{1, \ldots, n\}$ during a GROW or CHANGE step in **bartMachine**.

```
1: If $x_{ij}$ is missing, send it $\longleftarrow$; if it is present and $x_{ij} \leq x^*_{ij}$, send it $\longleftarrow$, otherwise $\longrightarrow$.
2: If $x_{ij}$ is missing, send it $\longrightarrow$; if it is present and $x_{ij} \leq x^*_{ij}$, send it $\longleftarrow$, otherwise $\longrightarrow$.
3: If $x_{ij}$ is missing, send it $\longleftarrow$; if it is present, send it $\longrightarrow$.
```

3.3. Principled variable selection

Our package also implements the variable selection procedures developed in Bleich *et al.* (2013), which is best applied to data problems where the number of covariates influencing the response is small relative to the total number of covariates. We give a brief summary of the procedures here.

In order to select variables, we make use of the "variable inclusion proportions," the proportion of times each predictor is chosen as a splitting rule divided by the total number of splitting rules appearing in the model (see Section 4.6 for more details). The variable selection procedure can be outlined as follows:

1. Compute the model's variable inclusion proportions.

2. Permute the response vector, thereby breaking the relationship between the covariates and the response. Rebuild the model and compute the "null" variable inclusion proportions. Repeat this a number of times to create a null permutation distribution.

3. Three selection rules are can be used depending on the desired stringency of selection:

   (a) Local Threshold: Include a predictor $\boldsymbol{x}_k$ if its variable inclusion proportion exceeds the $1 - \alpha$ quantile of its own null distribution.

   (b) Global Max Threshold: Include a predictor $\boldsymbol{x}_k$ if its variable inclusion proportion exceeds the $1 - \alpha$ quantile of the distribution of the maximum of the null variable inclusion proportions from each permutation of the response.

   (c) Global SE Threshold: Select $\boldsymbol{x}_k$ if its variable inclusion proportion exceeds a threshold based from its own null distribution mean and SD with a global multiplier shared by all predictors.

The Local procedure is the least stringent in terms of selection and the Global Max procedure the most. The Global SE procedure is a compromise, but behaves more similarly to the Global Max. Bleich *et al.* (2013) demonstrate that the best procedure depends on the underlying sparsity of the problem, which is often unknown. Therefore, the authors include an additional procedure that chooses the best of these thresholds via cross-validation and this method is also implemented in **bartMachine**. Examples of these procedures for variable selection are provided in Section 4.10.

4. Regression Features

We illustrate the package features by using both real and simulated data, focusing first on regression problems.

4.1. Computing parameters

We first set some computing parameters. In this exploration, we allow up to 5GB of RAM for the Java heap (although we never used more than 1GB during this paper's exploration)[2] and we set the number of computing cores available for use to 4.

```
R> library("bartMachine")
R> set_bart_machine_memory(5000)
R> set_bart_machine_num_cores(4)
```

The following Sections 4.2 – 4.10 use a dataset obtained from UCI (Bache and Lichman 2013). The $n = 201$ observations are automobiles and the goal is to predict each automobile's price from 25 features (15 continuous and 10 nominal), first explored by Kibler, Aha, and Albert (1989).[3] This dataset also contains missing data.

4.2. Model building

We are now are ready to construct a **bartMachine** model. The default hyperparameters generally follow the recommendations of Chipman *et al.* (2010) and provide a ready-to-use algorithm for many data problems. Our hyperparameter settings are $m = 50$,[4] $\alpha = 0.95$, $\beta = 2$, $k = 2$, $q = 0.9$, $\nu = 3$, and probabilities of the GROW / PRUNE / CHANGE steps is 39% / 39% /44%. We set the number of burn-in Gibbs samples to 250 and number of post-burn-in samples to 1,000. We default the missing data feature to be off. We default the covariates to be equally important *a priori*. Other parameters and their defaults can be found in the package's online manual. Below is a default **bartMachine** model. Here, $\boldsymbol{X}$ denotes automobile attributes and $\boldsymbol{y}$ denotes the log price of the automobile.

```
R> bart_machine = bartMachine(X, y)
Building BART for regression ... evaluating in sample data...done
```

If one wishes to see the iterations of the Gibbs sampler of Equation 3, the flag `verbose` can be set to "TRUE". One can see more debug information from the Java program by setting the flag `debug_log` to true and the program will print to `unnamed.log` in the current working directory. We now inspect the model object to query its in-sample performance and to be reminded of the input data and model hyperparameters.

Since the response was considered continuous, we employ **bartMachine** for regression. The dimensions of the design matrix are given. Note that we dropped 45 observations that contained missing data (which we will retain in Section 4.9). We then have a recording of the MSE for

[2] Note that the maximum amount of memory can be set only *once* at the beginning of the R session (a limitation of **rJava** since only one Java Virtual Machine can be initiated per session), but the number of cores can be respecified at any time.

[3] We first preprocess the data. We first drop one of the nominal predictors (car company) due to too many categories (22). We then coerce two of the of the nominal predictors to be continuous. Further, the response variable, price, was logged to reduce right skew in its distribution.

[4] In contrast to Chipman *et al.* (2010), we recommend this default as a good starting point rather than $m = 200$ due to our experience experimenting with the "RMSE by number of trees" feature found in later in this section. Performance is often similar and computational time and memory requirements are dramatically reduced.

```
R> bart_machine
Bart Machine v1.0b for regression

training data n = 160 and p = 46
built in 1 secs on 4 cores, 50 trees, 250 burn in and 1000 post. samples

sigsq est for y beforehand: 0.014
avg sigsq estimate after burn-in: 0.00886

in-sample statistics:
 L1 = 9.03
 L2 = 0.8
 rmse = 0.07
 Pseudo-Rsq = 0.9741
p-val for shapiro-wilk test of normality of residuals: 0.01785
p-val for zero-mean noise: 0.97692
```

Figure 2: The summary for the default **bartMachine** model built with the automobile data

the OLS model and our average estimate of σ_e^2. We are then given in-sample statistics on error. Pseudo-R^2 is calculated via $1 - SSE/SST$. Also provided are outputs from tests of the error distribution being mean centered and normal. In this case, we cannot conclude normality of the residuals using the Shapiro-Wilk test.

We can also obtain out-of-sample statistics to assess level of overfitting by using k-fold cross validation. Using 10 folds we find:

```
R> k_fold_cv(X, y, k_folds = 10)
$L1_err    $L2_err           $rmse            $PseudoRsq
[1] 22.63155     [1] 5.202831      [1] 0.1803266    [1] 0.831917
```

The Pseudo-R^2 being lower out-of-sample versus in-sample suggests evidence that **bartMachine** is slightly overfitting (note also that the training sample during cross-validation is 10% smaller).

It may also be of interest how the number of trees m affects performance. One can examine how out-of-sample predictions vary by the number of trees via

```
R> rmse_by_num_trees(bart_machine, num_replicates = 20)
```

and the output is shown in Figure 3.

It seems that increasing $m > 50$ does not result in any substantial increase in performance. We can now try to build a better **bartMachine** by grid-searching over a set of hyperparameter combinations, including m (for more details, see BART-cv in Chipman *et al.* 2010). The default grid search is small and it can be customized by the user.

```
R> bart_machine_cv = bartMachineCV(X, y)
...
BART CV win: k: 2 nu, q: 10, 0.75 m: 200
```

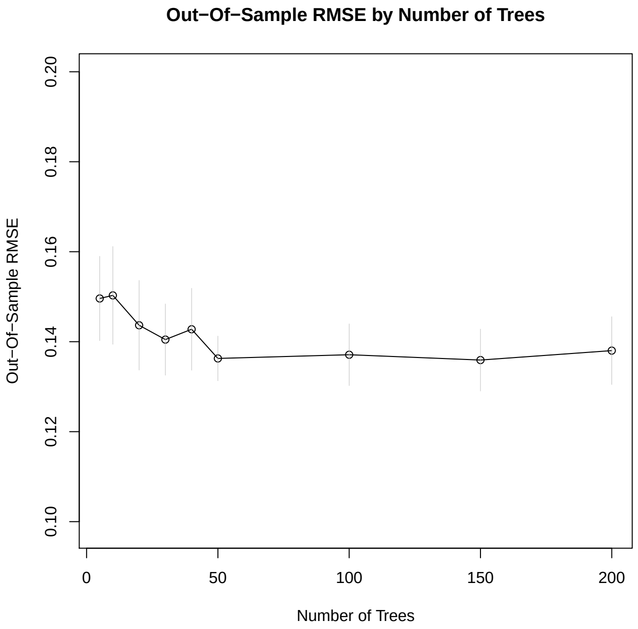


Figure 3: Out-of-sample predictive performance by number of trees

This function returns the "winning" model, which is the one with lowest out-of-sample RMSE over a 5-fold cross-validation. Here, the cross-validated **bartMachine** model has slightly better in-sample performance (L1 = 8.18, L2 = 0.68 and Pseudo-R^2 = 0.978) as well as slightly better out-of-sample performance:

```
R> k_fold_cv(X, y, k_folds = 10, k = 2, nu = 3, q = 0.9, num_trees = 200)
$L1_err          $L2_err          $rmse               $PseudoRsq
[1] 21.21557     [1] 4.517916     [1] 0.1680386       [1] 0.8540439
```

Predictions are handled with the `predict` function:

```
predict(bart_machine_cv, X[1 : 14, ])
   9.479963  9.775766  9.799110 10.050041  9.659138  9.697902  9.873622
   9.931972  8.550436  8.688874  8.794351  8.647986  8.690300  9.029066
```

We also include a convenience method `bart_predict_for_test_data` that will predict and return out-of-sample error metrics when the test outcomes are known.

4.3. Model destroying

As noted in the introduction to Section 3, **bartMachine** objects persist in Java for the entirety of an R session. These model objects can use a substantial amount of RAM and it is prudent to release this memory when it is no longer needed. Simply removing a **bartMachine** in R does *not* destroy the Java object and release the RAM (resulting in a memory leak).

Therefore, we provide a utility function `destroy_bart_machine` that cleans up the Java object. This function should be called when a **bartMachine** object is no longer needed and *before* removing or overwriting the R variable. Since we no longer are using the original `bart_machine` object, we now release its memory via:

```
destroy_bart_machine(bart_machine)
```

4.4. Assumption checking

The package includes features that assess the plausibility of the BART model assumptions. Checking the mean-centeredness of the noise is addressed in the summary output of Figure 2 and is simply a one-sample t-test of the average residual value against a null hypothesis of true mean zero. We assess both normality and heteroskedasticity via:

```
R> check_bart_error_assumptions(bart_machine_cv)
```

This will display a window similar to Figure 4 which contains a QQ-plot (to assess normality) as well as a residual-by-predicted plot (to assess homoskedasticity). It appears that the errors are most likely normal and homoskedastic.

In addition to the model assumptions, BART requires convergence of its Gibbs sampler. Figure 5 displays four types of convergence diagnostics.

4.5. Credible intervals and prediction intervals

An advantage of BART is that if we believe the priors and model assumptions, the Bayesian probability model and corresponding burned-in Gibbs samples provide the approximate posterior distribution of $f(\boldsymbol{x})$. Thus, one can compute uncertainty estimates via quantiles of the posterior samples. These provide Bayesian "credible intervals" which are intervals for the conditional expectation function, $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{X}]$.

Another useful uncertainty interval can be computed for individual predictions by combining uncertainty from the conditional expectation function with the systematic, homoskedastic normal noise produceed by $\mathcal{E}$. Since we have draws from the posterior of the conditional expectation distribution and concomitant draws from the posterior of the variance distribution, we can simulate the distribution of the response by first drawing a pair $\left(f(\boldsymbol{x})_n, \sigma_n^2\right)$ via a draw from the burned-in Gibbs chain. Then, a posterior predictive sample $y^\star$ is drawn from $\mathcal{N}\left(f(\boldsymbol{x})_n,\, \sigma_n^2\right)$. This process is repeated many times (our default is 1000). The prediction interval is then provided by the appropriate quantiles of the posterior predictive samples.

Below is an example of how both types of intervals are computed in the package (for the 100th observation of the training data):

```
R> calc_credible_intervals(bart_machine_cv, X[100, ], ci_conf = 0.95)
     ci_lower_bd ci_upper_bd
[1,]    8.725202    8.971687
R> calc_prediction_intervals(bart_machine_cv, X[100, ], pi_conf = 0.95)
     pi_lower_bd pi_upper_bd
[1,]    8.631243     9.06353
```

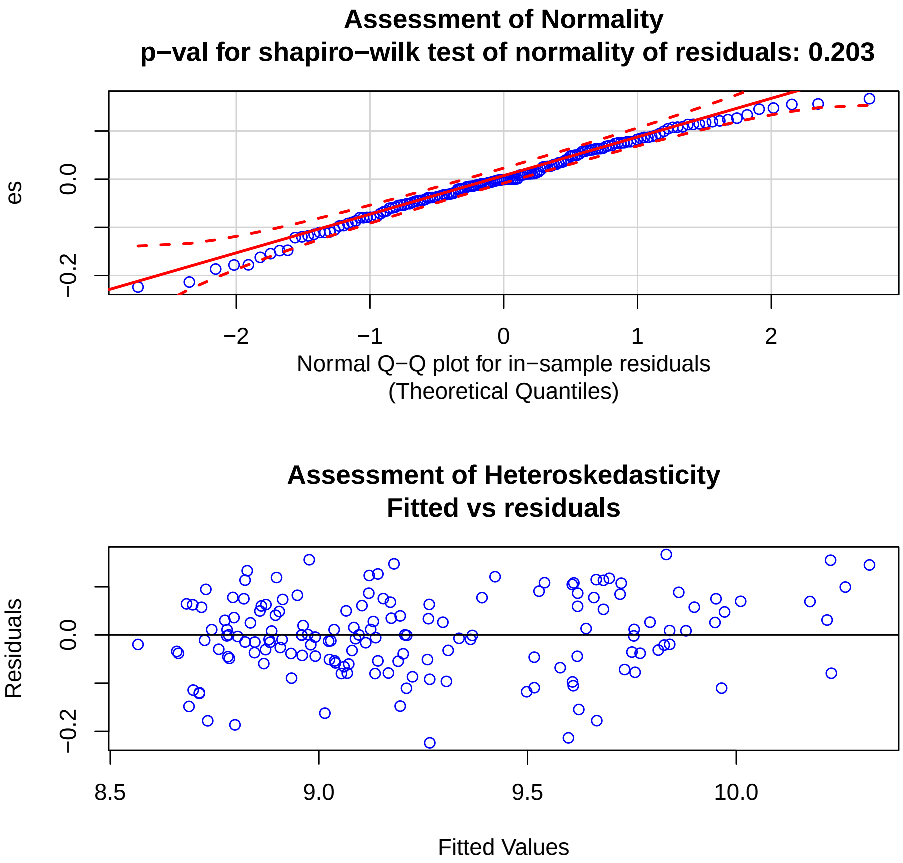


Figure 4: Test of normality of errors using QQ-plot and the Shapiro-Wilk test (top), residual plot to assess heteroskedasticity (bottom).

Note that the prediction intervals are wider than the credible intervals because they reflect the uncertainty from the error term.

We can then plot these intervals in sample:

```
R> plot_y_vs_yhat(bart_machine_cv, credible_intervals = TRUE)
R> plot_y_vs_yhat(bart_machine_cv, prediction_intervals = TRUE)
```

Figure 6a shows how our prediction fared against the original response (in-sample) with 95% credible intervals. Figure 6b shows the same prediction versus the original response plot now with 95% prediction intervals.

4.6. Variable importance

After a **bartMachine** model is built, it is natural to ask the question: which variables are most important? This is assessed by examining the splitting rules in the m trees across the post burn-in gibbs samples which are known as "inclusion proportions" (Chipman *et al.* 2010). The inclusion proportion for any given predictor represents the proportion of times that variable is

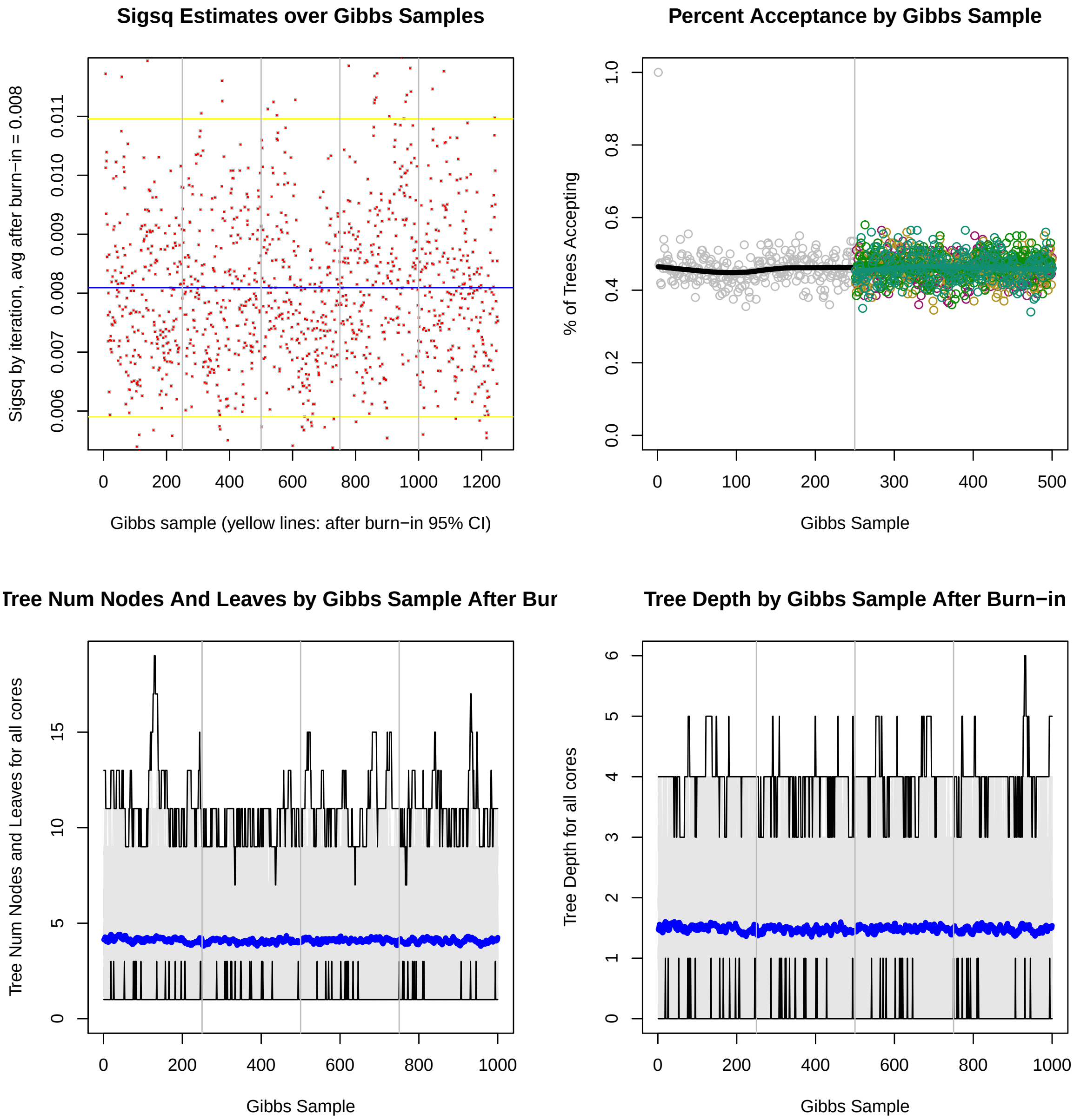
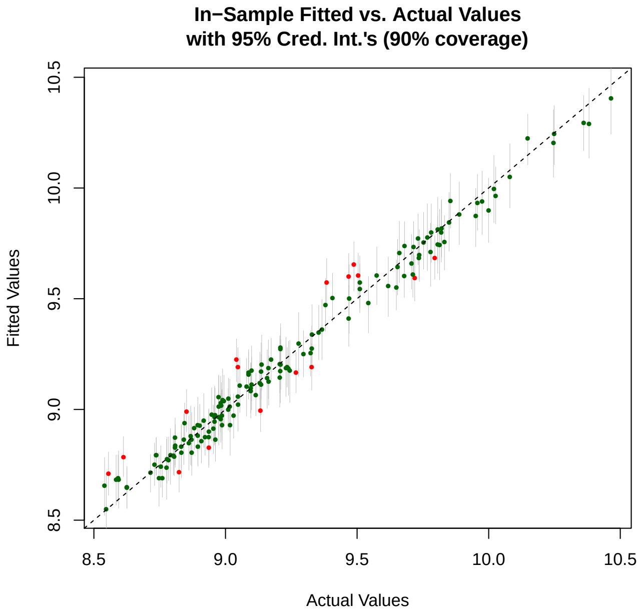
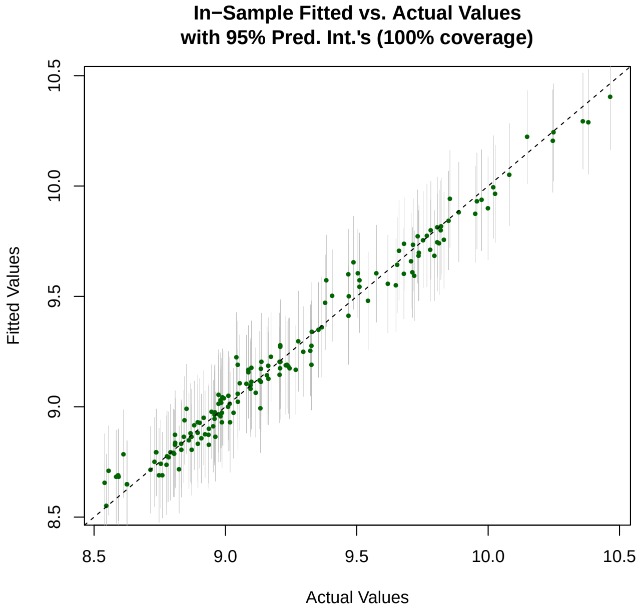


Figure 5: Convergence diagnostics for the cross-validated **bartMachine** model. Top left: σ^2 by Gibbs sample. Samples to the left of the first vertical grey line are burn-in from the first computing core's Gibbs sample chain. The four subsequent plots separated by grey lines are the post-burn-in samples from each of the four computing cores employed during model construction. Top right: percent acceptance of Metropolis-Hastings proposals across the m trees where each point plots one iteration. Points before the grey vertical line illustrate burn-in samples and points after illustrate post burn-in. Each computing core is colored differently. Bottom left: average number of leaves across the m trees by sample (post burn-in only where computing cores separated by vertical grey lines). Bottom right: average tree depth across the m trees by sample (post burn-in only where computing cores separated by vertical grey lines).



(a) Segments illlustrate credible intervals



(b) Segments illlustrate prediction intervals

Figure 6: Fitted versus actual response values for the automobile dataset. Segments are 95% credible intervals (a) or 95% prediction intervals (b). Green dots indicate the true response is within the stated interval and red dots indicate otherwise. Note that the percent coverage in (a) is not expected to be 95% because the response includes a noise term.

chosen as a spliting rule out of all splitting rules among the posterior draws of the sum-of-trees model. Figure 7 illustrates the inclusion proportions for all variables obtained via:

```
R> investigate_var_importance(bart_machine_cv, num_replicates_for_avg = 20)
```

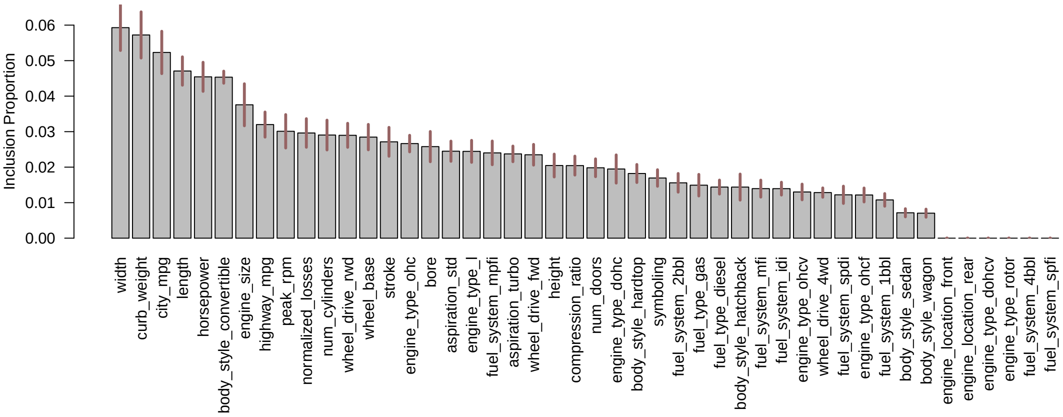


Figure 7: Average variable inclusion proportions in the cross-validated **bartMachine** model for the automobile data averaged over 100 model constructions to obtain stable estimates across many posterior modes in the sum-of-trees distribution (as recommended in Bleich *et al.* 2013). The segments atop the bars represent 95% confidence intervals. The eight predictors with inclusion proportions of zero feature identically one value (after missing data was dropped).

Actual selection of variables *significantly* affecting the response is addressed conceptually in Section 3.3 and examples are provided in Section 4.10.

4.7. Variable effects

It is also natural to ask: does $\boldsymbol{x}_j$ affect the response, controlling for other variables in the model? This is roughly analogous to the t-test in ordinary least squares regression of no linear effect of $\boldsymbol{x}_j$ on $\boldsymbol{y}$ while controlling for $\boldsymbol{x}_{-j}$. The null hypothesis here is the same but the linearity constraint is relaxed. To test this, we employ a permutation approach where we record the observed Pseudo-R^2 from the **bartMachine** model built with the original data. Then we permute the $\boldsymbol{x}_j$th column, thereby destroying any relationship between $\boldsymbol{x}_j$ and $\boldsymbol{y}$, construct a new duplicate **bartMachine** model from this permuted design matrix and record a "null" Pseudo-R^2. We then repeat this process to obtain a null distribution of Pseudo-R^2's. Since the alternative hypothesis is that $\boldsymbol{x}_j$ has an effect on $\boldsymbol{y}$ in terms of predictive power, our p value is the proportion of null Pseudo-R^2's greater than the observed Pseudo-R^2, making our procedure a natural one-sided test. Note, however, that this test is conditional on the BART model and its selected priors being true, similar to the assumptions of the linear model.

If we wish to test if a set of covariates $A \subset \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p\}$ affect the response after controlling for other variables, we repeat the procedure outlined in the above paragraph by permuting the columns of A in every null sample. We do not permute each column separately, but instead permute as a unit in order to perserve collinearity structure. This is roughly analogous to the partial F-test in ordinary least squares regression.

If we wish to test if *any* of the covariates matter in predicting $\boldsymbol{y}$, we simply permute $\boldsymbol{y}$ during the null sampling. This procedure breaks the relationship between the response and the predictors but does not alter the existing associations between predictors. This is roughly analogous to the omnibus F-test in ordinary least squares regression.

At $\alpha = 0.05$, Figure 8a demonstrates an insignificant effect of the variable `width` of car on price. Even though `width` is putatively the "most important" variable as measured by proportions of splits in the posterior sum-of-trees model (Figure 7), note that this is largely an easy prediction problem with many collinear predictors. Figure 8b shows the results of a test of the putatively most important categorical variable, `body style` (which involves permuting the categories, then dummifying the levels to preserve the structure of the variable). We find a marginally significant effect ($p = 0.0495$). A test of the top ten most important variables is convincingly significant (Figure 8c). For the omnibus test, Figure 8d illustrates an extremely statistically significant result, as would be expected. The code to run these tests is shown below (output suppressed).

```
R> cov_importance_test(bart_machine_cv, covariates = c("width"))
R> cov_importance_test(bart_machine_cv, covariates = c("body_style"))
R> cov_importance_test(bart_machine_cv, covariates = c("width",
  "curb_weight", "city_mpg", "length", "horsepower", "body_style",
  "engine_size", "highway_mpg", "peak_rpm", "normalized_losses"))
R> cov_importance_test(bart_machine_cv)
```

4.8. Partial dependence

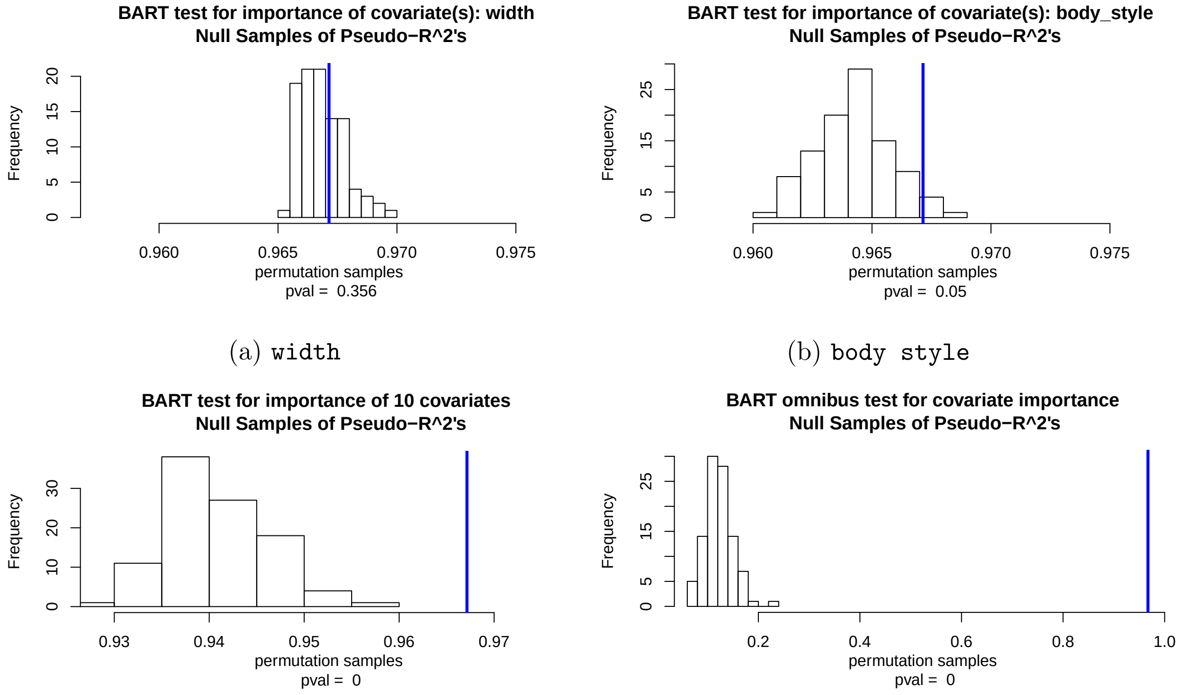
A data analyst may also be interested in understanding how $\boldsymbol{x}_j$ affects the response on average, after controlling for other predictors. This can be examined using Friedman (2001)'s Partial Dependence Function (PDP),

$$f_j(\boldsymbol{x}_j) = \mathbb{E}_{\boldsymbol{x}_{-j}}\left[f(\boldsymbol{x}_j, \boldsymbol{x}_{-j})\right] = \int f(\boldsymbol{x}_j, \boldsymbol{x}_{-j}) \mathrm{dP}\left(\boldsymbol{x}_{-j}\right), \tag{4}$$

where $\boldsymbol{x}_{-j}$ denotes all variables except $\boldsymbol{x}_j$. The PDP of predictor $\boldsymbol{x}_j$ gives the average value of f when $\boldsymbol{x}_j$ is fixed and $\boldsymbol{x}_{-j}$ varies over its marginal distribution, $\mathrm{dP}\left(\boldsymbol{x}_{-j}\right)$. As neither the true model f nor the distribution of the predictors $\mathrm{dP}\left(\boldsymbol{x}_{-j}\right)$ are known, we estimate Equation 4 by computing

$$\hat{f}_j(\boldsymbol{x}_j) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(\boldsymbol{x}_j, \boldsymbol{x}_{-j,i}) \tag{5}$$

where n is the number of observations in the training data and $\hat{f}$ denotes the **bartMachine** model. Since BART provides an estimated posterior distribution, we can plot credible bands for the PDP function. In Equation 5, the $\hat{f}$ can be replaced with a function that calculates the qth quantile of the post-burned-in Gibbs samples for $\hat{y}$. Figure 9a plots the PDP along with the 2.5%ile and the 97.5%ile for the variable `horsepower`. By varying over most of the



(a) `width`

(b) `body style`

(c) The 10 most split-on variables

(d) All covariates

Figure 8: Tests of covariate importance conditional on the cross-validated **bartMachine** model. All tests performed with 100 null samples.

range of `horsepower`, the price is predicted to increase by about $1000. Figure 9b plots the PDP along with the 2.5%ile and the 97.5%ile for the variable `stroke`. This predictor seemed to be relatively unimportant according to Figure 7 and the PDP confirms this, with a very small, yet nonlinear average partial effect. The code for both plots is below.

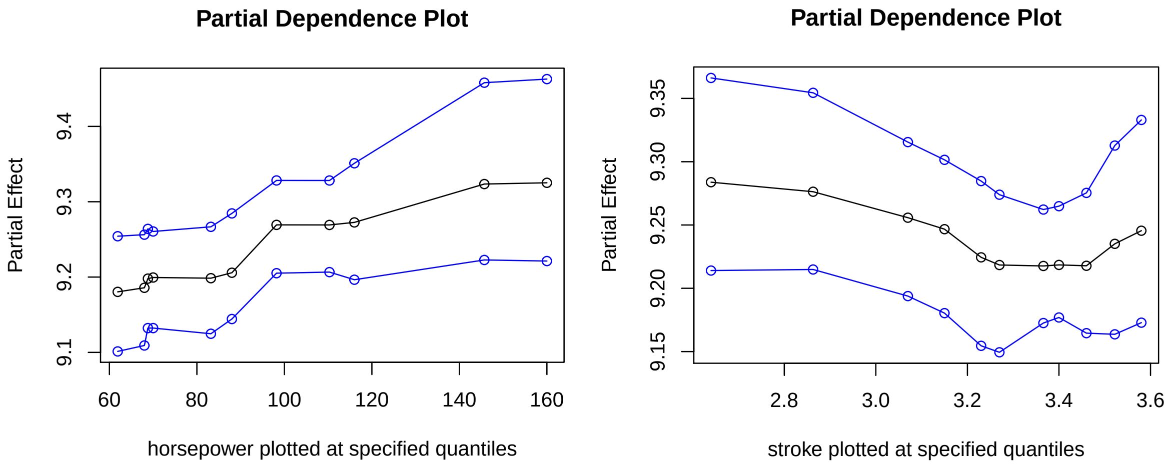
```
R> pd_plot(bart_machine_cv, j = "horsepower")
R> pd_plot(bart_machine_cv, j = "stroke")
```

4.9. Incorporating missing data

The procedure for incorporating missing data was introduced in Section 3.2. We now build a **bartMachine** model using this procedure below:

```
R> bart_machine = bartMachine(X, y, use_missing_data = TRUE,
     use_missing_data_dummies_as_covars = TRUE)
R> bart_machine
Bart Machine v1.0b for regression

Missing data feature ON
training data n = 201 and p = 50
built in 1.3 secs on 1 core, 50 trees, 250 burn-in and 1000 post. samples
```



(a) `horsepower`

(b) `stroke`

Figure 9: PDPs plotted in black and 95% credible intervals plotted in blue for variables in the automobile dataset. Points plotted are at the 5%ile, 10%ile, 20%ile, . . . , 90%ile and 95%ile of the values of the predictor. Lines plotted between the points approximate the PDP by linear interpolation.

```
sigsq est for y beforehand: 0.016
avg sigsq estimate after burn-in: 0.01055

in-sample statistics:
 L1 = 12.77
 L2 = 1.28
 rmse = 0.08
 Pseudo-Rsq = 0.9746
p-val for shapiro-wilk test of normality of residuals: 0.6638
p-val for zero-mean noise: 0.95693
```

Note that we now use the complete data set including the 41 observations for which there were missing features. Also note that p has now increased from 46 to 51. The five new predictors are dummy variables which indicate missingness constructed from the predictors which exhibited missingness (due to the `use_missing_data_dummies_as_covars` parameter being set to true). These variables are important if missingness *itself* shifts the response, as was the case in models explored in Kapelner and Bleich (2013). One way to test if this type of missingness is important is to run a covariate test (Section 4.7) on these new 5 covariates:

```
> cov_importance_test(bart_machine, covariates = c("M_normalized_losses",
     "M_bore", "M_stroke", "M_horsepower", "M_peak_rpm"))
BART test for importance of 5 covariates....p_val = 0.673
```

From the p value, we cannot conclude that these variables have an effect on the response.

This seems reasonable given that only 5 predictors featured missingness among only 40 observations. This suggests that it may be appropriate to shrink the model by leaving out these missing dummies, but still allow for missingness to be incorporated into the split rules:

```
R> bart_machine = bartMachine(X, y, use_missing_data = TRUE)
```

The procedure of Section 3.2 also natively incorporates missing data during prediction. Missingness will yield larger credible intervals. In the example below, we suppose that the `curb_weight` and `symboling` values were unavailable for 20th automobile.

```
R> x_star = X[20, ]
R> calc_credible_intervals(bart_machine, x_star, ci_conf = 0.95)
     ci_lower_bd ci_upper_bd
[1,]    8.650093    8.824515
R> x_star[c("curb_weight", "symboling")] = NA
R> calc_credible_intervals(bart_machine, x_star, ci_conf = 0.95)
     ci_lower_bd ci_upper_bd
[1,]    8.622582    8.978313
```

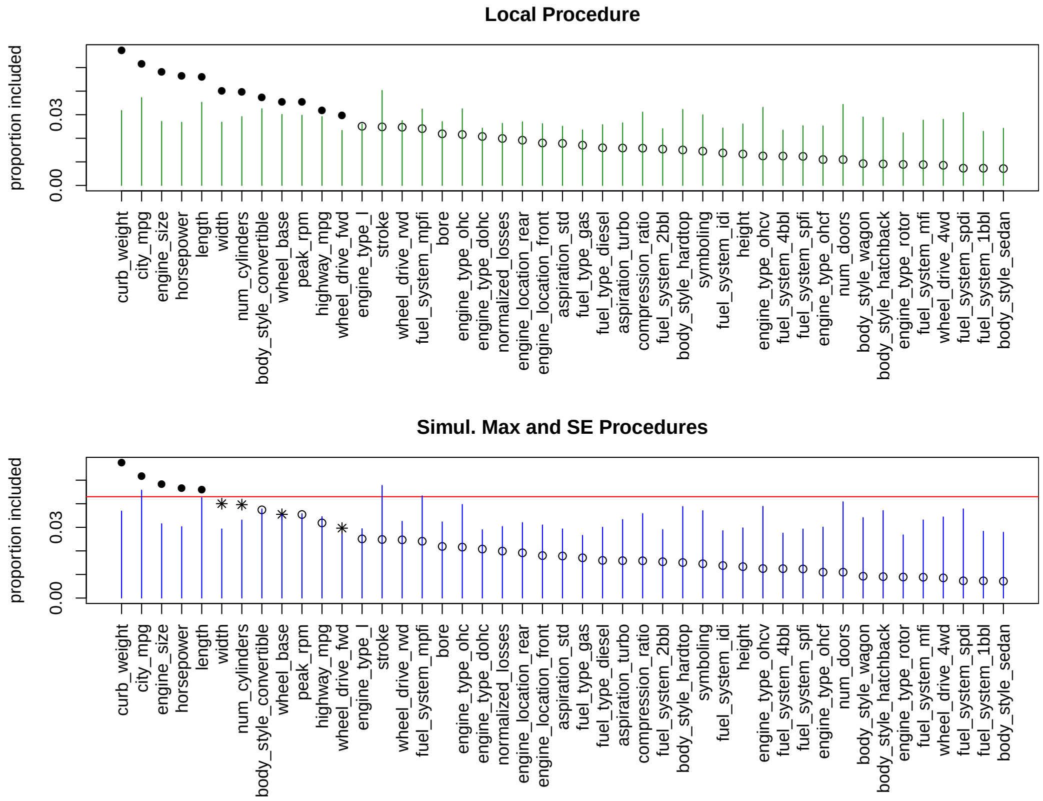
4.10. Variable selection

In this section we demonstrate the principled variable selection procedure introduced in Section 3.3. The following code will select variables based on the three thresholds and also displays the plot in Figure 10.[5]

```
R> vs = var_selection_by_permute_response_three_methods(bart_machine,
            bottom_margin = 10, num_permute_samples = 10)
R> vs$important_vars_local_names
  "curb_weight"  "city_mpg" "engine_size"   "horsepower"
  "length"       "width"    "num_cylinders" "body_style_convertible"
  "wheel_base"   "peak_rpm" "highway_mpg"   "wheel_drive_fwd"
R> vs$important_vars_global_max_names
  "curb_weight" "city_mpg"    "engine_size" "horsepower"  "length"
R> vs$important_vars_global_se_names
  "curb_weight"   "city_mpg"      "engine_size" "horsepower" "length"
  "width"         "num_cylinders" "wheel_base"  "wheel_drive_fwd"
```

Usually, "Global Max" and "Global SE" perform similarly, as they are both more stringent in selection. However, in many situations it will not be clear to the data analyst which threshold is most appropriate. The "best" procedure can be chosen via cross-validation on out-of-sample RMSE as follows:

[5] By default, variable selection is performed individually on dummy variables for a factor. The variable selection procedures return the permutation distribution and an aggregation of the dummy variables' inclusion proportions can allow for variable selection to be performed on an entire factor.


Local Procedure

Simul. Max and SE Procedures

Figure 10: Visualization of the three variable selection procedures outlined in Section 3.3 with $\alpha = 0.05$. The top plot illustrates the “Local” procedure. The green lines are the threshold levels determined from the permutation distributions that must be exceeded for a variable to be selected. The plotted points are the variable inclusion proportions for the observed data (averaged over five duplicate **bartMachine** models). If the observed value is higher than the green bar, the variable is included and is displayed as a solid dot; if not, it is not included and it is displayed as an open dot. The bottom plot illustrates both the “Global SE” and “Global Max” thresholds. The red line is the cutoff for “Global Max” and variables pass this threshold are displayed as solid dots. The blue lines represent the thresholds for the “Global SE” procedure. Variables that exceed this cutoff but not the “Global Max” threshold are displayed as asterisks. Open dots exceed neither threshold.

```
var_selection_by_permute_response_cv(bart_machine)
$best_method
[1] "important_vars_local_names"

$important_vars_cv
 [1] "body_style_convertible" "city_mpg"               "curb_weight"
 [4] "engine_size"            "engine_type_ohc"        "horsepower"
 [7] "length"                 "num_cylinders"          "peak_rpm"
[10] "wheel_base"             "wheel_drive_fwd"        "wheel_drive_rwd"
[13] "width"
```

On this dataset, the "best" approach (as defined by out-of-sample prediction error) is the "Local" procedure.

The following Sections 4.11 and 4.12 demostrate additional features using Friedman (1991)'s function:

$$\boldsymbol{y} = 10 \sin \pi \boldsymbol{x}_1 \boldsymbol{x}_2 + 20(\boldsymbol{x}_3 - .5)^2 + 10\boldsymbol{x}_4 + 5\boldsymbol{x}_5 + \boldsymbol{\mathcal{E}}, \qquad \boldsymbol{\mathcal{E}} \sim \mathcal{N}_n\left(\boldsymbol{0},\, \sigma^2 \boldsymbol{I}\right). \tag{6}$$

4.11. Informed prior information on covariates

Bleich *et al.* (2013) propose a method for incorporating informed prior information about the predictors into the BART model. This can be achieved by modifying the prior on the splitting rules as well as the corresponding calculations in the Metropolis-Hastings step. In particular, covariates believed to influence the response can be proposed more often as candidates for splitting rules. Useful prior information can aid in both variable selection and prediction tasks. We illustrate the impact of a correctly informed prior in the context of the Friedman function (Equation 6). We include the 5 predictors which influence the response as well as 95 that do not.

All that is required is a specification of relative weights for each predictor. These are converted internally to probabilities. We assign 5 times the weight to the 5 true covariates of the model relative to the 95 useless covariates.

```
R> prior = c(rep(5, times = 5), rep(1, times = 95))
```

We now sample 500 observations from the Friedman function and construct a default **bartMachine** model as well as a **bartMachine** model with the informed prior and compare their performance on a test set of another 500 observations.

```
R> bart_machine = bartMachine(X, y)
R> bart_machine_informed = bartMachine(X, y, cov_prior_vec = prior)

R> bart_predict_for_test_data(bart_machine, Xtest, ytest)$rmse
[1] 1.661159
R> bart_predict_for_test_data(bart_machine_informed, Xtest, ytest)$rmse
[1] 1.232925
```

There is a substantial improvement in out-of-sample predictive performance when a properly informed prior is used.

Note that by default the prior vector down-weights the indicator variables that result from dummifying factors so that the total set of dummy variables has the same weight as a continuous covariate.

4.12. Interaction effect detection

In Section 4.6, we explored using variable inclusion proportions to understand the relative influences of different covariates. A similar procedure can be carried out for examining interaction effects within a BART model. This question was initially explored in Damien, Dellaportas, Polson, and Stephens (2013) where an interaction was considered to exist between two variables if they both appeared in at least one splitting rule in a given tree. We refine the definition of an interaction as follows.

We first begin with a $p \times p$ matrix of zeroes. Within a given tree, for each split rule variable j, we look at all split rule variables of child nodes, k, and we increment the j, k element of the matrix. Hence variables are considered to interact in a given tree *only if* they appear together in a contiguous downward path from the root node to a terminal node. Note that a variable may interact with itself (when fitting a linear effect, for instance). Since there is no order between the parent and child, we then add the j, k counts together with the k, j counts (if $j \neq k$). Summing across trees and Gibbs samples gives the total number of interactions for each pair of variables from which relative importance can be assessed.

We demonstrate interaction detection on the Friedman function using 10 covariates using the code below:

```
R> interaction_investigator(bart_machine, num_replicates_for_avg = 25,
     num_var_plot = 10, bottom_margin = 5)
```

Shown in Figure 11 are the ten most important interactions in the model. The illustration is averaged over many model constructions to obtain stable estimates across many posterior modes in the sum-of-trees distribution. Notice that the interaction between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ dominates all other terms, as **bartMachine** is correctly capturing the single true interaction effect in Equation 6. Choosing which of these interactions *significantly* affect the response is not addressed in this paper. The methods suggested in Section 3.3 may be applicable here and we consider this to be fruitful future work.

5. Classification Features

In this section we highlight the features that differ from the regression case when the response is dichotomous. The illustrative dataset consists of 332 Pima Indians obtained from the UCI repository. Of the 332 subjects, 109 were diagnosed with diabetes, the binary response variable. There are seven continuous predictors which are body metrics such as blood pressure, glucose concentration, etc. and there is no missing data.

Building a **bartMachine** model for classification has the same computing parameters except that q, ν cannot be specified since there is no longer a prior on σ^2 (see Section 2.3). We first build a cross-validated model below.

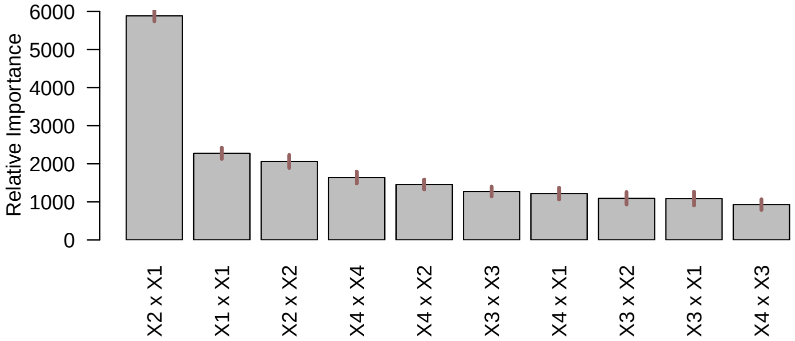


Figure 11: The top 10 average variable interaction counts (termed "relative importance") in the default **bartMachine** model for the Friedman function data averaged over 25 model constructions. The segments atop the bars represent 95% confidence intervals.

```
R> bart_machine_cv = bartMachineCV(X, y)
... BART CV win: k: 3 m: 50
R> bart_machine_cv
Bart Machine v1.0b for classification

training data n = 332 and p = 7
built in 0.5 secs on 4 cores, 50 trees, 250 burn-in and 1000 post. samples

confusion matrix:

           predicted No predicted Yes model errors
actual No       211.000         12.00        0.054
actual Yes       41.000         68.00        0.376
use errors        0.163          0.15        0.160
```

Classification models have an added hyperparameter, `prob_rule_class`, which is the rule for determining if the probability estimate is great enough to be classified into the positive category. We can see above that the **bartMachine** at times predicts "NO" for true "YES" outcomes and we suffer from a 37.6% error rate for this outcome. We can try to mitigate this error by lowering the threshold to increase the number of "YES" labels predicted:

```
R> bartMachine(X, y, prob_rule_class = 0.3)
Bart Machine v1.0b for classification

training data n = 332 and p = 7
built in 0.5 secs on 4 cores, 50 trees, 250 burn-in and 1000 post. samples

confusion matrix:

           predicted No predicted Yes model errors
```

```
actual No       178.000       45.000        0.202
actual Yes       12.000       97.000        0.110
use errors        0.063        0.317        0.172
```

This lowers the model error to 11% for the “YES” class, but at the expense of increasing the error rate for the “NO” class. We encourage the user to cross-validate this rule based on the appropriate objective function for the problem at hand.

We can also check out-of-sample statistics:

```
R> oos_stats = k_fold_cv(X, y, k_folds = 10)
R> oos_stats$confusion_matrix
           predicted No predicted Yes model errors
actual No       203.000       20.000        0.090
actual Yes       47.000       62.000        0.431
use errors        0.188        0.244        0.202
```

Note that it is possible to predict both class labels and probability estimates for given observations:

```
R> predict(bart_machine_cv, X[1 : 2, ], type = "prob")
[1] 0.6253160 0.1055975
R> predict(bart_machine_cv, X[1 : 2, ], type = "class")
[1] "Yes" "No"
```

When using the covariate tests of Section 4.7,total misclassification error becomes the statistic of interest instead of Pseudo-R^2. The p value is calculated now as the proportion of null samples with *lower* misclassification error. Figure 12 illustrates the test showing that predictor `age` seems to matter in the prediction of `Diabetes`, controlling for other predictors.

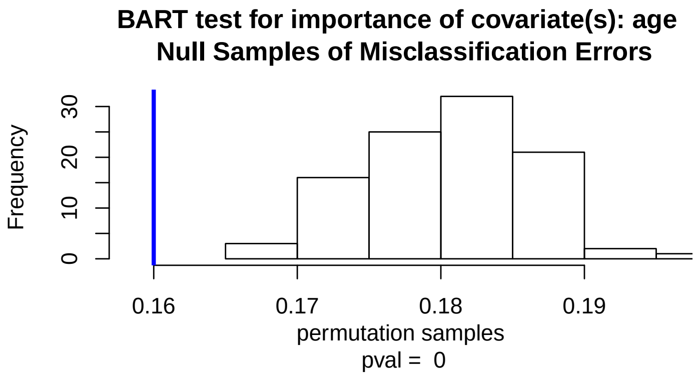


Figure 12: Test of covariate importance for predictor `age` on whether or not the subject will contract `Diabetes`.

The partial dependence plots of Section 4.8 are now scaled as probit of the probability estimate. Figure 13 illustrates that as glucose increases, the probability of contracting `Diabetes` increases linearly on a probit scale.

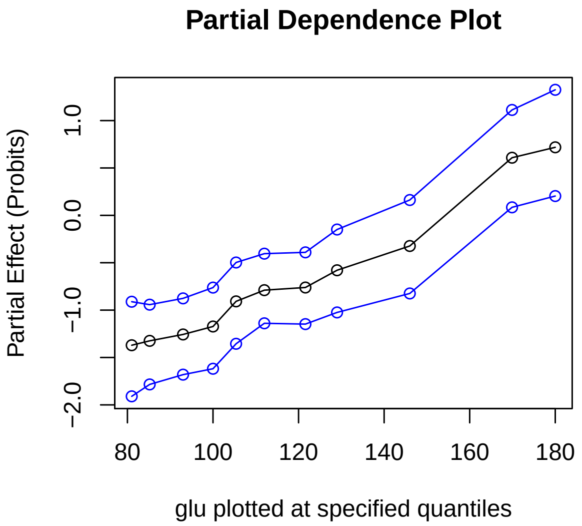


Figure 13: PDP for predictor `glu`. The blue lines are 95% credible intervals.

Credible intervals are implemented for classification **bartMachine** and are displayed on the probit scale. Note that the prediction intervals of Section 4.5 do not exist for classification.

```
R> calc_credible_intervals(bart_machine_cv, X[1 : 2, ])
     ci_lower_bd ci_upper_bd
[1,]  0.34865355   0.8406097
[2,]  0.01686486   0.2673171
```

Other functions work similarly to regression except those that plot the responses and those that explicitly depend on RMSE as an error metric.

6. Discussion

This article introduced **bartMachine**, a new R package which implements Bayesian additive regression trees. The goal of this package is to provide a fast, extensive and user-friendly implementation accessible to a wide range of data analysts, and increase the visibility of BART to a broader statistical audience. We hope we have provided organized, well-documented open-source code and we encourage the community to make innovations on this package.

Replication

The code for **bartMachine** is located at `http://github.com/kapelner/bartMachine` under the GPL3 and MIT licenses. Results, tables, and figures found in this paper can be replicated via the scripts located in the `bart_package_paper` folder within the `git` repository.

Acknowledgements

We thank Richard Berk, Andreas Buja, Zachary Cohen, Ed George, Alex Goldstein, Shane

Jensen, Abba Krieger, and Robert DeRubeis for helpful discussions. We thank Simon Urbanek for his generous help with **rJava**. Adam Kapelner acknowledges support from the National Science Foundation's Graduate Research Fellowship Program.

References

Albert J, Chib S (1993). "Bayesian Analysis of Binary and Polychotomous Response Data." *Journal of the American Statistical Association*, **88**(422), 669–679.

Bache K, Lichman M (2013). "UCI Machine Learning Repository." URL http://archive.ics.uci.edu/ml.

Blattenberger G, Fowles R (2014). "Avalanche Forecasting: Using Bayesian Additive Regression Trees (BART)." In *Demand for Communications Services–Insights and Perspectives*, pp. 211–227. Springer-Verlag.

Bleich J, Kapelner A, Jensen S, George E (2013). "Variable Selection Inference for Bayesian Additive Regression Trees." *ArXiv e-prints*.

Breiman L (2001). "Statistical Modeling: The Two Cultures." *Statistical Science*, **16**(3), 199–231.

Chipman H, George E, McCulloch R (2010). "BART: Bayesian Additive Regressive Trees." *The Annals of Applied Statistics*, **4**(1), 266–298.

Chipman H, McCulloch R (2010). *BayesTree: Bayesian Methods for Tree Based Models*. R package version 0.3-1.1, URL http://CRAN.R-project.org/package=BayesTree.

Damien P, Dellaportas P, Polson N, Stephens D (2013). *Bayesian Theory and Applications*, pp. 455–464. First edition. Oxford University Press.

Ding Y, Simonoff J (2010). "An Investigation of Missing Data Methods for Classification Trees Applied to Binary Response Data." *The Journal of Machine Learning Research*, **11**, 131–170.

Eliashberg J (2010). *Green-lighting Movie Scripts: Revenue Forecasting and Risk Management*. Ph.D. thesis, University of Pennsylvania.

Friedman J (1991). "Multivariate Adaptive Regression Splines." *The annals of statistics*, **19**, 1–67.

Friedman J (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics*, **29**(5), 1189–1232.

Friedman J (2002). "Stochastic Gradient Boosting." *Computational Statistics & Data Analysis*, **38**(4), 367–378.

Gelman A, Carlin J, Stern H, Rubin D (2004). *Bayesian Data Analysis*. Second edition. Chapman & Hall / CRC.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **6**, 721–741.

Hastie T, Tibshirani R (2000). "Bayesian Backfitting." *Statistical Science*, **15**(3), 196–213.

Hastings W (1970). "Monte Carlo Sampling Methods using Markov Chains and their Applications." *Biometrika*, **57**(1), 97–109.

Kapelner A, Bleich J (2013). "Prediction with Missing Data via Bayesian Additive Regression Trees." *ArXiv e-prints*.

Kibler D, Aha D, Albert M (1989). "Instance Based Prediction of Real Valued Attributes." *Computational Intelligence*, **5**, 51.

Kindo B, Wang H, Pe E (2013). "MBACT - Multiclass Bayesian Additive Classification Trees." pp. 1–29. arXiv:1309.7821v1.

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." *R News*, **2**(3), 18–22.

Pratola M, Chipman H, Higdon D, McCulloch R, Rust W (2013). "Parallel Bayesian Additive Regression Trees." *Technical report*, University of Chicago.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Taddy M, Gramacy R, Polson N (2011). "Dynamic Trees for Learning and Design." *Journal of the American Statistical Association*, **106**(493), 109–123.

Twala B, Jones M, Hand D (2008). "Good Methods for Coping with Missing Data in Decision Trees." *Pattern Recognition Letters*, **29**(7), 950–956.

Urbanek S (2013). *rJava: Low-level R to Java interface*. R package version 0.9-6, URL http://CRAN.R-project.org/package=rJava.

Zhou Q, Liu JS (2008). "Extracting Sequence Features to Predict Protein–DNA Interactions: a Comparative Study." *Nucleic acids research*, **36**(12), 4137–4148.

A. Sampling new trees

This section provides details on the implementation of Equation 3 (steps 1, 3, ..., $2m-1$), the Metropolis-Hastings step for sampling new trees. Recall from Section 2.2 that trees can be altered via growing new daughter nodes from an existing terminal node, pruning two terminal nodes such that their parent becomes terminal, or changing the splitting rule in a node.

Below is the Metropolis ratio (Gelman *et al.* 2004, p.291) where the parameter sampled is the tree and the data is the responses unexplained by other trees denoted by $\boldsymbol{R}$. We denote the new, proposal tree with an asterisk and the original tree without the asterisk.

$$r = \frac{\mathbb{P}\left(\text{🌲}_* \to \text{🌲}\right)}{\mathbb{P}\left(\text{🌲} \to \text{🌲}_*\right)} \frac{\mathbb{P}\left(\text{🌲}_* \mid \boldsymbol{R}, \sigma^2\right)}{\mathbb{P}\left(\text{🌲} \mid \boldsymbol{R}, \sigma^2\right)} \tag{7}$$

We accept a draw from the posterior distribution of trees if a draw from a standard uniform distribution is less than the value of r. Immediately we note that it is difficult (if not impossible) to calculate the posterior probabilities for the trees themselves. Instead, we employ Bayes' Rule,

$$\mathbb{P}\left(\text{🌲} \mid \boldsymbol{R}, \sigma^2\right) = \frac{\mathbb{P}\left(\boldsymbol{R} \mid \text{🌲}, \sigma^2\right)\mathbb{P}\left(\text{🌲} \mid \sigma^2\right)}{\mathbb{P}\left(\boldsymbol{R} \mid \sigma^2\right)},$$

and plug the result into Equation 7 to obtain:

$$r \quad = \quad \underbrace{\frac{\mathbb{P}\left(\text{🌲}_* \to \text{🌲}\right)}{\mathbb{P}\left(\text{🌲} \to \text{🌲}_*\right)}}_{\text{transition ratio}} \quad \times \quad \underbrace{\frac{\mathbb{P}\left(\boldsymbol{R} \mid \text{🌲}_*, \sigma^2\right)}{\mathbb{P}\left(\boldsymbol{R} \mid \text{🌲}, \sigma^2\right)}}_{\text{likelihood ratio}} \quad \times \quad \underbrace{\frac{\mathbb{P}\left(\text{🌲}_*\right)}{\mathbb{P}\left(\text{🌲}\right)}}_{\text{tree structure ratio}} \quad .$$

Note that the probability of the tree structure is independent of σ^2.

The goal of this section is to explicitly calculate r for all possible tree proposals — GROW, PRUNE and CHANGE. For each proposal, the calculations are organized into separate sections detailing each of the three ratios — transition, likelihood and tree structure. Note that our actual implementation uses the following expressions in log form for numerical accuracy.

A.1. Grow proposal

Transition ratio

Transitioning from the original tree to a new tree involves growing two daughter nodes from a current terminal node:

$$\begin{aligned}
\mathbb{P}\left(\text{🌲} \to \text{🌲}_*\right) \quad &= \quad \mathbb{P}\left(\text{GROW}\right)\mathbb{P}\left(\text{selecting } \eta \text{ to grow from}\right) \times \\
&\qquad \mathbb{P}\left(\text{selecting the } j\text{th attribute to split on}\right) \times \\
&\qquad \mathbb{P}\left(\text{selecting the } i\text{th value to split on}\right) \\
&= \quad \mathbb{P}\left(\text{GROW}\right)\frac{1}{b}\frac{1}{p_{\text{adj}}(\eta)}\frac{1}{n_{j\cdot\text{adj}}(\eta)}.
\end{aligned} \tag{8}$$

We chose one of the current b terminal nodes which we denote the ηth node, and then we pick an attribute and split point. $p_{\text{adj}}(\eta)$ denotes the number of predictors left available to split on. This can be less than p if certain predictors do not have two or more unique values once the data reaches the ηth node. For example, this regularly occurs if a dummy variable was split on in some node higher up in the lineage. $n_{j\cdot\text{adj}}(\eta)$ denotes the number of *unique* values left in the pth attribute after adjusting for parents' splits.

Transitioning from the new tree back to the original tree involves pruning that node:

$$\mathbb{P}\left(\text{🌲}_* \to \text{🌲}\right) = \mathbb{P}\left(\text{PRUNE}\right)\mathbb{P}\left(\text{selecting } \eta \text{ to prune from}\right) = \mathbb{P}\left(\text{PRUNE}\right)\frac{1}{w_2^*}$$

where w_2^* denotes the number of second generation internal nodes (nodes with two terminal daughter nodes) in the new tree. Thus, the full transition ratio is:

$$\frac{\mathbb{P}\left(\text{🌲}_* \to \text{🌲}\right)}{\mathbb{P}\left(\text{🌲} \to \text{🌲}_*\right)} = \frac{\mathbb{P}\left(\text{PRUNE}\right)}{\mathbb{P}\left(\text{GROW}\right)}\frac{b\; p_{\text{adj}}(\eta)\; n_{j\cdot\text{adj}}(\eta)}{w_2^*}.$$

Note that when there are no variables with more two or more unique values, the probability of GROW is set to zero and the step will be automatically rejected.

Likelihood ratio

To calculate the likelihood, the tree structure determines which responses fall into which of the b terminal nodes. Thus,

$$\mathbb{P}\left(R_1, \ldots, R_n \mid \text{🌲}, \sigma^2\right) = \prod_{\ell=1}^{b} \mathbb{P}\left(R_{\ell_1}, \ldots, R_{\ell_{n_\ell}} \mid \sigma^2\right)$$

where each term on the right hand side is the probability of responses in one of the b terminal nodes, which are independent by assumption. The R_ℓ's denote the data in the ℓth terminal node and where n_ℓ denotes how many observations are in each terminal node and $n = \sum_{\ell=1}^{b} n_\ell$.

We now find an analytic expression for the node likelihood term. Remember, if the mean in each terminal node, which we denote μ_ℓ, was known, then we would have $R_{\ell_1}, \ldots, R_{\ell_{n_\ell}} \mid \mu_\ell, \sigma^2 \overset{iid}{\sim} \mathcal{N}\left(\mu_\ell, \sigma^2\right)$. BART requires μ_ℓ to be margined out, allowing the Gibbs sampler in Equation 3 to avoid dealing with jumping between continuous spaces of varying dimensions (Chipman *et al.* 2010, page 275). Recall that one of the BART model assumptions is a prior on the average value of $\mu \sim \mathcal{N}\left(0, \sigma_\mu^2\right)$ and thus,

$$\mathbb{P}\left(R_{\ell_1}, \ldots, R_{\ell_{n_\ell}} \mid \sigma^2\right) = \int_{\mathbb{R}} \mathbb{P}\left(R_{\ell_1}, \ldots, R_{\ell_{n_\ell}} \mid \mu_\ell, \sigma^2\right)\mathbb{P}\left(\mu_\ell; \sigma_\mu^2\right) d\mu_\ell$$

which can be shown via completion of the square or convolution to be

$$\begin{aligned}\mathbb{P}\left(R_{\ell_1}, \ldots, R_{\ell_{n_\ell}} \mid \sigma^2\right) \quad = \quad & \frac{1}{\left(2\pi\sigma^2\right)^{n_\ell/2}}\sqrt{\frac{\sigma^2}{\sigma^2 + n_\ell\sigma_\mu^2}} \times \\ & \exp\left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n_\ell}\left(R_{\ell_i} - \bar{R}_\ell\right)^2 - \frac{\bar{R}_\ell^2 n_\ell^2}{n_\ell + \frac{\sigma^2}{\sigma_\mu^2}} + n_\ell\bar{R}_\ell^2\right)\right)\end{aligned} \tag{9}$$

where $\bar{R}_\ell$ denotes the mean response in the node and R_{ℓ_i} denotes the observations $i = 1 \ldots n_\ell$ in the node.

Since the likelihoods are solely determined by the terminal nodes, the proposal tree differs from the original tree by only the selected node to be grown, denoted by ℓ, which becomes two daughters after the GROW step denoted by ℓ_L and ℓ_R. Hence, the likelihood ratio becomes:

$$\frac{\mathbb{P}\left(\boldsymbol{R} \mid \mathcal{T}_*, \sigma^2\right)}{\mathbb{P}\left(\boldsymbol{R} \mid \mathcal{T}, \sigma^2\right)} = \frac{\mathbb{P}\left(R_{\ell_{L,1}}, \ldots, R_{\ell_{L,n_{\ell,L}}} \mid \sigma^2\right) \mathbb{P}\left(R_{\ell_{R,1}}, \ldots, R_{R,\ell_{n_{\ell,R}}} \mid \sigma^2\right)}{\mathbb{P}\left(R_{\ell_1}, \ldots, R_{\ell_{n_\ell}} \mid \sigma^2\right)} \tag{10}$$

Plugging Equation 9 into Equation 10 three times yields the ratio for the GROW step:

$$\sqrt{\frac{\sigma^2\left(\sigma^2 + n_\ell \sigma_\mu^2\right)}{\left(\sigma^2 + n_{\ell_L}\sigma_\mu^2\right)\left(\sigma^2 + n_{\ell_R}\sigma_\mu^2\right)}} \exp\left(\frac{\sigma_\mu^2}{2\sigma^2}\left(\frac{\left(\sum_{i=1}^{n_{\ell_L}} R_{\ell_L,i}\right)^2}{\sigma^2 + n_{\ell_L}\sigma_\mu^2} + \frac{\left(\sum_{i=1}^{n_{\ell_R}} R_{\ell_R,i}\right)^2}{\sigma^2 + n_{\ell_R}\sigma_\mu^2} - \frac{\left(\sum_{i=1}^{n_\ell} R_{\ell,i}\right)^2}{\sigma^2 + n_\ell \sigma_\mu^2}\right)\right)$$

where n_{ℓ_L} and n_{ℓ_R} denote the number of data points in the newly grown left and right daughter nodes.

Tree structure ratio

In Section 2.1 we discussed the prior on the tree structure (where the splits occur) as well as the tree rules. For the entire tree,

$$\mathbb{P}\left(\mathcal{T}\right) = \prod_{\eta \in H_{\text{terminals}}} \left(1 - \mathbb{P}_{\text{SPLIT}}\left(\eta\right)\right) \prod_{\eta \in H_{\text{internals}}} \mathbb{P}_{\text{SPLIT}}\left(\eta\right) \prod_{\eta \in H_{\text{internals}}} \mathbb{P}_{\text{RULE}}\left(\eta\right)$$

where $H_{\text{terminals}}$ denotes the set of terminal nodes and $H_{\text{internals}}$ denotes the internal nodes.

Recall that the probability of splitting on a given node η is $\mathbb{P}_{\text{SPLIT}}\left(\eta\right) = \alpha / \left(1 + d_\eta\right)^\beta$. The probability is controlled by two hyperparameters, α and β, and d_η is the depth (number of parent generations) of node η. When assigning a rule, recall that BART picks from all available attributes and then from all available unique split points. Using the notation from the transition ratio section, $\mathbb{P}_{\text{RULE}}\left(\eta\right) = 1/p_{\text{adj}}(\eta) \times 1/n_{j \cdot \text{adj}}(\eta)$.

Once again, the original tree features a node η that was selected to be grown. The proposal tree differs with two daughter nodes denoted η_L and η_R. We can now form the ratio:

$$\begin{aligned}
\frac{\mathbb{P}\left(\mathcal{T}_*\right)}{\mathbb{P}\left(\mathcal{T}\right)} &= \frac{\left(1-\mathbb{P}_{\text{SPLIT}}\left(\eta_L\right)\right)\left(1-\mathbb{P}_{\text{SPLIT}}\left(\eta_R\right)\right)\mathbb{P}_{\text{SPLIT}}\left(\eta\right)\mathbb{P}_{\text{RULE}}\left(\eta\right)}{\left(1-\mathbb{P}_{\text{SPLIT}}\left(\eta\right)\right)} \\
&= \frac{\left(1-\frac{\alpha}{\left(1+d_{\eta_L}\right)^{\beta}}\right)\left(1-\frac{\alpha}{\left(1+d_{\eta_R}\right)^{\beta}}\right)\frac{\alpha}{\left(1+d_{\eta}\right)^{\beta}}\frac{1}{p_{\text{adj}}(\eta)}\frac{1}{n_{j\cdot\text{adj}}(\eta)}}{1-\frac{\alpha}{\left(1+d_{\eta}\right)^{\beta}}} \\
&= \alpha\frac{\left(1-\frac{\alpha}{\left(2+d_{\eta}\right)^{\beta}}\right)^2}{\left(\left(1+d_{\eta}\right)^{\beta}-\alpha\right)p_{\text{adj}}(\eta)n_{j\cdot\text{adj}}(\eta)}
\end{aligned}$$

The last line follows from algebra and using the fact that the depth of the grown nodes is the depth of the parent node incremented by one ($d_{\eta_L} = d_{\eta_R} = d_\eta + 1$).

A.2. Prune proposal

A prune proposal is the "opposite" of a grow proposal. Prune selects a node with two daughters and removes them. Thus, each ratio will be approximately the inverse of the ratios found in the previous section concerning the grow proposal. Note also that prune steps are not considered in trees that consist of a single root node.

Transition ratio

We begin with transitioning from the original tree to the proposal tree:

$$\mathbb{P}\left(\mathcal{T}\rightarrow\mathcal{T}_*\right) = \mathbb{P}\left(\text{PRUNE}\right)\mathbb{P}\left(\text{selecting } \eta \text{ to prune from}\right) = \mathbb{P}\left(\text{PRUNE}\right)\frac{1}{w_2}$$

where w_2 denotes the number of parent nodes that have two daughters but no grand-daughters. To transition in the opposite direction, we are obligated to grow from node η. This is similar to Equation 8 except the proposed tree has one less terminal node due to the pruning of the original tree, resulting in a $1/(b-1)$ term:

$$\mathbb{P}\left(\mathcal{T}_*\rightarrow\mathcal{T}\right) = \mathbb{P}\left(\text{GROW}\right)\frac{1}{b-1}\frac{1}{p_{\text{adj}}(\eta^*)}\frac{1}{n_{j^*\cdot\text{adj}}(\eta^*)}.$$

Thus, the transition ratio is:

$$\frac{\mathbb{P}\left(\mathcal{T}_*\rightarrow\mathcal{T}\right)}{\mathbb{P}\left(\mathcal{T}\rightarrow\mathcal{T}_*\right)} = \frac{\mathbb{P}\left(\text{GROW}\right)}{\mathbb{P}\left(\text{PRUNE}\right)}\frac{w_2}{(b-1)p_{\text{adj}}(\eta^*)n_{j^*\cdot\text{adj}}(\eta^*)}.$$

Likelihood ratio

This is simply the inverse of the likelihood ratio for the grow proposal:

$$\begin{aligned}\frac{\mathbb{P}\left(\boldsymbol{R} \mid \text{🌲}_*, \sigma^2\right)}{\mathbb{P}\left(\boldsymbol{R} \mid \text{🌲}, \sigma^2\right)} &= \sqrt{\frac{\left(\sigma^2 + n_{\ell_L}\sigma_\mu^2\right)\left(\sigma^2 + n_{\ell_R}\sigma_\mu^2\right)}{\sigma^2\left(\sigma^2 + n_\ell \sigma_\mu^2\right)}} \times \\ &\exp\left(\frac{\sigma_\mu^2}{2\sigma^2}\left(\frac{\left(\sum_{i=1}^{n_\ell} R_{\ell,i}\right)^2}{\sigma^2 + n_\ell\sigma_\mu^2} - \frac{\left(\sum_{i=1}^{n_{\ell_L}} R_{\ell_L,i}\right)^2}{\sigma^2 + n_{\ell_L}\sigma_\mu^2} - \frac{\left(\sum_{i=1}^{n_{\ell_R}} R_{\ell_R,i}\right)^2}{\sigma^2 + n_{\ell_R}\sigma_\mu^2}\right)\right).\end{aligned}$$

Tree structure ratio

This is also simply the inverse of the tree structure ratio for the grow proposal:

$$\frac{\mathbb{P}\left(\text{🌲}_*\right)}{\mathbb{P}\left(\text{🌲}\right)} = \frac{\left((1+d_\eta)^\beta - \alpha\right) p_{\text{adj}}(\eta^*) n_{j^*\cdot\text{adj}}(\eta^*)}{\alpha\left(1 - \frac{\alpha}{(2+d_\eta)^\beta}\right)^2}.$$

A.3. Change proposal

A change proposal involves picking an internal node and changing its rule by picking both a new available predictor to split on and a new valid split value among values of the selected predictor. Although this could be implemented for use in any internal node in the tree, for simplicity we limit our implementation to *singly* internal nodes: those that have two terminal daughter nodes and thus, no grand-daughters.

Transition ratio

The transition to a proposal tree is below:

$$\begin{aligned}\mathbb{P}\left(\text{🌲} \to \text{🌲}_*\right) &= \mathbb{P}\left(\text{CHANGE}\right)\mathbb{P}\left(\text{selecting node } \eta \text{ to change}\right) \times \\ &\quad \mathbb{P}\left(\text{selecting the new attribute to split on}\right) \times \\ &\quad \mathbb{P}\left(\text{selecting the new value to split on}\right)\end{aligned}$$

When calculating the ratio, the first three terms are shared in both numerator and denominator. The probability of selecting the new value to split on will differ as different split features have different numbers of unique values available. Thus we are left with

$$\frac{\mathbb{P}\left(\text{🌲}_* \to \text{🌲}\right)}{\mathbb{P}\left(\text{🌲} \to \text{🌲}_*\right)} = \frac{n_{j^*\cdot\text{adj}}(\eta^*)}{n_{j\cdot\text{adj}}(\eta)}$$

where $n_{j^*\cdot\text{adj}}(\eta^*)$ is the number of split values available under the proposal tree's splitting rule and $n_{j\cdot\text{adj}}(\eta)$ is the number of split values available under the original tree's splitting rule.

Likelihood ratio

The proposal tree differs from the original tree only in the two daughter nodes of the selected change node. These two terminal nodes have the unexplained responses apportioned differently. Denote $R_{1\cdot}$ as the residuals of the first daughter node and $R_{2\cdot}$ as the unexplained responses in the second daughter node. Thus we begin with:

$$\frac{\mathbb{P}\left(\boldsymbol{R} \mid \text{🌲}_*, \sigma^2\right)}{\mathbb{P}\left(\boldsymbol{R} \mid \text{🌲}, \sigma^2\right)} \quad = \quad \frac{\mathbb{P}\left(R_{1^*,1}, \ldots, R_{1^*,n_{1^*}} \mid \sigma^2\right) \mathbb{P}\left(R_{2^*,1}, \ldots, R_{2^*,n_{2^*}} \mid \sigma^2\right)}{\mathbb{P}\left(R_{1,1}, \ldots, R_{1,n_1} \mid \sigma^2\right) \mathbb{P}\left(R_{2,1}, \ldots, R_{2,n_2} \mid \sigma^2\right)}$$

where the responses denoted with an asterisk are the responses in the proposal tree's daughter nodes.

Substituting Equation 9 four times and using algebra, the following expression is obtained for the ratio:

$$\sqrt{\frac{\left(\frac{\sigma^2}{\sigma_\mu^2} + n_1\right)\left(\frac{\sigma^2}{\sigma_\mu^2} + n_2\right)}{\left(\frac{\sigma^2}{\sigma_\mu^2} + n_1^*\right)\left(\frac{\sigma^2}{\sigma_\mu^2} + n_2^*\right)}} \times$$
$$\exp\left(\frac{1}{2\sigma^2}\left(\frac{\left(\sum_{i=1}^{n_{1^*}} R_{1^*,i}\right)^2}{n_{1^*} + \frac{\sigma^2}{\sigma_\mu^2}} + \frac{\left(\sum_{i=1}^{n_{2^*}} R_{2^*,i}\right)^2}{n_{2^*} + \frac{\sigma^2}{\sigma_\mu^2}} - \frac{\left(\sum_{i=1}^{n_1} R_{1,i}\right)^2}{n_1 + \frac{\sigma^2}{\sigma_\mu^2}} - \frac{\left(\sum_{i=1}^{n_2} R_{2,i}\right)^2}{n_2 + \frac{\sigma^2}{\sigma_\mu^2}}\right)\right)$$

which simplifies to

$$\exp\left(\frac{1}{2\sigma^2}\left(\frac{\left(\sum_{i=1}^{n_{1^*}} R_{1^*,i}\right)^2 - \left(\sum_{i=1}^{n_{1^*}} R_{1,i}\right)^2}{n_1 + \frac{\sigma^2}{\sigma_\mu^2}} + \frac{\left(\sum_{i=1}^{n_{1^*}} R_{2^*,i}\right)^2 - \left(\sum_{i=1}^{n_{1^*}} R_{2,i}\right)^2}{n_2 + \frac{\sigma^2}{\sigma_\mu^2}}\right)\right)$$

if the number of responses in the children do not change in the proposal ($n_1 = n_1^*$ and $n_2 = n_2^*$).

Tree structure ratio

The proposal tree has the same structure as the original tree. Thus we only need to take into account the changed node's daughters:

$$\frac{\mathbb{P}\left(\text{🌲}_*\right)}{\mathbb{P}\left(\text{🌲}\right)} = \frac{\left(1 - \mathbb{P}_{\text{SPLIT}}\left(\eta_{1^*}\right)\right)\left(1 - \mathbb{P}_{\text{SPLIT}}\left(\eta_{2^*}\right)\right)\mathbb{P}_{\text{SPLIT}}\left(\eta_*\right)\mathbb{P}_{\text{RULE}}\left(\eta_*\right)}{\left(1 - \mathbb{P}_{\text{SPLIT}}\left(\eta_1\right)\left(1 - \mathbb{P}_{\text{SPLIT}}\left(\eta_2\right)\right)\right)\mathbb{P}_{\text{SPLIT}}\left(\eta\right)\mathbb{P}_{\text{RULE}}\left(\eta\right)}.$$

The probability of splits remain the same because the daughter nodes are at the same depths. Thus we only need to consider the ratio of the probability of the rules. Once again, the probability of selecting the new value to split on will differ as different split features have different numbers of unique values available. We are left with

$$\frac{\mathbb{P}\left(\text{🌳}_*\right)}{\mathbb{P}\left(\text{🌳}\right)} = \frac{n_{j\cdot\text{adj}}(\eta)}{n_{j^*\cdot\text{adj}}(\eta^*)}.$$

Note that this is the inverse of the transition ratio. Hence, for the change step, only the likelihood ratio needs to be computed to determine the Metropolis-Hastings ratio r.

B. Bakeoff

We baked off nine regression data sets and assessed out-of-sample RMSE using 10-fold cross-validation. The results are displayed in Table 1.

Dataset Name	**bartMachine**	**BayesTree**	**randomForest**
boston	4.66	4.59	4.58
triazine	0.04*	0.05	0.04
ozone	48.02	47.75	48.77
baseball	0.02	0.02*	0.02
wine.red	0.73	0.69*	0.76
ankara	2.62	2.61	3.46
wine.white	0.49	0.49	0.54
pole	4.00	3.65	11.56
compactiv	46.17*	47.92	40.74

Table 1: Average of 20 replicates of out-of-sample RMSE values on 9 datasets using three machine learning algorithms. Asterisks indicate a significant difference between **bartMachine** and **BayesTree** at a significance level of 5% with a Bonferroni correction. Comparisons with **randomForest**'s performance were not conducted.

We conclude that the implementation outlined in this paper performs approximately the same as the previous implementation with regards to predictive accuracy.

Affiliation:

Adam Kapelner, Justin Bleich
Department of Statistics
The Wharton School of the University of Pennsylvania
3730 Walnut Street
Philadelphia, PA, 19104
E-mail: {kapelner, jbleich}@wharton.upenn.edu
URL: http://scholar.google.com/citations?user=TzgMmnoAAAAJ
URL: http://scholar.google.com/citations?user=JbpcbD8AAAAJ