BART: Bayesian Additive Regression Trees

Robert McCulloch

School of Mathematical and Statistical Sciences
Arizona State University

September 29, 2016

Joint with
Hugh Chipman (Acadia University)
Ed George (University of Pennsylvania)

We want to "fit" the fundamental model:

$$Y_i = f(X_i) + \epsilon_i$$

BART is a Markov Monte Carlo Method that draws from

$$f \mid (x, y)$$

We can then use the draws as our inference for f.

To get the draws, we will have to:

- Put a prior on f.
- Specify a Markov chain whose stationary distribution is the posterior of f.

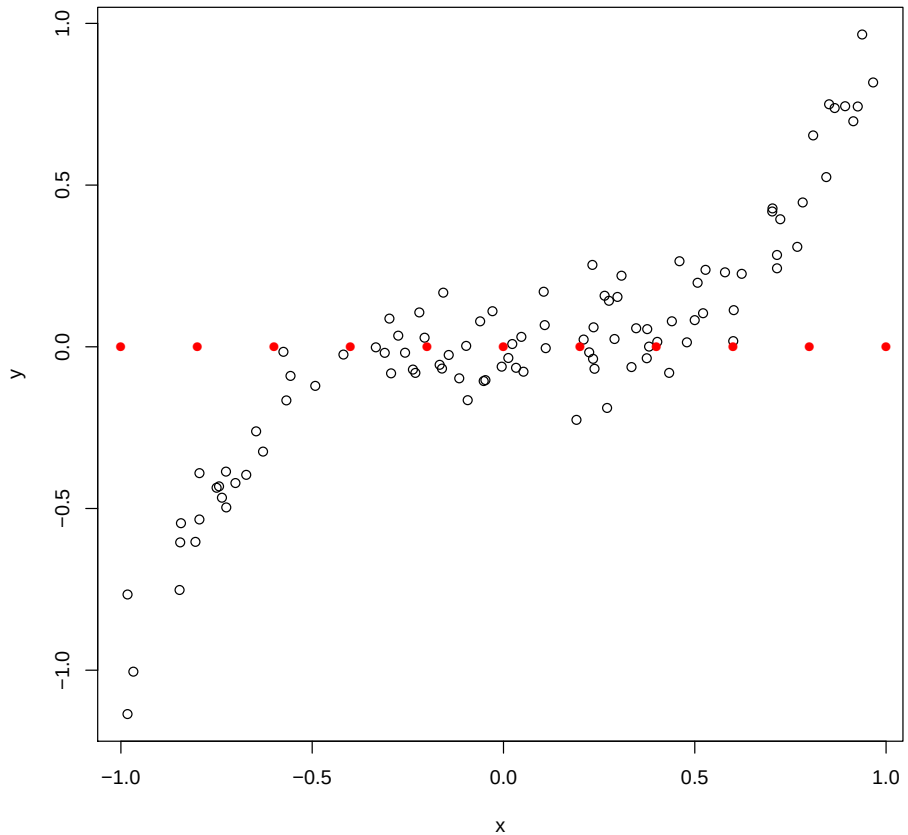
Simulate data from the model:

$$Y_i = x_i^3 + \epsilon_i \quad \epsilon_i \sim N(0, \sigma^2) \ \ iid$$

```
-----------------------------------------------
n = 100
sigma = .1
f = function(x) {x^3}
set.seed(14)
x = sort(2*runif(n)-1)
y = f(x) + sigma*rnorm(n)
xtest = seq(-1,1,by=.2)
-----------------------------------------------
```

Here, *xtest* will be the *out of sample* x values at which we wish to infer f or make predictions.

```
-------------------------------------------------
plot(x,y)
points(xtest,rep(0,length(xtest)),col='red',pch=16)
-------------------------------------------------
```



Red is xtest.

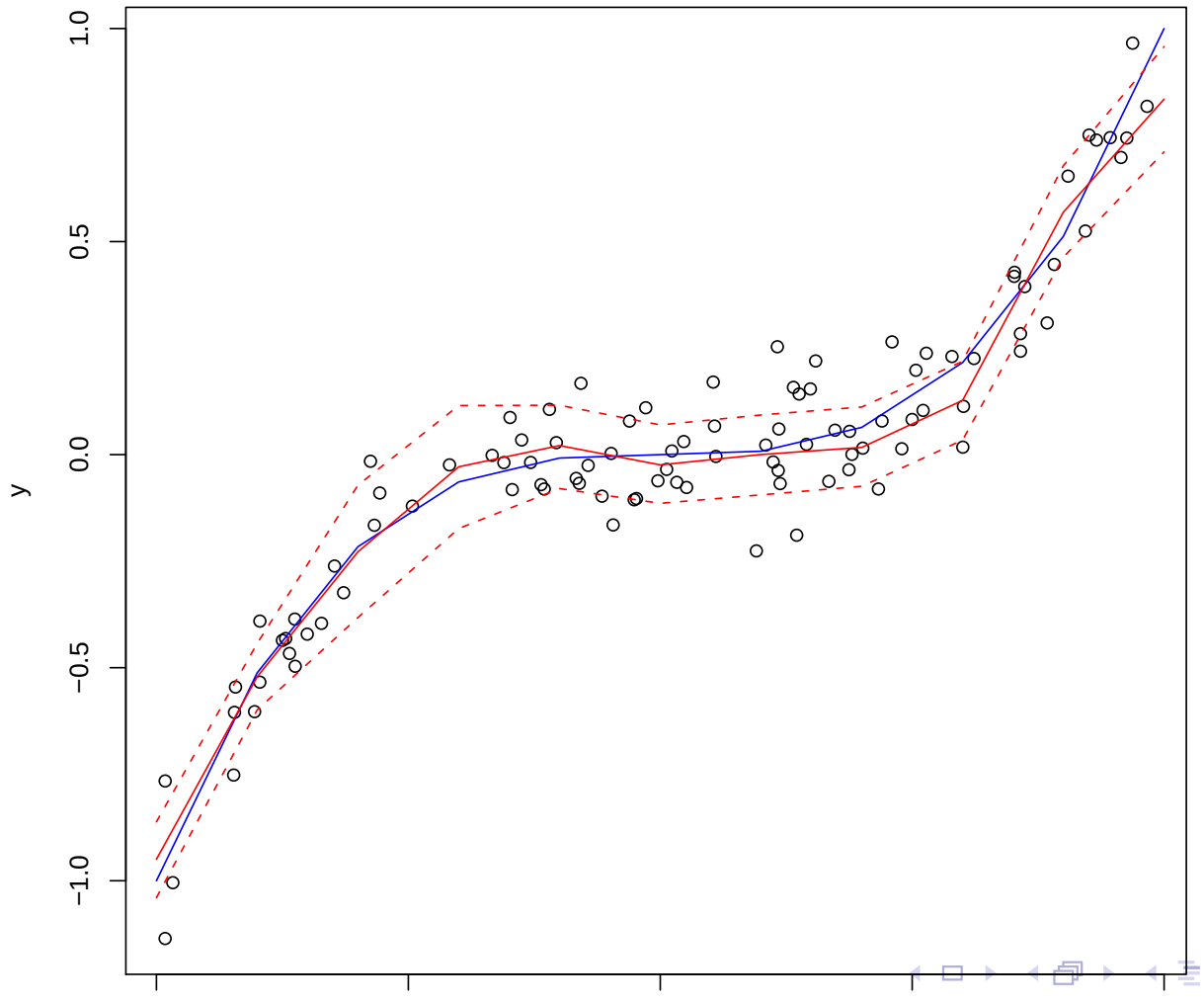
```
-------------------------------------------------
library(BayesTree)
rb = bart(x,y,xtest)
length(xtest)
[1] 11
dim(rb$yhat.test)
[1] 1000   11
-------------------------------------------------
```

The (i, j) element of yhat.test is

the i^{th} draw of f evaluated at the j^{th} value of xtest.

1,000 draws of f, each of which is evaluated at 11 xtest values.

```
-------------------------------------------------
plot(x,y)
lines(xtest,xtest^3,col="blue")
lines(xtest,apply(rb$yhat.test,2,mean),col="red")
qm = apply(rb$yhat.test,2,quantile,probs=c(.05,.95))
lines(xtest,qm[1,],col="red",lty=2)
lines(xtest,qm[2,],col="red",lty=2)
-------------------------------------------------
```

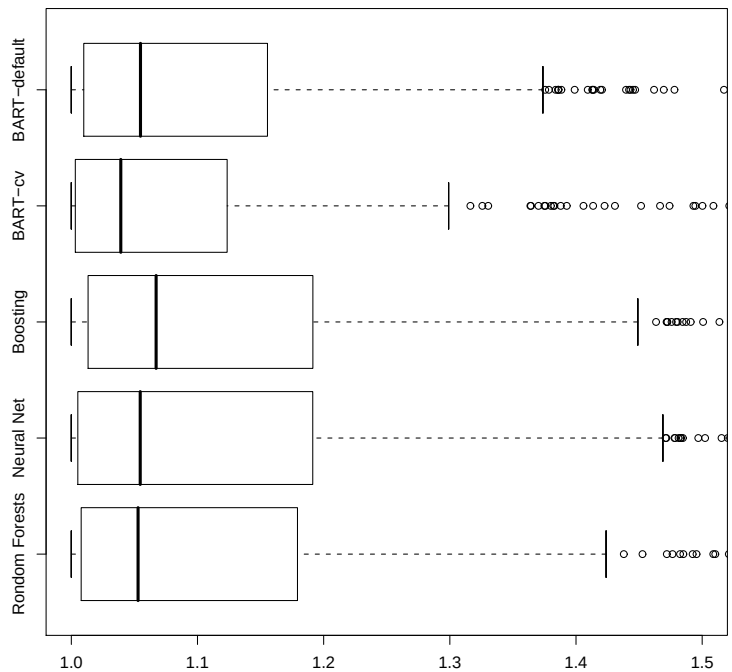


Example: Out of Sample Prediction

Did out of sample predictive comparisons on 42 data sets.
(*thanks to Wei-Yin Loh!!*)

- p=3 − 65, $n = 100 - 7,000$.
- for each data set 20 random splits into 5/6 train and 1/6 test
- use 5-fold cross-validation on train to pick hyperparameters (except BART-default!)
- gives 20*42 = 840 **out-of-sample predictions**, for each prediction, divide rmse of different methods by the smallest

+ each boxplots represents 840 predictions for a method
+ 1.2 means you are 20% worse than the best
+ BART-cv best
+ BART-default (use default prior) does amazingly well!!

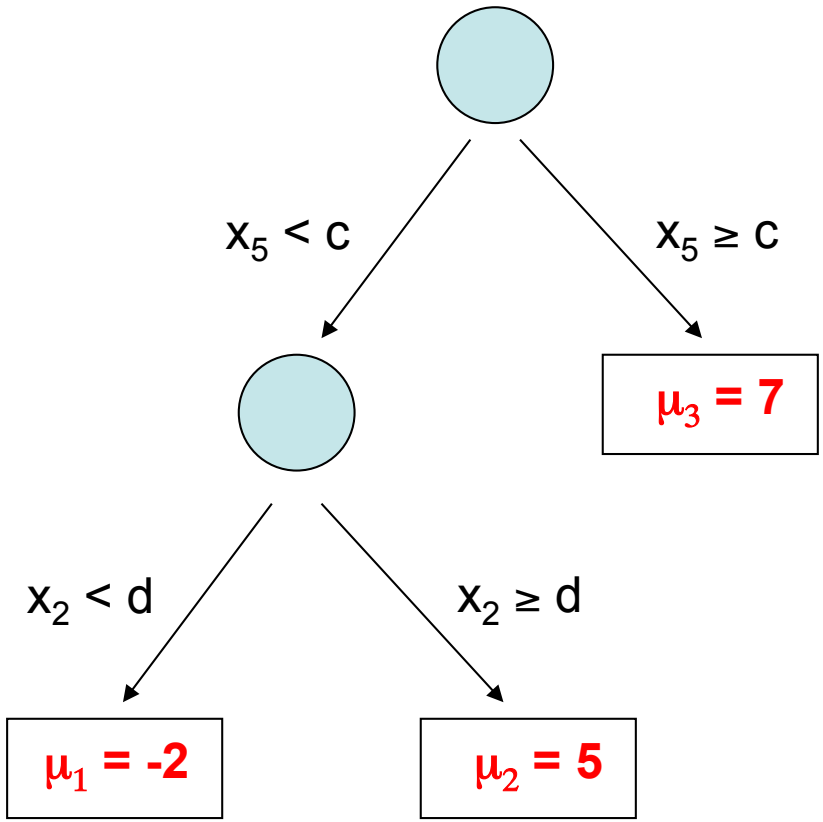


A Regression Tree Model

Let T denote the tree structure including the decision rules.

Let $M = \{\mu_1, \mu_2, \ldots, \mu_b\}$ denote the set of bottom node μ's.

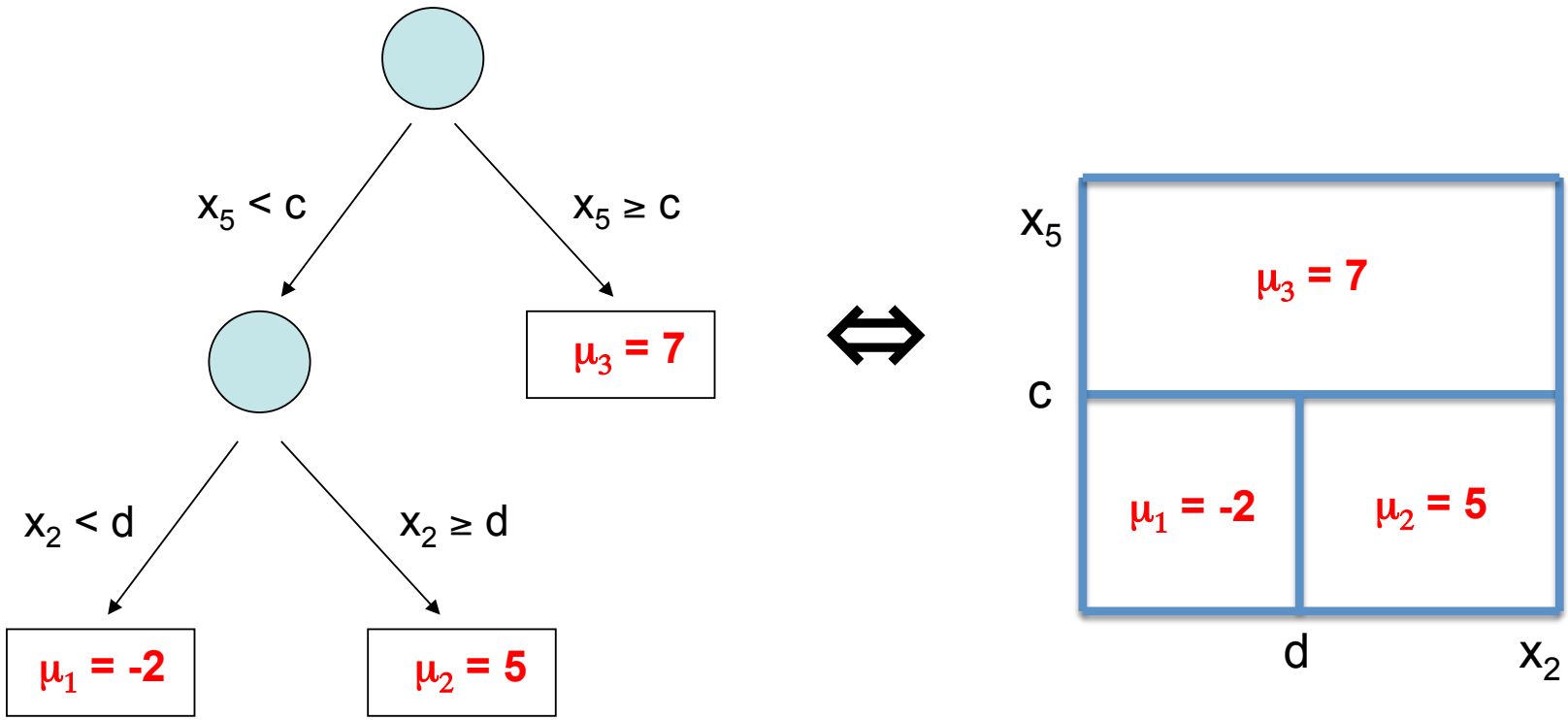
Let $g(x;\theta)$, $\theta = (T, M)$ be a regression tree function that assigns a μ value to x.



A single tree model:

$$y = g(x;\theta) + \epsilon.$$

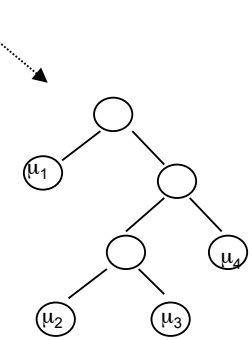
A coordinate view of $g(x;\theta)$



Easy to see that $g(x;\theta)$ is just a step function.

The BART Model

$$Y = g(x;T_1,M_1) + g(x;T_2,M_2) + \ldots + g(x;T_m,M_m) + \sigma z, \quad z \sim N(0,1)$$



$m = 200, 1000, \ldots,$ big, $\ldots$.

$f(x \mid \cdot)$ is the sum of all the corresponding μ's at each bottom node.

Such a model combines additive and interaction effects.

Complete the Model with a Regularization Prior

$$\pi(\theta) = \pi((T_1, M_1), (T_2, M_2), \ldots, (T_m, M_m), \sigma).$$

π wants:

- Each T small.
- Each μ small.
- "nice" σ (smaller than least squares estimate).

We refer to π as a regularization prior because it keeps the overall fit small.

In addition, it keeps the contribution of each $g(x; T_i, M_i)$ model component small.

Consider the prior on μ.
Let θ denote all the parameters.

$$f(x \mid \theta) = \mu_1 + \mu_2 + \cdots \mu_m.$$

Let $\mu_i \sim N(0, \sigma_\mu^2)$, iid.

$$f(x \mid \theta) \sim N(0, m\,\sigma_\mu^2).$$

In practice we often, unabashadly, use the data by first centering and then choosing σ_μ so that

$$f(x \mid \theta) \in (y_{min}, y_{max})$$

with high probability:

$$\sigma_\mu^2 \propto \frac{1}{m}.$$

BART MCMC

$$Y = g(x;T_1,M_1) + \ldots + g(x;T_m,M_m) + \sigma z$$
plus
$$\pi((T_1,M_1),\ldots.(T_m,M_m),\sigma)$$

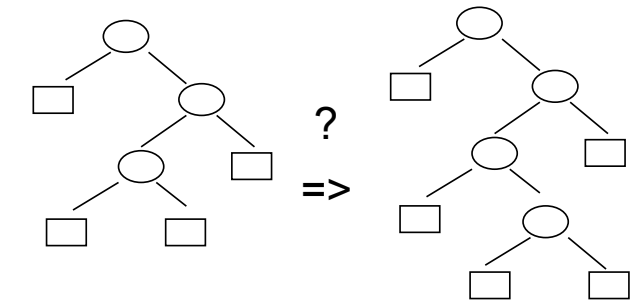
First, it is a "simple" Gibbs sampler:

$$\begin{aligned} (T_i, M_i) \quad &| \quad (T_1, M_1, \ldots, T_{i-1}, M_{i-1}, T_{i+1}, M_{i+1}, \ldots, T_m, M_m, \sigma) \\ \sigma \quad &| \quad (T_1, M_1, \ldots, \ldots, T_m, M_m) \end{aligned}$$

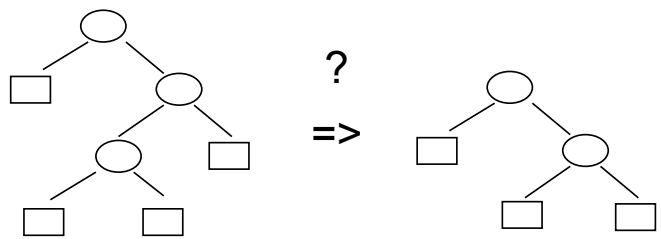
To draw $(T_i, M_i) \,|\, \cdot$ we subract the contributions of the other trees from both sides to get a simple one-tree model.

We integrate out M to draw T and then draw $M \,|\, T$.

To draw T we use a Metropolis-Hastings with Gibbs step.
We use various moves, but the key is a "birth-death" step.



propose a more complex tree



propose a simpler tree

$$Y = g(x;T_1,M_1) + \ldots + g(x;T_m,M_m) + \sigma z$$
plus
$$\pi((T_1,M_1),\ldots.(T_m,M_m),\sigma)$$

Connections to Other Modeling Ideas:

Bayesian Nonparametrics:

- Lots of parameters to make model flexible.
- A strong prior to shrink towards a simple structure.
- BART shrinks towards additive models with some interaction.

Dynamic Random Basis:

- $g(x; T_1, M_1), g(x; T_2, M_2), \ldots, g(x; T_m, M_m)$ are dimensionally adaptive.

Gradient Boosting:

- Overall fit becomes the cumulative effort of many weak learners.

$$Y = g(x;T_1,M_1) + \ldots + g(x;T_m,M_m) + \sigma z$$
plus
$$\pi((T_1,M_1),\ldots.(T_m,M_m),\sigma)$$

Some Distinguishing Feastures of BART:

BART is NOT Bayesian model averaging of single tree model.

Unlike Boosting and Random Forests, BART updates a set of m trees over and over, *stochastic search*.

Choose m large for flexible estimation and prediction.

Choose m smaller for variable selection

- fewer trees forces the x's to compete for entry.

The Friedman Simulated Example

$$y = f(x) + Z, \quad Z \sim N(0,1).$$

$$f(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - .5)^2 + 10x_4 + 5x_5.$$

$n = 100$.
Add 5 irrelevant x's ($p = 10$).
$x_i \sim \text{uniform}(0,1)$.
$\hat{f}(x)$ is the posterior mean.

Compute out of sample RMSE using 1,000 simulated $x \in R^{10}$.

$$\text{RMSE} = \sqrt{\frac{1}{1000}\sum_{i=1}^{1000}(f(x_i) - \hat{f}(x_i))^2}$$

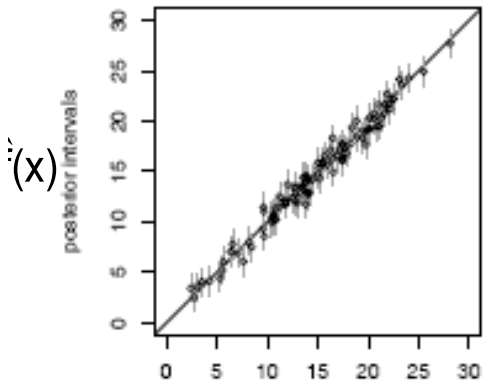
Method	average RMSE	se(RMSE)
Random Forests	2.655	0.025
Linear Regression	2.618	0.016
Neural Nets	2.156	0.025
Boosting	2.013	0.024
MARS	2.003	0.060
BART-cv	1.787	0.021
BART-default	1.759	0.019

Method	Parameter	Values considered
Boosting	# boosting iterations	`n.trees`= $1, 2, \ldots, 2000$
	Shrinkage (multiplier of each tree added)	`shrinkage`= 0.01, 0.05, 0.10, 0.25
	Max depth permitted for each tree	`interaction.depth`= 1,2,3,4
Neural Nets	# hidden units	`size`= 10, 15, 20, 25, 30
	Decay (penalty coef on sum-squared weights)	`decay`= 0.50, 1, 1.5, 2, 2.5
	(Max # optimizer iterations, # restarts)	fixed at `maxit`= 1000 and 5
Random Forests	# of trees	`ntree`= 200, 500, 1000
	# variables sampled to grow each node	`mtry`= 3, 5, 7, 10
MARS	GCV penalty coefficient	`gcv`= 1, 2, ..., 8
BART -cv	Sigma prior: (ν, q) combinations	(3,0.90), (3,0.99), (10,0.75)
	μ Prior: k value for σ_μ	1, 1.5, 2, 2.5, 3
	(# trees m, iterations used, burn-in iterations)	fixed at (200, 1000,500)
BART -default	Sigma prior: (ν, q) combinations	fixed at (3,0.90)
	μ Prior: k value for σ_μ	fixed at 2
	(# trees m, iterations used, burn-in iterations)	fixed at (200, 1000,500)

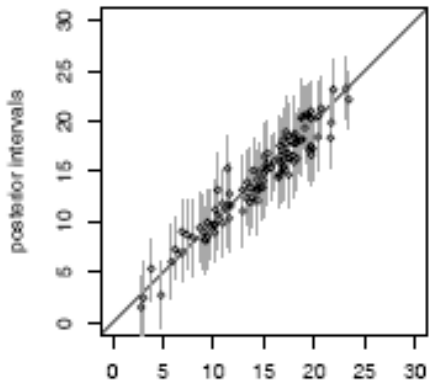
Table 1: Operational parameters for the various competing models. `Names` in last column indicate parameter names in R.

Results for one draw.

95% posterior intervals vs true f(x)

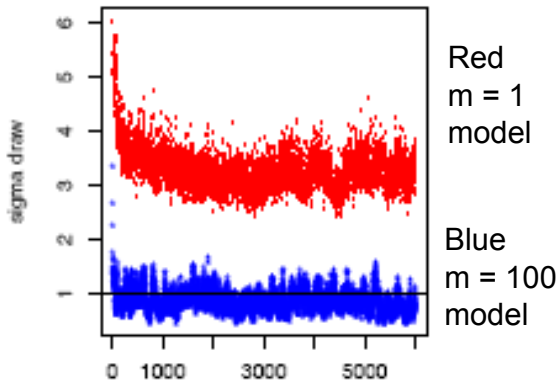


in-sample f(x)



out-of-sample f(x)

σ draws

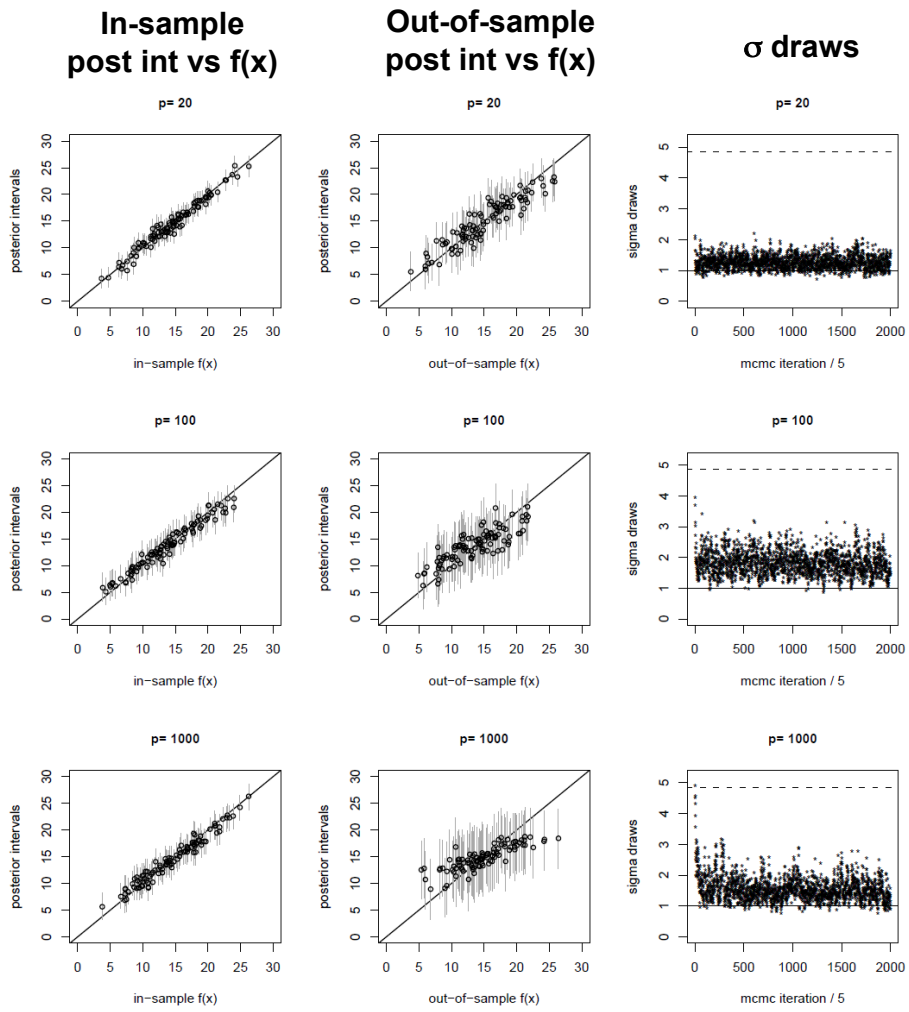


MCMC iteration

Frequentist coverage rates of 90% posterior intervals:
in sample: 87%
out of sample: 93 %.

Added many useless x's to Friedman's example

With only 100 observations on y and 1000 x's, BART yielded "reasonable" results !!!!



20 x's

100 x's

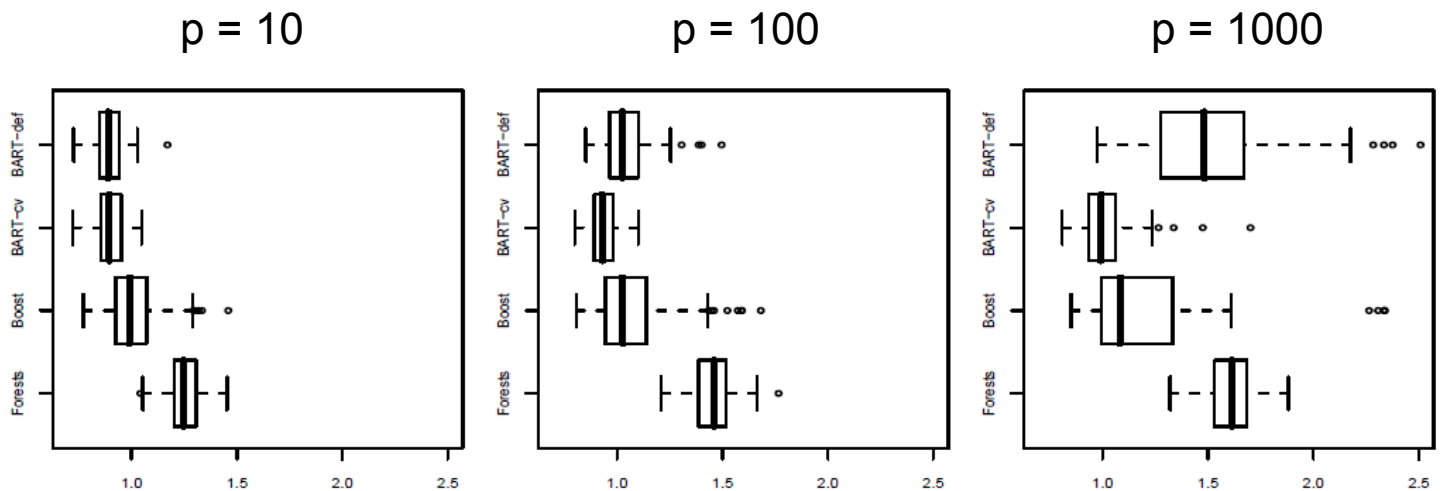
1000 x's

Big p, small n.

$n = 100$.

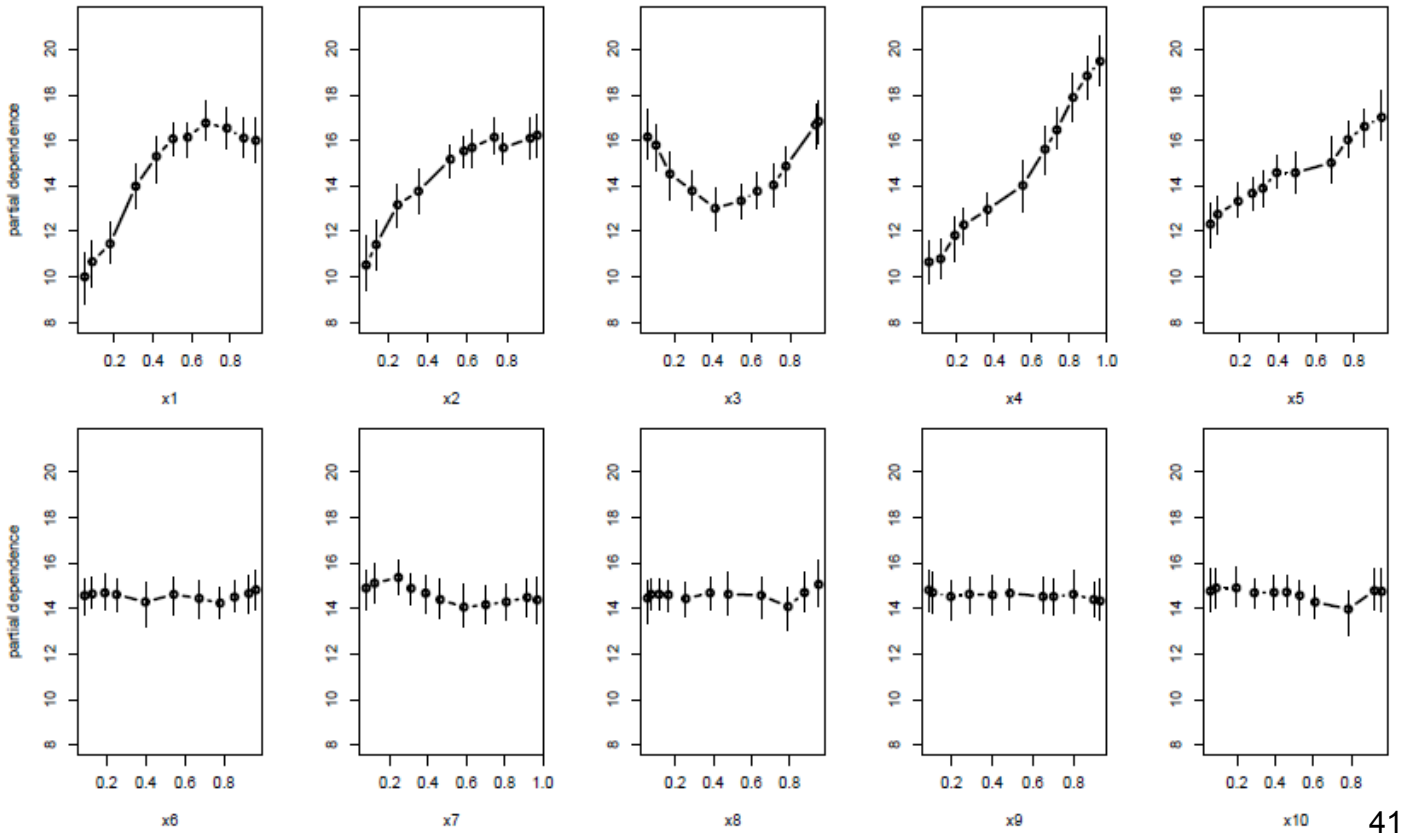
Compare BART-default,BART-cv,boosting, random forests.

Out of sample RMSE.

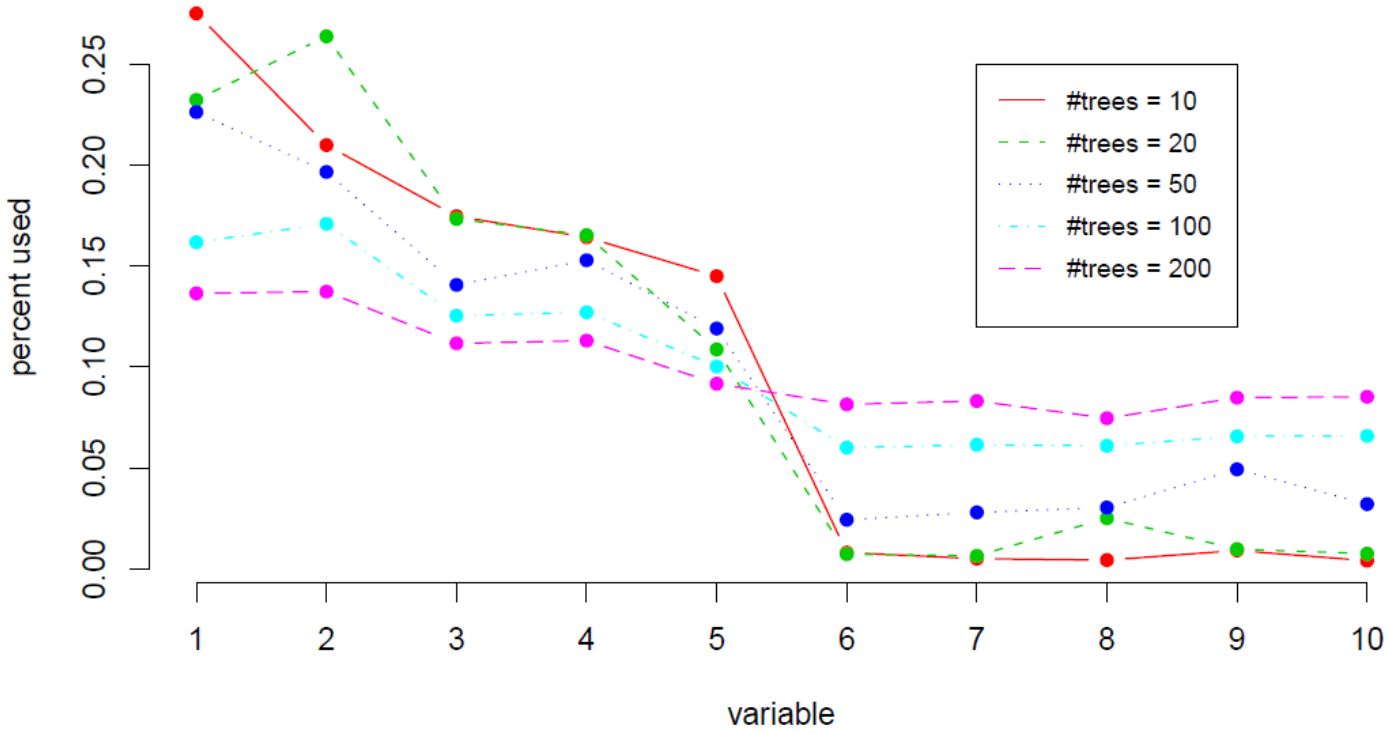


Partial Dependence plot:

Vary one x and average out the others.



Variable selection, frequency with which a variable is used.



Example: Drug Discovery

Goal: To predict the "activity" of a compound against a biological target.

That is: $y = 1$ means drug worked (compound active), 0 means it does not.

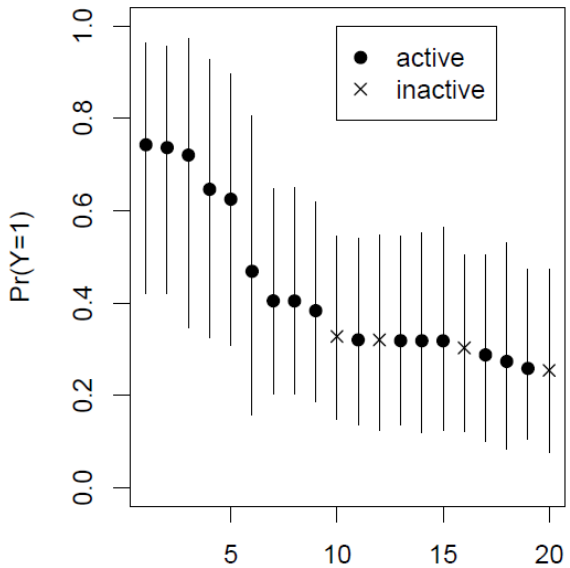
Easy to extend BART to binary y using Albert & Chib.

$n = 29,3744 \rightarrow$ $14,687$ train, $14,687$ test.
$p = 266$ characterizations of the compound's molecular structure.

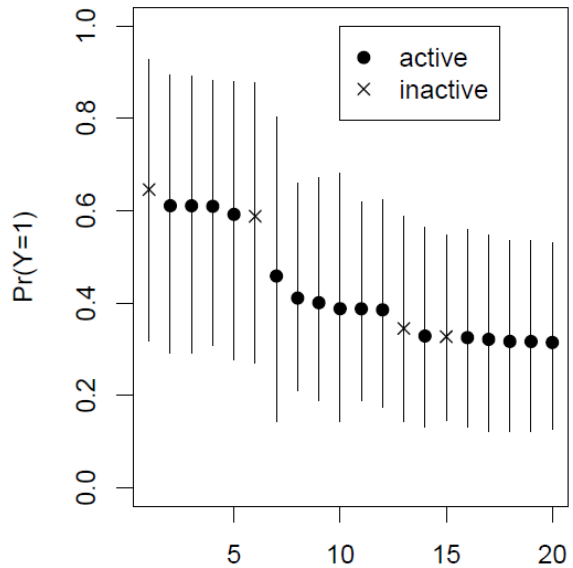
Again, out-of-sample prediction competitive with other methods, compared to neural-nets, boosting, random forests, support vector machines.

20 compounds with highest $Pr(Y = 1 \mid x)$ estimate.
90% posterior intervals for $Pr(Y = 1 \mid x)$.

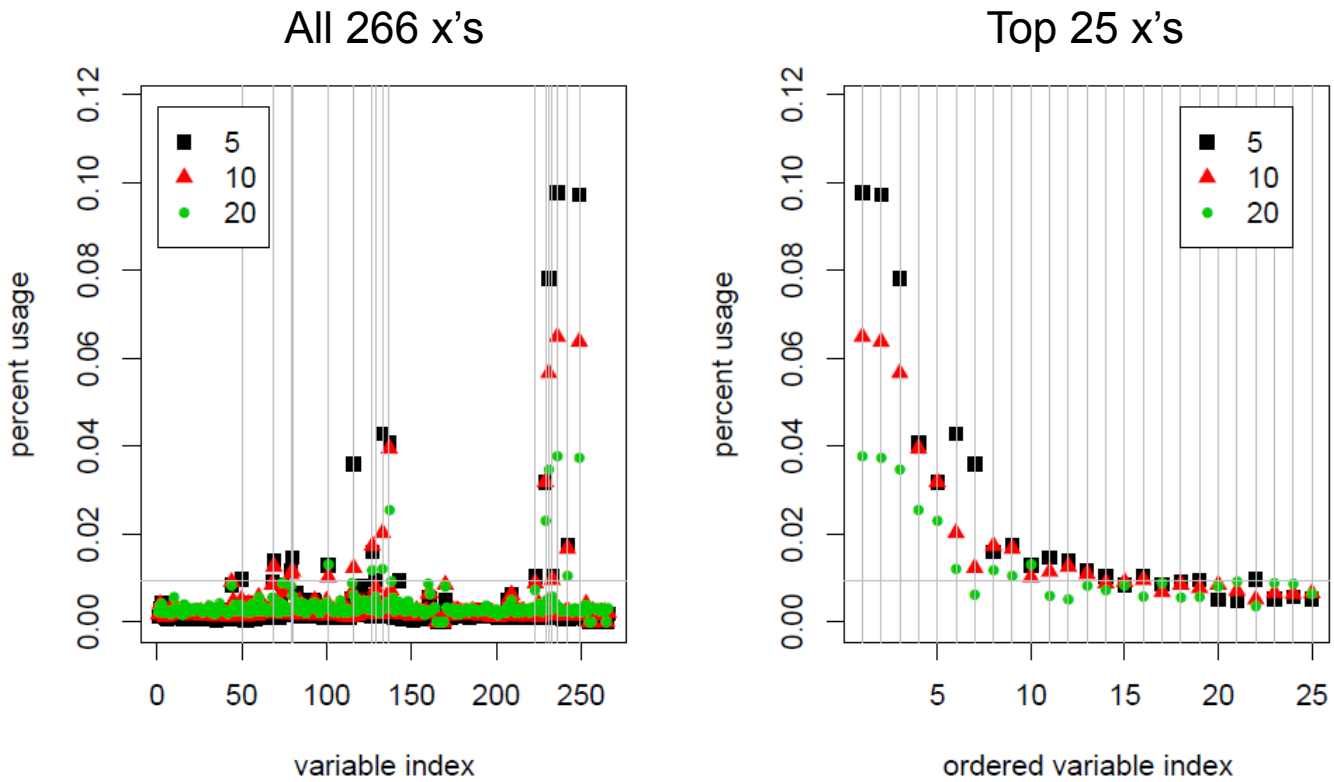
In-sample



Out-of-Sample



Variable selection.



Current Work

Nonparametric modeling of the error distribution (with Paul Damien)

Multinomial outcomes (with Nick Polson).

More on priors and variable-selection.

Constrain the multivariate function to be monotonic
(with Tom Shively)
- Tom has a *beautiful* cross-dimensional, constrained, slice-sampler.

Recode with MPI to make it faster!!

With Dave Higdon, James Gattiker, and Matt Pratola at Los Alamos National Labs..

Dave came to me and said, "we tried your stuff (the R package) on the analysis of computer experiments and it seemed promising but it is too slow".

1. Rewrote code so that it is leaner.
2. Used MPI to compute.

	num obs	new-parallel	new-serial	old
1	1000	7	9	43
2	2000	8	18	95
3	3000	9	28	149
4	4000	10	36	204
5	5000	12	45	262
6	10000	18	90	547
7	50000	70	439	NA
8	100000	138	902	NA
9	500000	904	6410	NA

With 10,000 observations the new algorithm is $547/90 = 6$ times faster than the old algorithm.

The parallel version is $90/18 = 5$ times faster than the serial version (with 7 cores).

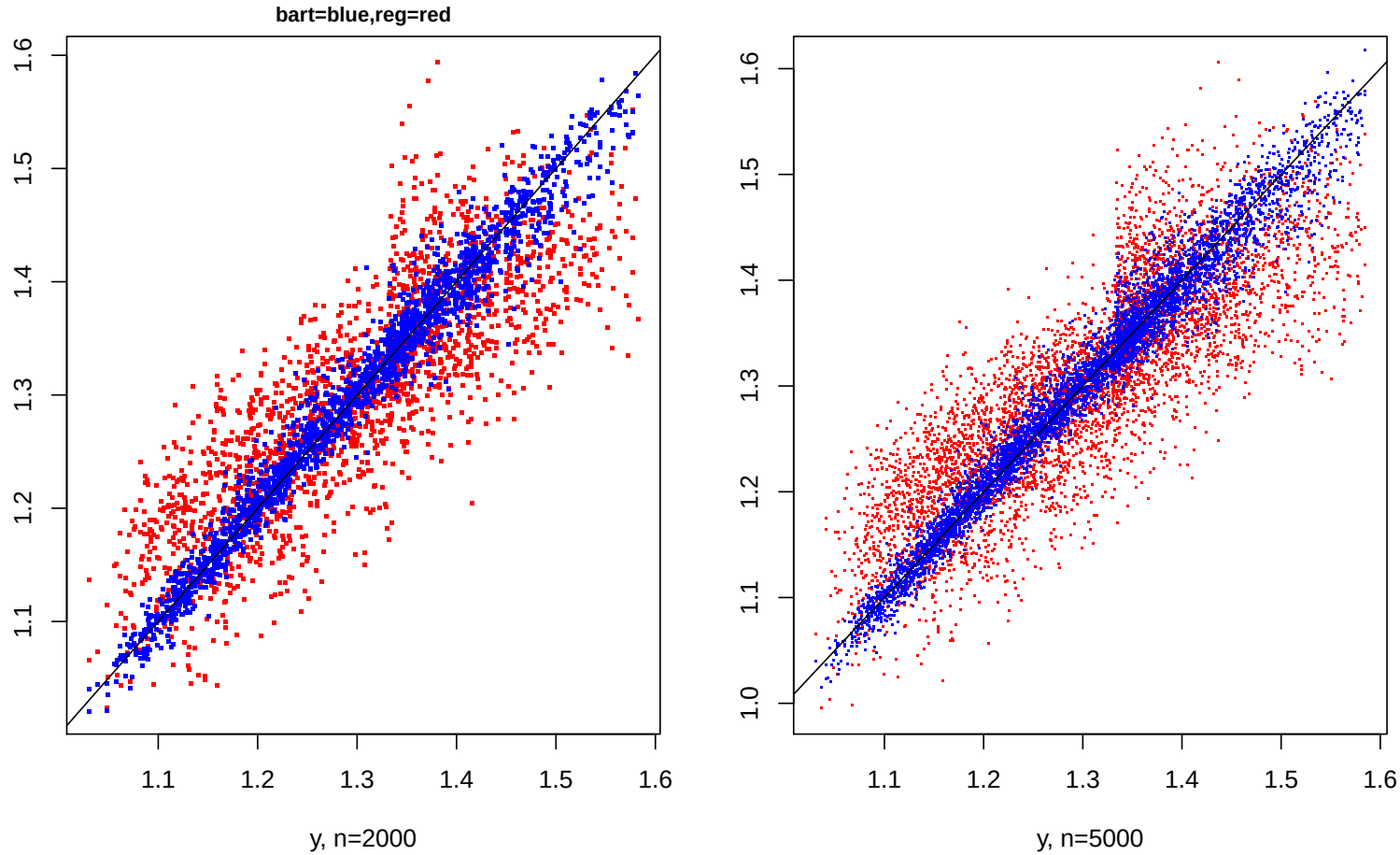
Thus, the parallelized new algorithm is 30 times faster than the old BART algorithm (available in the R package BayesTree).

With 500,000 observations, the old algorithm cannot be run on the machine being used. The parallel version is $6410/904 = 7$ times faster than the serial version. Recall that we are using 7 cores to do the basic computations.

linear in the number of cores!!

100,000 observations, $p = 251$.
For regression, $\text{cor}(y, \hat{y}) = .84$ for BART $= .99$.

Blue is BART, red is least-squares.



3. Parallelize prediction, so we can do all the stuff we want!