Introduction to the bartMachine R package

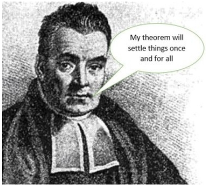
Saint Louis R User Group

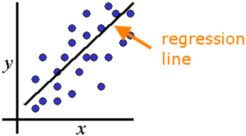
John Snyder

April 11, 2019

What is BART?

Bayesian **A**dditive **R**egression **T**rees





How it works

- Ensemble method which is the sum of many shallow trees.
- Complexity is regularized via Bayesian "priors."
  - This frees us from ad hoc decisions
- Uses "Bayesian Backfitting"
  - Each tree is sequentially exposed to the residuals when all other trees are used to predict

Results:

- Each tree describes a tiny amount of the structure
- The Bayesian structure means variation is fully quantified.
  - Intervals, p-values, and model selection become possible
- Outperforms many common models in out of sample prediction.

Powerful Predictive Performance

- Test RMSE of 100 random datasets simulated from various nonlinear functions (added noise with s=1)

Function	BART	XGBoost*	Random Forest*	Linear Reg(lol)
Friedman	1.08	1.21	1.64	2.61
Mirsha's Bird	1.53	2.78	2.90	26.59
Weird Exp	1.04	1.05	1.07	6.08
Linear	1.025	1.032	1.034	1.004

Details in Simulation.R

- bartMachine is relatively unknown
  - xgboost: ~43k downloads per month
  - randomForest: ~88k downloads per month
  - bartMachine: ~2k downloads per month
- No full featured python library

Package Overview:

- Based in Java
- Functions for Cross Validation
- Model fitting:
  - Is done in parallel[1]
  - Can incorporate missing data
- Lots of fun statistical things
  - Credible interval calculation
  - Diagnostic plots/tests
- Model free variable selection
- Interaction detection
- Export fit trees

[1] MCMC sampling is used, so speedups during model fitting aren't great

Installation and loading steps

1. Google “How to install rJava on [your OS]”
2. Do that
3. Run the following

```r
install.packages("bartMachine")
```

To load the package with:

- 10GB of memory
- All but one core available for compute

```r
options(java.parameters = "-Xmx10g")
library(bartMachine)
numcores <- parallel::detectCores()
set_bart_machine_num_cores(numcores - 1)
```

Boston Data

We will consider the Boston dataset from the MASS package

```r
data(Boston)
Boston %>% round(digits=2) %>% head
```

```
##   crim zn indus chas  nox   rm  age  dis rad tax ptratio  black lstat medv
## 1 0.01 18  2.31    0 0.54 6.58 65.2 4.09   1 296    15.3 396.90  4.98 24.0
## 2 0.03  0  7.07    0 0.47 6.42 78.9 4.97   2 242    17.8 396.90  9.14 21.6
## 3 0.03  0  7.07    0 0.47 7.18 61.1 4.97   2 242    17.8 392.83  4.03 34.7
## 4 0.03  0  2.18    0 0.46 7.00 45.8 6.06   3 222    18.7 394.63  2.94 33.4
## 5 0.07  0  2.18    0 0.46 7.15 54.2 6.06   3 222    18.7 396.90  5.33 36.2
## 6 0.03  0  2.18    0 0.46 6.43 58.7 6.06   3 222    18.7 394.12  5.21 28.7
```

- Target: Predict median home value(medv)
- Features include:
    1. lstat: Proportion of low income individuals
    2. rm: Average umber of rooms per dwelling
    3. age: proportion of old homes
    4. etc. . .

Fitting BART

```r
y <- Boston$medv
X <- Boston %>% dplyr::select(-"medv")

#Fit BART model
bart.model <- bartMachine(X,y,
                          num_trees = 200,
                          num_burn_in = 1000,
                          num_iterations_after_burn_in = 5000)
```

BART is fit with MCMC, which requires a "burnin" set of initial iterations which are discarded.

- NOTE: with 10 cores, each thread would fit $1000 + 5000/10$ or 1500 iterations

The BART object

```
> bart.model
bartMachine v1.2.3 for regression

training data n = 506 and p = 13
built in 19.4 secs on 31 cores, 200 trees, 1000 burn-in and 5000 post.

sigsq est for y beforehand: 21.938
avg sigsq estimate after burn-in: 2.70319

in-sample statistics:
 L1 = 430.35
 L2 = 654.72
 rmse = 1.14
 Pseudo-Rsq = 0.9847
p-val for shapiro-wilk test of normality of residuals: 0
p-val for zero-mean noise: 0.98467
```

Did we overfit?

K fold CV using our fit model

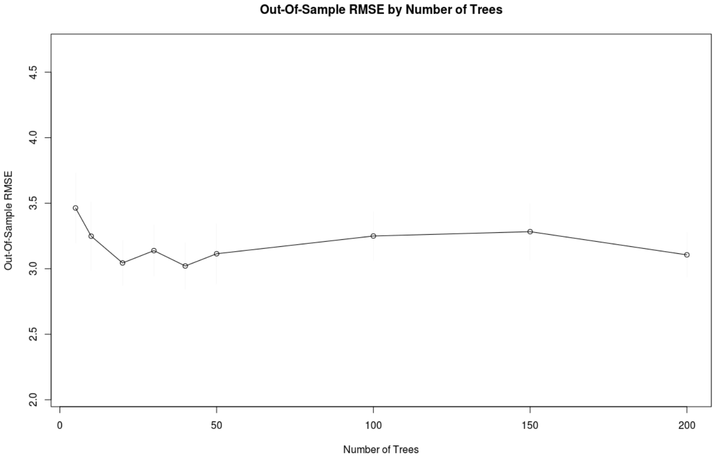
```
k_fold_cv(X, y, k_folds = 10,
          num_trees = 200,
          num_burn_in = 1000,
          num_iterations_after_burn_in = 5000)
```

```
..........
$y_hat
  [1] 24.932807 20.522438 31.149051 36.071931...

$L1_err
[1] 1022.238

$L2_err
[1] 4624.004

$rmse
[1] 3.02297

$PseudoRsq
[1] 0.8917508

$folds
  [1]  8 10  3  1 ....
```

Select the number of Trees

```
rmse_by_num_trees(bart.model, num_replicates = 20)
```

- This fits the model 20 times for a number of various numbers of trees
- Aggregates out of sample RMSE



Cross Validate

```
bart.model.cv <- bartMachineCV(X, y,
                               num_burn_in = 1000,
                               num_iterations_after_burn_in = 5000)
```

- To automatically cross validate, we just need to add a CV to the end of the function!
- Will fit a variety of models and return the best one.
  - Similar to the caret package
- This takes a while.
- bart.model.cv will be the model object for the best fit

Cross Validate

```
> bart.model.cv$cv_stats
      k nu    q num_trees oos_error % diff with lowest
 [1,] 2  3 0.90        50  2.860489           0.000000
 [2,] 3  3 0.90        50  2.922613           2.171807
 [3,] 2  3 0.99       200  2.924967           2.254080
 [4,] 2  3 0.90       200  2.933799           2.562845
 [5,] 2 10 0.75       200  2.956046           3.340573
 [6,] 2 10 0.75        50  3.036727           6.161102
 [7,] 3 10 0.75        50  3.040292           6.285746
 [8,] 3  3 0.99        50  3.060013           6.975171
 [9,] 3  3 0.90       200  3.086522           7.901900
[10,] 3  3 0.99       200  3.093631           8.150404
[11,] 3 10 0.75       200  3.123639           9.199475
[12,] 5  3 0.90        50  3.140318           9.782571
[13,] 5  3 0.99        50  3.158856          10.430626
[14,] 5 10 0.75        50  3.215616          12.414893
[15,] 2  3 0.99        50  3.342410          16.847512
[16,] 5  3 0.90       200  3.446951          20.502168
[17,] 5  3 0.99       200  3.447560          20.523454
[18,] 5 10 0.75       200  3.459553          20.942727
```

Cross Validate

```
k_fold_cv(X, y, k_folds = 5,
          k=2, nu=3, q=.90, num_trees = 50,
          num_burn_in = 1000,
          num_iterations_after_burn_in = 5000)
```

```
.....

$rmse
[1] 2.92

$PseudoRsq
[1] 0.9138275
```

This is better than before!

Variable Importance

```
investigate_var_importance(bart.model.cv)
```

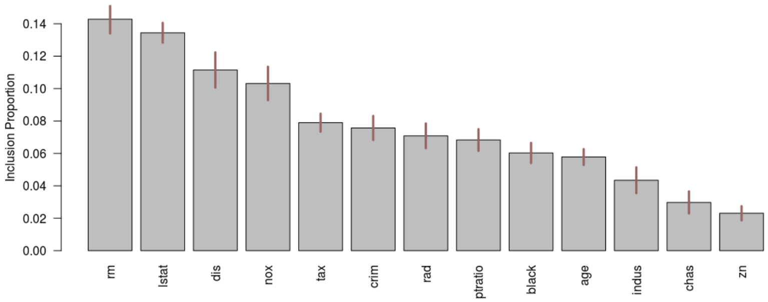


Figure 2

Joint Variable Imortance

```
interaction_investigator(bart.model.cv)
```

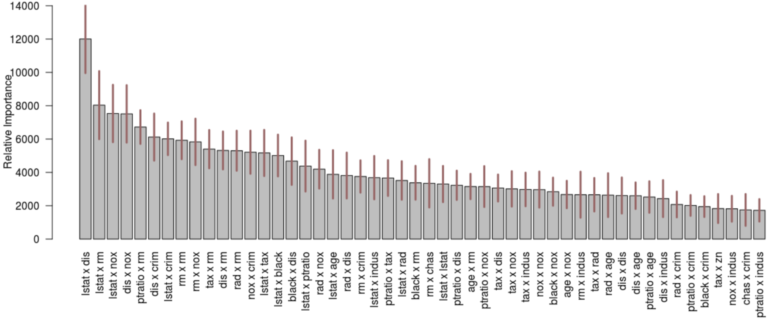


Figure 3

Model Selection

```
VarSel <- var_selection_by_permute(bart.model.cv,bottom_margin = 5)
VarSel$important_vars_local_names
```

```
[1] "rm"    "lstat" "dis"   "nox"
```

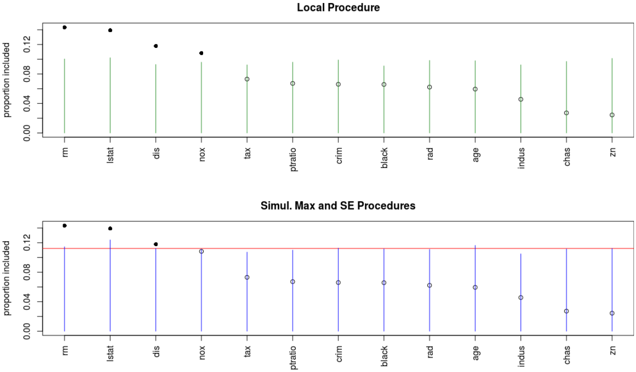


Figure 4

Partial Dependence plots!

```
cov_importance_test(bart.model.cv, covariates = "lstat")
pd_plot(bart.model.cv, j = "lstat")
```

```
P-Value = 0
```

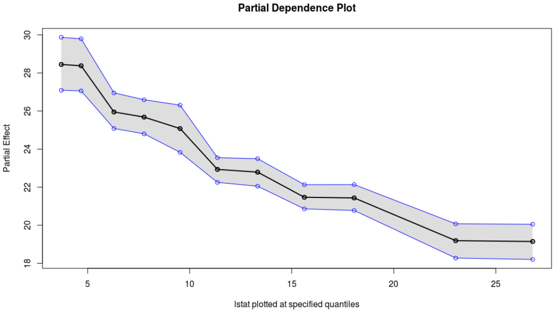


Figure 5

- Low income areas have a negative relationship with home value

Partial Dependence plots!

```
cov_importance_test(bart.model.cv, covariates = "rm")
pd_plot(bart.model.cv, j = "rm")
```

```
P-Value = 0
```

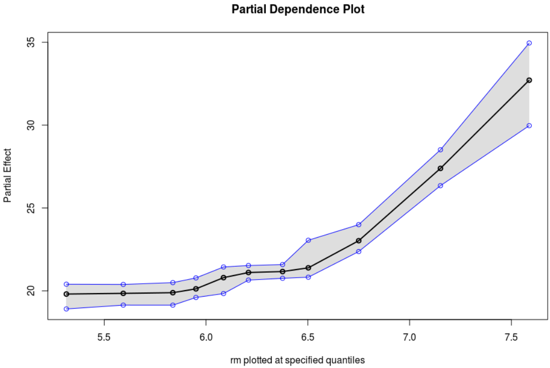


Figure 6

- Number of rooms has a positive relationship with home value

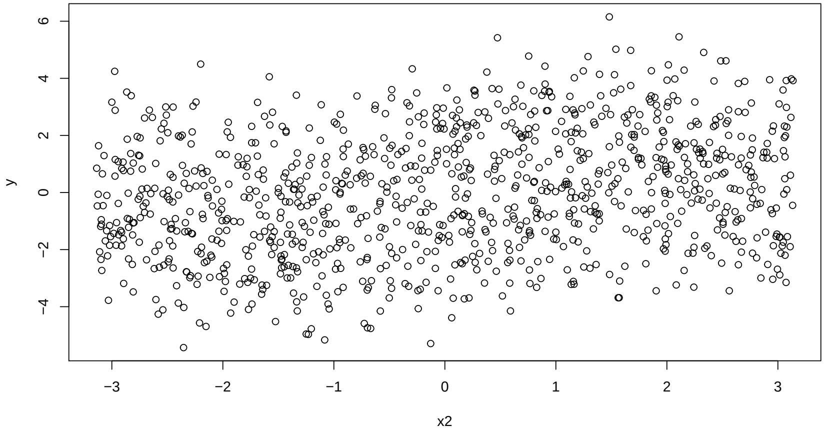
Conclusions supported by regressions

```r
RegressMod <- lm(formula = medv ~ ., data = Boston)
summary(RegressMod)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

Aside: Partial Dependence

```r
nsim <- 1000
x1 <- runif(nsim,-3.14,3.14)
x2 <- runif(nsim,-3.14,3.14)

y <- x1 + sin(x2) + rnorm(nsim)
plot(x2,y)
```



Aside: Partial Dependence

```
bart.model <- bartMachine(data.frame(x1,x2),y,
                          num_trees = 200,
                          num_burn_in = 1000,
                          num_iterations_after_burn_in = 4000)
pd_plot(bart.model, j = "x2")
```

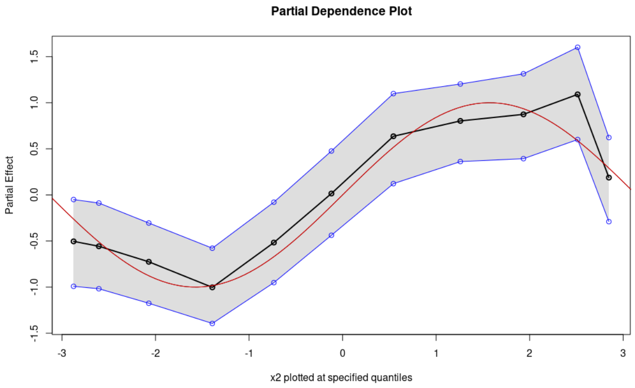


Figure 7

Notable Arguments

The bartMachine function has a number of arguments. Default values are listed.

1. use_missing_data = FALSE
   - Uses rows with missing data *without* imputing
2. mem_cache_for_speed = TRUE
3. serialize = FALSE
   - Used to store the java object
   - save(bart.model, file="MyBart.Rdata")
4. cov_prior_vec = NULL
   - Place informative priors on variable inclusion
5. If y is a factor, classification is performed automatically.

Other Statistical Things: Credible/Prediction intervals!

```
MeanDF <- colMeans(X)
MeanDF %>% head
```

```
   crim     zn  indus  chas   nox    rm    age   dis
1 3.614 11.364 11.137 0.069 0.555 6.285 68.575 3.795
    rad     tax ptratio   black  lstat
1 9.549 408.237  18.456 356.674 12.653
```

```
predict(bart.model.cv, newdata = MeanDF)
calc_credible_intervals(bart.model.cv, MeanDF, ci_conf = 0.95) %>% round(digits=2)
calc_prediction_intervals(bart.model.cv, MeanDF, pi_conf = 0.95) %>% round(digits=2)
```

```
[1] 23.60462

     ci_lower_bd ci_upper_bd
[1,]       17.69        29.3
```

Other Statistical Things: Assumption Checking

```
check_bart_error_assumptions(bart.model.cv)
```

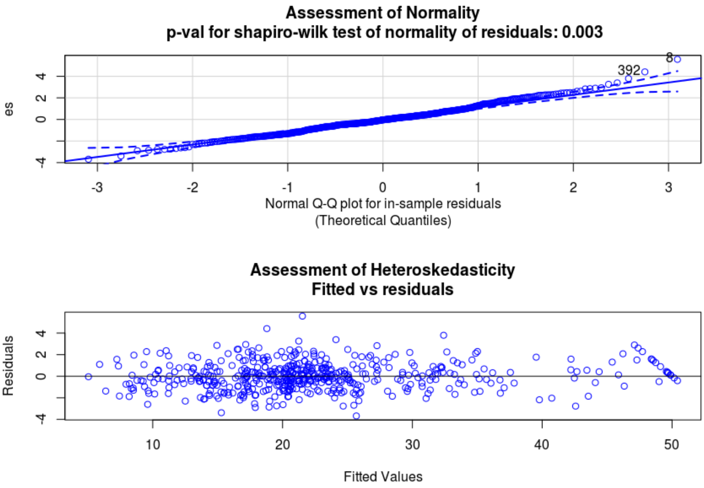
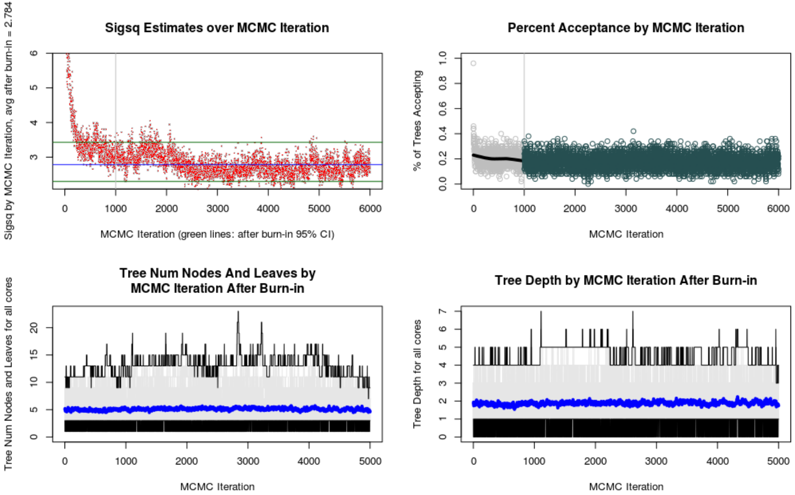


Figure 8

Other Statistical Things: Convergence Diagnostics

```
plot_convergence_diagnostics(bart.model.cv)
```

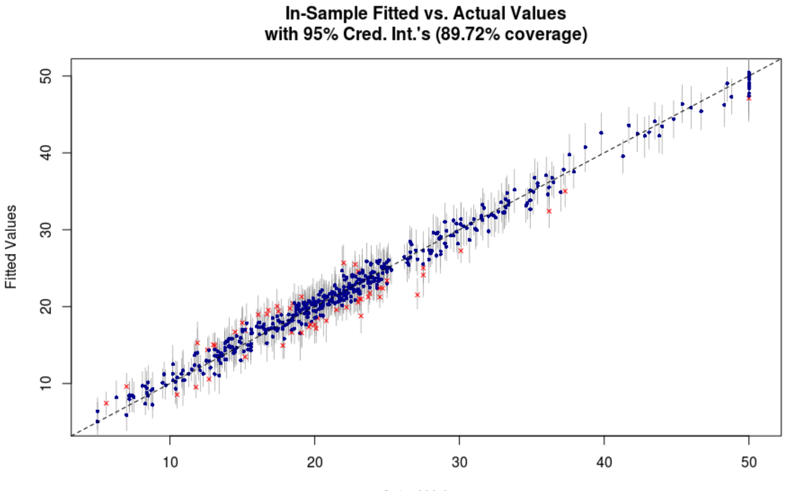
MCMC must “converge”



Other Statistical Things: Observations vs Predictions

```
plot_y_vs_yhat(bart.model.cv, credible_intervals = TRUE)
```

Looking for a 1:1 relationship



John's Final Thought

- BART is a powerful technique which brings many advantages
  - At the expense of computational efficiency.

```r
# n is 10000, p is 100
bart.model <- bartMachine(X,y,
                          num_trees = 100,
                          num_burn_in = 1000,
                          num_iterations_after_burn_in = 5000,
                          mem_cache_for_speed = TRUE)
```

- Statistical advantages are numerous
- Package authors aggregated many academic works on BART
- Great for small to mid sized data
- Good results with removing expected variation and feeding residuals into BART.