Assignment 3

Chelsey Hvingelby 35583153

March 18, 2019

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

Q1

Reading in the data.

```
salaries <- read.csv("salaries.csv", header = TRUE)
#head(salaries)
```

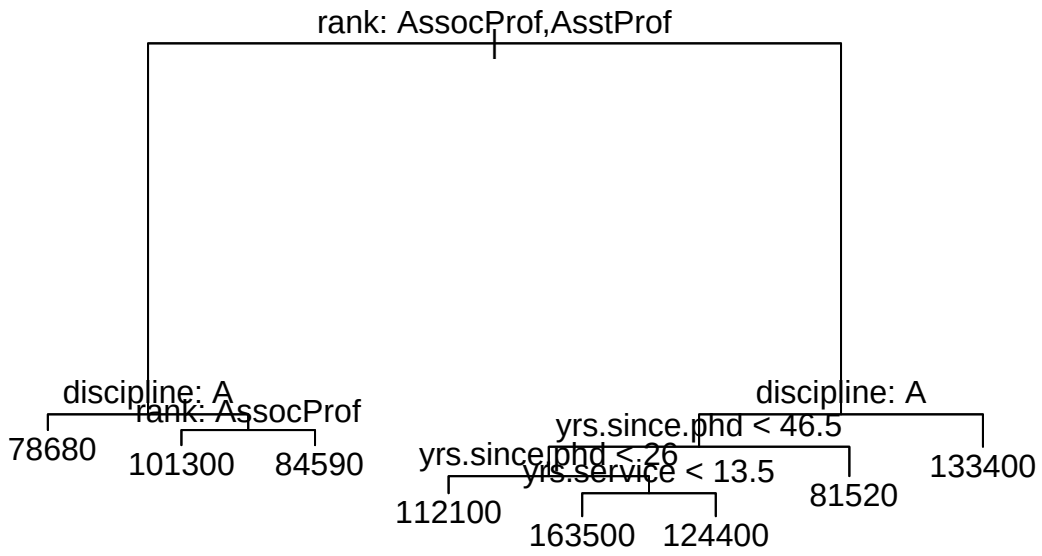
a)

Create a regression tree for a professor's salary given the remainder of the variables in the data set. Provide the tree, including labels.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.5.3
```

```
salarytree <- tree(salary~., data=salaries)
plot(salarytree)
text(salarytree, pretty=0)
```



b)

The nodes that eventually result from the splot "yrs.since.phd < 46.5" are a pretty strange result because this split shows that the salary is significantly higher if it has been less than 46.5 years since obtaining a PhD. This contradicts my initial thinking that more years since obtaining your PhD would result in more experience and hence a higher salary. What is really interesting is that the same is not true about the split "yrs.since.phd < 26". This split tells us that if it has been less than 26 years since you obtained your PhD, you will make less than someone with more experience (more time since obtaining their PhD).

One possible explanation for this strange result on the split "yrs.since.phd < 46.5" is that the professors who obtained their PhD longer than 46.5 years ago are less enthusiastic about doing research or bringing on graduate students. They may then recieve a salary cut for not publishing. Another possible explanation would be that they are only working part time as they may be preparing for retirement.

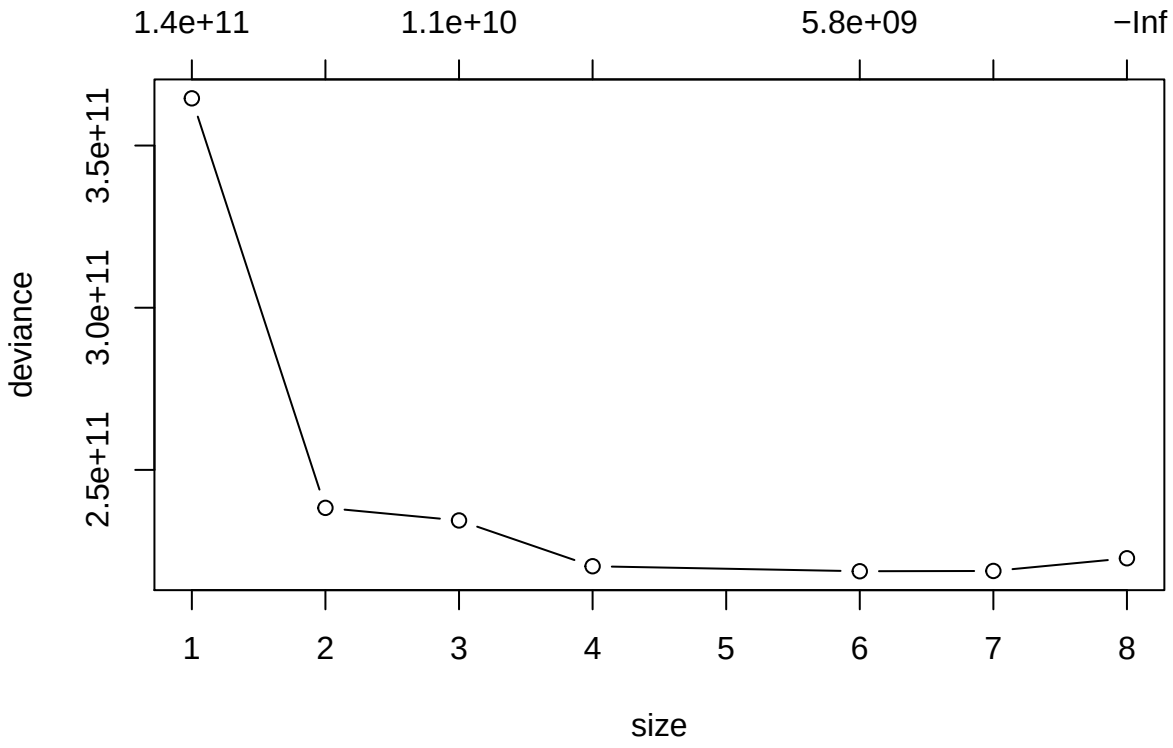
c)

Based on this tree, as an early career professor (Assistant or Associate) you would make more money in a theoretical department. This can be seen on the "discipline: A" split on the left hand side of the tree.

d)

Perform 20-fold cross-validation using cv.tree. Plot the resulting object. How many terminal nodes does cross-validation suggest?

```
set.seed(6421)
cv.salarytree <- cv.tree(salarytree, FUN = prune.tree, K = 20)
plot(cv.salarytree, type="b")
```




```
#How many terminal nodes does cross-validation suggest?
cv.salarytree$dev
```

```
## [1] 222749582678 218837618667 218751273169 220290119067 234412250039
## [6] 238295262199 364550608892
```

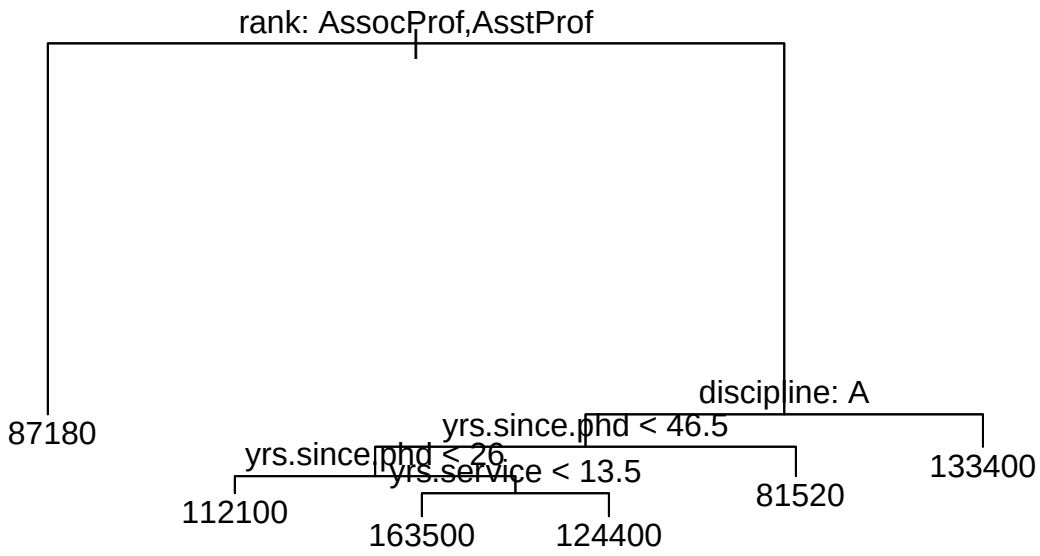
```
cv.salarytree$size
```

```
## [1] 8 7 6 4 3 2 1
```

```
which.min(cv.salarytree$dev)
```

```
## [1] 3
```

```
#This suggests that the third element of the vector is the best number of terminal nodes.  In other wor
p.salarytree <- prune.tree(salarytree, best=6)
plot(p.salarytree)
text(p.salarytree, pretty = 0)
```




```
summary(p.salarytree)
```

```
##
## Regression tree:
## snip.tree(tree = salarytree, nodes = 2L)
## Variables actually used in tree construction:
## [1] "rank"          "discipline"    "yrs.since.phd" "yrs.service"
## Number of terminal nodes:  6
## Residual mean deviance:  480500000 = 1.879e+11 / 391
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -65830  -14380   -2676       0   12610   98150
```

e)

Give the predicted salary for me, assuming I was at this university. That is, what is the predicted salary for an Assistant Professor, in an aplied department, who got their PhD in 2012 (5 years ago), has 4 years of service, and is male. Use the tree and your brain, or enter the data into R and use the predic() function.

```
predict(p.salarytree, salaries, type = "tree")
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 397 3.633e+11 113700
##    2) rank: AssocProf,AsstProf 131 2.208e+10  87180 *
##    3) rank: Prof 266 2.036e+11 126800
```

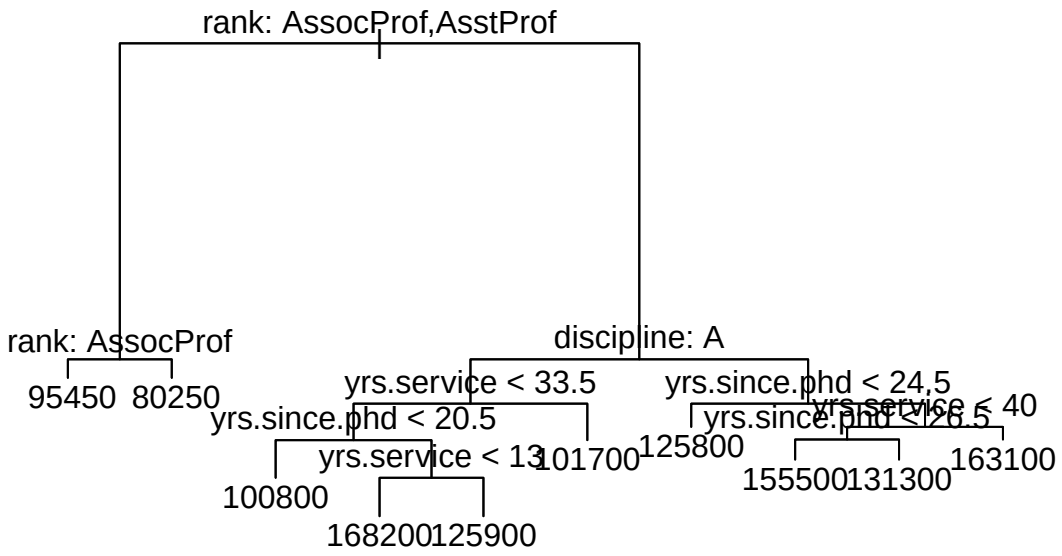
```
##      6) discipline: A 131 1.016e+11 119900
##       12) yrs.since.phd < 46.5 124 8.925e+10 122100
##         24) yrs.since.phd < 26 42 1.331e+10 112100 *
##         25) yrs.since.phd > 26 82 6.961e+10 127200
##           50) yrs.service < 13.5 6 5.428e+09 163500 *
##           51) yrs.service > 13.5 76 5.565e+10 124400 *
##       13) yrs.since.phd > 46.5 7 1.458e+09  81520 *
##      7) discipline: B 135 8.996e+10 133400 *
```

Given the tree and the above output, Dr. Andrews has a predicted salary at this University of $87180.

f)

```
#Setting up the training and testing set
set.seed(763)
trainindex <- sample(1:nrow(salaries), 200)
proftrain <- salaries[trainindex, ]
proftest <- salaries[-trainindex, ]

#Fit the model to the training dataset
salarytreetrain <- tree(salary~., data=proftrain)
plot(salarytreetrain)
text(salarytreetrain, pretty=0)
```




```
predict(salarytreetrain, salaries, type = "tree")
```

```
## node), split, n, deviance, yval
```

```
##       * denotes terminal node
##
##  1) root 397 3.634e+11 114300
##    2) rank: AssocProf,AsstProf 131 2.237e+10  88670
##      4) rank: AssocProf 64 1.221e+10  95450 *
##      5) rank: AsstProf 67 4.428e+09  80250 *
##    3) rank: Prof 266 2.036e+11 126700
##      6) discipline: A 131 1.020e+11 118200
##       12) yrs.service < 33.5 98 6.982e+10 125100
##         24) yrs.since.phd < 20.5 26 6.903e+09 100800 *
##         25) yrs.since.phd > 20.5 72 5.752e+10 131800
##           50) yrs.service < 13 8 1.210e+10 168200 *
##           51) yrs.service > 13 64 4.431e+10 125900 *
##       13) yrs.service > 33.5 33 3.090e+10 101700 *
##      7) discipline: B 135 9.011e+10 134500
##       14) yrs.since.phd < 24.5 68 2.926e+10 125800 *
##       15) yrs.since.phd > 24.5 67 6.293e+10 142200
##         30) yrs.service < 40 60 5.040e+10 138900
##           60) yrs.since.phd < 26.5 11 5.859e+09 155500 *
##           61) yrs.since.phd > 26.5 49 3.940e+10 131300 *
##         31) yrs.service > 40 7 1.339e+10 163100 *
```

Based on the model fit to the training set, the predicted salary for Dr. Andrews is $80250.

Calculating the MSE of the test set

```
set.seed(763)
##Remove the 6th (salary) column of the test set
proftest_predictors <- proftest[,-6]
salarytestMSE <- sum((proftest$salary-predict(salarytreetrain, proftest_predictors))^2)/length(proftest$
salarytestMSE
```

```
## [1] 616355582
```

This is huge!

g)

```
set.seed(474)
salarybag <- randomForest(salary~., data = salaries, importance = TRUE, mtry = 5)
salarybag
```

```
##
## Call:
##  randomForest(formula = salary ~ ., data = salaries, importance = TRUE,      mtry = 5)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 571054059
##                     % Var explained: 37.6
```

Calculate the MSE

```
salarybag
```

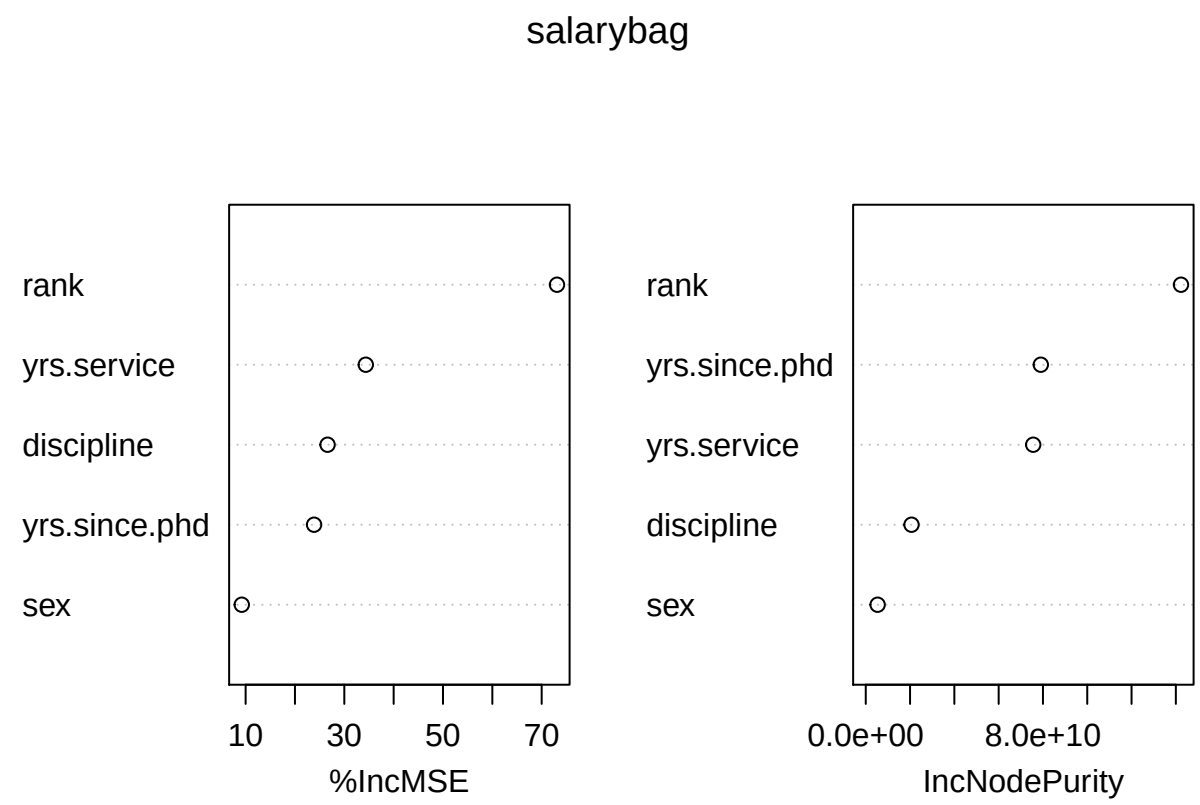
```
##
## Call:
```

```
##  randomForest(formula = salary ~ ., data = salaries, importance = TRUE,      mtry = 5)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 571054059
##                     % Var explained: 37.6
```

The MSE is given by 571054059. This is lower than the MSE in the previous question.

Look at the variable importance

```
varImpPlot(salarybag)
```



The most important variable, as can be seen from the above variable importance plots is rank. This means that the majority of the variation in salary can be explained by the rank of the professor.

h)

```
set.seed(474)
salaryRF <- randomForest(salary~., data = salaries, importance = TRUE)
salaryRF
```

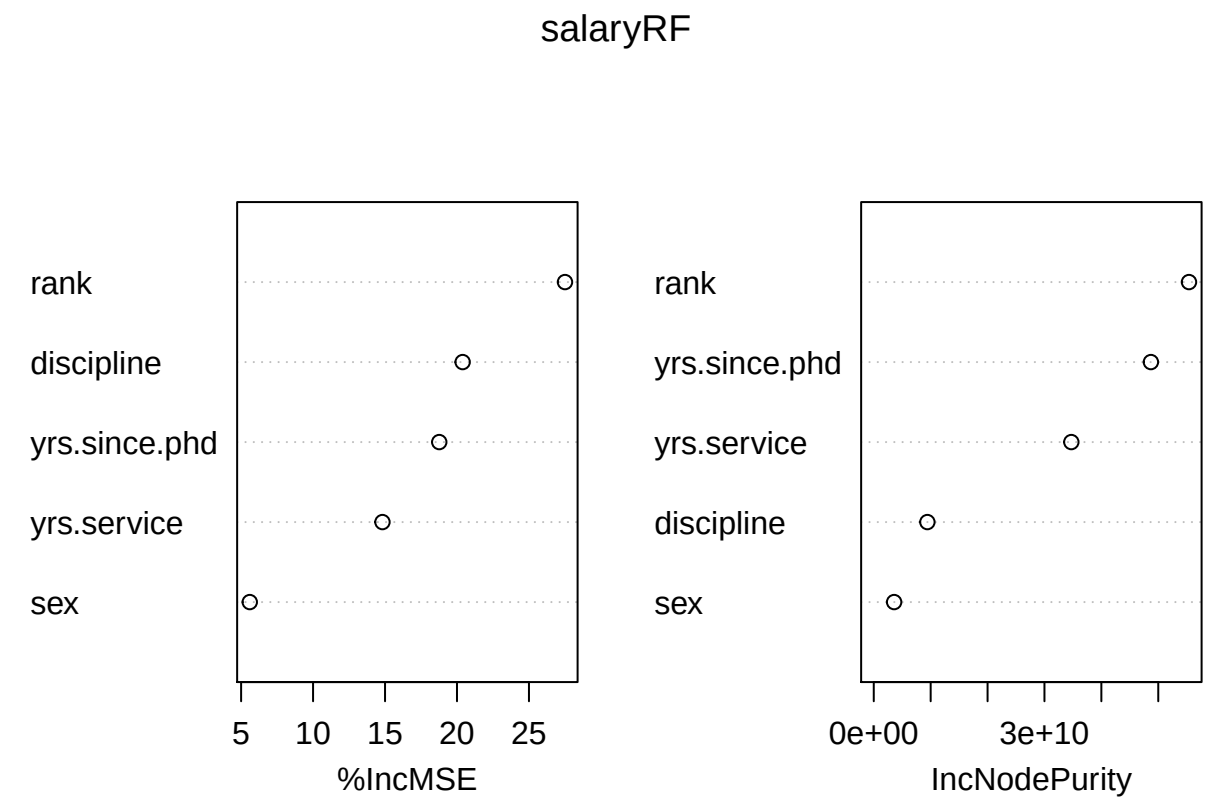
```
##
## Call:
##  randomForest(formula = salary ~ ., data = salaries, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 534252850
##                     % Var explained: 41.62
```

The MSE of the random forest model is 534252850. This is lower than both of the two previous models.

Look at the variable importance

```
varImpPlot(salaryRF)
```



The most important variable, as can be seen from the above variable importance plots is still rank. This means that the majority of the variation in salary can be explained by the rank of the professor.

For the above, recall that the only difference between bagging trees and random forests is that we randomly remove predictors from consideration at each binary split (random forest). This means that our trees have individually less information to grow from, which leads to a larger variety of trees and therefore less correlationa cross trees and therefore more stability and better predictions on average.

The best mtry input is the square root of the number of predictor variables (5). Hence I will try random forests with 2 predictor variables.

```
set.seed(474)
salaryRF <- randomForest(salary~., data = salaries, importance = TRUE, mtry=2)
salaryRF
```

```
##
## Call:
##  randomForest(formula = salary ~ ., data = salaries, importance = TRUE,      mtry = 2)
##                Type of random forest: regression
```

```
##                       Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 505203490
##                     % Var explained: 44.79
```

Note that the MSE for random forests with two splits provides the lowest MSE out of all previously investigated models.

Q2)

The second MSE value (139.8054) is lower because it is using the same data for test and train. The carrun model was developed on the cars dataset. However, we are also trying to predict using the same cars data agian. Therefore, the predictions will be close to the model and the mean squared error is less, as expected.

If new data was used, it would be expected that the MSE would be closer to that of the previous MSEs.

Q3

```
data <- cars
head(data)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

Since the cars data set only has two variables (speed and distance), the predictor variable in this case is speed and the response variable is distance.
If we had more than one response variable, we would be able to conclude that "random forests" was being performed because we would be splitting on a subset of the data. However, since there is only one predictor variable in this dataset, we cannot conclude if "random forests" or "bagging" is being used. Random Forest and bagging converge to the same algorithm when only one predictor variable is available.

Random Forests randomly removes a number of the predictors from consideration at each split. This effectively decreases the dependency (or correlation) across the trees, decreasing the variabce of the averaged model and therefore increasing stability. With only one predictor, Random Forests is not able to randly remove variables from consideration and therefore has no advantage over bagging.

Q4

Loading in the data

```
wineData <- read.csv("winequality-red.csv", header=TRUE)
#head(wineData)
dim(wineData)
```

```
## [1] 1599   12
```

```
attach(wineData)
```

First I will set up a testing and training set for the data. Since there are 1599 observations, and I am choosing to split approximately 70-30, I will use 1120 points for my training set and 479 observations for my testing

set.

```
set.seed(1010101)
#Setting up the training and testing set
trainindex <- sample(1:nrow(wineData), 1120)
winetrain <- wineData[trainindex, ]
winetest <- wineData[-trainindex, ]
```

Linear Models

Here I am fitting a multiple linear model.

```
set.seed(6573)
linearmod <- lm(quality~., data = wineData)
summary(linearmod)
```

```
##
## Call:
## lm(formula = quality ~ ., data = wineData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.68911 -0.36652 -0.04699  0.45202  2.02498
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.197e+01  2.119e+01   1.036   0.3002
## fixed.acidity         2.499e-02  2.595e-02   0.963   0.3357
## volatile.acidity     -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
## citric.acid          -1.826e-01  1.472e-01  -1.240   0.2150
## residual.sugar        1.633e-02  1.500e-02   1.089   0.2765
## chlorides            -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
## free.sulfur.dioxide   4.361e-03  2.171e-03   2.009   0.0447 *
## total.sulfur.dioxide -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
## density              -1.788e+01  2.163e+01  -0.827   0.4086
## pH                   -4.137e-01  1.916e-01  -2.159   0.0310 *
## sulphates             9.163e-01  1.143e-01   8.014 2.13e-15 ***
## alcohol               2.762e-01  2.648e-02  10.429  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.648 on 1587 degrees of freedom
## Multiple R-squared:  0.3606, Adjusted R-squared:  0.3561
## F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16
```

I will now apply backwards selection to determine what variables to include in the linear model.

```
set.seed(82989)
linearmod <- lm(quality~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar + chlorides + f
#summary(linearmod)
linearmod <- lm(quality~volatile.acidity + citric.acid + residual.sugar + chlorides + free.sulfur.dioxi
#summary(linearmod)
linearmod <- lm(quality~volatile.acidity + citric.acid + chlorides + free.sulfur.dioxide + total.sulfur
#summary(linearmod)
linearmod <- lm(quality~volatile.acidity + chlorides + free.sulfur.dioxide + total.sulfur.dioxide + pH +
summary(linearmod)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + pH + sulphates + alcohol, data = wineData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.68918 -0.36757 -0.04653  0.46081  2.02954
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.4300987  0.4029168  10.995  < 2e-16 ***
## volatile.acidity     -1.0127527  0.1008429 -10.043  < 2e-16 ***
## chlorides            -2.0178138  0.3975417  -5.076 4.31e-07 ***
## free.sulfur.dioxide   0.0050774  0.0021255   2.389    0.017 *
## total.sulfur.dioxide -0.0034822  0.0006868  -5.070 4.43e-07 ***
## pH                   -0.4826614  0.1175581  -4.106 4.23e-05 ***
## sulphates             0.8826651  0.1099084   8.031 1.86e-15 ***
## alcohol               0.2893028  0.0167958  17.225  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6477 on 1591 degrees of freedom
## Multiple R-squared:  0.3595, Adjusted R-squared:  0.3567
## F-statistic: 127.6 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Without printing out all of the intermediate steps, the highest p-value initially belonged to density, and hence density was removed from the model. Then fixed.acidity, residual.sugar, and citric acid. The summary of the resulting multiple linear regression model, containing the predictor variables volatile.acidity, chlorides free.sulfur.dioxide, total.sulfur.dioxide, pH, sulphates, and alcohol, is printed above.

I will now perform cross validation on a multiple linear model with the predictor variables listed above.

```
set.seed(7861)
cvlm <- list()
msecv <- NA
for(i in 1:nrow(wineData)){
  #Fit the linear model
  cvlm[[i]] <- lm(quality[-i] ~ volatile.acidity[-i] + chlorides[-i] + free.sulfur.dioxide[-i] + total.s

  # Calculate MSE for ith model
  msecv[i] <- (predict(cvlm[[i]], newdata = data.frame(volatile.acidity[i] + chlorides[i] + free.sulfur
  #msecv[i]
}
#output mean of MSE
mean(msecv)
```

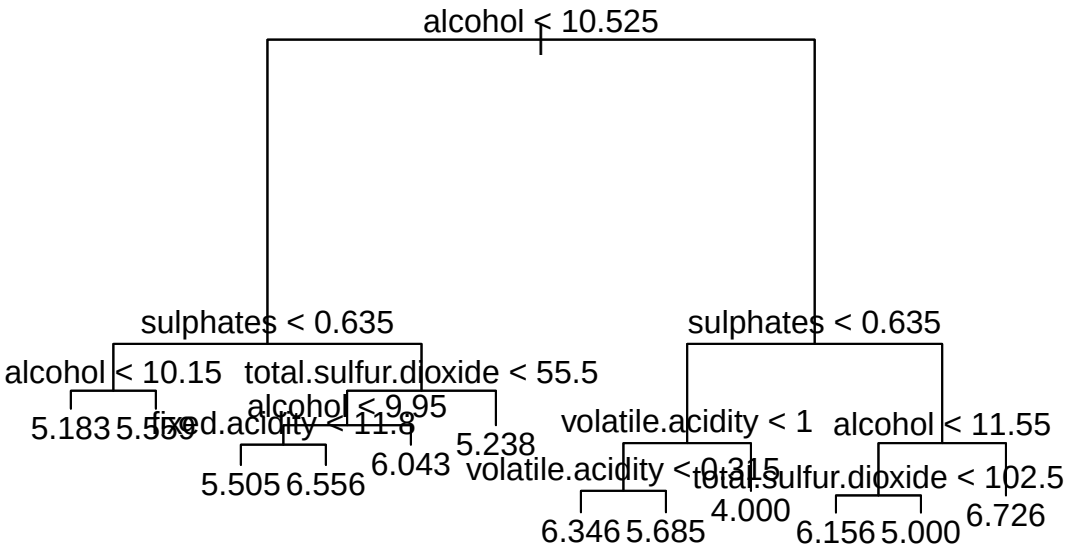
```
## [1] 1.025766
```

The MSE from the multiple linear model, after applying backwards selection, is 1.025766.

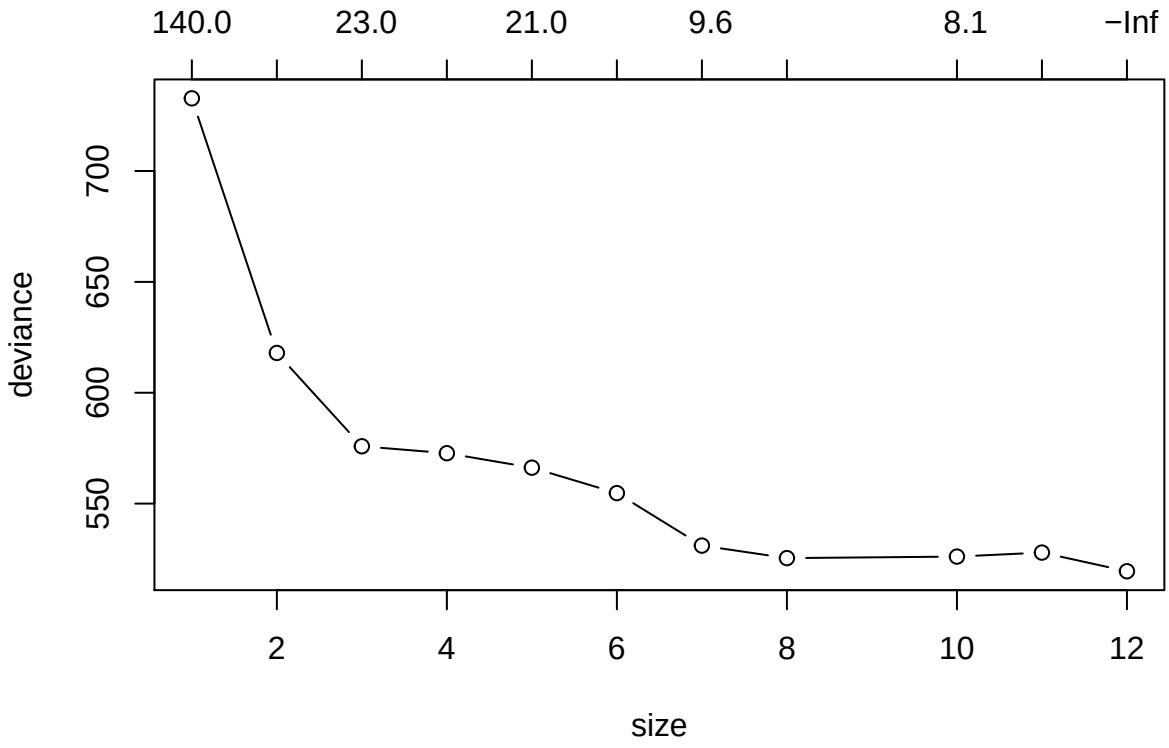
Trees

```
set.seed(64213)
library(tree)
```

```
winetree <- tree(quality~., data = winetrain)
plot(winetree)
text(winetree, pretty=0)
```




```
winetreeCV <- cv.tree(winetree, FUN = prune.tree, K = 20)
plot(winetreeCV, type = "b")
```




```
#How many terminal nodes does cross-validation suggest?
winetreeCV$dev
```

```
##  [1] 519.4974 527.9114 526.1054 525.4220 531.0681 554.7299 566.2105
##  [8] 572.7325 575.8601 617.9485 732.7742
```

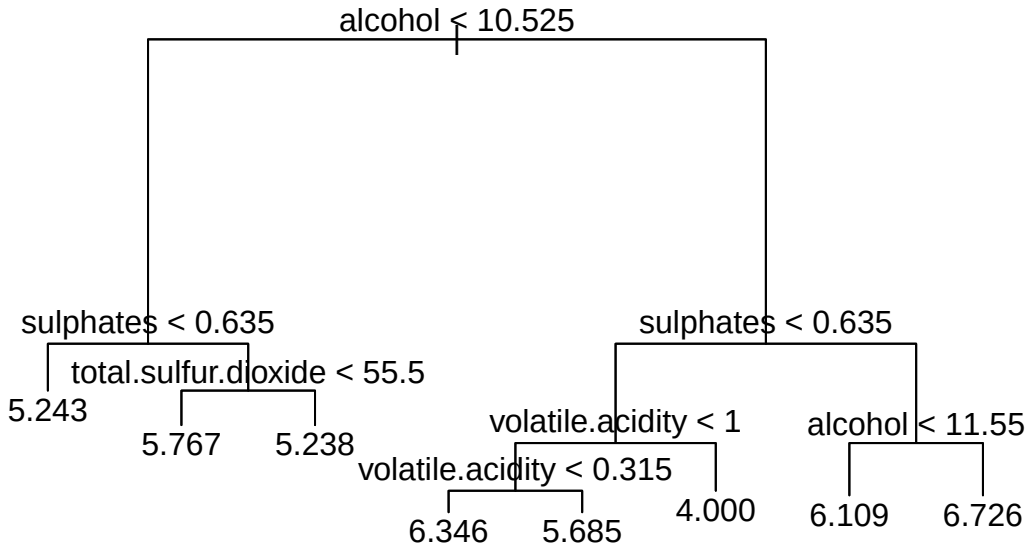
```
winetreeCV$size
```

```
##  [1] 12 11 10  8  7  6  5  4  3  2  1
```

```
which.min(winetreeCV$dev)
```

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

```
#Cross validation suggests 8 terminal nodes.
p.winetreeCV <- prune.tree(winetree, best=8)
plot(p.winetreeCV)
text(p.winetreeCV, pretty = 0)
```




```
summary(p.winetreeCV)
```

```
##
## Regression tree:
## snip.tree(tree = winetree, nodes = c(14L, 4L, 10L))
## Variables actually used in tree construction:
## [1] "alcohol"              "sulphates"            "total.sulfur.dioxide"
## [4] "volatile.acidity"
## Number of terminal nodes:  8
## Residual mean deviance:  0.412 = 458.1 / 1112
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.7670 -0.2430 -0.1088  0.0000  0.3151  2.2330
```

The above prints a simple, cross validated tree on the wine data set. The MSE on the cross-validated tree is given by 0.4054.

Random Forests

```
set.seed(1728)
wineRF <- randomForest(quality~., data = wineData, importance = TRUE)
wineRF
```

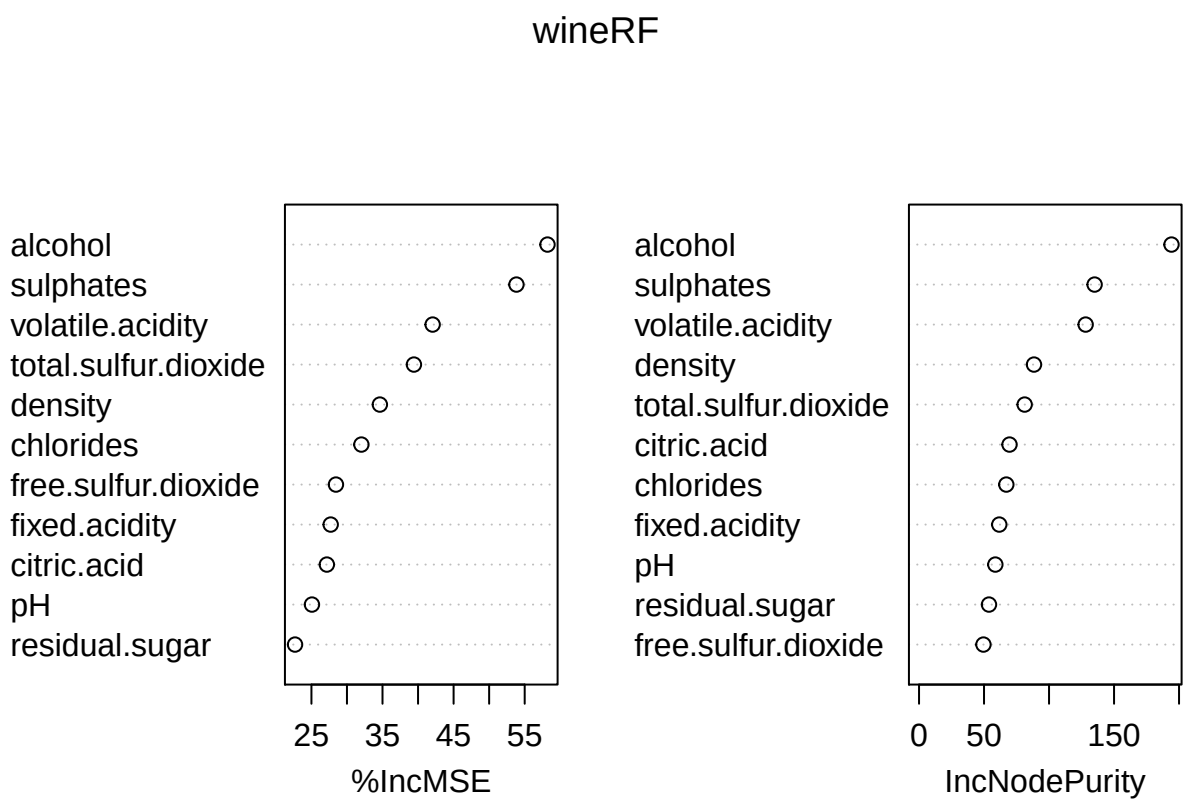
```
##
## Call:
##  randomForest(formula = quality ~ ., data = wineData, importance = TRUE)
```

```
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.3163553
##                     % Var explained: 51.46
```

Random Forest uses out-of-bag performance that is very similar to cross validation. Therefore, I did not perform cross validation manually on this algorithm. The long run MSE for random forests is 0.3163553

We can also look at the importance from the RF.

```
varImpPlot(wineRF)
```



The variable importance plot on random forests shoes that alcohol is the most important predictor variable that is used to predict quality of red wines.

Boosting Again

```
set.seed(237856)
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.5.3
```

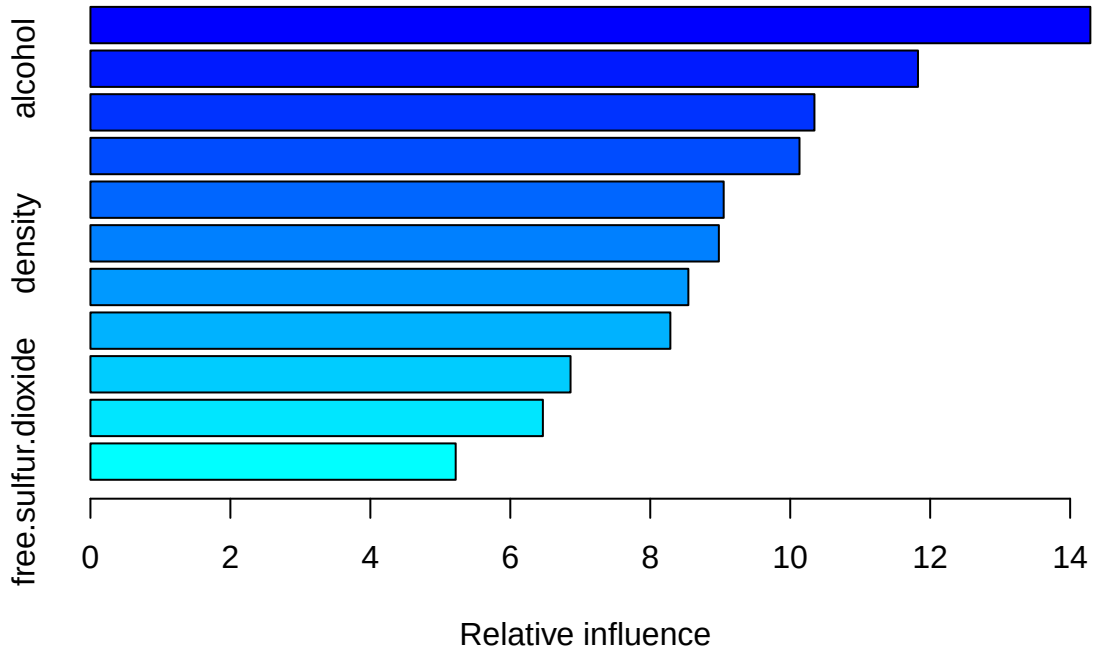
```
## Loaded gbm 2.1.5
```

```
wineData <- read.csv("winequality-red.csv", header = TRUE)
wine.boost = gbm(quality ~., data = winetrain, distribution = "multinomial", n.trees = 5000, interactio
```

```
wine.boost
```

```
## gbm(formula = quality ~ ., distribution = "multinomial", data = winetrain,
##     n.trees = 5000, interaction.depth = 1, cv.folds = 5)
## A gradient boosted model with multinomial loss function.
## 5000 iterations were performed.
## The best cross-validation iteration was 55.
## There were 11 predictors of which 11 had non-zero influence.
```

```
summary(wine.boost) #Summary gives a table of Variable Importance and a plot of Variable Importance
```




```
##                                     var   rel.inf
## volatile.acidity         volatile.acidity 14.289000
## alcohol                           alcohol 11.825806
## sulphates                       sulphates 10.345989
## chlorides                       chlorides 10.132758
## pH                                     pH  9.049130
## density                           density  8.980217
## total.sulfur.dioxide total.sulfur.dioxide  8.544244
## fixed.acidity               fixed.acidity  8.286954
## citric.acid                   citric.acid  6.860842
## residual.sugar             residual.sugar  6.465625
## free.sulfur.dioxide   free.sulfur.dioxide  5.219435
```

```
#get minimum MSE
min(wine.boost$cv.error)
```

```
## [1] 0.96118
```

The above Boosted Model is a Gradient Boosted Model. I generated 5000 trees with an interaction depth of 1 (so each tree has only one split). We can look at the feature importance plot and list to determine the most important variable volatile.acidity, followed by alcohol. I set the number of cross validation folds to be equal to 5. The minimum MSE for boosting is 0.954818 which is the best case scenario MSE.

LASSO

```
set.seed(4787)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.5.3
```

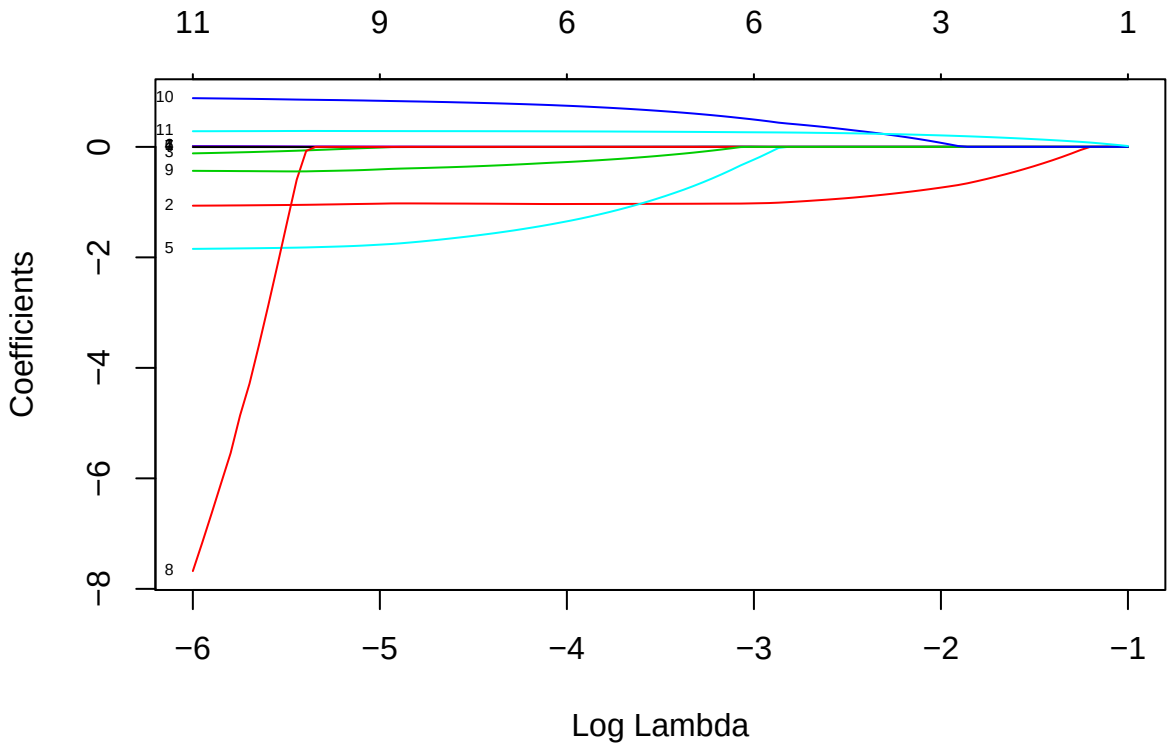
```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

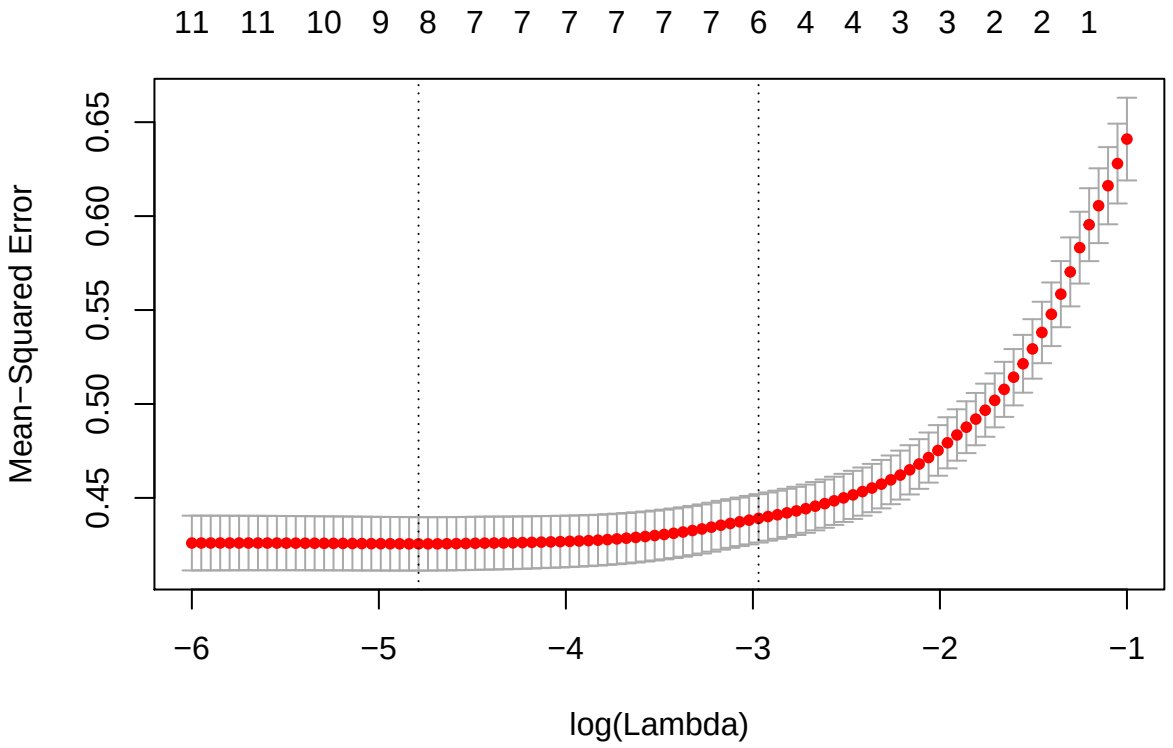
```
## Warning: package 'foreach' was built under R version 3.5.3
```

```
## Loaded glmnet 2.0-16
```

```
wineData <- read.csv("winequality-red.csv", header = TRUE)
x <- as.matrix(wineData[,-12])
grid <- exp(seq(-6, -1, length=100))
y <- wineData$quality
lasim <- cv.glmnet(x,y,alpha=1,lambda = grid)
plot(lasim$glmnet.fit,label=TRUE,xvar="lambda")
```



```
plot(lasim)
```




```
lammin <- lasim$lambda.min
lammin
```

```
## [1] 0.008330109
```

```
lam1se <- lasim$lambda.1se
lam1se
```

```
## [1] 0.05131886
```

```
lasimmin <- glmnet(x,y,alpha=1,lambda=lammin)
lasim1se <- glmnet(x,y,alpha=1,lambda=lam1se)
coef(lasimmin)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)            4.167098267
## fixed.acidity          .
## volatile.acidity      -1.023965287
## citric.acid            .
## residual.sugar         0.001457507
## chlorides             -1.715766626
## free.sulfur.dioxide    0.002488586
## total.sulfur.dioxide -0.002690497
## density                .
## pH                    -0.384104018
```

```
## sulphates             0.820910285
## alcohol               0.285217009
```

```
coef(lasim1se)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)           3.1590297223
## fixed.acidity         0.0010835904
## volatile.acidity     -1.0200658028
## citric.acid           .
## residual.sugar        .
## chlorides            -0.1839647551
## free.sulfur.dioxide   .
## total.sulfur.dioxide -0.0008916136
## density               .
## pH                    .
## sulphates             0.4829678160
## alcohol               0.2634607296
```

```
#Finding the MSE
lasim$cvm.mse
```

```
## NULL
```

```
msepls = lasim$cvm[lasim$lambda == lammin]
msepls
```

```
## [1] 0.4255299
```

```
#coeftab <- cbind(c(20.00, 3.00, -2.00), #coef(linmod), coef(rrsimmin), coef(lasimmin))
#colnames(coeftab) <- c("TRUE", "LM", "RidgeRegMin", "LassoMin")
#round(coeftab, 2)
```

Using a cross-validated Lasso, the MSE is 0.425529.

Comparing The MSEs

The multiple linear model performed the worst with an MSE of 1.025766. This model was likely too simple for the data. The second worst MSE belonged to the boosting model which hadd an MSE of 0.954818. Lasso had an MSE of 0.425529 while the cross-validated tree had a MSE of 0.4054. Random forest provided the lowest MSE with a value of 0.3163553.

Therefore, random forests is most likely to provide the lowest MSE in the long-run.

If I were consulting a company on this data set, I would choose to the cross-validated tree as the model to use. Even though this does not have as low of an MSE as the random forest, it is more interpretable. Random forest is pretty much a black box, but the tree is easy to look at the plot and be able to interpret what is going on. I believe this would benefit the company.