Homework #5

Justin Robinette

September 25, 2018

No collaborators for any problem

Problem #1, Part A: The **BostonHousing** dataset reported by Harrison and Rubinfeld (1978) is available as data.frame package **mlbench** (Leisch and Dimitriadou, 2009). The goal here is to predict the median value of owner-occupied homes in USD 1000's (medv variable) based on other predictors in the dataset. Use this dataset to do the following.

Construct a regression tree using rpart(). The following need to be included in your discussion:
- How many nodes did your tree have?
- Did you prune the tree?
- Did it decrease the number of nodes?
- What is the prediction error (calculate MSE)?
- Provide the predicted vs. observed plot.
- Plot the final tree.

Results: As we see from *Figure 1.1*, our initial regression tree (from **rpart()**) has 9 nodes. *Figure 1.2* shows the tree, as created by the model. *I've included a plot using* ***ggdendrogram*** *per homework instructions.*

As this is my first time working with decision trees, I did go through the process from the text to prune the tree. *Figure 1.3* shows the Relative Error and Complexity Parameters by number of nodes. As we can see, the optimal number of nodes is 9. *Figure 1.4* summarizes the plot in *Figure 1.3*. Again, we see that the optimal number of nodes is 9. *There is an analogous ggplot per homework instructions*.

For the practice, as I mentioned above, I pruned the tree to get the number of nodes corresponding to the best 'CP' value. This returns an identical tree to our original, as we cann see in *Figure 1.5*. *I've included a plot using* ***ggdendrogram*** *per homework instructions.*

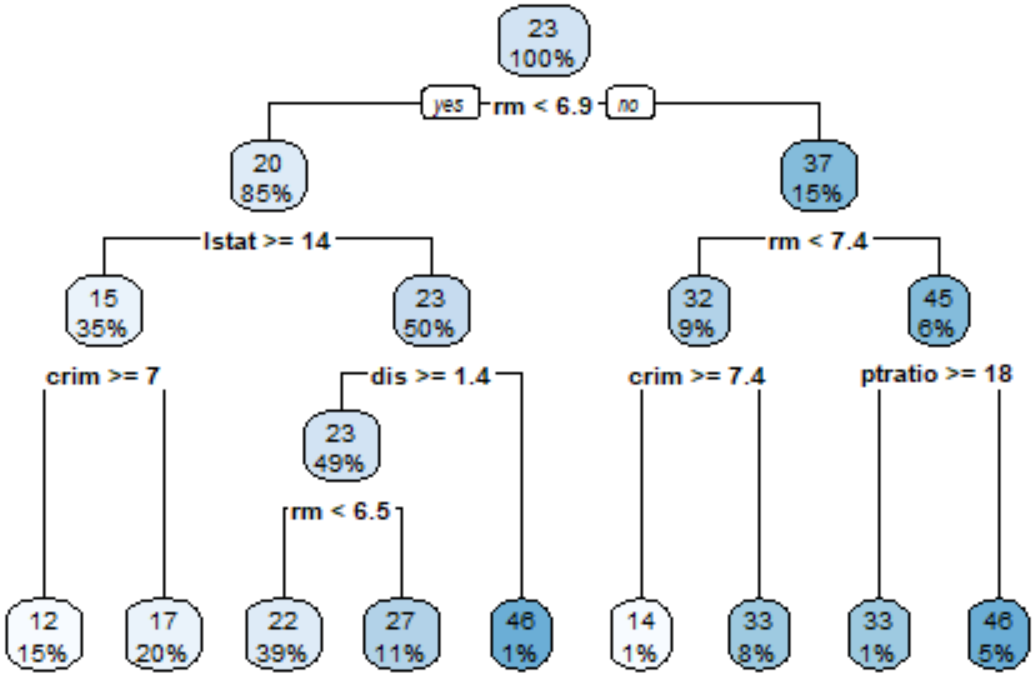
The predicted error (MSE) is **12.71556**, as seen in *Figure 1.6*.

Lastly, I've provided a plot showing the relationship between predicted 'Median Value' and the observed values of the same variable. As we see from this relationship, plotted in *Figure 1.7*, there is a considerable amount of variation (error) in the prediction success of the model. *A comparable base R plot is included*.

Figure 1.1: Number of Nodes in Original Model

Nodes
9

Figure 1.2: Original Tree



Original Tree - ggplot

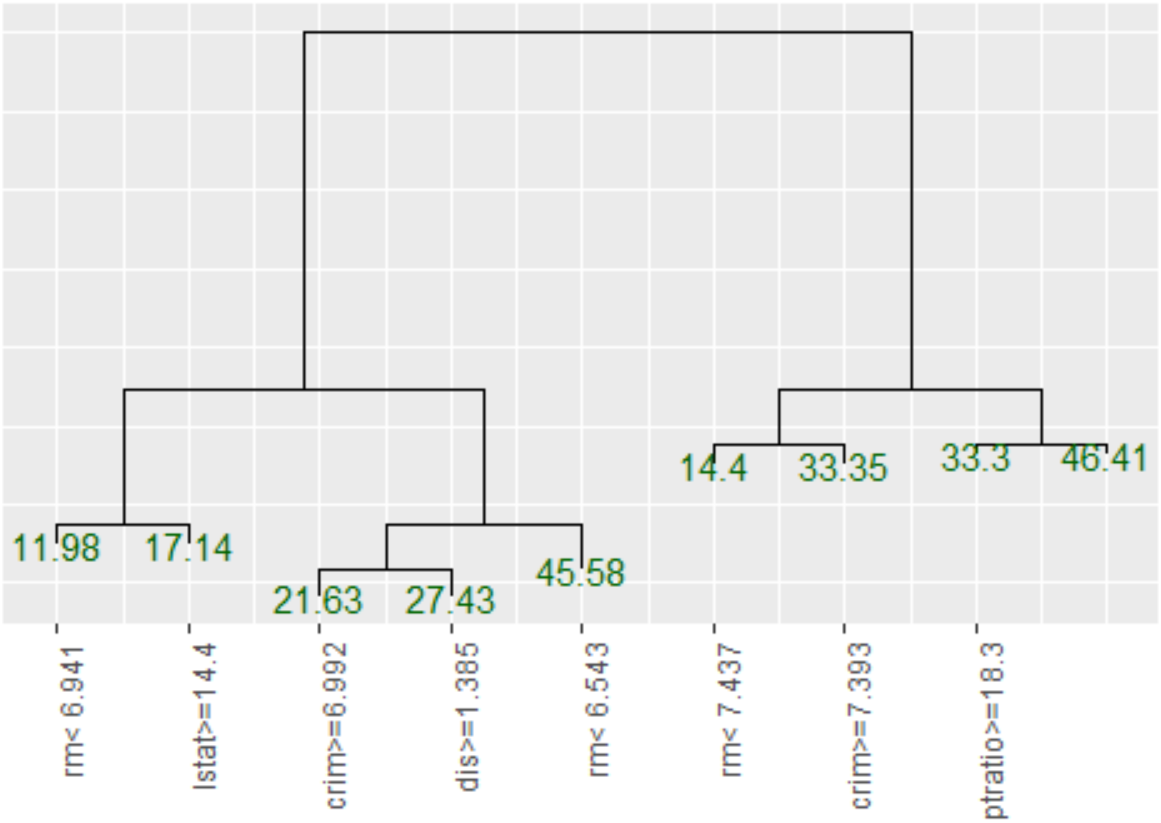
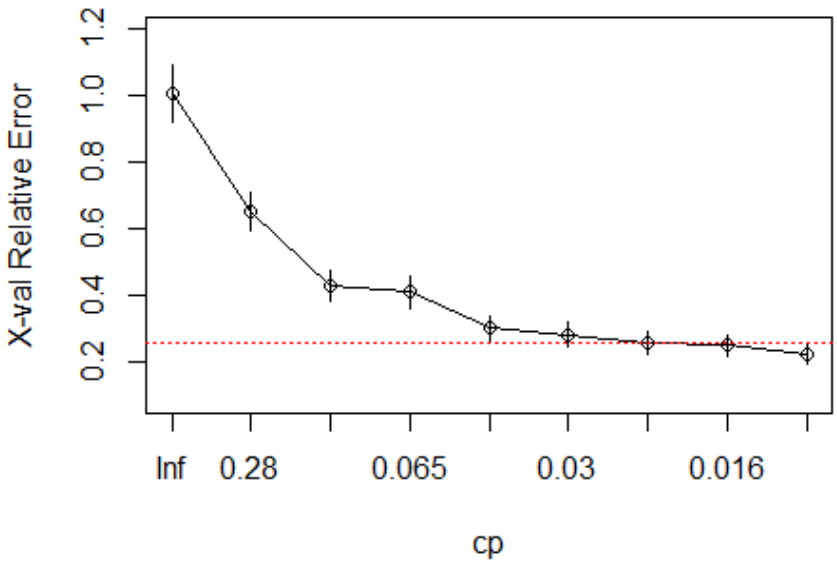


Figure 1.3: CP and 'X-Error' by # Nodes



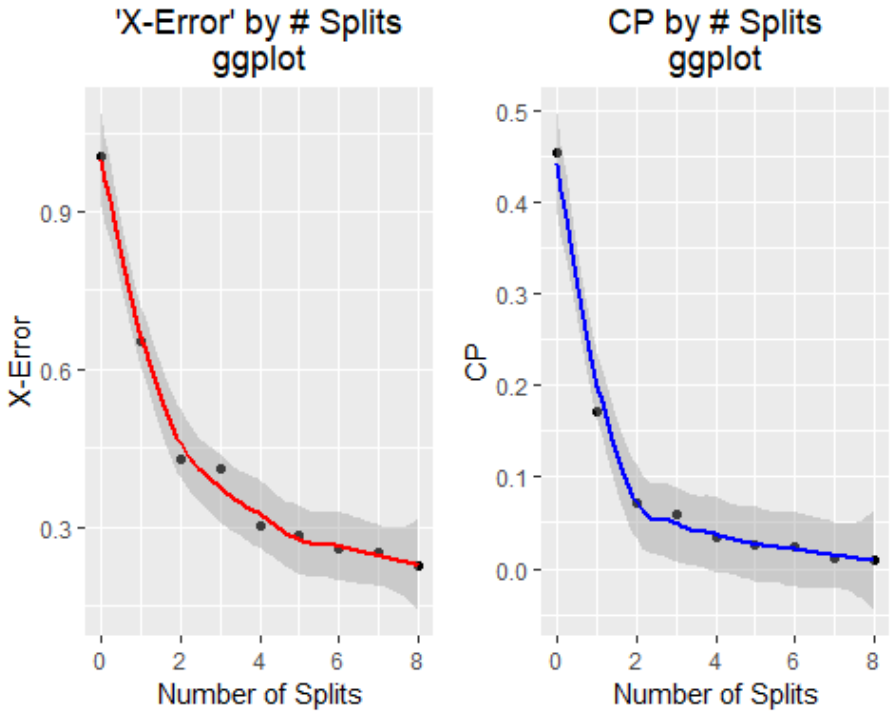
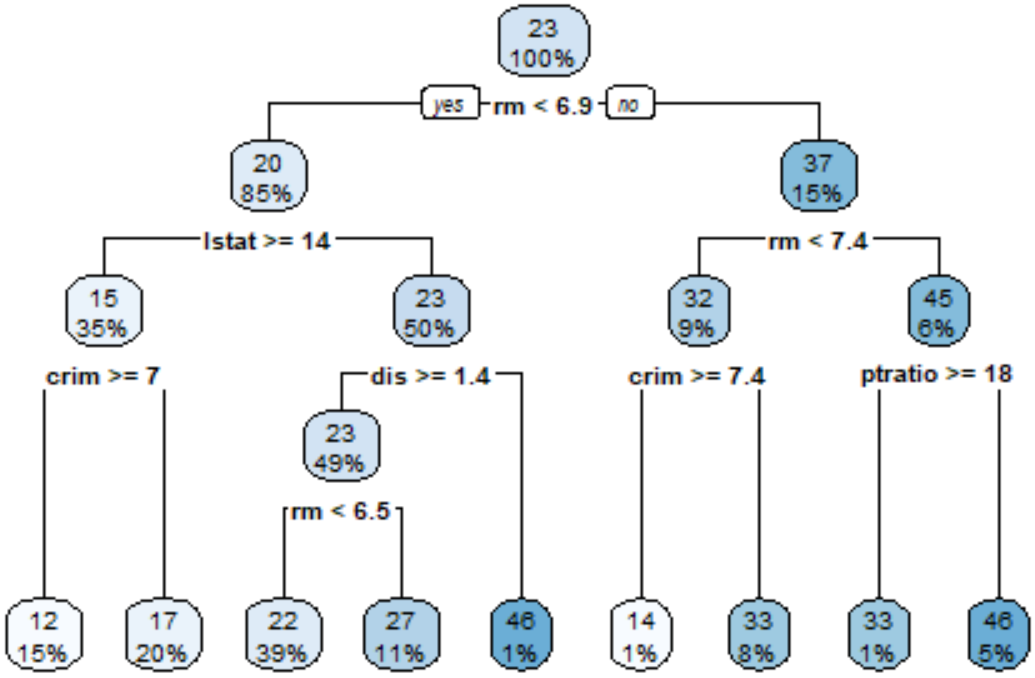


Figure 1.4: CP by Node Number

	Complexity Parameter
1	**0.4527442**
2	**0.1711724**
3	**0.0716578**
4	**0.0590015**
5	**0.0337559**
6	**0.0266130**
7	**0.0235724**
8	**0.0108593**
9	**0.0100000**

Figure 1.5: Pruned Tree (Identical to Original)



Pruned Tree (Identical to Original) - ggplot

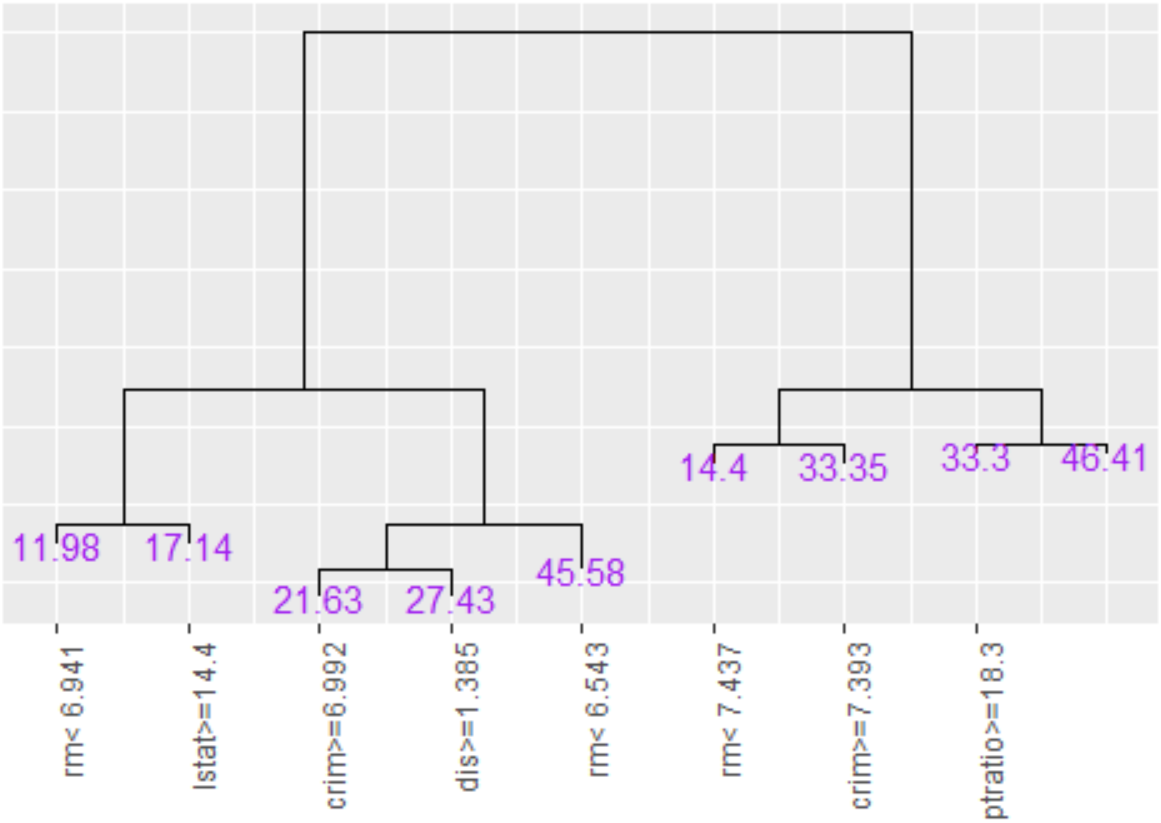
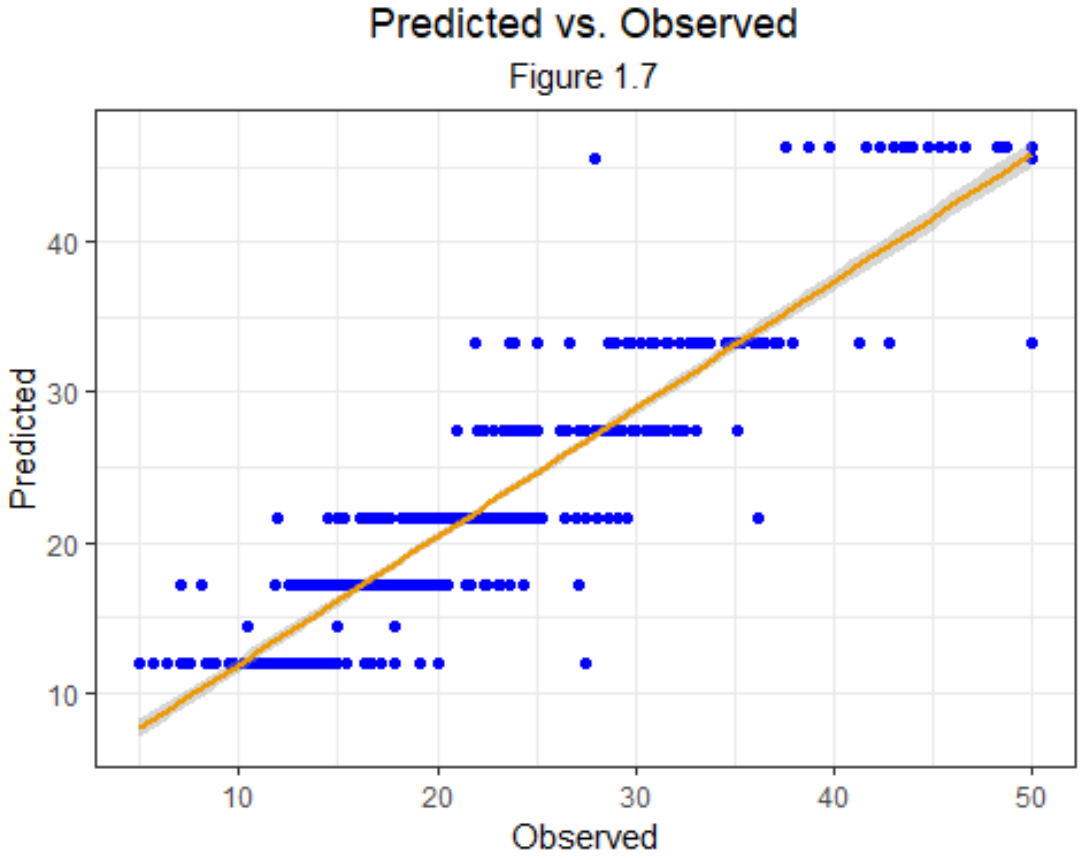
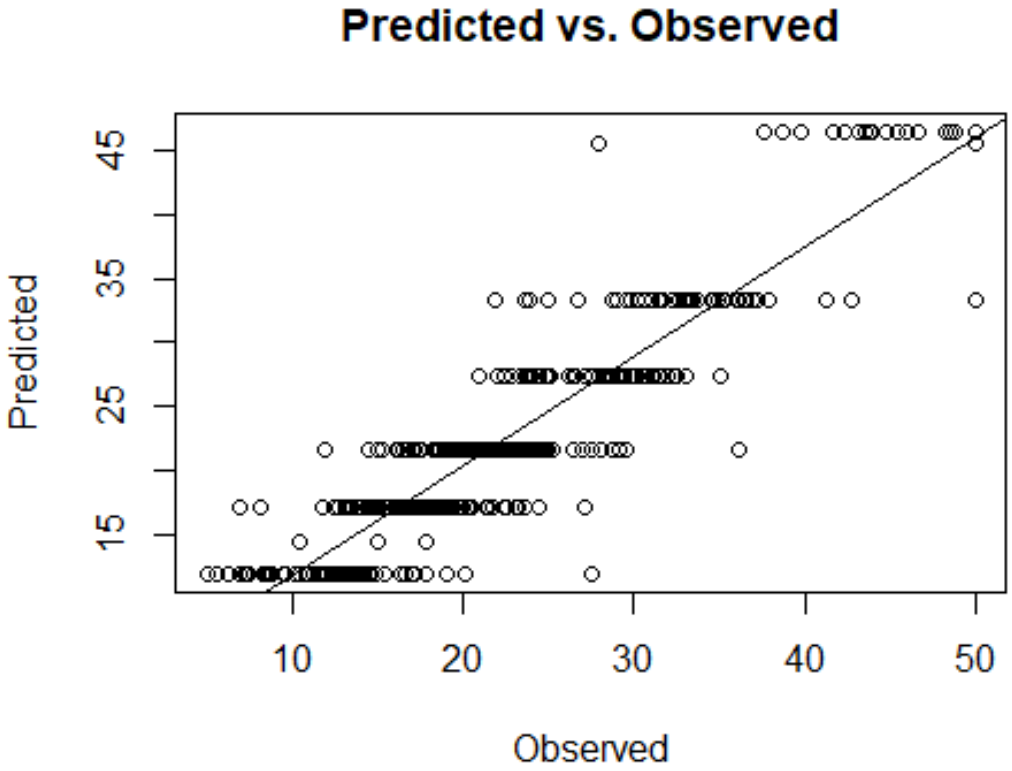


Figure 1.6: Predicted Error of Optimal Model

Model MSE
12.71556





Problem #1, Part B: Perform bagging with 50 trees. Report the prediction error (MSE). Provide the predicted vs. observed plot.

Results: Here I've performed bagging with 50 trees. The first summary I wanted to take a look at was the variable chosen as the root node. As we see in *Figure 1.8*, the root node is not consistently one variable over the rest. This explains part of the reason why we have such a considerable amount of variation in *Figure 1.7* above.

Next, we take a look at *Figure 1.9* shows the predicted error (MSE) with bagging of **16.24467**. As we can see, this is a higher MSE than we saw with the model produced in part A of this exercise.

Lastly, in *Figure 1.10* we see the relationship between 'Predicted' and 'Observed' Median Values from *BostonHousing*. Similar to *Figure 1.7*, we see a high amount of variation between the predicted and actual values. As always, a similar base R plot is included for comparison.

Figure 1.8: Variable Frequency as Root of Tree[i]

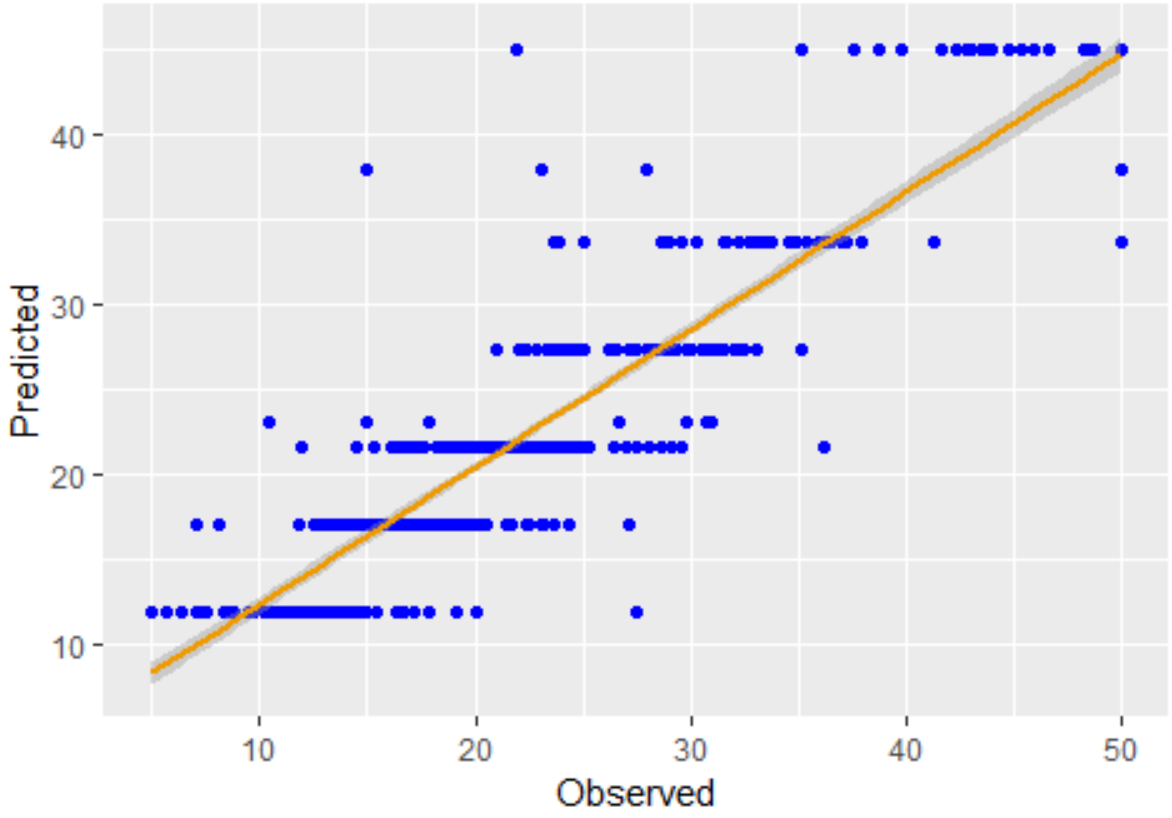
Root Node	Frequency
lstat	**19**
rm	**31**

Figure 1.9: Predicted Error of Bagging

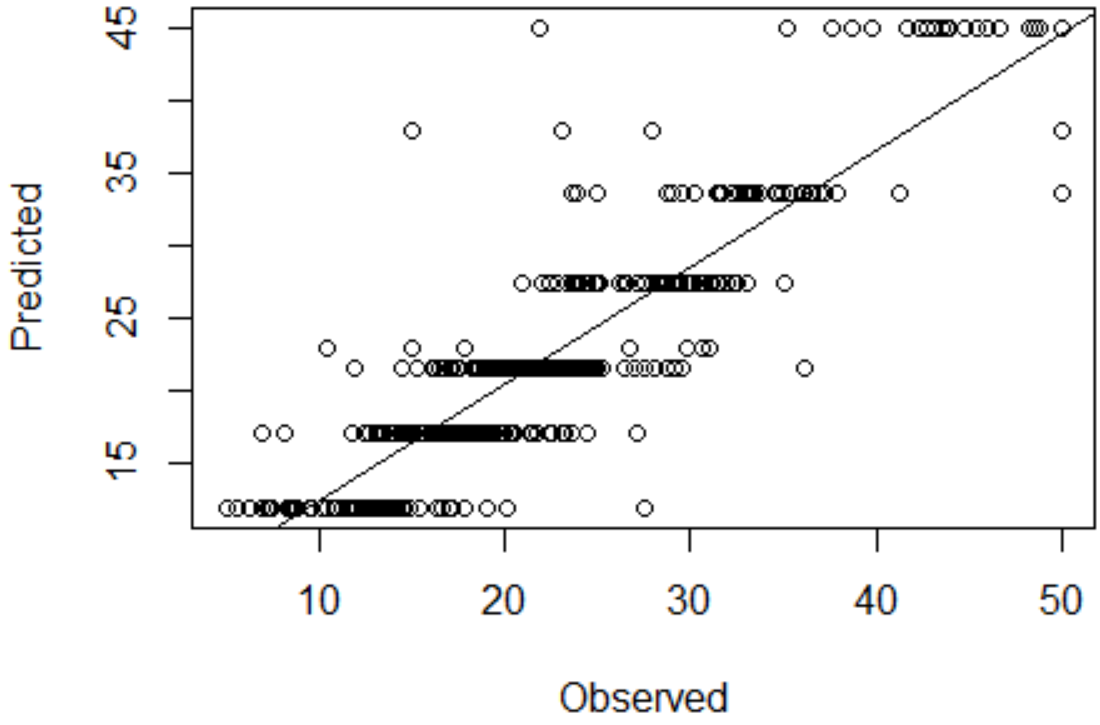
Bagging MSE
16.24467

Predicted vs. Observed - Bagging

Figure 1.10



Predicted vs. Observed - Bagging



Problem #1, Part C: Use randomForest() function in R to perform bagging. Report prediction error (MSE). What it the same as (b)? If they are different, what do you think caused it? Provide the predicted vs. observed plot.

Results: Here I used randomForest to perform bagging. In keeping with the previous exercise, I set ntree = 50. As we can see, from *Figure 1.11*, the predicted error (MSE) with this method is **10.96093**. This is an improvement from the bagging method used in part B (**16.24467**). I would think the difference from part B is due to the the way randomForest works which is to use a random sample of the variables and observations to build the trees. This difference in approach should explain why there is a different error rate in this instance.

Figure 1.12 shows that our relationship between 'Actual' and 'Predicted' median values has become less variable by using randomForest to perform bagging. Here we have a much more linear relationship than we've had in the prior 2 examples. A comparable base R plot is shown as well.

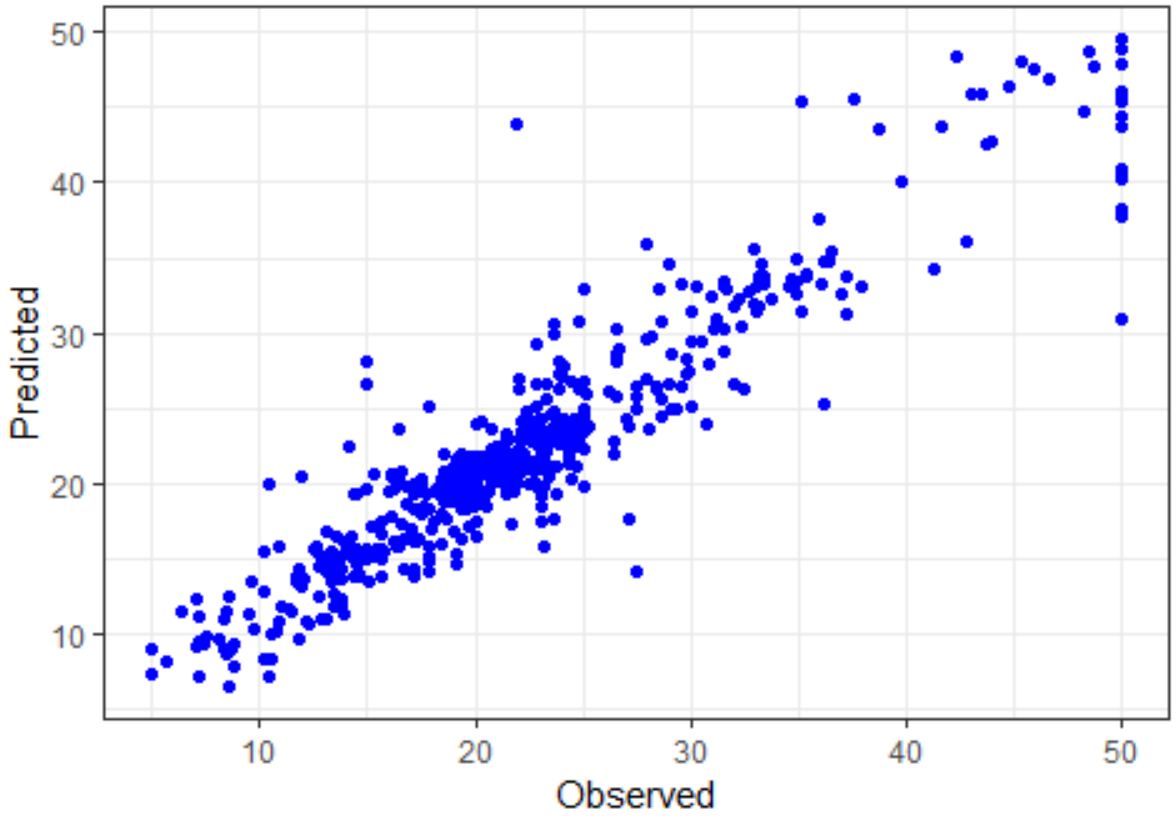
Figure 1.13 uses randomForest's **importance()** function to visually represent the importance of each Predictor Variable. As we see, after the first two variables, 'rm' and 'lstat', there is a huge drop off in importance. 'rm' stands for the average number of rooms per dwelling and 'lstat' shows the percentage of lower status of the population. I found the ranking system and difference in importance between the top 2 and remaining 11 predictor variables to be interesting here.

Figure 1.11: Predicted Error of Bagging

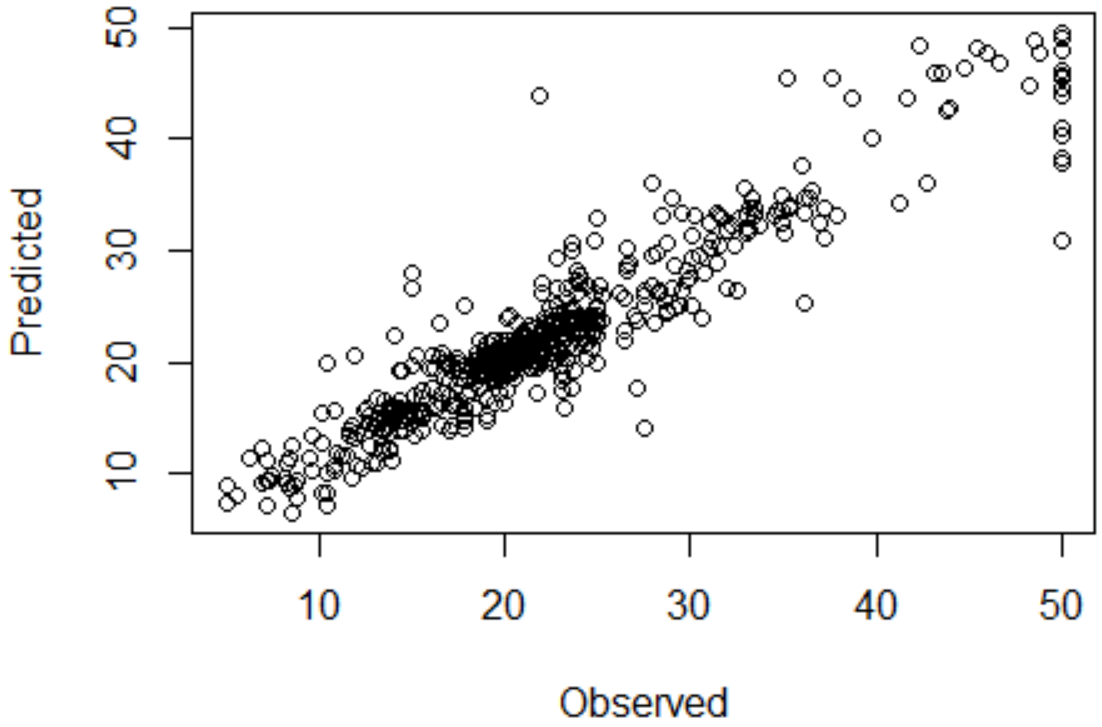
randomForest Bagging MSE
10.96093

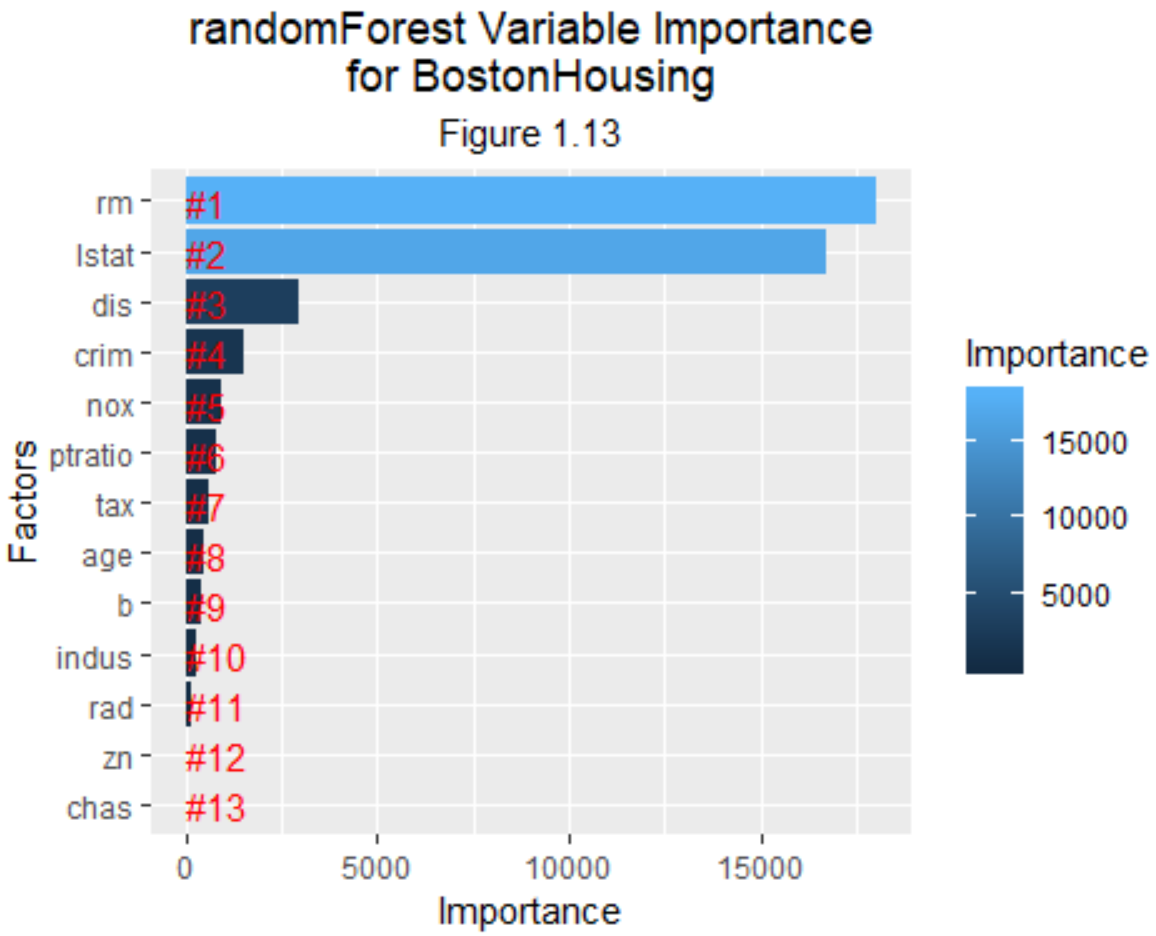
Predicted vs. Observed - randomForest Bagging

Figure 1.12



Predicted vs. Observed -randomForest Bagging





Problem #1, Part D: Use randomForest() function in R to perform random forest. Report the prediction error (MSE). Provide the predicted vs. observed plot. For this we do not need to change mtry.

Results: For this exercise we've used randomForest and calculated MSE while using the default 'ntree' and 'mtry' value for randomForest. This default 'mtry' is calculated with the following formula (rounded down):

$$\sqrt{(}\boldsymbol{n_{predictorvariables}})$$

$$\boldsymbol{mtry} = \sqrt{\mathbf{13}} = \mathbf{3.61} = \mathbf{3}$$

As we see from *Figure 1.14*, the MSE is lower using the default 'mtry' value (3) than it was using an 'mtry' value of 50. *Figure 1.15* shows a similarly close relationship between actual and predicted median values that we saw in *Figure 1.12* above.

Again, I've included a plot showing the importance of each predictor variable. *Figure 1.16* summarizes this data. Similar to *Figure 1.13* above, 'rm' and 'lstat' are the two most important variables. Contrary to *Figure 1.13*, *Figure 1.16* below shows a higher importance placed on other variables - namely 'ptratio', 'nox', 'indus' and 'dis'. These variables correspond to the following: - pupil-teacher ratio by town - nitric oxides concentration (parts per 10 million) - proportion of non-retail business acres per town - weighted distances to five Boston employment centres

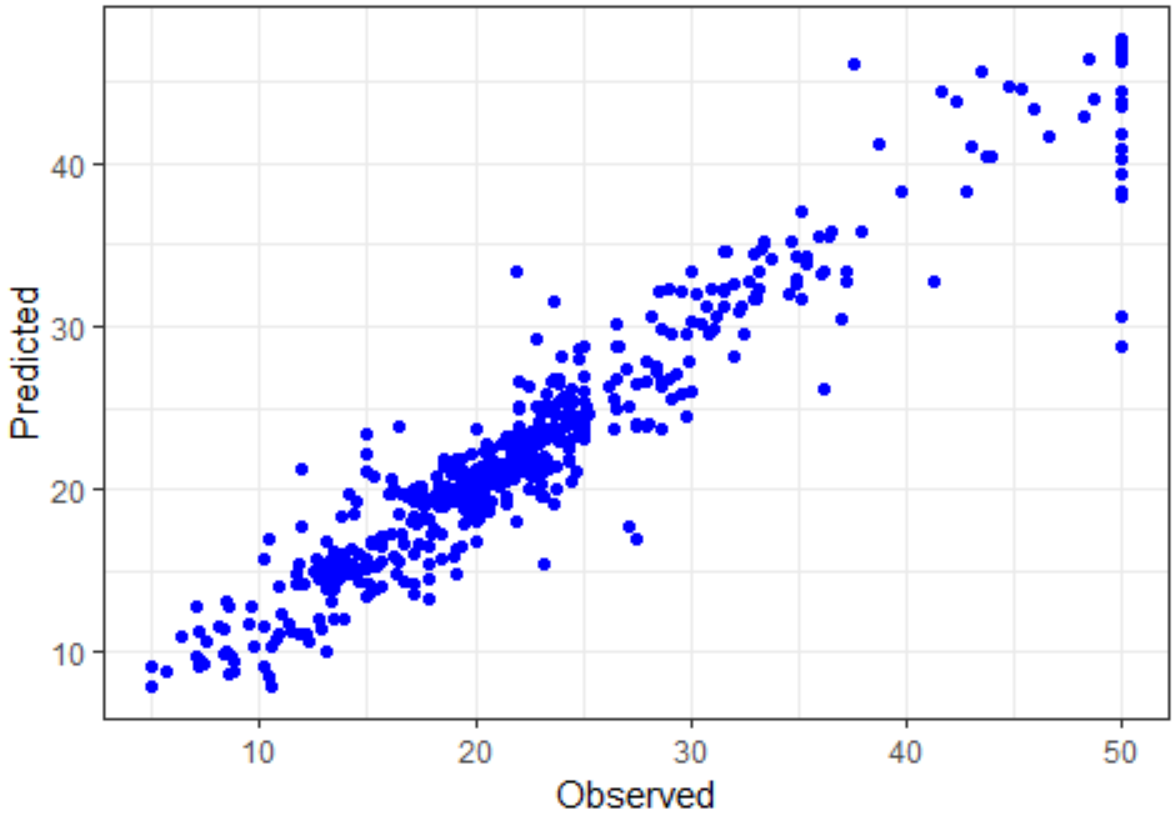
Based on the above descriptions, I find it easy to see why 'ptratio', 'indus', and 'dis' have a higher influence on median value. The relative importance of 'nox' is interesting and may warrant more independent research on my part.

Figure 1.14: Predicted Error of randomForest

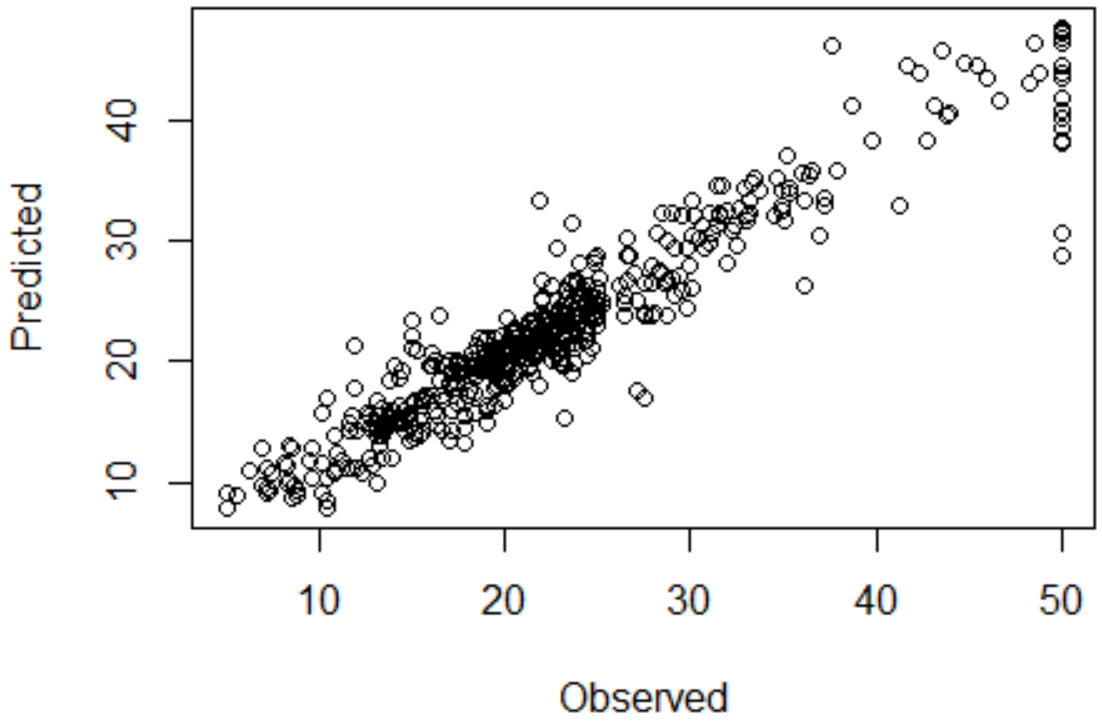
randomForest MSE
9.600825

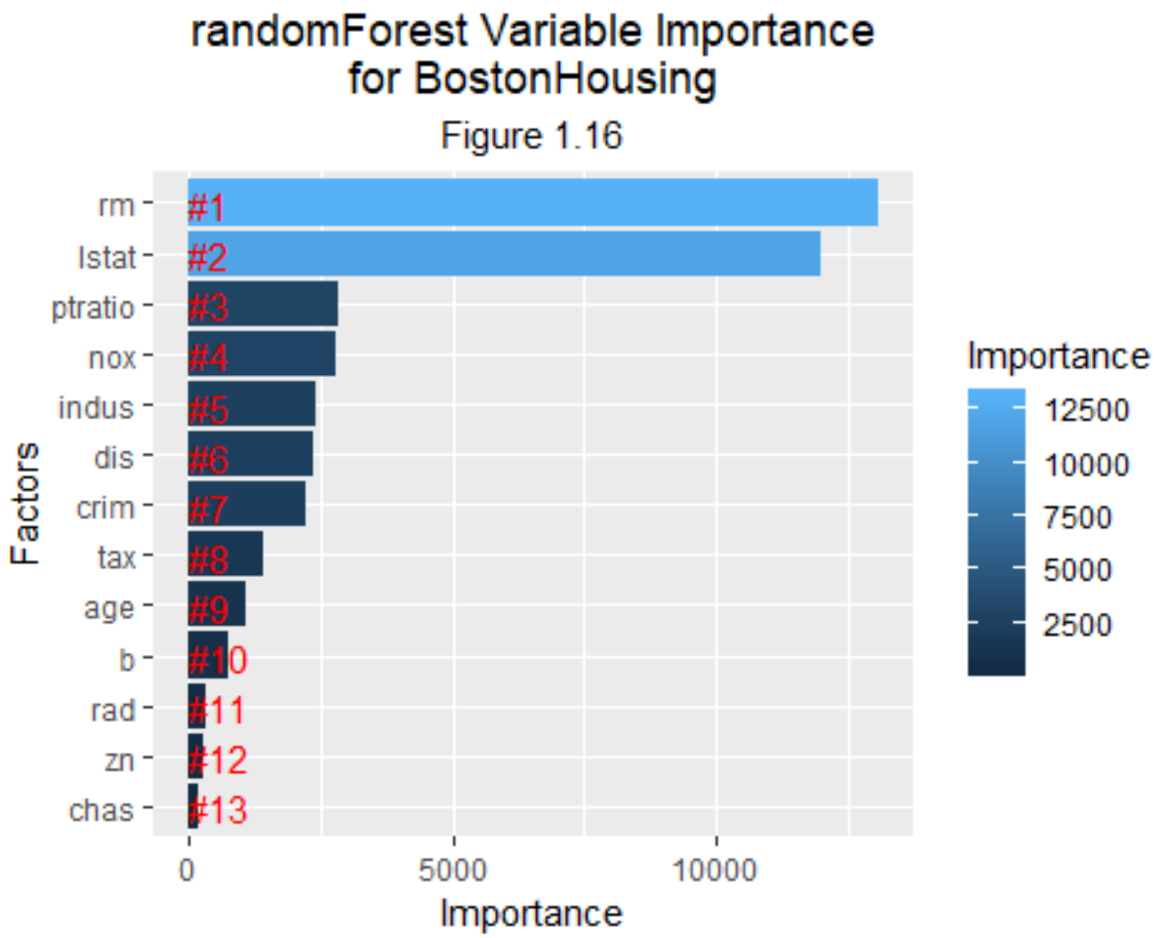
Predicted vs. Observed - randomForest

Figure 1.15



Predicted vs. Observed - randomForest





Problem #1, Part E: Provide a table containing each method and associated MSE. Which method is more accurate?

Results: *Figure 1.17* shows the summary of each method and its associated Mean Square Error ordered by accuracy. The most accurate method is the randomForest method with the default value used for 'mtry'. The least favorable method, as we can see, was the rpart bagging method with 50 trees.

Figure 1.17: Comparison of Error Rate by Method

Method	MSE
rF_default	**9.600825**
rF_13	**10.960931**
rpart	**12.715559**
rpart_bagging	**16.244674**

Problem #2, Part A: Consider the glacoma data (data = **"GlaucomaM"**, package = **"TH.data"**).

Build a logistic regression model. Note that most of the predictor variables are highly correlated. Hence, logistic regression model using the whole set of variables will not work here as it is sensitive to correlation.

glac_glm <- glm(Class ~ ., data = GlaucomaM, family = "binomial")
warning messages – variable selection needed

The solution is to select variables that seem to be important in predicting the response and using those in modeling process using GLM. One way to do this is by looking at the relationship between the response and predictors using graphical or numerical summaries - this tends to be tedious. Secondly, we can use a formal variable selection approach. The *step()* function will do this in R. Using *step*, choose any direction for variable selection and fit logistic regression model. Discuss the model and error rate.

Do not print out the summaries of every single model built using variable selection. That will end up being dozens of pages long and not worth reading through. Your discussion needs to include the direction you chose. You may only report on the final model, the summary of that model, and the error rate associated with that model.

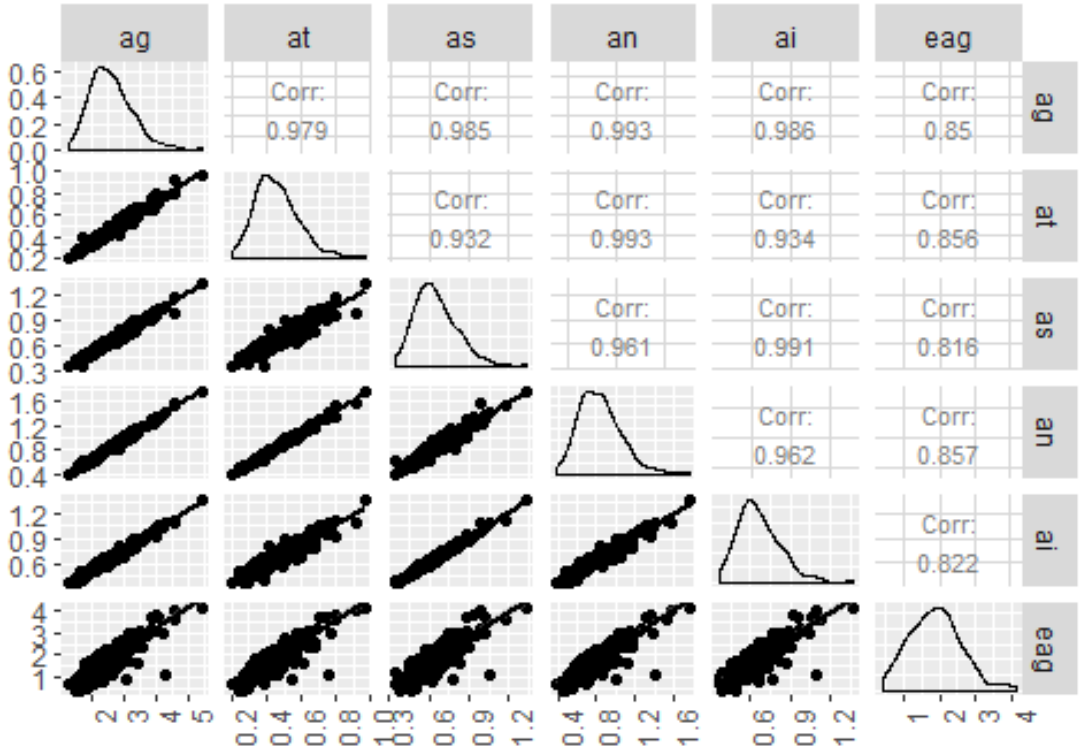
Results: The exercise tells us that building a logistic regression model with *Class* as the response variable and the remaining 62 variables as *Class'* treatment produces a warning due to multicollineary. First, we can take a look at a couple of correlation plots to visualize some of the variable correlations.

Figure 2.1 looks at the first 6 'independent' variables and the correlations between them. We see the correlation is very strong among these variables, as the question stated they would be. *Figure 2.2* shows another sample that depicts the strong correlation among many of the variables in the data set. I've included comparable base R plots for review.

In the interest of space, I will not go through all 62 predictor variables, but it is safe to say that there are strong correlations between treatment variables. Next, I will do model selection to build a model that avoids the **dummy variable trap**.

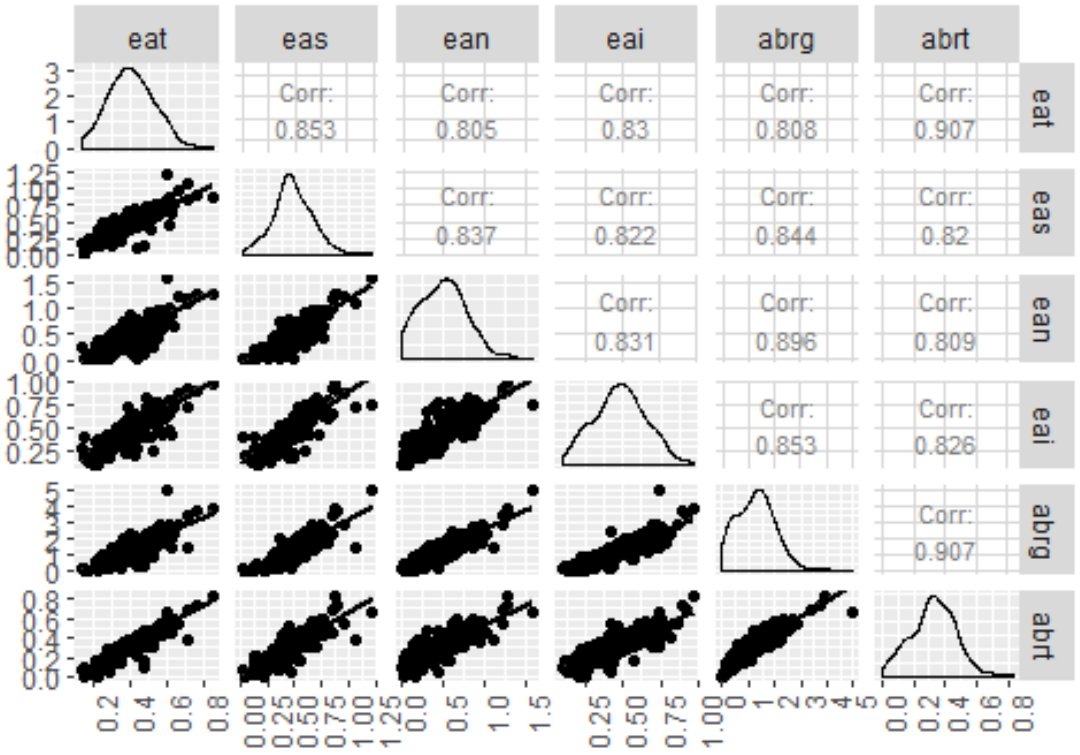
First Sample of GlaucomaM Variable Correlation

Figure 2.1

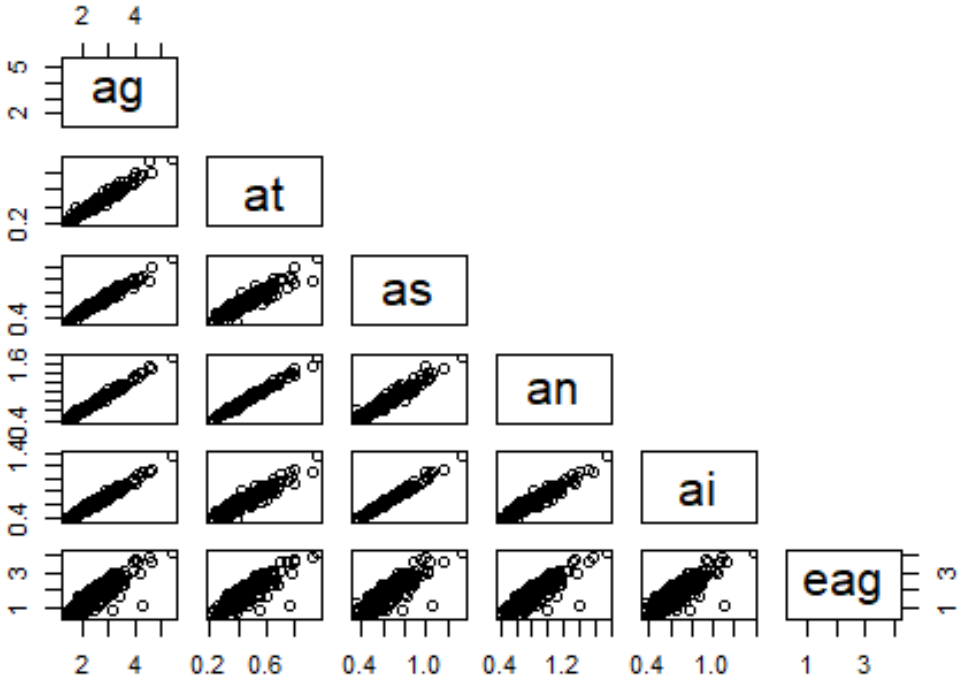


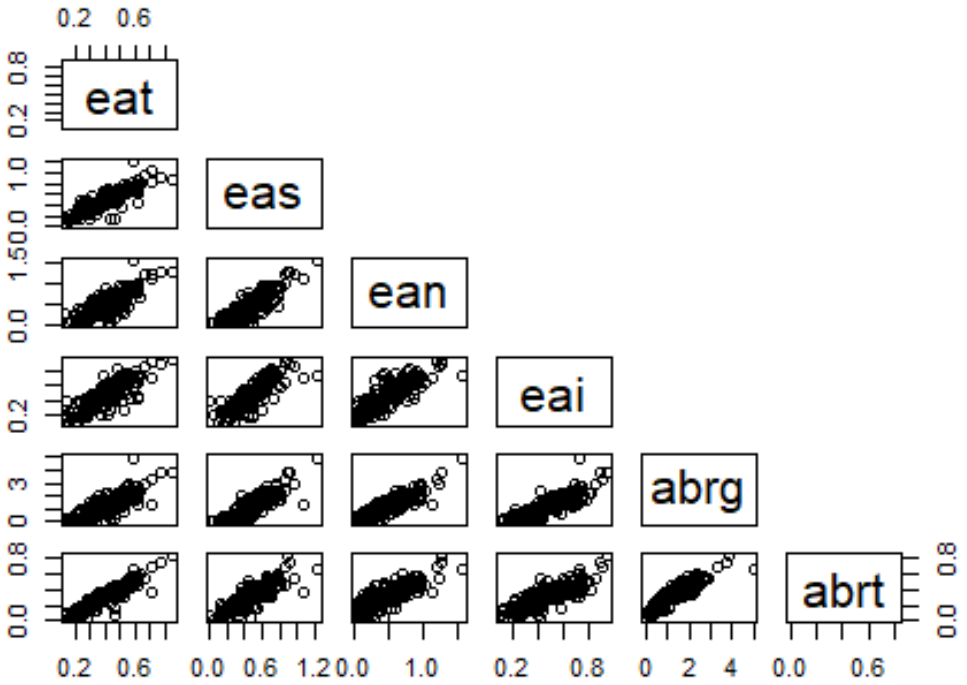
Second Sample of GlaucomaM Variable Correlation

Figure 2.2



First Sample of GlaucomaM Variable Correlation-base R





Results, con't: To remove highly correlated variables, first I took the tedious approach. I removed any that have a correlation greater than 0.6 (or less that -0.6) with another treatment variable. Once these highly correlated treatment variables were removed, I created training(70%) and test(30%) sets to protect against overfitting.

Following this form of variable selection, I fit a logistic regression model using the variables that had made my arbitrary 'cut line'.

Next, for comparison, I created stepwise models beginning with all of the original 62 independent variables. These stepwise models use the *step()* function's *backward* and *forward* methodologies.

Once I had the 3 models, I created *Figure 2.3* to compare the AIC values of the three models (manual, backward, forward). As we see, using the **step** function improves the model with the *backward* stepwise method being superior to *forward*, according to AIC. The *manual* step method produces a model with a higher (worse) AIC value.

Figure 2.4 shows a comparison of error rate. Despite the difference in AIC from the above figure, we see that the *forward* step actually performs better on the test data.

Lastly, I've included the formulas from each model in *Figure 2.5*. First we see the original model that includes all possible predictor variables. Next, we see the model that was derived from my manual variable selection process of removing all variables that had a correlation with another variable that was greater than the absolute value of 0.6. The *backward* and *forward* models are next showing that we have fewer variables using the *forward* stepwise method.

Based on the model's simplicity and improved accuracy on the test dataset, the best model appears to be the one created by the *forward* stepwise model.

Figure 2.3: AIC Comparison

Manual Var-Select AIC	Backward Step AIC	Forward Step AIC
165.7153	**40.00001**	**85.42116**

Figure 2.4: Error Rates by Stepwise Methods

Manual Var-Select Error	Backward Step Error	Forward Step Error
0.2711864	**0.2033898**	**0.1016949**

```
## Figure 2.5: Formulae Summary
##
## Full Model Independent Variables-->
## ag + at + as + an + ai + eag + eat + eas + ean + eai + abrg + abrt + abrs + abrn + abri + hi
c + mhcg + mhct + mhcs + mhcn + mhci + phcg + phct + phcs + phcn + phci + hvc + vbsg + vbst + v
bss + vbsn + vbsi + vasg + vast + vass + vasn + vasi + vbrg + vbrt + vbrs + vbrn + vbri + varg
+ vart + vars + varn + vari + mdg + mdt + mds + mdn + mdi + tmg + tmt + tms + tmn + tmi + mr +
rnf + mdic + emd + mv
## --- 62 Predictor Variables ---
##
## Manual Variable Selection Model Independent Variables-->
## ag + mhct + hvc + vasg + vast + mv
## --- 6 Predictor Variables ---
##
## Backward Step Model Independent Variables-->
## eas + abrs + hic + mhcn + mhci + phci + vbsg + vbss + vbsn + vbsi + vasg + vbrg + vbrt + vbr
s + vbrn + vbri + varg + vars + mdic
## --- 19 Predictor Variables ---
##
## Forward Step Model Independent Variables-->
## varg + phci + vass + tmg + abrn + vars + mdi + mdt + mdg + varn + vbsn + mhcs
## --- 12 Predictor Variables ---
```

Problem #2, Part B: Build a logistic regression model with K-fold cross-validation (k = 10). Report the error rate.

Results: The first step here is to create a cost function to include in the model creation. We utilized our formula, with k=10 and the cost function to derive the error rate. *Figure 2.6* shows us the error rate for our logistic regression model is **0.559322** which is surprisingly high.

Figure 2.6: Error Rate with K-Fold K=10

Error Rate
0.559322

Problem #2, Part C: Find a function (package in R) that can conduct the "adaboost" ensemble modeling. Use it to predict glaucoma and report error rate.

Results: Here we use the **fastAdaboost** library to conduct *adaboost* modeling. We set the 'nIter' = 500 and calculated the error.

Figure 2.7 summarizes the error we received from the boosting model, which is **0.1525424**

Figure 2.7: Error Rate with Adaboost

Error Rate
0.1525424

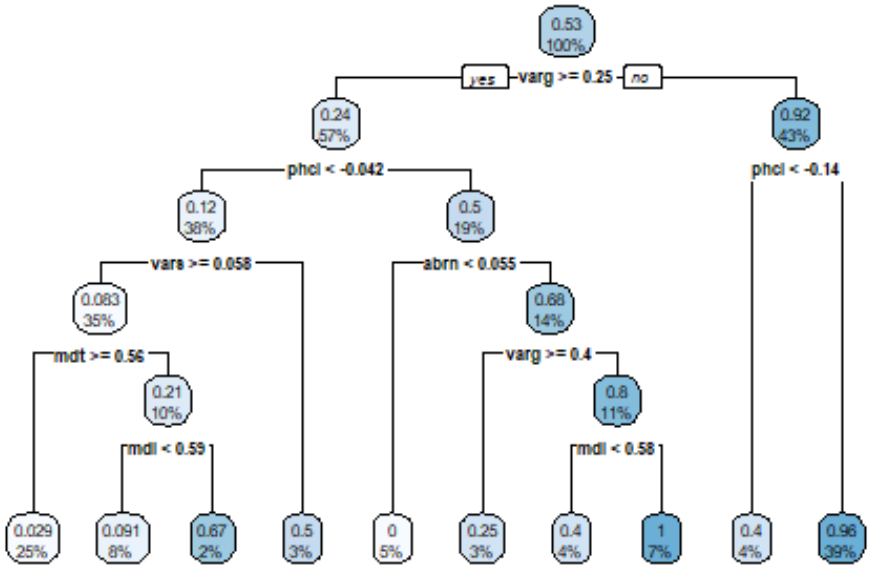
Problem #2, Part D: Report the error rates based on single tree, bagging, and random forest. (A table would be great for this).

Results - Single Tree: Here we fit a model with the predictor variables that we chose during the variable selection process. We plotted the original tree in *Figure 2.8. I've included a plot using* ***ggdendrogram*** *per homework instructions.* From the tree, we see the first two splits are along the variables of *varg* and *phci*.

Next we used *plotcp* to take a look at the CP and X-Error by the number of nodes This plot is summarized by *Figure 2.9*. From this we can see that 2 has the lowest error while 3 has the 2nd lowest.

Using CP and X-Error, we pruned the tree to the optimal level and printed that new tree in *Figure 2.10*. *Figure 2.11* gives out error rate, which is **0.1694915**. *I've included a plot using* ***ggdendrogram*** *per homework instructions.*

Figure 2.8: Original Tree



Original Tree - ggplot

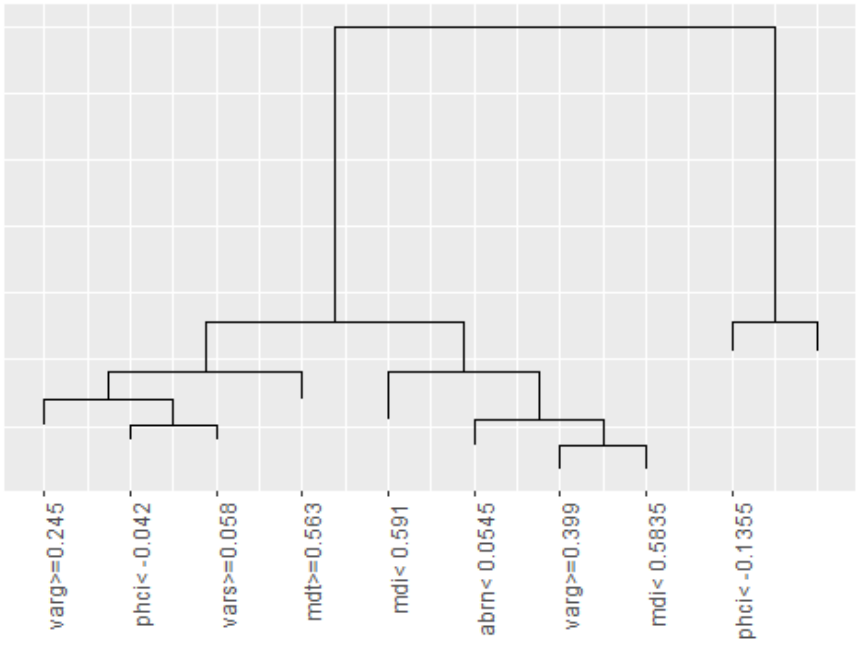
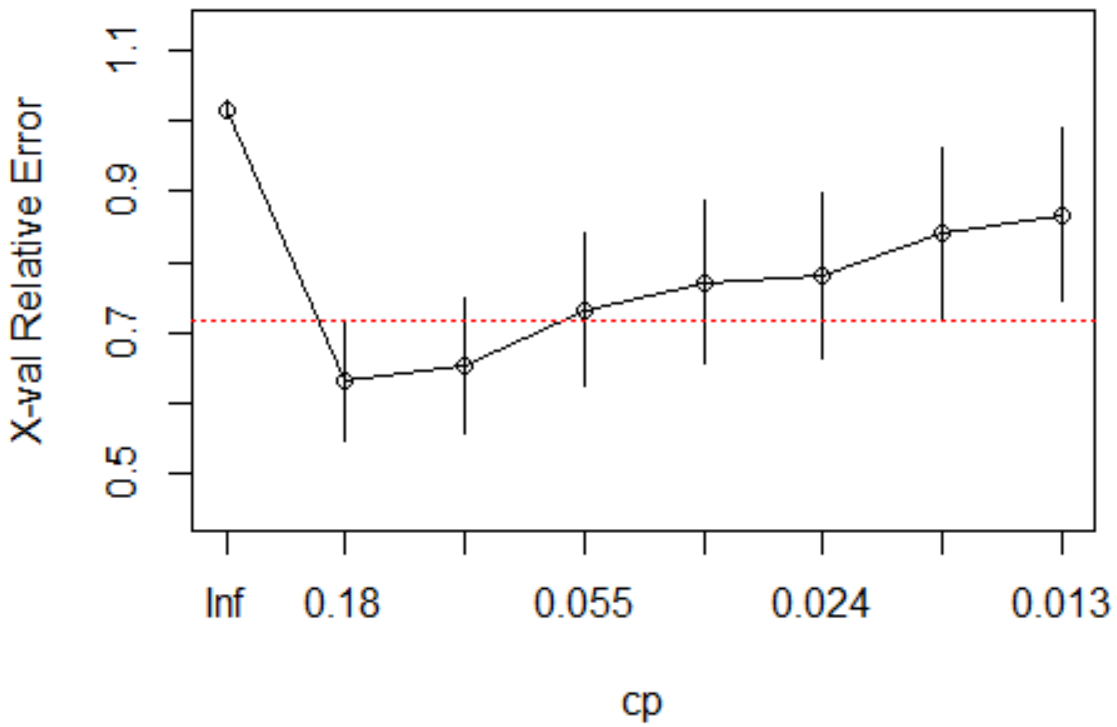
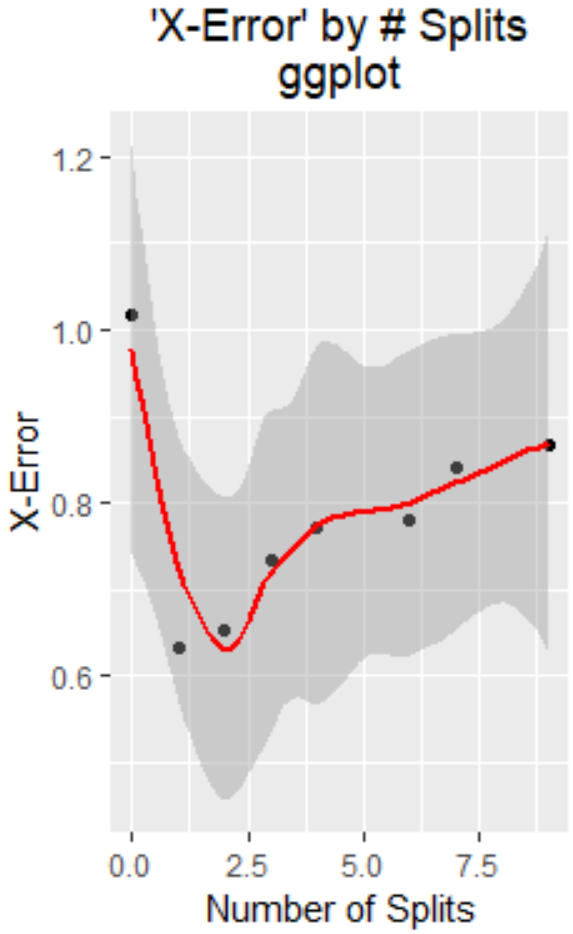


Figure 2.9: CP and 'X-Error' by # Nodes





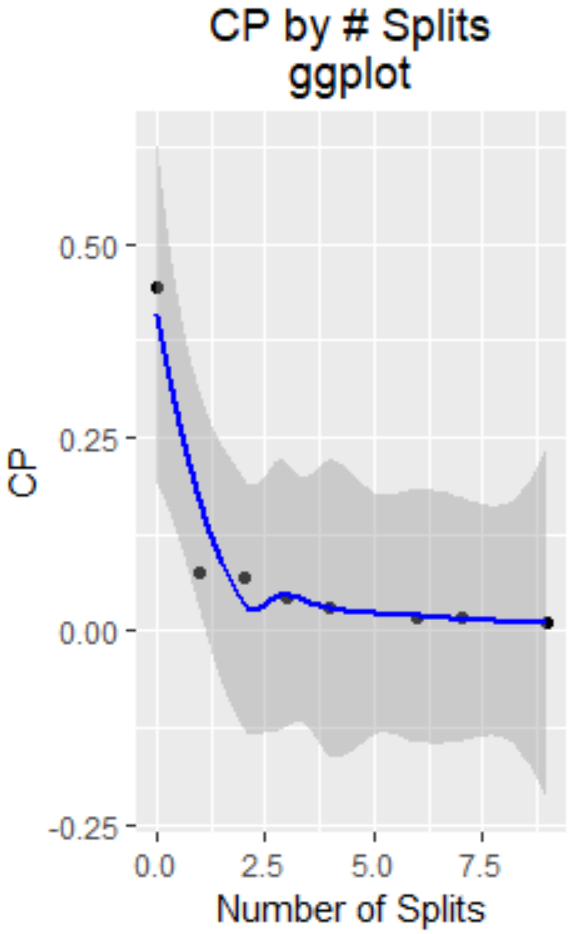
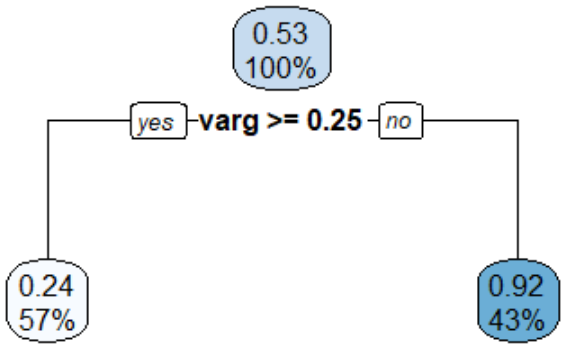


Figure 2.9: CP and Error by Node Number

	Complexity Parameter	Error
1	0.4443739	1.0168999
2	0.0751888	0.6311372
3	0.0702224	0.6515114
4	0.0425294	0.7326193
5	0.0316001	0.7708394
6	0.0187972	0.7807153
7	0.0164259	0.8421212
8	0.0100000	0.8670110

Figure 2.10: Pruned Tree



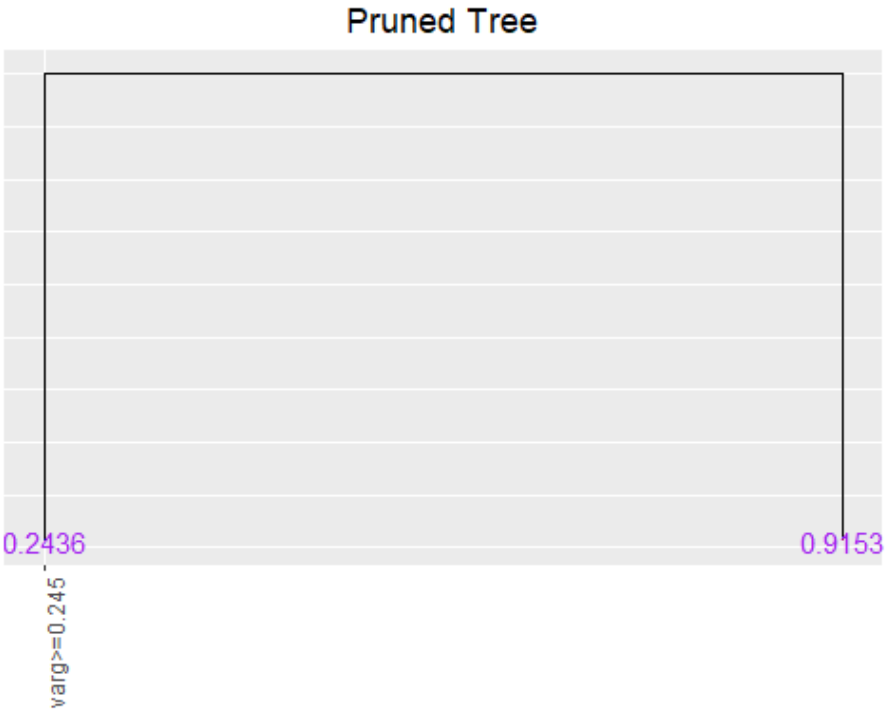


Figure 2.11: Error Rate with Single Tree

Error Rate
0.1694915

Results - Bagging: Here I used bagging with a length of 25 to calculate my error rate using my variables chosen during the variable selection portion of this assignment.

The nice thing about bagging is, as it iterates through, it shows which variable was chosen as the root node for this iteration. This information is summarized in *Figure 2.12*. Here, as we saw with the 'one tree' version above, *varg* and *phci* are again important variables. We also can see that *vars* and *tmg* were chosen in multiple interations as the root node.

Figure 2.13 shows our error rate from bagging as **0.2542373**.

Figure 2.12: Variable Frequency as Root of Tree[i]

Root Node	Frequency
phci	1
tmg	2
varg	12
vars	10

Figure 2.13: Error Rate with Bagging

Error Rate
0.2542373

Results - Random Forest: In this exercise, we use Random Forest functionality to predict GlaucomaM's *Class* variable. In this example, we get an error rate equal to **0.1355932**. This value is shown in *Figure 2.14*.

Figure 2.14: Error Rate with randomForest

Error Rate
0.1355932

Problem #2, Part E: Write a conclusion comparing the above results (use a table to report models and corresponding error rates). Which is the best model?

Results: The best models are the *Forward Stepwise Model* and the *Random Forest* model. By far the worst performing model is the K-Folds model with k=10. Bagging with 'trees' = 25 also performs relatively worse than the other models.

Figure 2.15: Summary of Error Rates by Model

Forward Step	K-10 Folds	Adaboost	Single Tree	Bagging	Random Forest
0.1016949	0.559322	0.1525424	0.1694915	0.2542373	0.1355932

Problem #2, Part F: From the above analysis, which variables seem to be important in predicting Glaucoma?

Results: The two best performing models are the Forward Step and Random Forest methods. *Figure 2.16* re-summarizes the model, showing which independent variables were used in the creation of the superior model.

Since these are the same variables used in the Random Forest model, I've plotted the importance by variable to see which are most important in predicting GlaucomaM's 'Class' variable. *Figure 2.17* shows that the most important predictor variables are: *varg*, *vars* and *phci*.

To further scrutinize this, we look at our results from 'bagging' that show which variables were chosen as the root node and the frequency in which they were chosen. *Figure 2.18* seems to confirm that the three variables mentioned above are among the predictors most important to predicting Glaucoma.

```
## Figure 2.16: Forward Stepwise Formuula -->
## varg + phci + vass + tmg + abrn + vars + mdi + mdt + mdg + varn + vbsn + mhcs
```

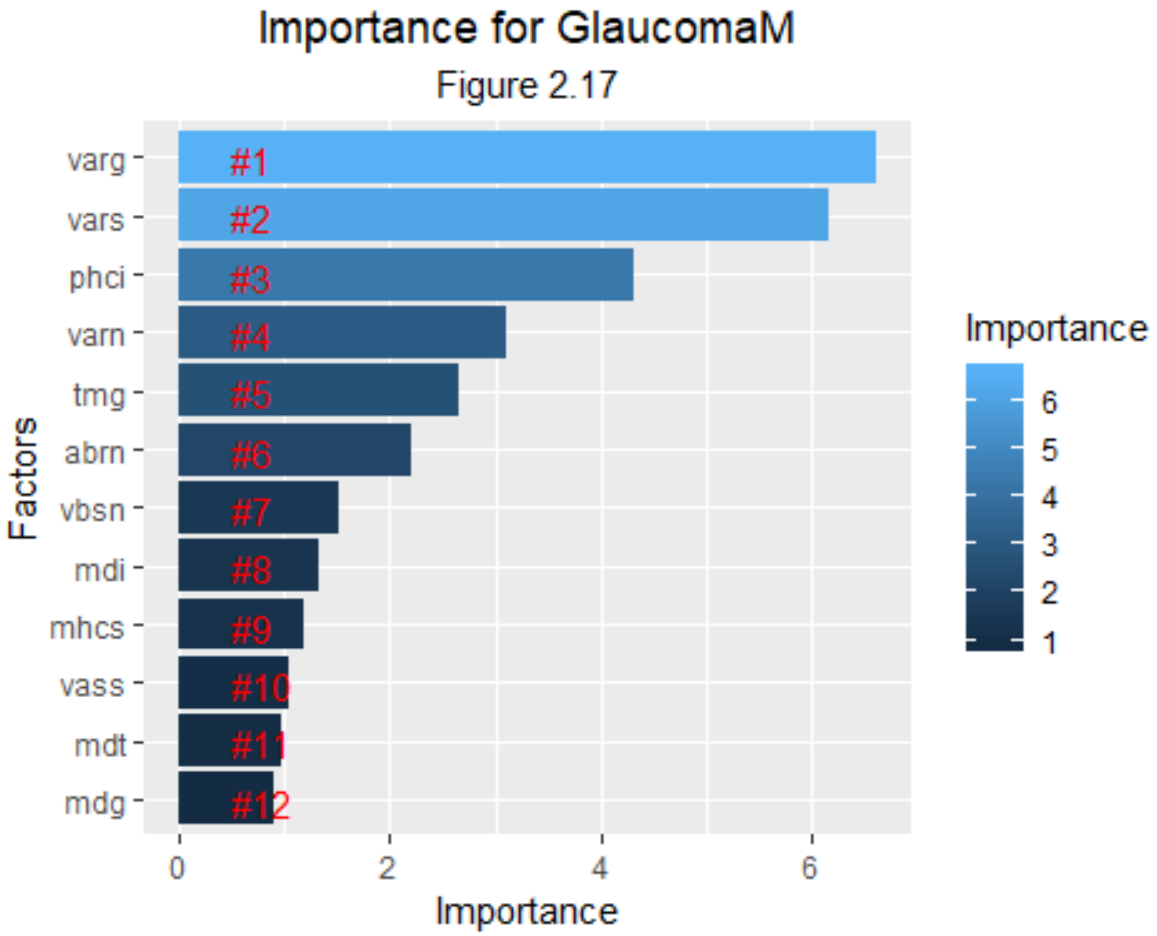


Figure 2.18: Variable Frequency as Root of Tree[i]

Root Node	Frequency
phci	**1**
tmg	**2**
varg	**12**
vars	**10**