ISEN 613 603 – FALL 2020

Group Project

Final Project Report

Anshul Sareen, Jay Shah, Shipra Tyagi, Vaibhav Khanna

# Executive Summary

**To:** President X

**From:** Data Consultant Y

**Subject:** Prediction of current rate of COVID-19 spread

The goal of this project is to predict the rate of spread of COVID-19 with respect to 66 features related to mobility changes in places of activity like retail and recreation places, transit stations, grocery and pharmacy stores, workplaces, parks and residential areas. The data for the mobility changes in each of these 6 distinct places of activity are provided for 11 preceding days.

According to this data, the current average rate of COVID-19 spread (Rt) is about 1.18 with a mean percentage change in mobility for 11 days at -16%, -0.4%, 105%, 9.5%, -29% and -11.5% for retail and recreation places, grocery and pharmacy stores, parks, residential areas, workplaces and transit stations respectively. These values are within the 5% margin for the past 11 days data.

According to the data, the average rate of spread of COVID-19 is as low as 0.85 and as high as 2.0, considering the data for mobility changes for all days.

The changes in mobility for retail and recreation places range from -65% to 17% (range: 82%), and for grocery and pharmacy stores, from -42% to 32% (range: 74%). The percentage change in mobility for parks are as high as 298% and as low as -18% (range: 318%) and for residential areas, they range from -2% to 24% (range: 26%). For workplaces, they vary from -73% to 4% (range: 77%) and for transit stations, they vary from as low as -42% to as high as 14% (range: 56%).

The data from the top 5% rows for the rate of COVID-19 spread has predictors whose average mobility change values are 10% for retail and recreation places, 7% for grocery and pharmacy stores, 44% for parks, 7% for transit stations, 2% for workplaces and -1% for residential areas over the past 11 days. These values give an average rate of spread of COVID-19 equal to 1.93 which is approximately 2% lower than the mean value of 1.97 for the Rt values in top 5% of the dataset.

From our analysis, using multiple linear regression with forward selection and k-fold cross validation, random forest and bagging methods, the **random forest model produced the lowest error on our training dataset at 0.0054 and the test dataset at 0.0063**. The random forest model had 10 statistically significant predictors, namely, workplaces percent change from baseline on the 9th, 8th, 7th, 6th, 5th, 3rd and 2nd days, and retail and recreation percent change from baseline on the 4th, 3rd and 1st days, which were important in influencing the rate of spread of COVID-19.

For this project, various machine learning models were run between all 4 group members and the top 3 models were then selected among them. The executive summary and the 3 technical summaries, too, were distributed evenly among the 4 group members to combine into a final report.

# Model 1: Multiple Linear Regression with K-Fold Cross-Validation

Processing the high dimensional data as the present data set with 66 predictors, 1 response and only 152 rows can be very challenging. Therefore, the algorithms which rely on Euclidean distance as the measure of distance between 2 points does not solve the purpose. On the other hand, algorithms such as linear or multiple regression can easily be overfit to data and thus require careful parameter selection followed by careful hyperparameter tuning. Hence, we have employed multiple linear regression with k-fold cross-validation.

* **Implementation and Predictors:**

The implementation of the technique is quite simple. First, we have declared an empty list to which we will be appending the relevant features. We start by selecting one feature and calculating the metric value for each feature on cross-validation dataset. The feature offering best metric value is selected and appended to list of features. In this assignment, we have compared Adjusted R-square, BIC, Cp and Cross-Validation error as the metrics. The process has reiterated one selected from the previous iteration and the other one selected from the set of all features which does not have the already chosen features.

We then ran the model implementing k-fold cross validation to get the top performing predictors/ variables which have minimum mean error. The outcome was that there were 5 predictors in total with minimum mean CV error with forward subset selection, least collinearity and lowest p values.

**[workplaces\_percent\_change\_from\_baseline\_9, workplaces\_percent\_change\_from\_baseline\_4, workplaces\_percent\_change\_from\_baseline\_7, grocery\_and\_pharmacy\_percent\_change\_from\_baseline\_7, parks\_percent\_change\_from\_baseline\_1]**

* **Final Model Summary:**
* **Residual Standard Error:** The residual standard error is 0.1574 on 146 degrees of freedom after running the final model.
* **Multiple R-squared: 0.7296**
* **Adjusted R-squared: 0.7203:** The value of adjusted r-square indicates that 72% of the variation in the response value is due to the final 5 predictors identified in final model
* **F-Statistic** is 78.79 on 5 predictors with 146 degrees of freedom.
* **P-value: < 2.2e-16**

As we can see in the above summary that combination of F-value and p-value, where F-value is very high, and p-value is very small and hence suggests that we can easily reject the null hypothesis.

1. **workplaces\_percent\_change\_from\_baseline\_9:** The estimated coefficient is 0.0063272 which shows that the predictor and the response has a positive relationship. The 95% confidence interval of coefficient is [0.0084998,0.0041546]. This implies that the number of coronavirus cases has decreased since the changes have taken place in the workplace 9 days ago which is to be expected. The p value of the coefficient is 3.51e-08 this means that **workplaces\_percent\_change\_from\_baseline\_9** is statistically significant.
2. **workplaces\_percent\_change\_from\_baseline\_4:** The estimated coefficient is 0.0051064 which shows that the predictor and the response has a positive relationship. The 95% confidence interval of coefficient is [0.0073098,0.002903]. This implies that the number of coronavirus cases has decreased since the changes have taken place in the workplace 4 days ago which is to be expected. The p value of the coefficient is 7.85e-06. This means that **workplaces\_percent\_change\_from\_baseline\_4** is statistically significant.
3. **workplaces\_percent\_change\_from\_baseline\_7:** The estimated coefficient is 0.0060390 which shows that the predictor and the response has a positive relationship. The 95% confidence interval of coefficient is [0.0082794,0.0037986]. This implies that the number of coronavirus cases has decreased since the changes have taken place in the workplace 7 days ago which is to be expected. The p value of the coefficient is less than 2.75e-07 this means that **workplaces\_percent\_change\_from\_baseline\_7** is statistically significant.
4. **grocery\_and\_pharmacy\_percent\_change\_from\_baseline\_7**: The estimated coefficient is -0.0066103 which shows that the predictor and the response has a negative relationship. The 95% confidence interval of coefficient is [0.0099625,-0.0032581]. This implies that the number of coronavirus cases has increased since the changes have taken place in the grocery and pharmacy 7 days ago which is to be expected. The p value of the coefficient is 0.000124 this means that

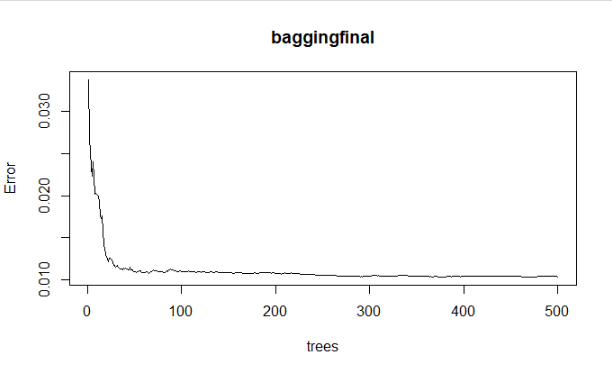
**grocery\_and\_pharmacy\_percent\_change\_from\_baseline\_7** is statistically significant.

1. **parks\_percent\_change\_from\_baseline\_1 :** The estimated coefficient is -0.0008690 which shows that the predictor and the response has a positive relationship. The 95% confidence interval of coefficient is [-0.0007015,-.0010365]. This implies that the number of coronavirus cases has increased since the changes have taken place in the parks 1 day ago which is to be expected. The p value of the coefficient is 6.98e-07 this means that **parks\_percent\_change\_from\_baseline\_1** is statistically significant.

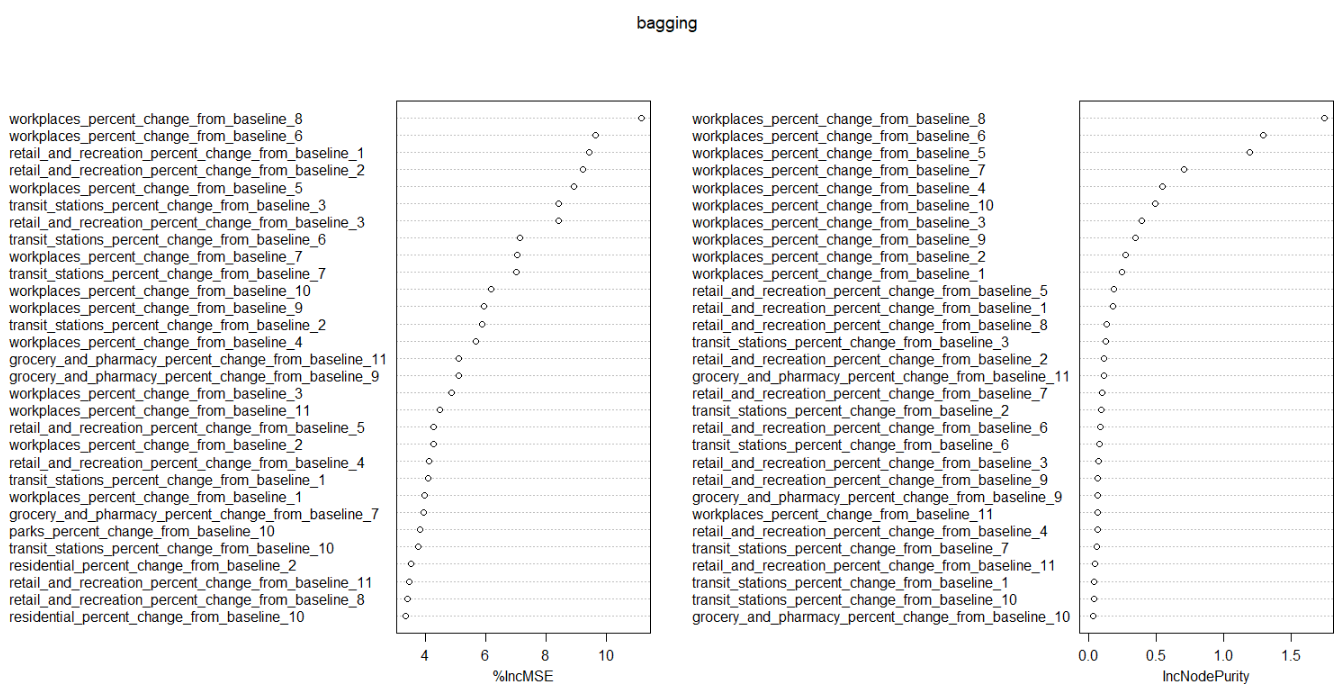
# Model 2: Bagging

Bagging, also known as Bootstrap Aggregating, is one of the ensemble machine learning algorithms which is used to improve the stability and accuracy of regression algorithms. It helps to reduce variance in the model and minimize overfitting. We can often optimize the performance by bagging 50-500 trees. Datasets with a few significant predictors often require less trees (50-200); whereas the datasets with multiple collinear predictors may need more. In the given dataset, we have 66 predictors that are highly collinear. We can see from the graph in Fig. 1 that the error stabilizes and is the lowest when we set ntree = 500 to plot 500 trees. We also set mtry = 66 as we want all the 66 available predictors randomly sampled as candidates in each split. An added advantage is that even if too many trees are used, it does not lead to overfitting.

In our case, we have partitioned the data into training (80%) and test (20%) datasets. Bagging was implemented on the training dataset containing 121 rows (out of 152 rows). The model was then run using the test dataset containing 31 rows.



We fit the bagging model on the training data using the randomForest() function for regression. This function is contained in the randomForest package in R. As mentioned above, we set mtry to 66 since we want all 66 variables to be randomly sampled as candidates at each split. Also, since the number of predictors in the dataset are higher and are highly collinear, we set ntree to 500 since we want to minimize the average error from these trees. We then predict the results of this model using the test dataset and estimate the mean square error for the test dataset which comes out to be around 0.0058 and the variance explained is 88.23%.



*Fig.2 – using varImpPlot() to visually check variable importance*

Further, we need to know the most important predictors according to bagging. We estimate these using the important() function in R, which is used to estimate the variable importance using 2 measures - %IncMSE and IncNodePurity. %incMSE indicates how much our model accuracy decreases if we leave out a variable. While IncNodePurity is a measure of variable importance based on the Gini impurity index used for the calculating the splits in trees.

Out of these two, %IncMSE is a more robust and informative measure. IncNodePurity is biased and should only be used if the extra computation time of calculating %IncMSE is unacceptable. Since it only takes ~5-25% extra time to calculate %IncMSE, this would almost never happen.

We visualize the importance of these predictors using varImpPlot() as seen in Fig. 2. The higher these values are for the predictors, the higher the importance of the predictor to our model.

Considering a threshold of 8 for %IncMSE, we see that the most significant predictors are -

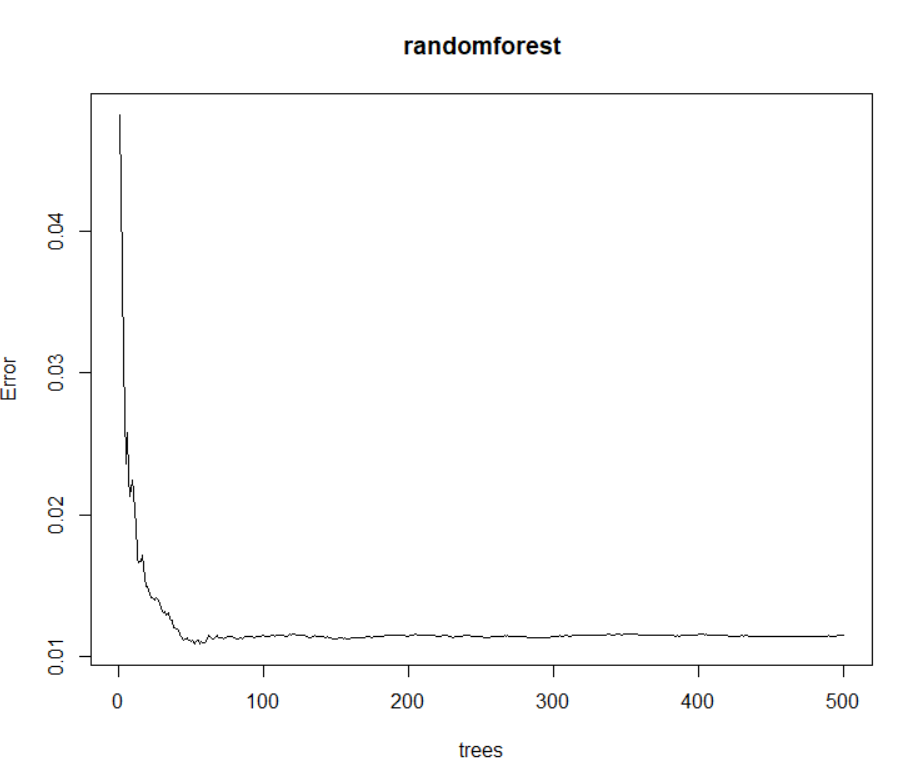
* workplace\_percent\_change\_from\_baseline\_8
* workplace\_percent\_change\_from\_baseline\_6
* retail\_and\_recreation\_ percent\_change\_from\_baseline\_1
* retail\_and\_recreation\_ percent\_change\_from\_baseline\_2
* workplace\_percent\_change\_from\_baseline\_5
* transit\_stations\_percent\_change\_from\_baseline\_3
* retail\_and\_recreation\_ percent\_change\_from\_baseline\_3

# Model 3: Random Forest Regression

Random Forest is a simple and flexible supervised learning algorithm, and it tends to produce good results even without tuning the parameters. Random Forest builds multiple trees, and this combination of models help to get more accurate and stable predictions.

Random Forrest is best among the classification method as it provides highest accuracy. We have used this method as it helps to achieve low bias and low variance by creating a number of trees from the subset of data and combining the output from all trees. It reduces overfitting and is very flexible as it can be used for both classification and regression problems. This method has a unique and automatic way of handling missing values. This algorithm is very stable and is less impacted by noise. We divided our data set such that the training data consists of 80% of the observations while the rest 20% belong to test data.

We begin by fitting the Random Forest model with our training dataset while the rest of the data is used to validate the trees. The **Mean Squared Error** comes out to be **0.0115** and the **Variance explained** is **85.76%**. MSE and Variance explained are calculated using Out of Bag Error estimation. After plotting our model, it can be observed that error rate drops as we increase the number of trees and average them.



We then used the predict function to get the predictions from our current model for our response. Overall RMSE is a measure of difference between precited and observed values. We have calculated the overall RMSE using formula {**mean((y.test - pred.randomforest)^2)**} and the value comes out to be **0.0054**.

Finally, we fit the Random Forest model on our entire data set and estimate the most important variables. In order to measure the importance of each predictor variable, importance() function is used by Random Forest model. The first measure it provides is **% Increase in MSE (%IncMSE)** which is calculated from permuting OOB data. To put simply, if a predictor is important in our current model, then permuting it with a random and realistic value will negatively affect our model.

**IncNodePurity** is a measure that chooses the best splits. It splits on the variable and calculates the total decrease in node impurities. The variables which are more important for the model have higher increase in node purities. We have used importance function to get **%IncMSE** and **IncNodePurity** and plotted these measures with **varImpPlot.**

|  |
| --- |
|  |

IncNodePurity is often biased and it is advised to use it only if computation time of %IncMSE is very large. Predictors with higher %IncMSE are more valuable for our model and hence we have considered a threshold of 6 to estimate our most important predictors.

The most important predictors are:

* Workplaces percent change from baseline 7
* Workplaces percent change from baseline 5
* Workplaces percent change from baseline 6
* Retail and Recreation percent change from baseline 1
* Workplaces percent change from baseline 3
* Workplaces percent change from baseline 8
* Workplaces percent change from baseline 9
* Workplaces percent change from baseline 2
* Retail and Recreation percent change from baseline 3
* Retail and Recreation percent change from baseline 4
* Workplaces percent change from baseline 10
* Transit stations percent change from baseline 7
* Workplaces percent change from baseline 4

Random Forest creates several trees and we have used 500 in our algorithm which can be complex and resource intensive. These disadvantages are compensated with its high accuracy, low bias, and low variance. We have achieved the **lowest overall error of 0.0054** with this model.

# Comparison of the Three Models

As discussed in the previous sections, the 3 final models selected are **Forward Selection Algorithm with K-Fold Cross-Validation, Random Forest method and Bagging method.**

The first method employed is highly interpretable but has low prediction accuracy. In this method, we ran the forward selection algorithm with cross-validation (k=10 folds). The output of this algorithm was 7 predictors. Multiple linear regression was performed on the complete dataset with these 7 predictors to enhance the accuracy of the model as it is expected to have better prediction accuracy with the complete training data. After this, the collinearity amongst predictors is determined using VIF. Two predictors are further removed from the model because one has high VIF values and the other has high p-value. The final model contains 5 predictors. The test minimum test error obtained from the forward selection is 0.029.

The Random forest method and bagging are less interpretable and have high prediction accuracy. Here the data was divided randomly into 80% training and 20% test data. Bagging was employed with parameters mtry = 66 and number of trees equal to 500. The output of this algorithm gave a low-test error of 0.0058. The Random forest method was employed on the same data. The advantage of random forest is that it decorrelates the regression trees. As a result, we have improved prediction accuracy at the cost of interpretability. The parameters used for Random Forest algorithm were mtry = 9 and ntree = 500. This model resulted in a test error of 0.0054.

Based on the test errors obtained from the above three models, the Random Forest and Bagging are producing similar results from the training data set. We are choosing Random Forest as our final algorithm because there is high collinearity amongst our set of 66 predictors and RF has an advantage over the bagging method by decorrelating the regression trees developed using the algorithm.

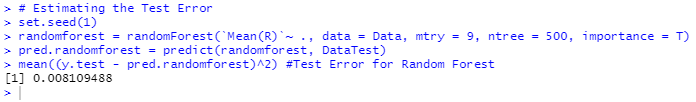
* **FINAL MODEL - Random Forest Regression:** The parameters developed using grid search on the training data were found to be mtry = 9 and ntree = 500. Using these parameters, the Random Forest algorithm was applied to the complete data set and the most important predictors are given using the Variable Importance Plot below.

The most important predictors were found to be:

|  |  |
| --- | --- |
| * Workplaces percent change from baseline 7 * Workplaces percent change from baseline 5 * Workplaces percent change from baseline 6 * Retail and Recreation percent change from baseline 1 * Workplaces percent change from baseline 3 * Workplaces percent change from baseline 8 * Workplaces percent change from baseline 9 * Workplaces percent change from baseline 2 * Retail and Recreation percent change from baseline 3 * Retail and Recreation percent change from baseline 4 * Workplaces percent change from baseline 10 * Transit stations percent change from baseline 7 * Workplaces percent change from baseline 4 |  |

# Evaluating the Test Data

The test data contains 50 rows in total with the 66 predictor variables and the response variable Mean(R). We ran our best model, i.e. random forest regression, on this dataset with the same parameter values for mtry = 9 and ntree = 500 which we used on the training dataset as well.



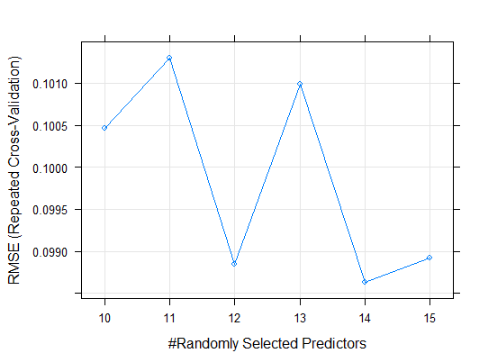
We obtained a **reported test error of 0.0081** using random forest regression as compared to 0.0054 on the training dataset. This is a reported increase of 0.0027 in the error rate.

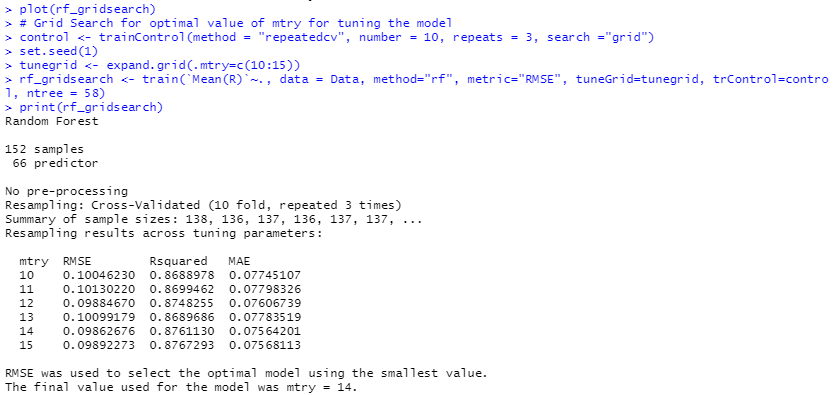
Since the testing error rate was higher, we decided to use hyper parameter tuning using grid search to optimize the values of mtry and ntree, which might help reduce the test error.

# Steps to Improve Best Model – Hyper Parameter Tuning

The first step in our hyper parameter tuning is to use grid search to find the optimal value of ‘mtry’. We will do this using the trainControl() function using the ‘repeatedcv’ resampling method. We will keep the number of folds to 10 and the number of complete sets of folds to compute at 3. Finally, we will set the search method to ‘grid’.

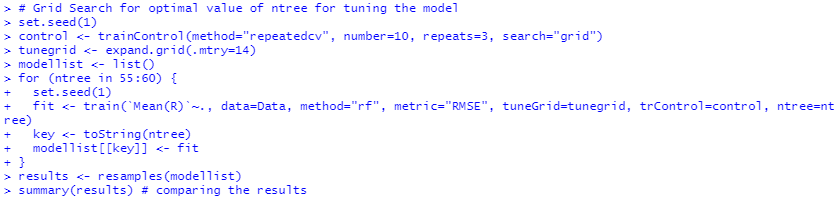
Next, using the train() function, we will train the model on the training data where the selection criteria for the optimal value of mtry will be the lowest RMSE value obtained for the mtry values. We ran the grid search, taking the mtry values in blocks of 5, and found that optimal values were obtained for mtry values between 10 and 15. On running this code, we find that the most optimal value for mtry comes out to be 14.

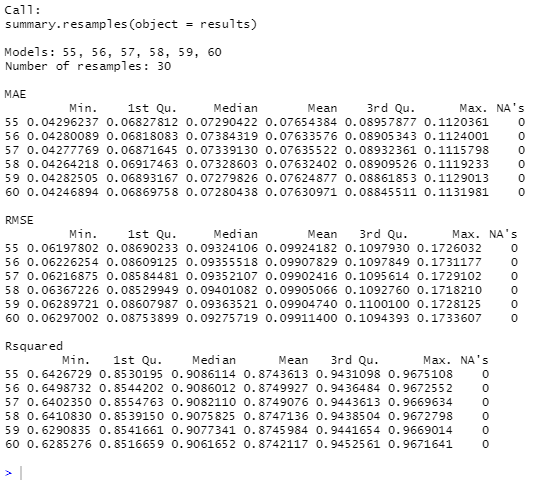




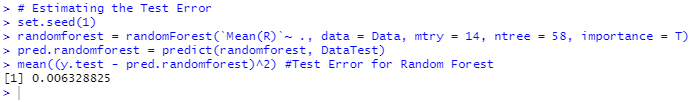
Now, using this optimal value of mtry = 14, we will try and compute the optimal value for ntree.

Similar to what we did for mtry, we use the trainControl() function where the selection criteria for the optimal value of ntree will be the lowest RMSE value. We also used the expand.grid() function to create a data frame to pass the value of mtry which is 14. We ran the grid search again, in blocks of 5, and found that optimal values were obtained for ntree values between 55 and 60.





On observing the values for ntree between 55 and 60 above, we see that for ntree = 57, 58 and 59, the mean RMSE values are lowest. Using these values to tune our random forest regression model, we found that the lowest test error was reported when value of ntree was 58.



**Final Result:** Our test error after hyper parameter tuning comes out to be **0.0063 for the test data**.