Introduction

This report makes use of real housing data to examine relationships between variables and to predict House Prices in the Sindian Dist., New Taipei City, Taiwan. The key prediction question for this report is; to what extent can House Prices in New Taipei City be predicted using house specific variables and local environment predictors? A secondary question is what variable(s) is/are most important in influencing the price of a House in the Sindian District of New Taipei City?

Being able to accurately predict house prices to is extremely exciting, not to mention relevant. Investors can conclude on the relative attractiveness of new residential opportunities and evaluate whether a particular house or suburb is over- or under-valued, which will hopefully improve returns. Homeowners can make use of calculations and predictions to cross-reference their own valuations improving the efficiency of local housing markets. Local governments can also make use of similar findings, discovering the best way to drive localised economic growth through valuation growth by examining how local factors influence prices, this could be useful when looking to regenerate particular boroughs, thus improving their citizens quality of life and financial security.

Data and Exploratory Analysis

The data used in this report is taken from the UCI Machine Learning Repository. It contains 414 observations with each observation made up of 7 variables. The variables used are as follows:

Variables

No.	Variable Name	Name in Code	Detail
1	Transaction Date	TransactionDate	From August 2021 to July 2013
2	House Age	HouseAge	From new (0) to 43.8 years old
3	Distance to the Nearest MRT Station	MRTDist	Measured in metres
4	Number of Convenience Stores within Walking Distances	StoreNum	Measured as an Integer
5	Latitude	Latitude	Measured in Degrees
6	Longitude	Longitude	Measured in Degrees
7	House Price per ping	HousePrice	Measured in NT$10,000/Ping, where Ping = 3.3m^2

The data set was downloaded already cleaned and with no missing values. The three strongest absolute correlations are -0.674 (MRTDist), 0.571 (StoreNum) and 0.546 (Latitude). These correlations suggest that these variables possess relatively strong predictive abilities. To build on this, the visual relationships between House Prices and its predictor variables will now be examined.

Figure 1 displays the prices Houses were sold for at different dates within the observation period. There is no obvious strong relationship between the date sold and the value of the property, emphasised by a weak positive correlation of 0.087.

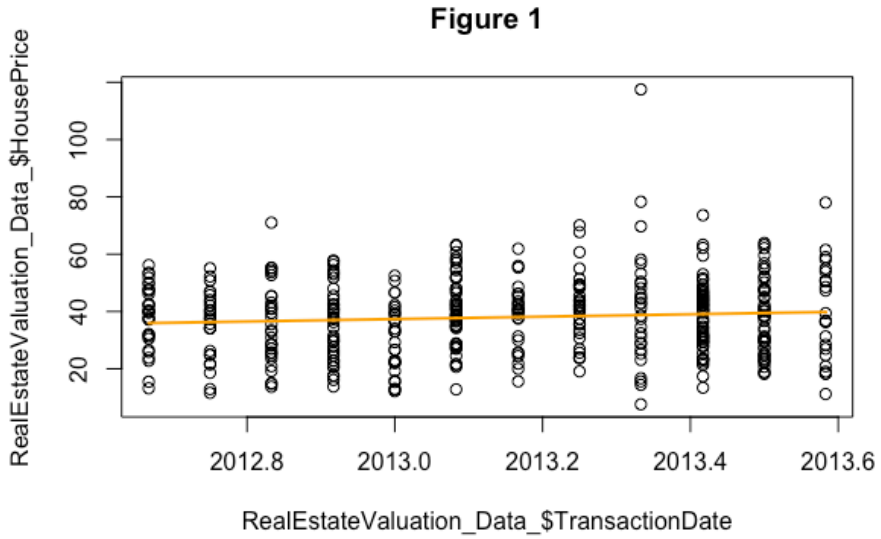


Figure 1 shows that Transaction Date (x-axis) and House Price (y-axis) have a slight positive relationship, shown by the slope of the linear line of best fit. This relationship suggests House Prices, on average, increased over the observation period. This is in line with expectation, due to inflation and economic growth.

Figure 2 shows the relationship between House Age and House Price. The pair of variables have a negative correlation of -0.211, suggesting as properties age their values fall proportionally, contrarily however we can see that this is only to a point. This non-linear relationship is expected. After a point property value rises, likely due to the increasing historic value of a property and perhaps due to the larger plots older Houses tend to have been built on.

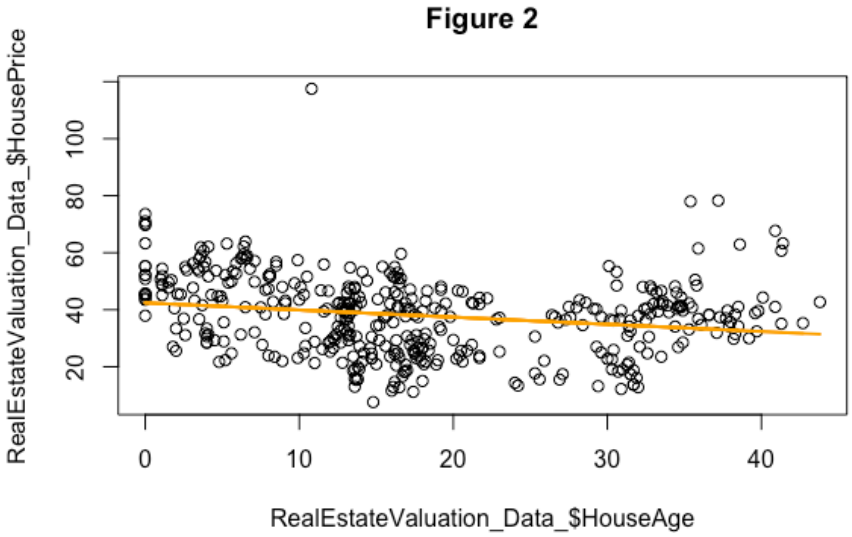


Figure 2 plots House Price on the y-axis and House Age on the x-axis, the linear line of best fit is downward sloping, implying a negative relationship. However, as stated this not a linear relationship.

The relationship between MRTDist and House Prices is shown in Figure 3. There is a distinct negative relationship between the two, reinforced by a correlation of -0.674, highlighting that the closer a property is to an MRT Station the higher value it can command. The relationship, however, appears to be non-linear. The plot shows a large clustering of sales over the observation period all within 1000 metres of an MRT Station, supported by Figure 4.

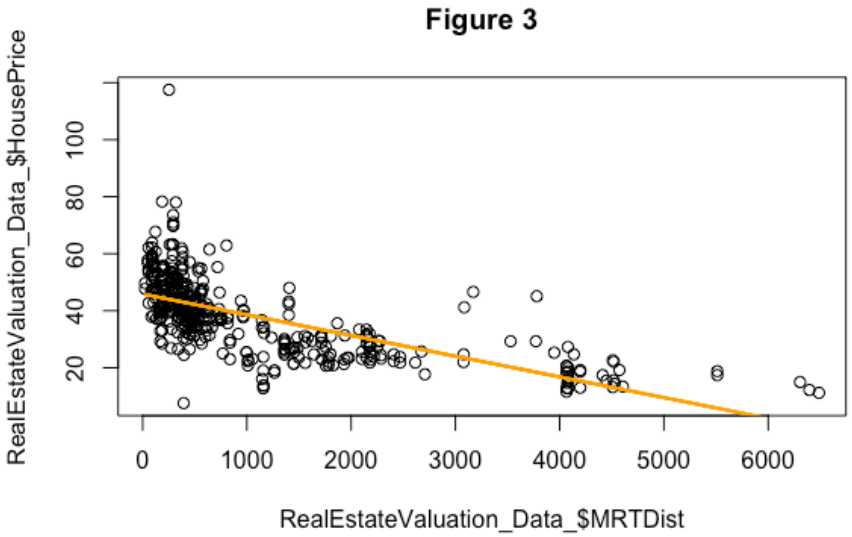


Figure 3 plots MRTDist on the x-axis with House Price on the y-axis. The variables have a strong negative correlation. However, it appears non-linear with prices levelling off at 4000m onwards.

Figure 4

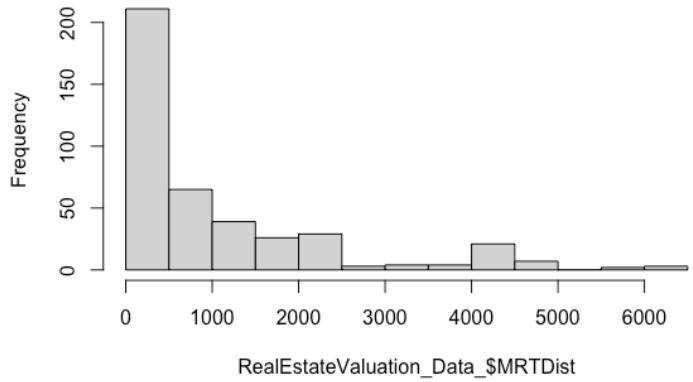


Figure 4 is a histogram presenting the distribution of distances from an MRT Station. There is a large cluster of properties between 0 and 1000m from a station. Fewer observations exist beyond 2500m, these properties are likely to be more rural and therefore not in as high demand as the properties closer to an MRT Station. The cluster suggests properties close to a station tend to have a higher occupancy turnover, which makes sense in a bustling and growing city.

Figure 5 shows the positive relationship between House prices and the number of convenience stores within walking distance of the property. The variables share a relatively strong correlation of 0.571, suggesting more local stores adds value to a property in the area observed. This general relationship is emphasised in Table 1, which shows the output of a box and whisker analysis, the median increases from 22.30, for zero stores, to 47.55, for 10 stores.

Figure 5

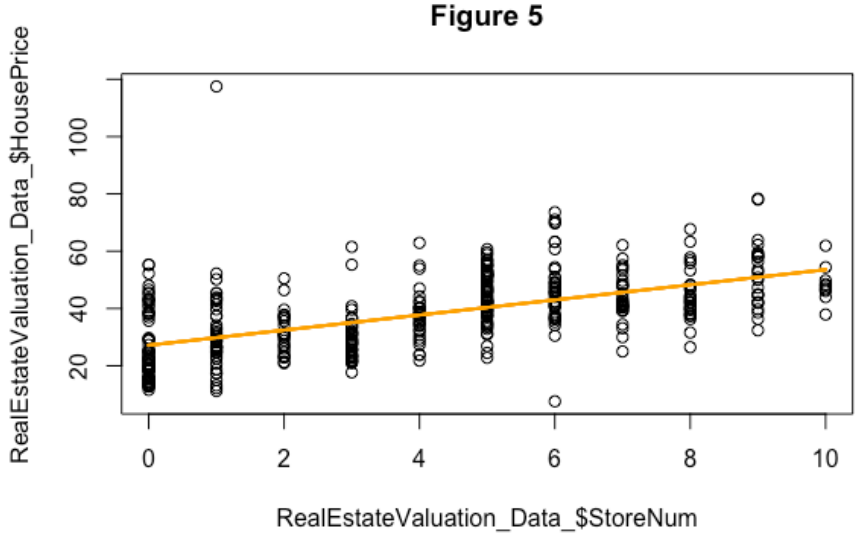


Figure 5 plots StoreNum on the x-axis with House Price on the y-axis. The linear line of best fit is strongly positive. Although this relationship is somewhat expected, it is interesting to see that there is no obvious limited to this relationship, one would expect too many shops may put off wealthier buyers due to the consequential noise and foot traffic near to the property.

Table 1

boxplot.Price.Store$stats												
	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	
[1,]	11.6	11.2	20.9	17.7	21.8	22.8	30.5	30	26.5	32.4	44	"Lower Whisker"
[2,]	17.45	24.7	25.3	23.6	32.35	39.35	40.6	40.4	39.5	42.2	46.1	"Lower Hinge 1ST Quartile"
[3,]	22.3	28.7	30.65	28.25	37.4	44.5	45.7	42	42.75	50.8	47.55	"Median"
[4,]	37.45	37.4	36.8	32.1	40.3	51.75	51	48.3	48.2	58.1	49.8	"Upper Hinger 3rd Quartile"
[5,]	55.3	52.2	50.5	40.8	47	60.7	63.3	57.4	57.8	78.3	54.4	"Upper Whisker"

Table 1 shows the output of a box and whisker analysis of StoreNum. The general trend of the median is upwards going from 22.30 (0) to 47.55 (10). Although the relationship is not always positive, with the median House value dropping a few times between the number of convenience stores, from 3 to 4 for example. However, this is the median and not the mean so difficult to conclude on.

The final two variables are latitude and longitude which combined give the geographical positioning of an observation. Figure 6 shows the locations of the properties sold. Both latitude and longitude share a relatively strong positive correlation with House prices, 0.546 and 0.523 respectively, suggesting that properties in the lower quadrant of figure 6, more rural when compared to a map, have lower values. This relationship however is clearly non-linear.

Figure 6

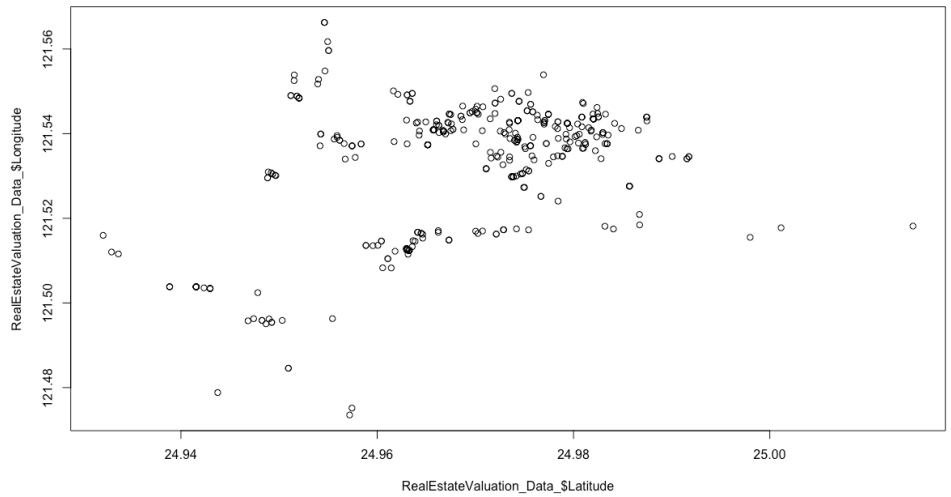


Figure 6 shows the geographical positioning of all the Houses sold over the observation period. With Latitude on the x-axis and Longitude on the y-axis. Rural properties are found in the lower left quadrant, due to the correlation of the variables with House Prices we can conclude that, generally, properties closer to the city centre and in more urban areas command higher values in New Taipei.

To assess the predictive ability of each independent variable a simple regression of each variable individually regressed against the dependent variable, House Prices, was calculated. The output from this analysis is shown in Table 2 below.

Table 2

Variable	R-squared	p-value	Significance
Transaction Date	0.007655	0.07537	< 95%
House Age	0.04434	1.56e-05	> 99.9%
MRT Distance	0.4538	2.2e-16	> 99.9%
Store Number	0.326	2.2e-16	> 99.9%
Latitude	0.2985	2.2e-16	> 99.9%
Longitude	0.2738	2.2e-16	> 99.9%

Table 2 shows the individual output of simple regressions of each predictor variable and House Price. 5 of the 6 variables are significant at regularly used confidence intervals.

Table 2 shows that five of the six predictor variables used in this report are extremely significant, demonstrated by their negligible p-values allowing us to reject the null hypothesis of zero and insignificance. Their significance is further emphasised by impressive individual R-squared values. Transaction Date is the only non-significant variable at regularly used confidence levels, 95% and above.

Machine Learning Methods

This report analysed various machine learning methods. Many of these methods are examined using training and test data sets. The training set for all methods is made up of 289 randomly chosen observations from the entire Real Estate Valuation data set, or 70% of all observations. The training set allows the methods to try understand and examine the various relationships between predictor variables and the dependent variable. These estimated relationships are then tested on the test set made up of the remaining 125 observations, or 30% of total observations. The accuracy of each method is determined by its R-squared coefficient and Mean Squared Error (MSE), a comparison of these parameters as well as intuition allow this report to conclude on the best methods for predicting House Prices in the Sindian District. R-squared in this report represents the proportion of House Price variance that is explained by the independent variables used in prediction. It is calculated using the following formula;

$$(\mathbf{1})\ R^2 = 1 - \frac{Residual\ Sum\ of\ Squares}{Total\ Sum\ of\ Squares} = 1 - \frac{RSS}{TSS}$$

Mean squared error (MSE) represents the average squared error of prediction, that is the difference between the predicted value from a method and the actual value squared. Its root shows the interval one could expect the true value of an observation to lie within, above and below, given an estimated value. The formula used to find MSE is as follows;

$$(\mathbf{2})\ MSE = \frac{1}{n}\sum_{i=0}^{n}(Y_i - \widehat{Y}_i)^2$$

Multiple Linear Regression

The first method used is the Multiple Linear Regression (MLR). This method uses all predictor variables outlined above to predict the dependent variable, House Price. It is an extension of the Ordinary Least Squares (OLS) method.

To test the predictive power of the multiple linear regression method, the training data was used to estimate an OLS model that minimises the difference between the observed values and the models fitted values by assigning linear Betas and an intercept. The coefficients of each variable were found to be:

Table 3

Coefficients:					
	Estimate	Std. Error	T value	Pr(>|t|)	
(Intercept)	-1.39E+04	8.99E+03	-1.544	0.1238	
TransactionDate	4.60E+00	1.95E+00	2.355	0.0192	*
HouseAge	-3.02E-01	4.82E-02	-6.268	1.37E-09	***
MRTDist	-4.22E-03	9.51E-04	-4.44	1.29E-05	***
StoreNum	1.10E+00	2.40E-01	4.595	6.54E-06	***
Latitude	2.65E+02	5.56E+01	4.757	3.14E-06	***
Longitude	-1.60E+01	6.51E+01	-0.245	0.8065	

Table 3 shows the coefficients assigned to the predictor variables from a MLR on the training data. All variables are significant apart from Longitude. Transaction Date is only significant at certain confidence intervals. The coefficients range in size but none are greater than 1.

This model was then applied to the test data and used to predict each observations House Price. The method did this with adequate accuracy, yielding an R-squared of 0.5919161and MSE of 65.24426. This means that the model collectively explains 59.2% of House Price variance in the test data set with a Root MSE (RMSE) of 8.077 units – or an expected percentage error of 21.3%. This R-squared, implying more accurate predictions than bad ones, is acceptable given the linearity assumptions of the model and the clear non-linearity of some of the predictor variables evident in the previous section.

In the hope of improving predictive accuracy a logarithmic transformation of House Prices was performed. The process outlined above was repeated. The outcome of was an R-squared of 0.6149192 and MSE of 61.56653. Compared to the previous model, transforming the independent variable improved predictive accuracy. Therefore, by taking the logarithm of House Prices the impacts of its non-normal distribution, shown in figure 7, are not as prevalent thus improving the model's accuracy.

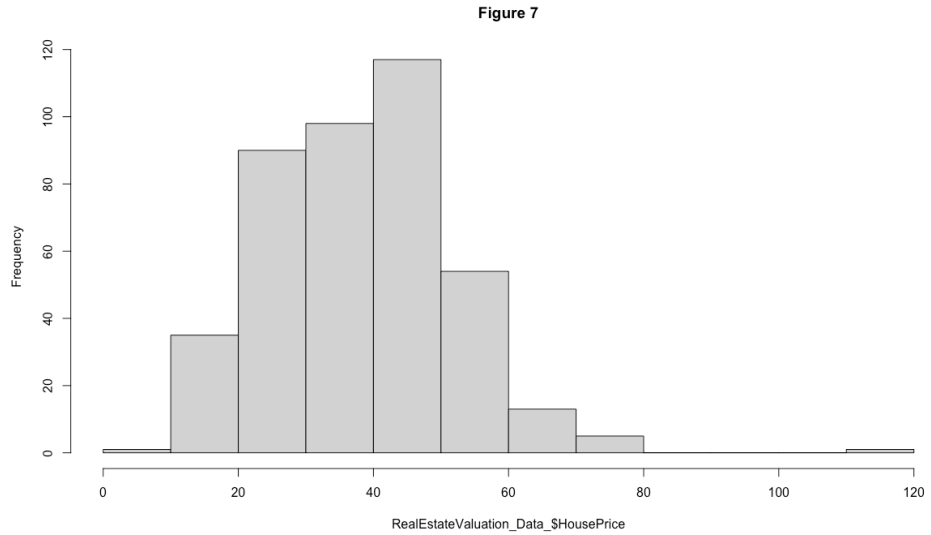


Figure 7 shows a histogram of House Prices. It is non-normal with a somewhat leptokurtic distribution with non-normal skew and kurtosis. Thus, it violates some of the normality assumptions of OLS used for MLRs. Logarithmic transformations improve this distribution and therefore the accuracy of the MLR method.

Subset Selection

Best subset selection builds upon the multiple linear regression model, it works by fitting each combination of the predictor variables at a given subset size and working out the one with the lowest RSS. The best model of each subset can then be compared with the other subsets to decide which model size and which predictor variables minimising RSS and maximise R-squared, thus leaving the model with the best predictive power and accuracy out of all the possible combinations. An issue with best subset selection is that it is very computationally expensive and can lead to overfitting, both due to the nature of its process. The best subset is selected according to which model minimises the Bayesian Information Criteria (BIC) and Mallows Cp (CP) whilst maximising adjusted R-squared. These parameters measure the accuracy of the model but penalise for additional variables.

Table 4 shows which variables were selected for the different model sizes using forward stepwise selection given the training set of data. Forward stepwise starts off with zero predictor variables and then calculates the best variable for a one variable model and then the best two variable model given the first variable and so on.

Table 4

##			TransactionDate	HouseAge	MRTDist	StoreNum	Latitude	Longitude
##	1	(1)	" "	" "	"*"	" "	" "	" "
##	2	(1)	" "	" "	"*"	" "	"*"	" "
##	3	(1)	" "	"*"	"*"	" "	"*"	" "
##	4	(1)	" "	"*"	"*"	"*"	"*"	" "
##	5	(1)	"*"	"*"	"*"	"*"	"*"	" "
##	6	(1)	"*"	"*"	"*"	"*"	"*"	"*"

Table 5 shows the process of forward stepwise selection. The table shows that, as expected, MRTDist is the best variable for a single factor model and that Longitude is not selected until the sixth and final model, reinforcing its weak predictive power.

Looking at the plots in figure 8, all the evaluative parameters are optimised in the five variable model suggesting this is the best sized model given these variables. These findings are similar for backward stepwise selection, which also finds the optimal model size to be five. Backward stepwise works in reverse to forward and starts off with all variables and then removes the weakest variables one by one until all are removed.

Figure 8

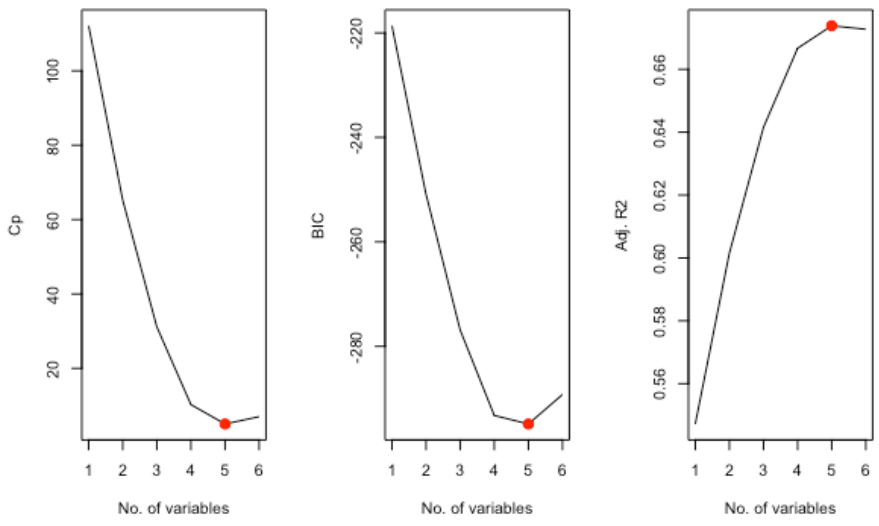


Figure 8 shows the various values of Cp, BIC and Adjusted R-squared for the different sized models examined using forward and backward stepwise subset selection. The red dots highlight that a 5-factor model is optimal.

Having established the 5-factor model is optimal given the training data, a MLR was run. Observing the five linear relationships from the training data allowed this report to predict the test observations, yielding an R-squared of 0.6992486 and a MSE of 61.45236. This shows that R-squared has increased by selecting the optimally sized model and its expected error has fallen to 20.6%.

Ridge and Lasso

Ridge and Lasso seek to improve the fit of a model by shrinking the coefficients of a regression's variables. Least squares methods seek to estimate the beta that minimises RSS, Ridge estimates the beta that minimises;

$$(3)\ RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

Where λ is a penalty parameter. As λ increases the flexibility of model is reduced, consequently reducing variance at the cost of increased bias. The ridge method reduces all coefficients at the same rate and so a higher λ will reduce all beta's instead of just the noisy or weaker variables. To find the optimal lambda, cross-validation was used on the training data for a range of lambda values. The lambda that minimised the cross-validation error was regarded as the optimal and used to predict the values of the test set. The optimal lambda for ridge was found to be 0.02984678, shown in figure 9. With the optimal lambda close to zero we can infer that the least squares model is good approximation for this data set.

Figure 9

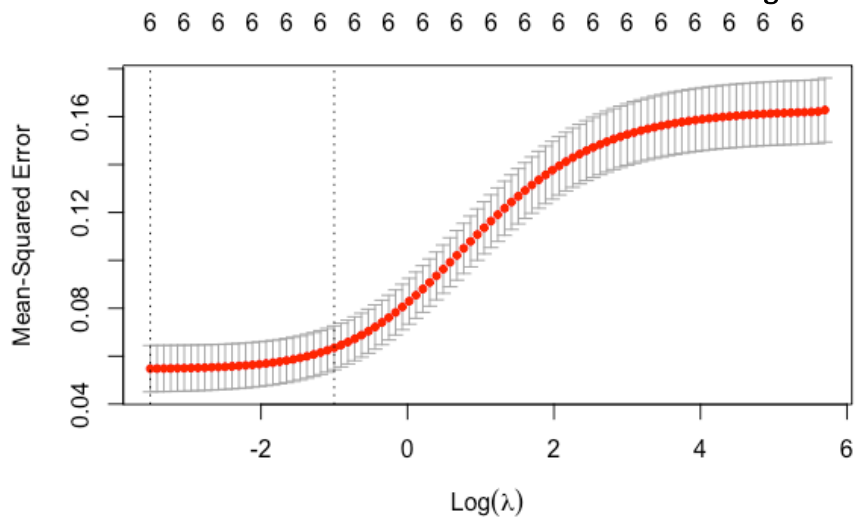


Figure 9 shows the range of training MSE for different values of lambda. The first dotted line shows the optimal lambda with the second showing the best lambda within a single standard deviation, this is equal to 0.3680.

Using the optimal lambda, the ridge coefficients are found to be:

Table 5

Ridge Coefficients	
(Intercept)	-6.82E+02
TransactionDate	1.20E-01
HouseAge	-7.09E-03
MRTDist	-1.12E-04
StoreNum	2.92E-02
Latitude	8.79E+00
Longitude	1.843442e+00

Table 5 shows the coefficients used by the ridge method for the optimal lambda. As expected, these are smaller than the initial MLR coefficients. These were then applied to the test data.

The R-squared when predicting the test observations using these coefficients is found to be 0.6972778 accompanied by a MSE of 62.69749, yielding an expected error of 20.8%. This suggests that the ridge method improves upon the MLR method but doesn't outperform the best subset model A marginally higher MSE compared to the log MLR is likely due to the variance-bias trade-off that Ridge faces.

The Lasso method also attempts to shrink coefficients, however, when its lambda is large enough it forces some coefficients to zero, unlike Ridge. This variable selection technique means that Lasso has better interpretation qualities compared to Ridge. The Lasso method attempts to find coefficients that minimise this equation;

$$(4)\ RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

Figure 10 shows the MSE found for each value of lambda, verified through cross-validation. MSE is minimised using all 6 variables and a lambda equal to 0.002135. This optimal lambda will be used to predict the test set.

Figure 10

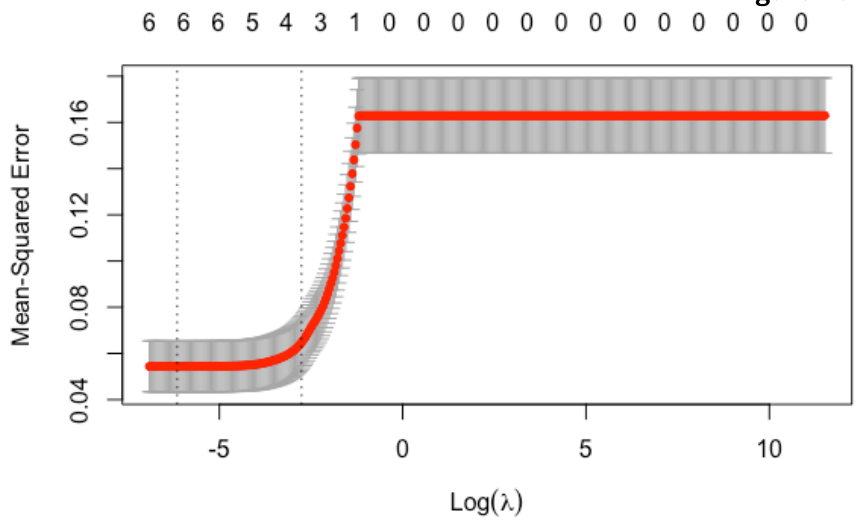


Figure 10 shows the training MSE for a range of log(λ). The first dotted line shows the optimal lambda, which is 0.002135. The second dotted line is the most sparce model within a standard deviation, this lambda is 0.0661947 where only 4 variables are used. This second model would be used if there were many variables but as 6 variables isn't computationally expensive all 6 variables will be estimated using the optimal lambda.

Using the minimum lambda, the optimal coefficients found through the Lasso method are:

Table 6

Lasso Coefficients	
(Intercept)	-4.85E+02
TransactionDate	1.23E-01
HouseAge	-7.40E-03
MRTDist	-1.37E-04
StoreNum	2.80E-02
Latitude	8.74E+00
Longitude	1.949798e-01

Table 6 shows the coefficients found for each variable using the optimal lambda. Similar to the Ridge output, the coefficients used in predicting the test data are smaller than the MLR regression.

R-squared is found to be 0.7001421, slightly higher than the Ridge's equivalent and higher than the MLR R-sqaured. MSE is found to be 61.43279, lower than Ridge and very similar to the Best Subset equivalent. This MSE yields a percentage error of 20.6%. Lasso's performance is in line with expectation, as it performs better on models with fewer predictor variables and with small coefficients. Both of these can be attributed to the Real Estate data set, especially when using the Log of House Prices as the independent variable.

PCR and PLS

The next two methods make use of dimension reduction techniques, which transform the predictor variables and perform a least squares method on these transformed variables. Principal Components Regressions (PCR) constructs "M" principal components into a model which is then regressed using the least squares method. Principal components are new variables that are made up of linear combinations of the original variables. The combinations are calculated in a way that new variables are uncorrelated and are made up of strongly correlated variables whose direction explains as much variation of the dependent variable as possible. PCR seeks to explain most of the variability in the data with as few principal components necessary while not undermining predictive accuracy.

Using cross-validation the optimal number of components was found to be 4, as this adequately minimised adjusted CV error and explained the variance of the dependent variable when applied to the training data, shown in figure 11. For the test data, these 4 principal components yielded an R-squared of 0.6768346 and a MSE of 67.50837. Worse than Best subset and Ridge & Lasso methods. This suggests that PCR main assumption, that the directions in which the predictor variables vary the most are associated with the House Price variation, was not fully met or that more components are needed to model the response.

Figure 11

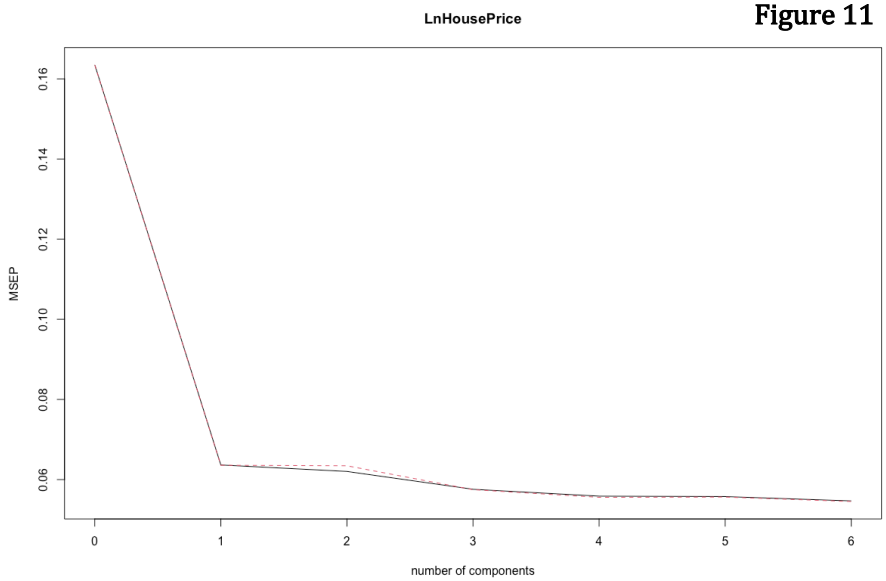


Figure 11 shows the Mean Squared Error of Prediction (MSEP) for a range of principal components, shown on the x-axis. The 4-component model performed best, other than the 6-component model, in terms of MSEP and R-squared. This will be used for prediction.

Partial Least Squares (PLS) improves upon on PCR's assumption by utilising the dependent variables when creating the new set of variables. PLS places higher weights on variables that are more strongly related to the response variable, in this case MRTDist. Using CV to determine the optimal number of principal components, 4 components was found to be the most appropriate. Testing this generated an R-squared of 0.6987883 and a MSE of 61.75218, or an expected error of 20.69%. Therefore, PLS's R-squared is higher than PCR's with a significantly lower MSE. It's improvement in MSE is likely a consequence of its supervised selection of principal components.

GAM

The Generalised Additive Model (GAM) method is an extension of the MLR, however, it allows for non-linear functions while maintaining additivity. This report tunes the splines of the three most important predictor variables as found from the best subset analysis above, these variables are MRTDist, StoreNum and House Age. Tuning splines involves fitting low-degree polynomials over different knots, or regions, of the various independent variables. By preforming cross-validation on the RSS for a range of degrees of freedom (DOF), using the training data the optimal DOF was found to be 11. Figure 12 shows the output of spline tuning.

Figure 12

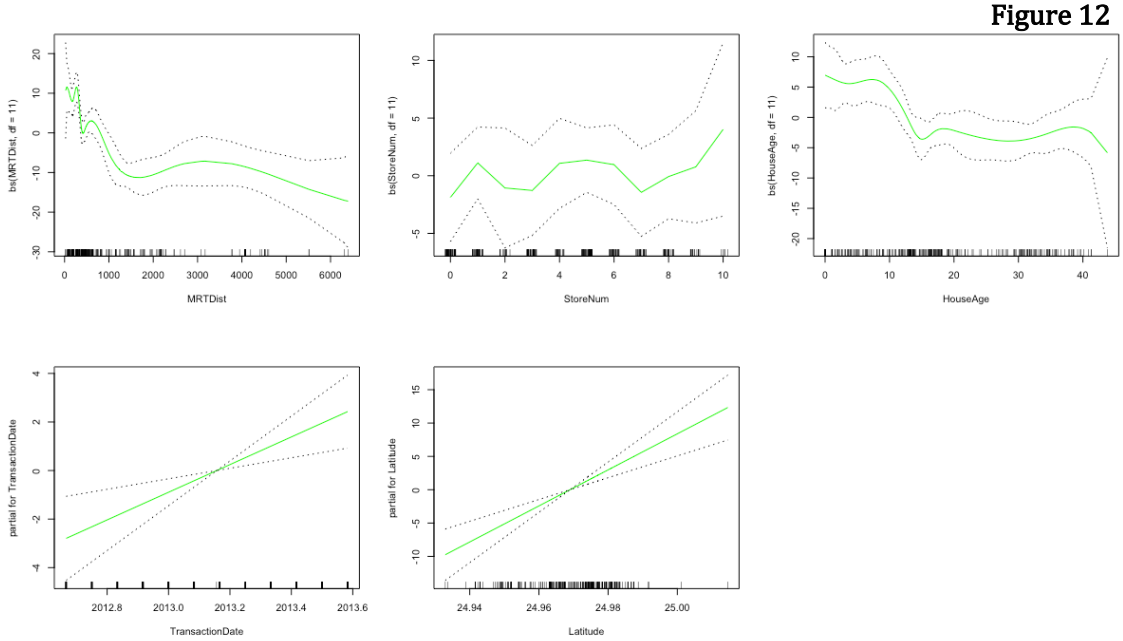


Figure 12 shows the result of tuning the splines of the three most important variables on the optimal model found from the Best Subset Selection model. The top 3 panels have been tuned, shown by their non-linear lines. Relaxing the linear assumption is meant to improve predictive ability. Splines are made smooth by constraining the first and second derivatives to continuousness.

Using this information and the GAM method, the model yielded an R-squared of 0.6677571 and a MSE of 53.11884. This shows that GAM is a good method at reducing expected error, which was 19.2% considerable the lowest so far, whilst maintaining an adequate R-squared. However, this could be a consequence of over-fitting which could lead to poor performance in the future.

Regression Tree

This report now makes use of tree-based methods. The Regression Tree method divides the predictor space into "J" distinct and non-overlapping regions, any observation that falls within that region is assigned the same predicted value, which is equal to the mean of the training observations observed to be within that region. Segmenting the predictor space is done through recursive binary splitting, this is a top-down approach and each split in the space is at the observation that generates the greatest possible reduction in RSS. This process is repeated until some stopping criteria is met. Figure 13 shows the output from using this simplistic method. This method yields an R-squared of 0.6367302 and a MSE of 58.07941.

Figure 13

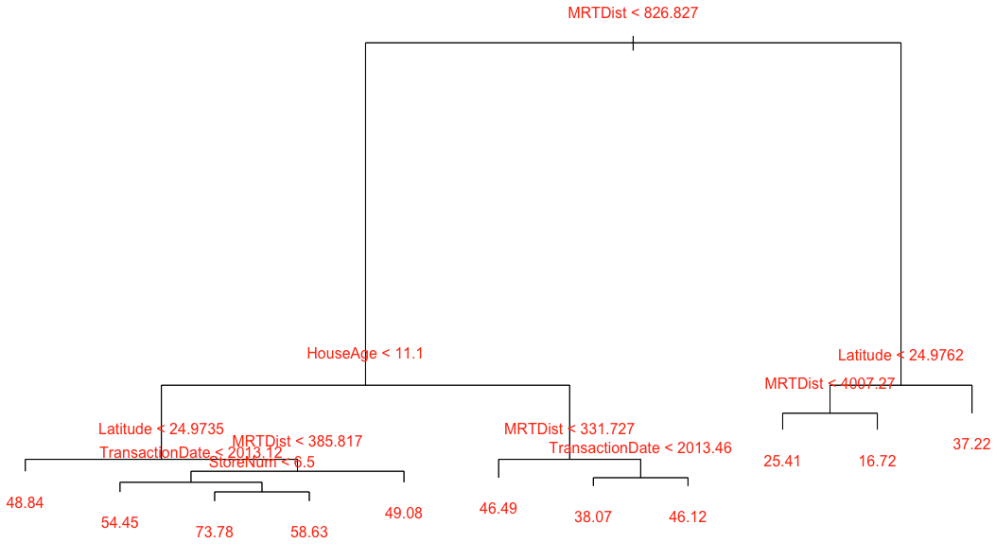


Figure 13 shows the result of a non-pruned regression tree. The terminal nodes outline the assigned value to any observation fitting within that region.

Tree pruning was then carried out as recursive binary splitting is likely to overfit the data. Weakest link pruning seeks to find a smaller tree with fewer splits that reduces variance at a cost of a little increase in bias, a penalty parameter controls the trade-off between the subtree's complexity and its bias. Conducting cross-validation revealed the optimally sized pruned tree, this was found to be 10. Using the optimal degree of complexity, running this pruned tree on the test data resulted in a slightly higher R-squared of 0.6475803 and with a lower MSE of 56.34469, the output of which is shown in figure 14. Therefore, tree pruning has proved to be effective in this instance as it has improved the amount of explained variance and also reduced the expected error of each prediction all while simplifying the complexity of the tree.

Figure 14

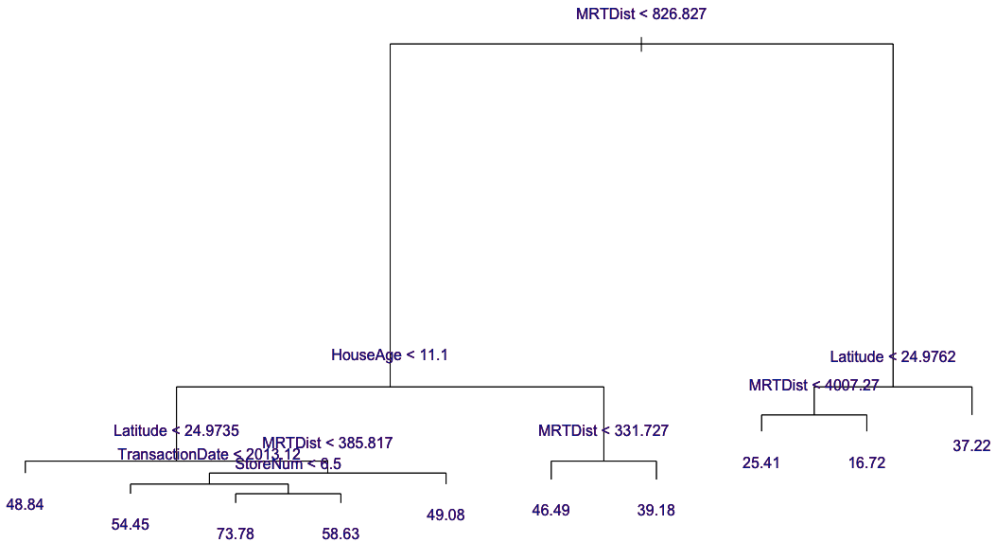


Figure 14 shows the result of tree-pruning, the optimal, pruned tree was found to have 10 terminal nodes compared to the original of 11. Again, each terminal node is the assigned value to any observation fitting its criteria.

Bagging, Random Forests, Boosting

The final methods are derivatives of tree-based prediction. Bagging is another name for Bootstrap aggregation which is used to reduce the variance of machine learning methods. Bagging uses the bootstrap method on the training data to create many different training sets and to build separate prediction models using each training set and then averages the result to improving predictive accuracy. The final bagging model is equal to:

$$(5)\ \widehat{f_{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f^{*b}}(x)$$

Where $\widehat{f^{*b}}$ represents each predicted tree. Each individual tree has high variance and low bias but by averaging these trees variance is reduced and predictive accuracy is improved. By generating 500 separate predictor trees and then average them, the test data was analysed and predicted. This yielded an R-squared of 0.7031235 and a MSE of 47.46447. A huge improvement on any method used so far as it has a considerably lower MSE, with an expected error of 18.14%, and explains the highest amount of House Price variation, 70.31%. This shows the power of bagging and the improved predictive power it can have over simple pruned regression trees.

Random Forests use a very similar process to Bagging but differs in that its predictive trees are far more de-correlated due to the fact that at each node only a pre-determined number of randomly sampled variables are tested rather than every predictor variable. This method generates more variable trees and thus greater variance that when averaged improves predictive power and lowers MSE. Using this method, having only 2 randomly selected predictor variables at each node proved most effective, generating a test MSE of only 40.31746, and consequently an expected error of 16.72%. It also dramatically improved the model's predictive power, increasing R-squared to 0.747826. This is likely due to the strengths of Random Forest on data sets with some very strong predictors, in this case MRTDist.

Boosting is the final method to be used on the Real Estate data set. Boosting works by growing trees sequentially, where each tree uses information from previously grown trees. Unlike bagging, the construction of a tree depends strongly on the trees already created, given an existing tree the next decision tree is fit to the residuals of the model, rather than the value of House Prices. Figure 15 shows the output of a for-loop estimating the various values of lambda, it found that test MSE was minimised when lambda was equal to 0.005623413, which suggests that many trees need to be used to improve results. Using 1000 trees and the optimal lambda the MSE on the test data was found to be 50.44264 with an R-squared of 0.6844959. These are significantly worse than the random forest method.

Figure 15

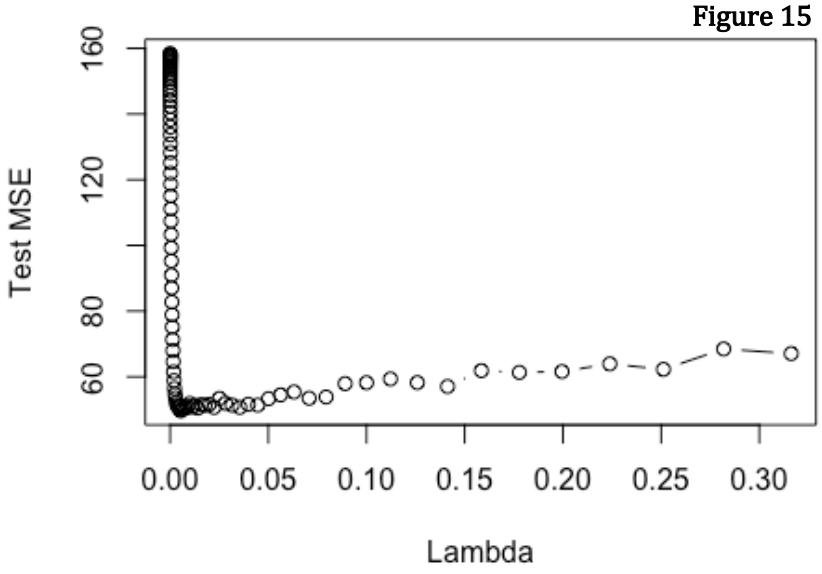


Figure 15 shows the output of a for-loop estimating MSE using a range of Lambda values on the Boosting method. It found that MSE drastically reduced initially then steadily increased from its bottom when Lambda is equal to 0.005623413. This minimum is used for the optimal boosting model.

Conclusion

We have now outlined the results of all 12 different machine learning methods. Figure 16 shows the various R-squared values for each method, all methods achieved adequate results, all explaining more variance in House Prices than they didn't.

Figure 16

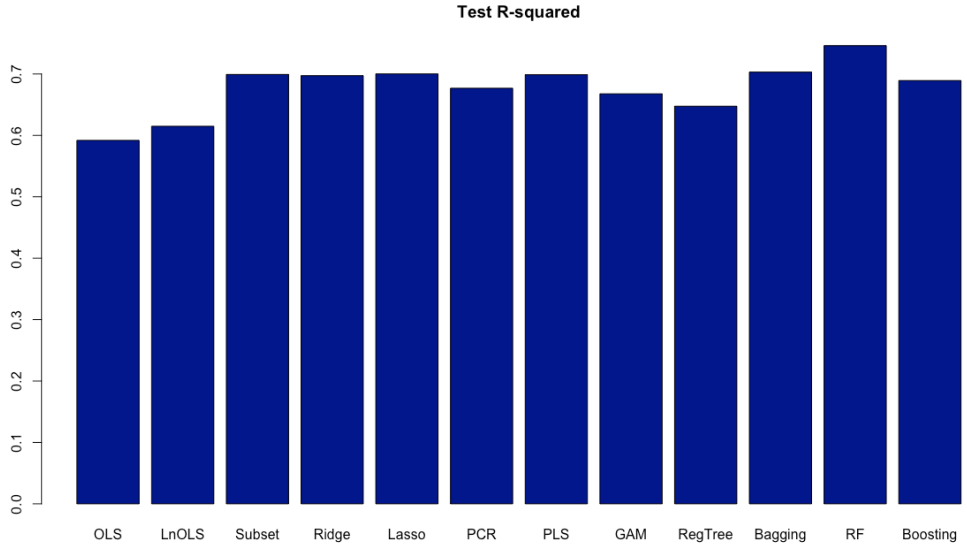


Figure 16 plots the R-squared of each method. They all achieved an R-squared of 0.5 and above with many above the 0.6 level. 3 had R-squareds above 0.7, these were Lasso, Bagging and Random Forest.

Test MSE results were more variable, shown in figure 17, with many clustering around the 20% expected error mark, meaning for a given observation the true value of a property could be 20% greater or lower than the predicted value, quite a considerable margin for error that does impact the usability of most of the methods. However, to conclude on this further assessment of the Sindian District's housing market should be undertaken, as an 20% discount on a properties intrinsic value may be regular.

Figure 17

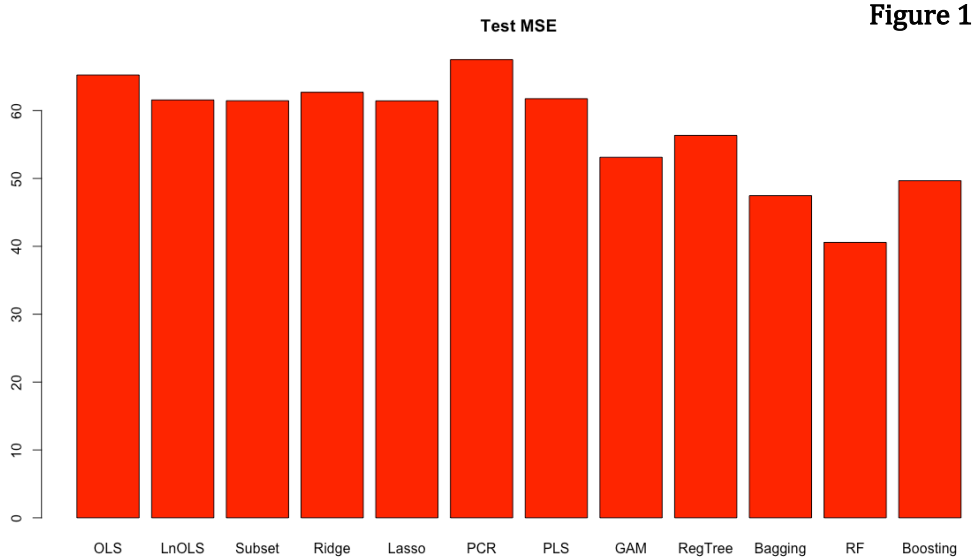


Figure 17 plots the Test MSE for each method. Random Forest performed the best by a considerable amount, with PCR performing the worst. The first 7 methods achieved an MSE of around 60 with the more advanced latter methods achieving 50 or less.

Table 7 outlines a simple ranking of each method. The MLR method using the original House Price values performed the worst, with an R-squared of 0.5919161 and expected error of 21.27%, as the simplest method applied to the data set this is unsurprising. Despite, being the poorest performing method, it still fits almost 60% of House Price variation which when considering the ease of interpretation make it an appropriate method to predict the data.

Table 7

Machine Learning Method Ranking 37.98019

Number	Method	R-squared	R-squared Ranking	MSE	Expected Error	MSE Ranking	Overall
1	MLR	0.5919161	12	65.24426	21.27%	11	12
2	Ln MLR	0.6149192	11	61.56653	20.66%	8	10
3	Best Subset Selection	0.6992486	4	61.45236	20.64%	7	5
4	Ridge	0.6972778	6	62.69749	20.85%	10	8
5	Lasso	0.7001421	3	61.43279	20.64%	6	5
6	PCR	0.6768346	8	67.50837	21.63%	12	10
7	PLS	0.6987883	5	61.75218	20.69%	9	7
8	GAM	0.6677571	9	53.11884	19.19%	4	7
9	Regression Tree	0.6475803	10	56.34469	19.76%	5	8
10	Bagging	0.7031235	2	47.46447	18.14%	2	2
11	Random Forest	0.7462013	1	40.57722	16.77%	1	1
12	Boosting	0.6893582	7	49.66527	18.56%	3	5

Table 7 plots the outcome of a simple ranking of all methods. It combines each MSE and R-squared value, ranking each then averaging them to produce a final rank. Random Forest was the best with Bagging second. Three methods drew for 3rd place. The first method, MLR, performed the worst overall with PCR second to last.

The Random Forest method performed best, by a notable margin. It explained almost 75% of House Price variance, very impressive given the simplicity of the predictor variables, with an expected error of 16.77%. By decorrelating each tree from the next Random Forest was able to drastically reduce variance whilst maintaining an incredible level of accuracy. However, although it performed the best, it is important to consider the difficulty of interpreting this method. This could mean for real world application perhaps Lasso or Best Subset Selection could be more appropriate as they are easier to interpret, assigning interpretable coefficients to each predictor variable.

To answer the primary prediction question, this report managed to predict House Prices in the Sindian District of New Taipei City with a good degree of accuracy using only house specific and local amenities predictors. Using the optimal method, houses could be predicted with only a 17% expected error either above or below the true value, this is with almost 75% of the test set data fitting the predicted model generated from the training data. To answer the secondary prediction question, a variable importance plot is used. Using the Random Forest model, Latitude was found to be the most important predictor variable followed by MRTDist and then House Age. Therefore, distance from an MRT station is very important in influencing House Prices, as well as its neighboured assume to be proxied by Latitude. Perhaps the New Taipei government could seek to increase Housing value by increasing the number of MRT stations in the city. However, their effect may be exhaustive, diminishing the more each variable is increased.

Appendix

ML-Final-Assignment-Final.R

matthewtlane

2021-01-19

```
# Final Assignment #

rm(list = ls())

library(MASS)
library(ISLR)
library(tree)
library(randomForest)

## randomForest 4.6-14

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

library(gbm)

## Loaded gbm 2.1.8

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.0-2

library(car)

## Loading required package: carData

library(leaps)
library(pls)

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

library(boot)

##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit
```

```
library(leaps)
library(splines)
library(gam)

## Loading required package: foreach

## Loaded gam 1.20

library(Hmisc)

## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:boot':
##
##     melanoma

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:boot':
##
##     aml

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

# Adding the labour data set to RStudio
library(readxl)
RealEstateValuation_Data_ <-
  read_excel("Real estate valuation data set.xlsx")
View(RealEstateValuation_Data_)

#renaming the variables so its easier to call upon in code
RealEstateValuation_Data_$TransactionDate <-
  (RealEstateValuation_Data_$`X1 transaction date`)
RealEstateValuation_Data_$HouseAge <-
  (RealEstateValuation_Data_$`X2 house age`)
```

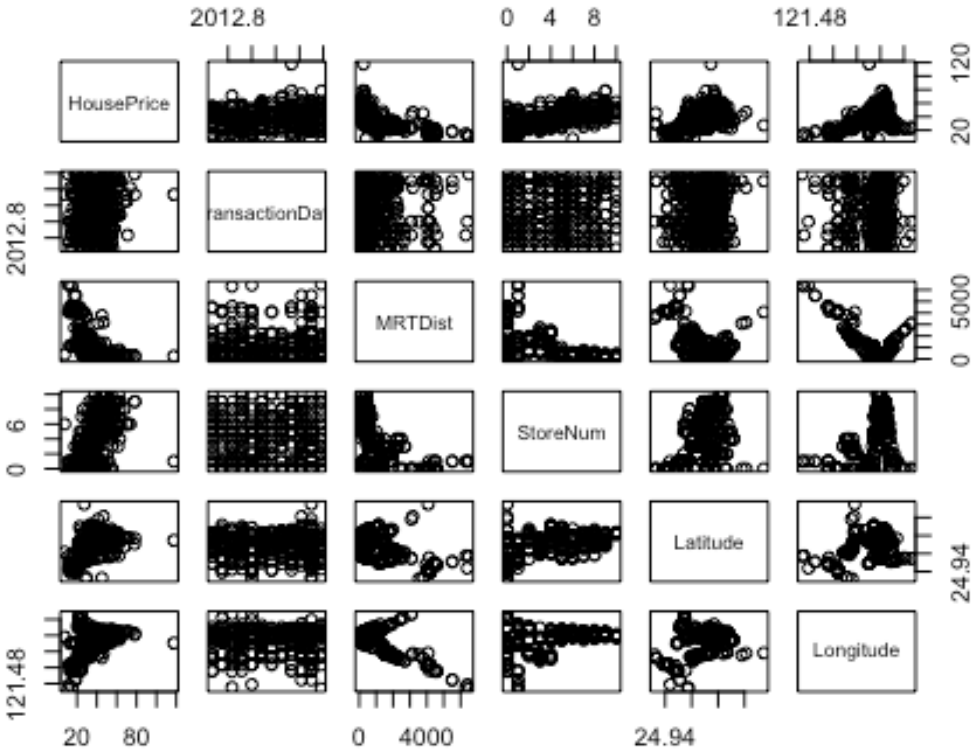
```
RealEstateValuation_Data_$MRTDist <-
  (RealEstateValuation_Data_$`X3 distance to the nearest MRT station`)
RealEstateValuation_Data_$StoreNum <-
  (RealEstateValuation_Data_$`X4 number of convenience stores`)
RealEstateValuation_Data_$Latitude <-
  (RealEstateValuation_Data_$`X5 latitude`)
RealEstateValuation_Data_$Longitude <-
  (RealEstateValuation_Data_$`X6 longitude`)
RealEstateValuation_Data_$HousePrice <-
  (RealEstateValuation_Data_$`Y house price of unit area`)
RealEstateValuation_Data_$LnHousePrice <-
  log(RealEstateValuation_Data_$`Y house price of unit area`)

#Getting rid of original column, as to not have duplicates
RealEstateValuation_Data_$`X1 transaction date` = NULL
RealEstateValuation_Data_$`X2 house age` = NULL
RealEstateValuation_Data_$`X3 distance to the nearest MRT station` = NULL
RealEstateValuation_Data_$`X4 number of convenience stores` = NULL
RealEstateValuation_Data_$`X5 latitude` = NULL
RealEstateValuation_Data_$`X6 longitude` = NULL
RealEstateValuation_Data_$`Y house price of unit area` = NULL

View(RealEstateValuation_Data_)


##########Data and Exploratory Analysis##########

describe(RealEstateValuation_Data_) #gives us: the top 5 max values, botto
m 5 minimum values, mean, quantiles, observations, missing observations.

## RealEstateValuation_Data_
##
##  9  Variables      414  Observations
## ---------------------------------------------------------------------------
---------
## No
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0      414        1    207.5    138.3    21.65    42.30
##      .25      .50      .75      .90      .95
##   104.25   207.50   310.75   372.70   393.35
##
## lowest :   1   2   3   4   5, highest: 410 411 412 413 414
## ---------------------------------------------------------------------------
---------
## TransactionDate
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0       12    0.991     2013   0.3239     2013     2013
##      .25      .50      .75      .90      .95
##     2013     2013     2013     2014     2014
##
## lowest : 2012.667 2012.750 2012.833 2012.917 2013.000
## highest: 2013.250 2013.333 2013.417 2013.500 2013.583
##
## Value      2012.667 2012.750 2012.833 2012.917 2013.000 2013.083 2013.1
```

```
67
## Frequency        30       27       31       38       28       46
25
## Proportion    0.072    0.065    0.075    0.092    0.068    0.111    0.0
60
##
## Value      2013.250 2013.333 2013.417 2013.500 2013.583
## Frequency        32       29       58       47       23
## Proportion    0.077    0.070    0.140    0.114    0.056
## ----------------------------------------------------------------------
---------
## HouseAge
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0      236        1    17.71    12.93    1.100    3.500
##      .25      .50      .75      .90      .95
##    9.025   16.100   28.150   34.670   37.735
##
## lowest :  0.0  1.0  1.1  1.5  1.7, highest: 40.9 41.3 41.4 42.7 43.8
## ----------------------------------------------------------------------
---------
## MRTDist
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0      259        1     1084     1205    90.46   157.61
##      .25      .50      .75      .90      .95
##   289.32   492.23  1454.28  2697.66  4082.01
##
## lowest :   23.38284   49.66105   56.47425   57.58945   82.88643
## highest: 4605.74900 5512.03800 6306.15300 6396.28300 6488.02100
## ----------------------------------------------------------------------
---------
## StoreNum
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0       11    0.986    4.094    3.371        0        0
##      .25      .50      .75      .90      .95
##        1        4        6        8        9
##
## lowest :  0  1  2  3  4, highest:  6  7  8  9 10
##
## Value          0     1     2     3     4     5     6     7     8     9
10
## Frequency     67    46    24    46    31    67    37    31    30    25
10
## Proportion 0.162 0.111 0.058 0.111 0.075 0.162 0.089 0.075 0.072 0.060
0.024
## ----------------------------------------------------------------------
---------
## Latitude
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0      234        1    24.97  0.01382    24.95    24.95
##      .25      .50      .75      .90      .95
##    24.96    24.97    24.98    24.98    24.99
##
## lowest : 24.93207 24.93293 24.93363 24.93885 24.94155
## highest: 24.99156 24.99176 24.99800 25.00115 25.01459
```

```
## ------------------------------------------------------------------------
---------
## Longitude
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0      232        1    121.5  0.01601    121.5    121.5
##      .25      .50      .75      .90      .95
##    121.5    121.5    121.5    121.5    121.5
##
## lowest : 121.4735 121.4752 121.4788 121.4846 121.4951
## highest: 121.5539 121.5548 121.5596 121.5617 121.5663
## ------------------------------------------------------------------------
---------
## HousePrice
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0      270        1    37.98    15.13    16.49    21.02
##      .25      .50      .75      .90      .95
##    27.70    38.45    46.60    54.94    59.17
##
## lowest :   7.6  11.2  11.6  12.2  12.8, highest:  71.0  73.6  78.0  78.
3 117.5
## ------------------------------------------------------------------------
---------
## LnHousePrice
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      414        0      270        1    3.567   0.4338    2.803    3.045
##      .25      .50      .75      .90      .95
##    3.321    3.649    3.842    4.006    4.080
##
## lowest : 2.028148 2.415914 2.451005 2.501436 2.549445
## highest: 4.262680 4.298645 4.356709 4.360548 4.766438
## ------------------------------------------------------------------------
---------

# Correlation Matrix
cor(RealEstateValuation_Data_) #The most correlated with HousePrice: MRTDi
st (negative), StoreNum (positive), Latitude (positive), longitude (positi
ve).

##                          No TransactionDate    HouseAge     MRTDist
## No               1.00000000    -0.048634447 -0.03280811 -0.01357349
## TransactionDate -0.04863445     1.000000000  0.01754234  0.06088009
## HouseAge        -0.03280811     0.017542341  1.00000000  0.02562205
## MRTDist         -0.01357349     0.060880095  0.02562205  1.00000000
## StoreNum        -0.01269895     0.009544199  0.04959251 -0.60251914
## Latitude        -0.01010966     0.035016305  0.05441990 -0.59106657
## Longitude       -0.01105928    -0.041065078 -0.04852005 -0.80631677
## HousePrice      -0.02858717     0.087529272 -0.21056705 -0.67361286
## LnHousePrice    -0.01644544     0.075495103 -0.18925434 -0.75233739
##                     StoreNum    Latitude   Longitude  HousePrice LnHous
ePrice
## No              -0.012698946 -0.01010966 -0.01105928 -0.02858717  -0.01
644544
## TransactionDate  0.009544199  0.03501631 -0.04106508  0.08752927   0.07
549510
```

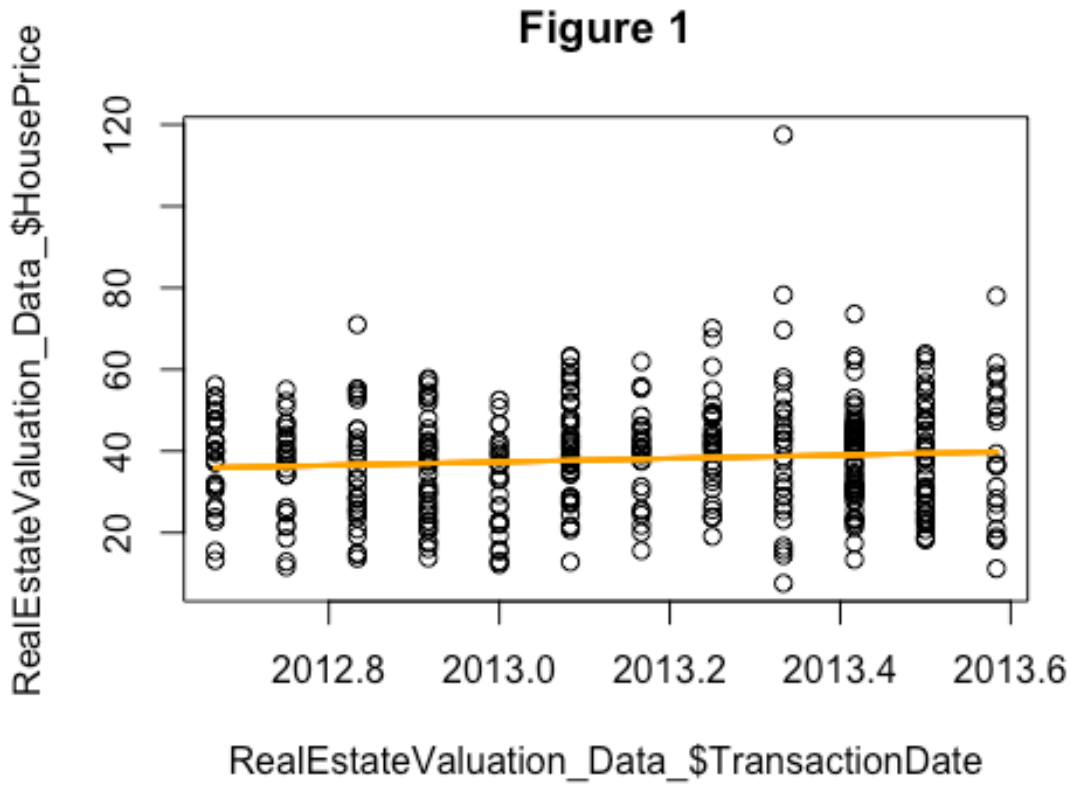
```
## HouseAge          0.049592513  0.05441990 -0.04852005 -0.21056705  -0.18
925434
## MRTDist          -0.602519145 -0.59106657 -0.80631677 -0.67361286  -0.75
233739
## StoreNum          1.000000000  0.44414331  0.44909901  0.57100491   0.59
882627
## Latitude          0.444143306  1.00000000  0.41292394  0.54630665   0.61
797918
## Longitude         0.449099007  0.41292394  1.00000000  0.52328651   0.59
448060
## HousePrice        0.571004911  0.54630665  0.52328651  1.00000000   0.96
205720
## LnHousePrice      0.598826265  0.61797918  0.59448060  0.96205720   1.00
000000
```

```
pairs(HousePrice ~ TransactionDate + MRTDist + StoreNum + Latitude + Longi
tude,
      data = RealEstateValuation_Data_)
```

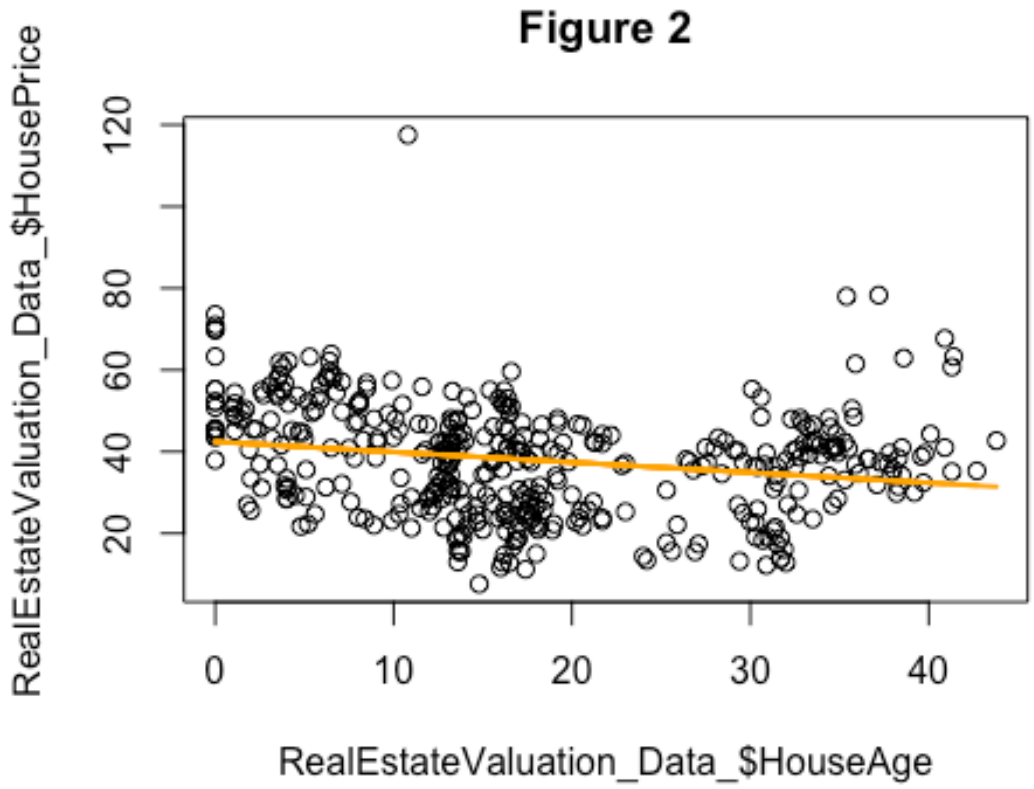



```
#Scatterplots of HousePrice with the explanatory variables to ascertain an
idea of a functional form.
par(mfrow = c(1, 1))

plot(
  RealEstateValuation_Data_$`TransactionDate`,
  RealEstateValuation_Data_$HousePrice
)
```

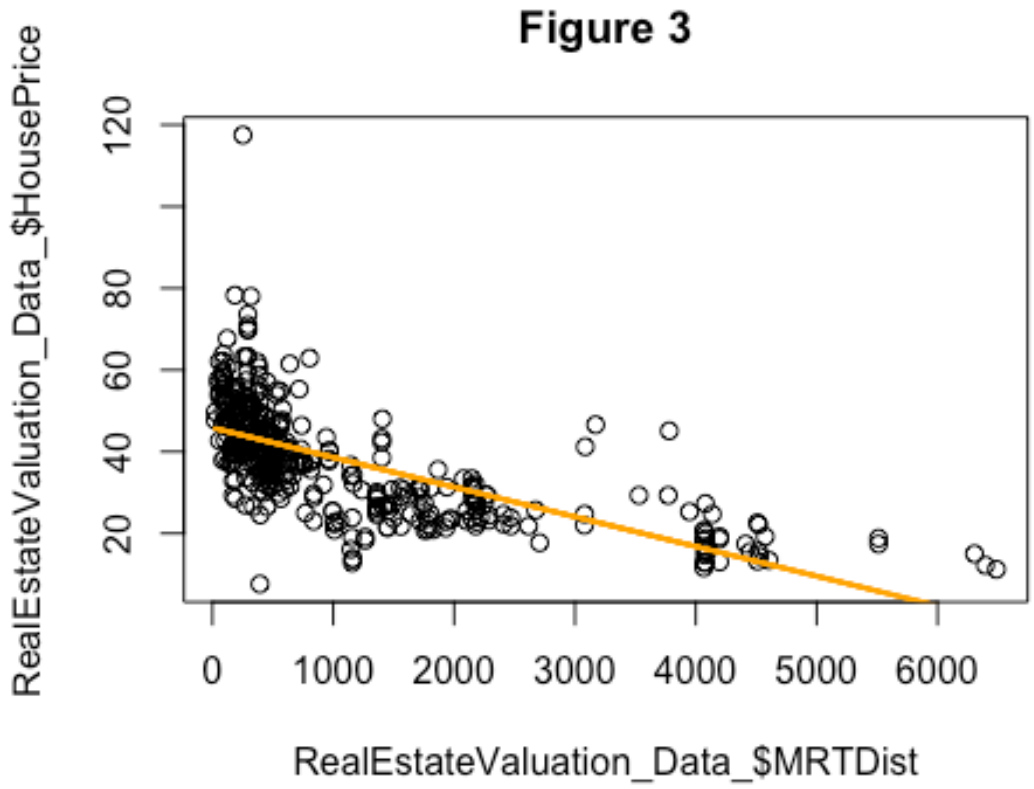
```r
lines(
  RealEstateValuation_Data_$`TransactionDate`,
  predict(
    lm(HousePrice ~ TransactionDate, data = RealEstateValuation_Data_)
  ),
  type = "l",
  col = "orange1",
  lwd = 2
)
title("Figure 1")
```




```r
plot(RealEstateValuation_Data_$`HouseAge`,
     RealEstateValuation_Data_$HousePrice) #Looks like a non-linear relati
onship. Different model may increase R squared.
lines(
  RealEstateValuation_Data_$`HouseAge`,
  predict(lm(HousePrice ~ HouseAge, data = RealEstateValuation_Data_)),
  type = "l",
  col = "orange1",
  lwd = 2
)
title("Figure 2")
```

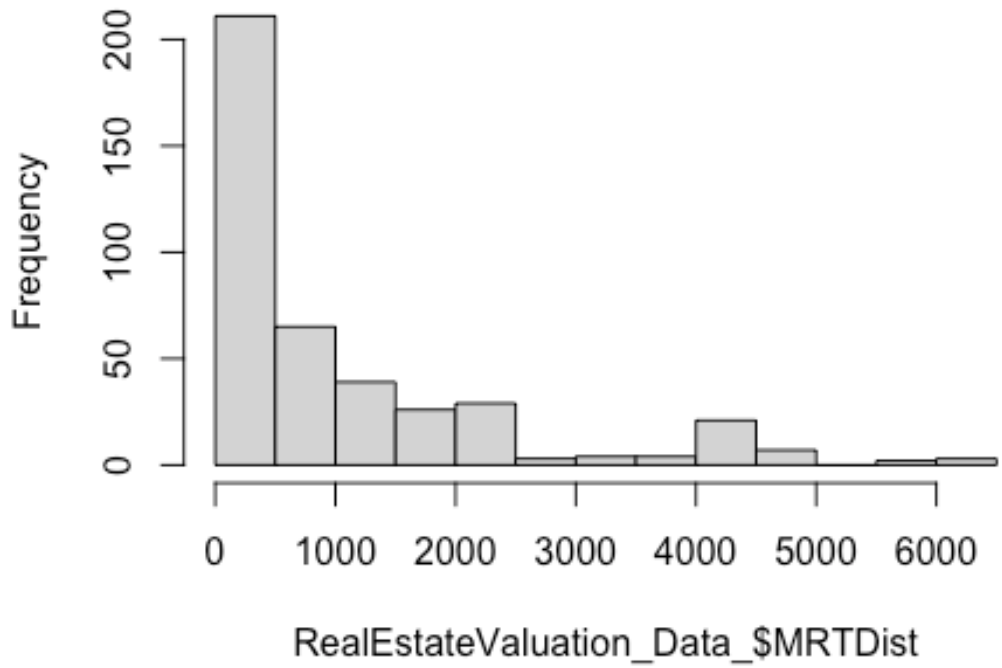



```
plot(x = RealEstateValuation_Data_$MRTDist, y = RealEstateValuation_Data_$
HousePrice) #Looks like a non-linear relationship - different method likel
y to increase R squared.
lines(
  RealEstateValuation_Data_$MRTDist,
  predict(lm(HousePrice ~ MRTDist, data = RealEstateValuation_Data_)),
  type = "l",
  col = "orange1",
  lwd = 2
)
title("Figure 3")
```

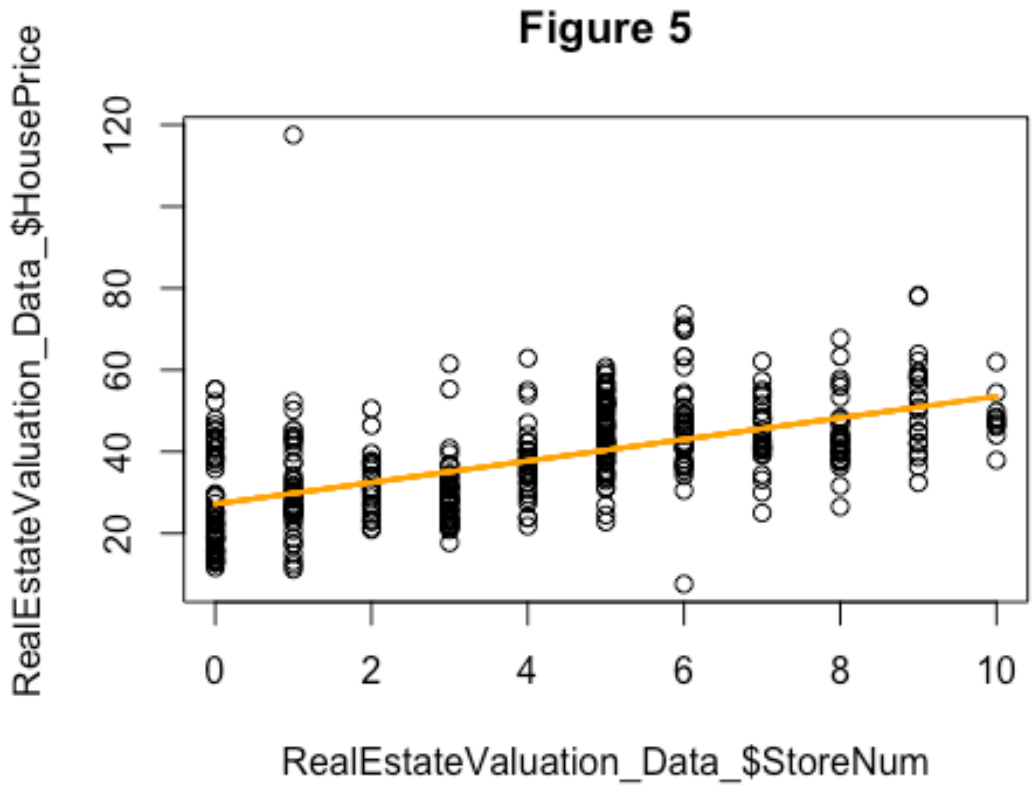



```
hist(RealEstateValuation_Data_$'MRTDist', main = paste("Figure 4"))
```

Figure 4

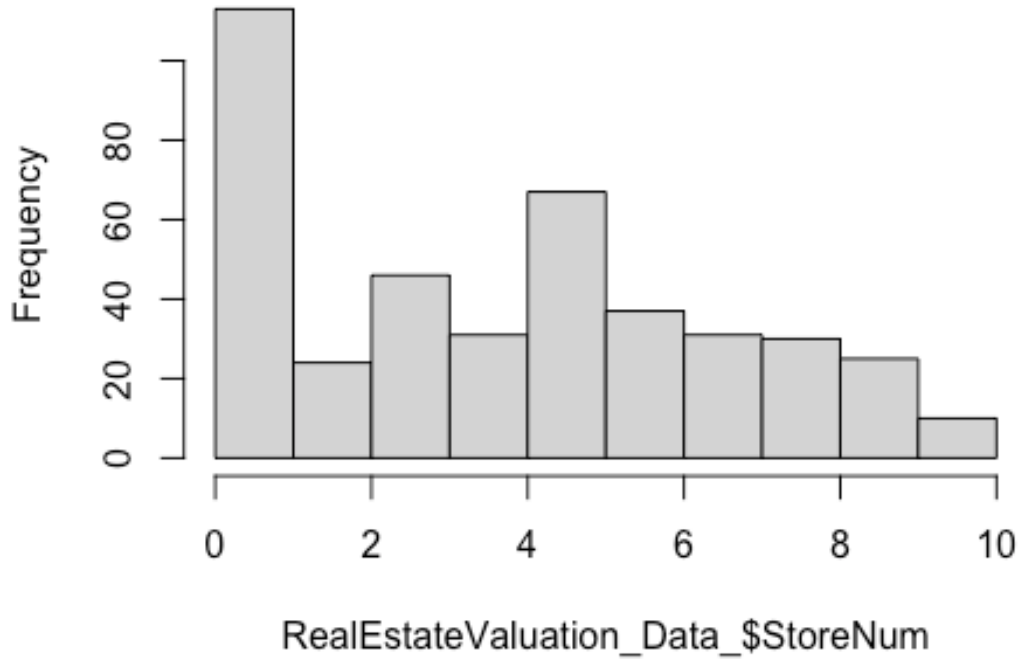



```
plot(RealEstateValuation_Data_$StoreNum,
     RealEstateValuation_Data_$HousePrice) #Looks like a linear model fits
it well.
lines(
  RealEstateValuation_Data_$StoreNum,
  predict(lm(HousePrice ~ StoreNum, data = RealEstateValuation_Data_)),
  type = "l",
  col = "orange1",
  lwd = 2
)
title("Figure 5")
```

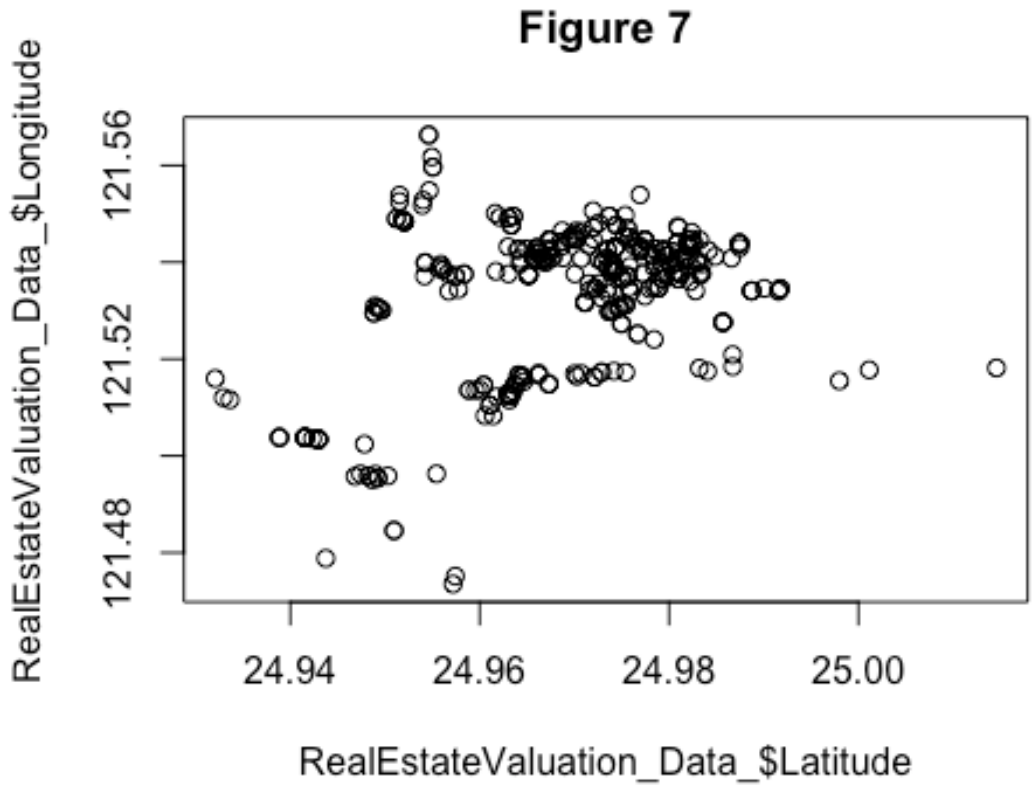



```
hist(RealEstateValuation_Data_$`StoreNum`, main = paste("Figure 6"))
```

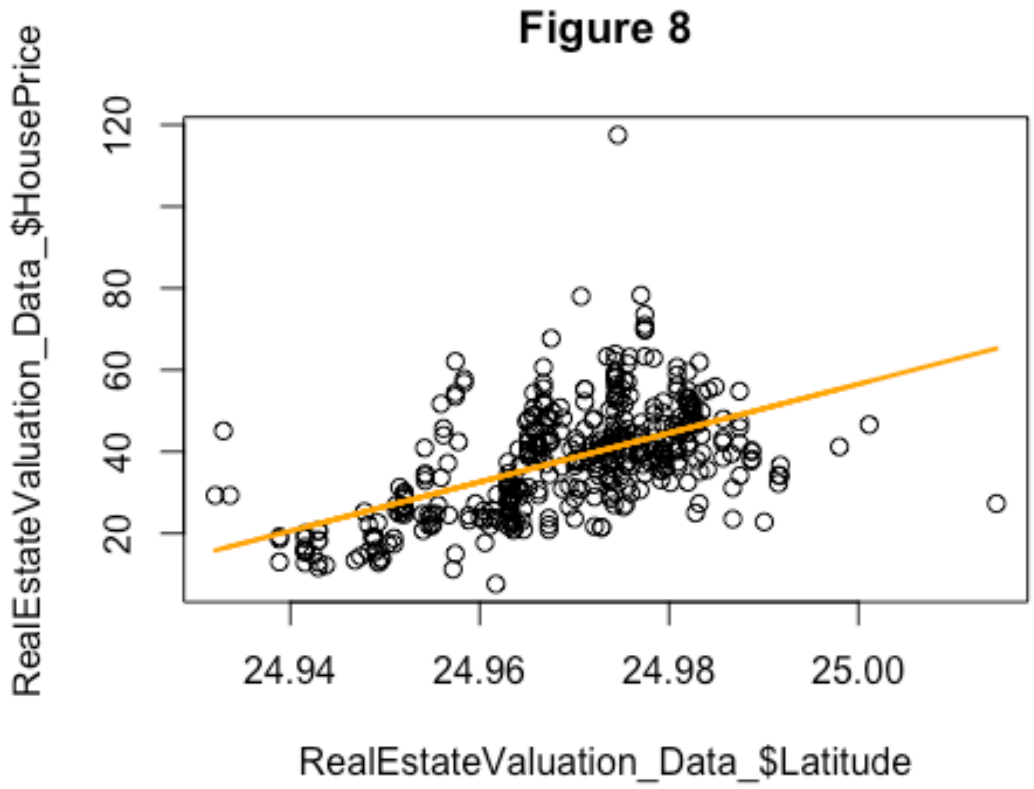
Figure 6



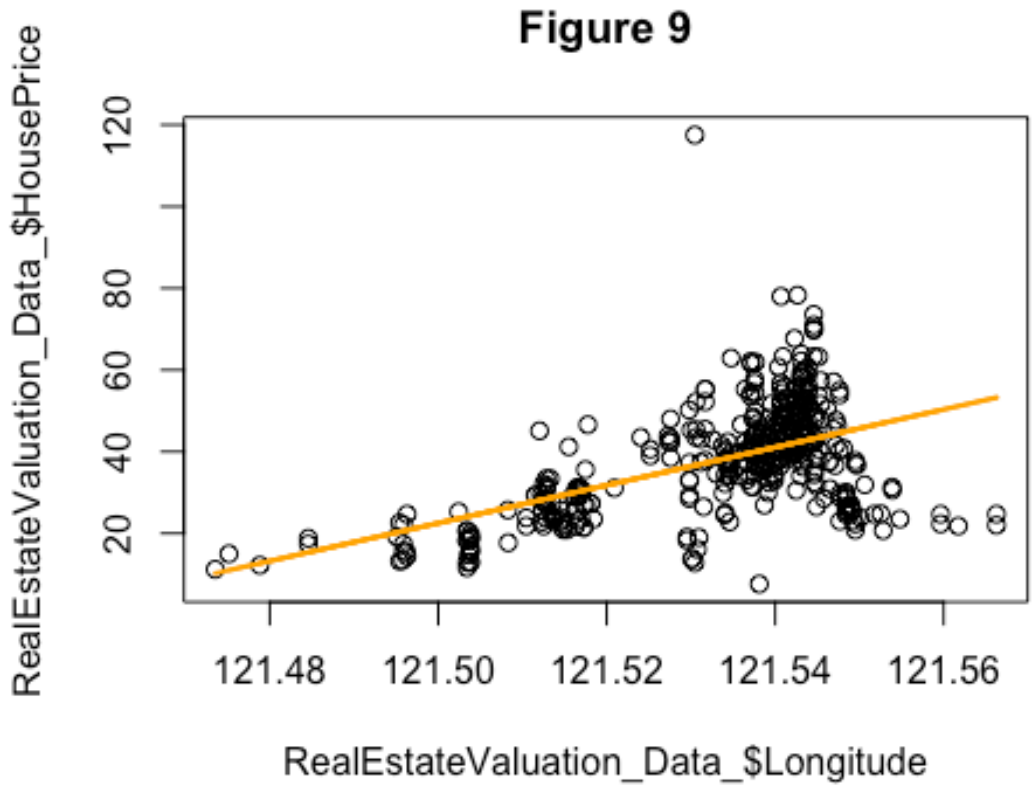

```
plot(RealEstateValuation_Data_$Latitude,
     RealEstateValuation_Data_$Longitude,
     data = RealEstateValuation_Data_)

## Warning in plot.window(...): "data" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "data" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "data" is
not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "data" is
not a
## graphical parameter

## Warning in box(...): "data" is not a graphical parameter

## Warning in title(...): "data" is not a graphical parameter

title("Figure 7")
```




```r
plot(RealEstateValuation_Data_$`Latitude`,
     RealEstateValuation_Data_$HousePrice)
lines(
  RealEstateValuation_Data_$`Latitude`,
  predict(lm(HousePrice ~ Latitude, data = RealEstateValuation_Data_)),
  type = "l",
  col = "orange1",
  lwd = 2
)
title("Figure 8")
```




```r
plot(RealEstateValuation_Data_$`Longitude`,
     RealEstateValuation_Data_$HousePrice)
lines(
  RealEstateValuation_Data_$`Longitude`,
  predict(lm(HousePrice ~ Longitude, data = RealEstateValuation_Data_)),
  type = "l",
  col = "orange1",
  lwd = 2
)
title("Figure 9")
```




```
#Simple linear regression to determine explanatory power by individual var
iables on house price (in linear form).

sm.trans = lm(HousePrice ~ TransactionDate, data = RealEstateValuation_Dat
a_)
summary(sm.trans) #very insignificant p-value rejected at 5% significance

##
## Call:
## lm(formula = HousePrice ~ TransactionDate, data = RealEstateValuation_D
ata_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.159 -10.082   0.923   8.528  78.741
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -8464.260   4767.178  -1.776   0.0765 .
## TransactionDate     4.223      2.368   1.783   0.0752 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.57 on 412 degrees of freedom
## Multiple R-squared:  0.007661,   Adjusted R-squared:  0.005253
## F-statistic: 3.181 on 1 and 412 DF,  p-value: 0.07524
```

```r
sm.age = lm(HousePrice ~ HouseAge, data = RealEstateValuation_Data_)
summary(sm.age) #Not much explanatory power - R squared is 0.04434 but P-v
alue is significant at all levels.

##
## Call:
## lm(formula = HousePrice ~ HouseAge, data = RealEstateValuation_Data_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.113 -10.738   1.626   8.199  77.781
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.43470    1.21098  35.042  < 2e-16 ***
## HouseAge    -0.25149    0.05752  -4.372 1.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.32 on 412 degrees of freedom
## Multiple R-squared:  0.04434,    Adjusted R-squared:  0.04202
## F-statistic: 19.11 on 1 and 412 DF,  p-value: 1.56e-05

sm.MRT = lm(HousePrice ~ MRTDist, data = RealEstateValuation_Data_)
summary(sm.MRT) #High explanatory power - R squared is 0.4538, P-value sig
nificant at all levels.

##
## Call:
## lm(formula = HousePrice ~ MRTDist, data = RealEstateValuation_Data_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.396  -6.007  -1.195   4.831  73.483
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 45.8514271  0.6526105   70.26   <2e-16 ***
## MRTDist     -0.0072621  0.0003925  -18.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.07 on 412 degrees of freedom
## Multiple R-squared:  0.4538, Adjusted R-squared:  0.4524
## F-statistic: 342.2 on 1 and 412 DF,  p-value: < 2.2e-16

sm.store = lm(HousePrice ~ StoreNum, data = RealEstateValuation_Data_)
summary(sm.store) # High explanatory power - R squared is 0.326, P-value s
ignificant at all levels.

##
## Call:
## lm(formula = HousePrice ~ StoreNum, data = RealEstateValuation_Data_)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -35.407  -7.341  -1.788   5.984  87.681
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  27.1811     0.9419   28.86   <2e-16 ***
## StoreNum      2.6377     0.1868   14.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.18 on 412 degrees of freedom
## Multiple R-squared:  0.326,  Adjusted R-squared:  0.3244
## F-statistic: 199.3 on 1 and 412 DF,  p-value: < 2.2e-16
```

```
sm.lat = lm(HousePrice ~ Latitude, data = RealEstateValuation_Data_)
summary(sm.lat) #Reasonably high explanatory power - R squared is 0.2985,
P-value is significant at all levels.
```

```
##
## Call:
## lm(formula = HousePrice ~ Latitude, data = RealEstateValuation_Data_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.969  -7.347  -1.392   5.685  76.184
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14917.68    1129.66  -13.21   <2e-16 ***
## Latitude       598.97      45.24   13.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.41 on 412 degrees of freedom
## Multiple R-squared:  0.2985, Adjusted R-squared:  0.2967
## F-statistic: 175.3 on 1 and 412 DF,  p-value: < 2.2e-16
```

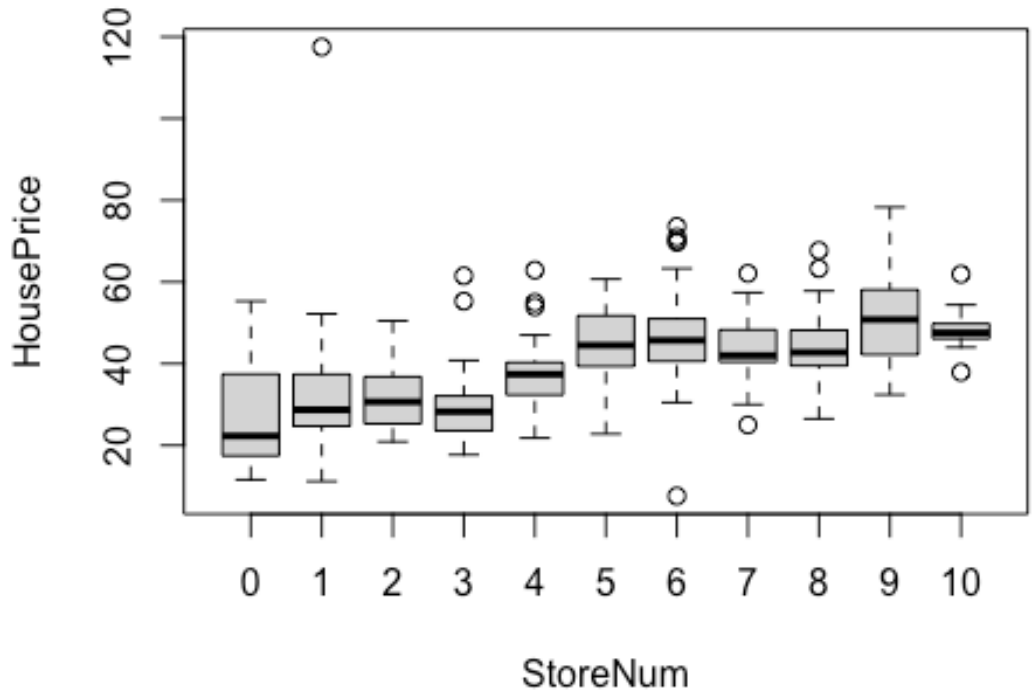
```
sm.long = lm(HousePrice ~ Longitude, data = RealEstateValuation_Data_)
summary(sm.long) #Reasonably high explanatory power - R squared is 0.2738,
P-value is significant at all levels.
```

```
##
## Call:
## lm(formula = HousePrice ~ Longitude, data = RealEstateValuation_Data_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.588  -5.693  -0.417   6.157  80.866
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -56345.57    4523.60  -12.46   <2e-16 ***
## Longitude      463.93      37.22   12.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 11.61 on 412 degrees of freedom
## Multiple R-squared:  0.2738, Adjusted R-squared:  0.2721
## F-statistic: 155.4 on 1 and 412 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
hist(RealEstateValuation_Data_$`HouseAge`)
hist(RealEstateValuation_Data_$'MRTDist')
hist(RealEstateValuation_Data_$`TransactionDate`)
hist(RealEstateValuation_Data_$`StoreNum`)
```


```
par(mfrow = c(1, 1))
boxplot.Price.Store <-
  boxplot(HousePrice ~ StoreNum, data = RealEstateValuation_Data_)
```




```
boxplot.Price.Store$stats
```

```
##       [,1] [,2]  [,3]  [,4]  [,5]  [,6] [,7] [,8]  [,9] [,10] [,11]
## [1,] 11.60 11.2 20.90 17.70 21.80 22.80 30.5 30.0 26.50  32.4 44.00
## [2,] 17.45 24.7 25.30 23.60 32.35 39.35 40.6 40.4 39.50  42.2 46.10
## [3,] 22.30 28.7 30.65 28.25 37.40 44.50 45.7 42.0 42.75  50.8 47.55
## [4,] 37.45 37.4 36.80 32.10 40.30 51.75 51.0 48.3 48.20  58.1 49.80
## [5,] 55.30 52.2 50.50 40.80 47.00 60.70 63.3 57.4 57.80  78.3 54.40
```

```
par(mfrow = c(1, 1))
plot(RealEstateValuation_Data_$Latitude,
     RealEstateValuation_Data_$Longitude,
     data = RealEstateValuation_Data_)
```

```
## Warning in plot.window(...): "data" is not a graphical parameter
```

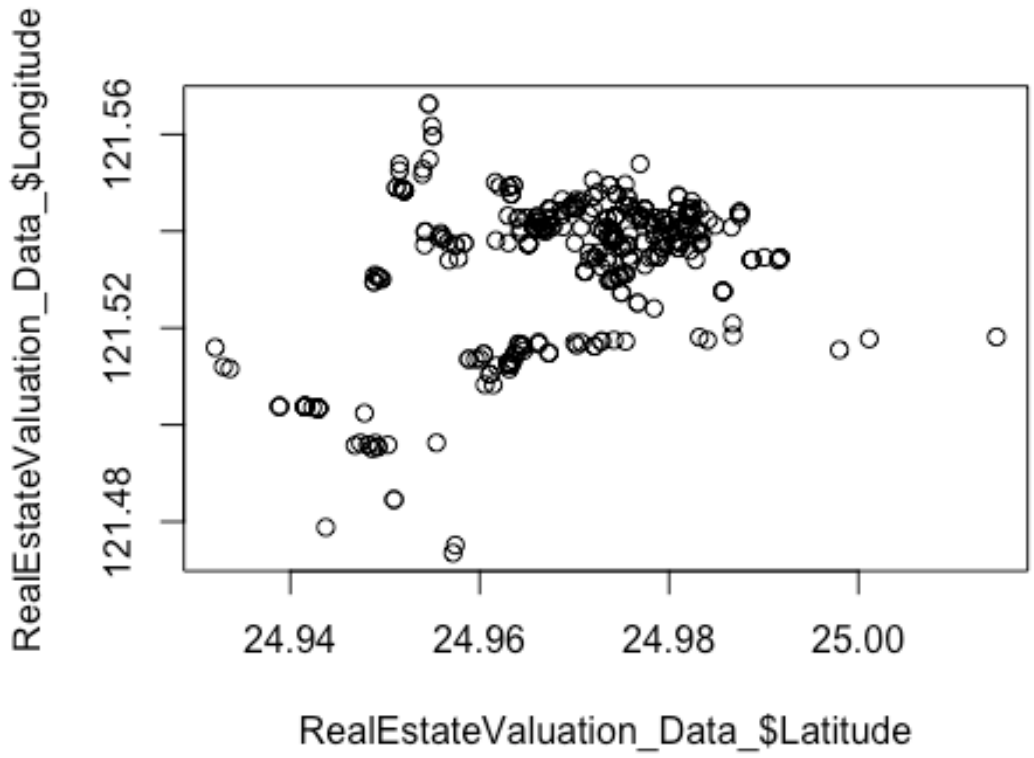
```
## Warning in plot.xy(xy, type, ...): "data" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "data" is
not a
## graphical parameter
```

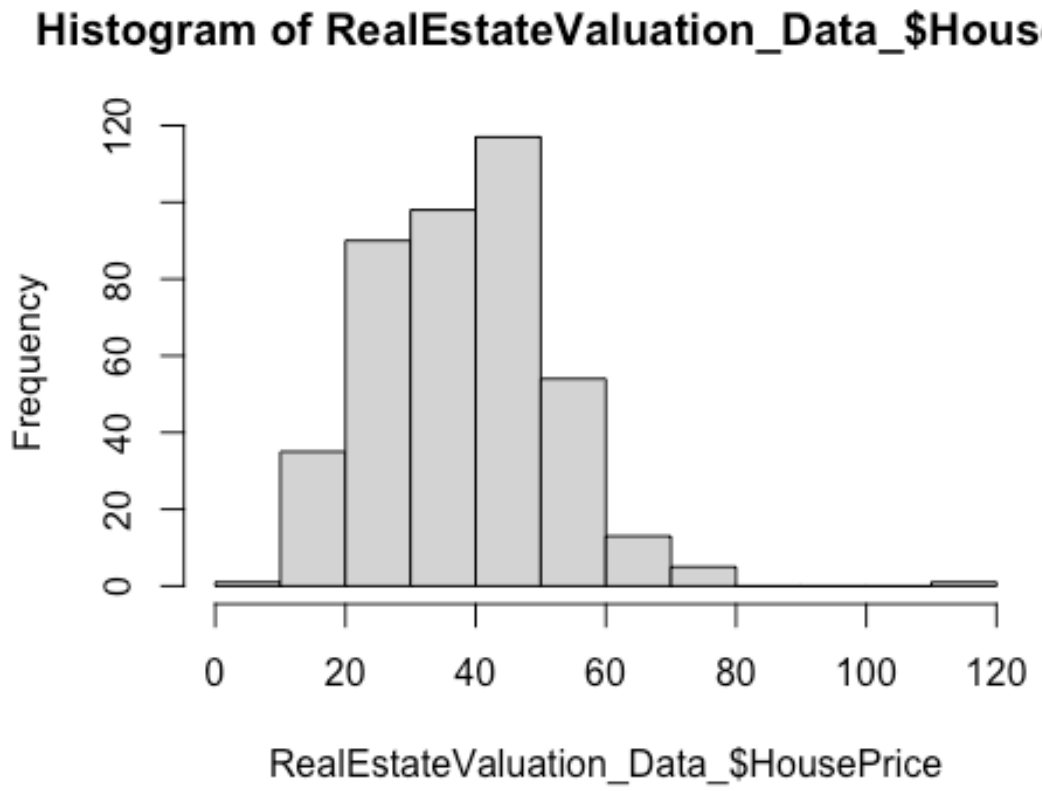
```
## Warning in axis(side = side, at = at, labels = labels, ...): "data" is
not a
## graphical parameter
```

```
## Warning in box(...): "data" is not a graphical parameter
```

```
## Warning in title(...): "data" is not a graphical parameter
```



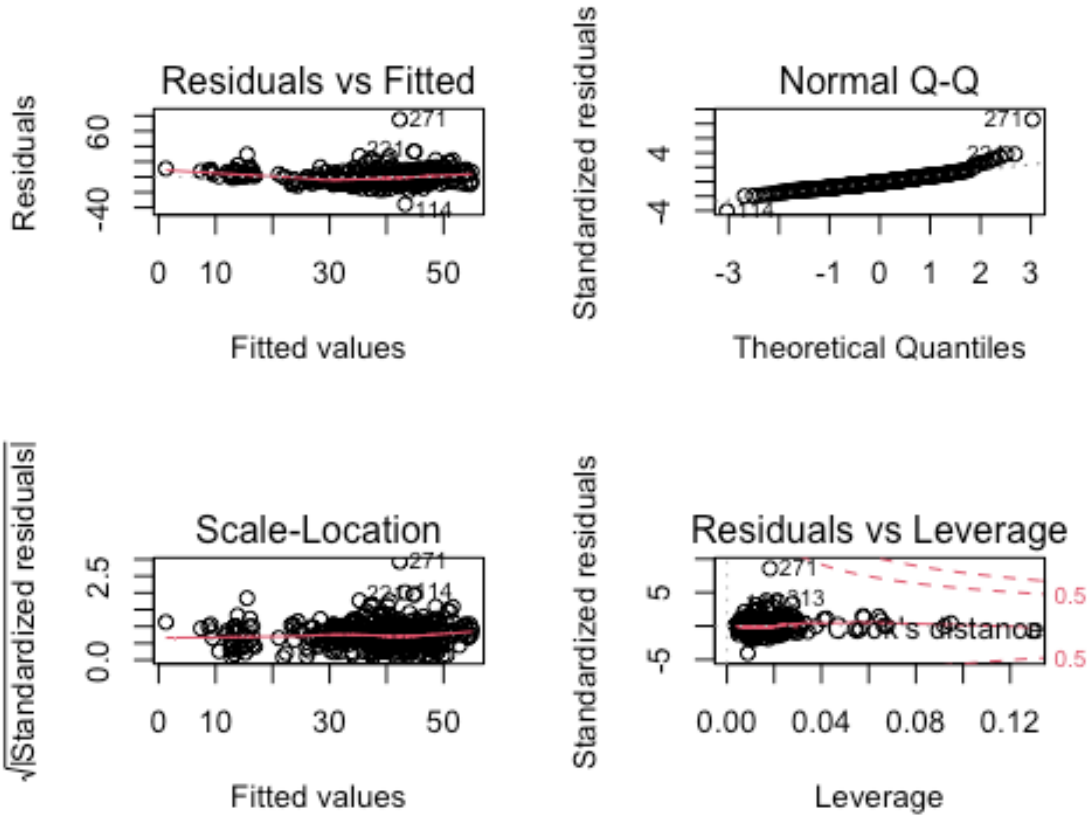

```
hist(RealEstateValuation_Data_$HousePrice)
```




```
########## Machine Learning Methods ##########

# Separating the data into train & test, decide on the method
set.seed(10)
train <-
  sample(1:289, 289) # Taking the first 289 observations(70% of the data f
or training)
REV.train <- RealEstateValuation_Data_[train, ] # 289 obvs
REV.test <- RealEstateValuation_Data_[-train, ] # 125 obvs

##### Multiple linear regression #####
set.seed(10)
lm.REV <-
  lm(HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitu
de + Longitude,
     data = RealEstateValuation_Data_)
summary(lm.REV)

##
## Call:
## lm(formula = HousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude + Longitude, data = RealEstateValuation_Data_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.667  -5.412  -0.967   4.217  75.190
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.444e+04  6.775e+03  -2.132  0.03364 *
## TransactionDate  5.149e+00  1.557e+00   3.307  0.00103 **
## HouseAge        -2.697e-01  3.853e-02  -7.000 1.06e-11 ***
## MRTDist         -4.488e-03  7.180e-04  -6.250 1.04e-09 ***
## StoreNum         1.133e+00  1.882e-01   6.023 3.83e-09 ***
## Latitude         2.255e+02  4.457e+01   5.059 6.38e-07 ***
## Longitude       -1.243e+01  4.858e+01  -0.256  0.79820
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.858 on 407 degrees of freedom
## Multiple R-squared:  0.5824, Adjusted R-squared:  0.5762
## F-statistic:  94.6 on 6 and 407 DF,  p-value: < 2.2e-16
```

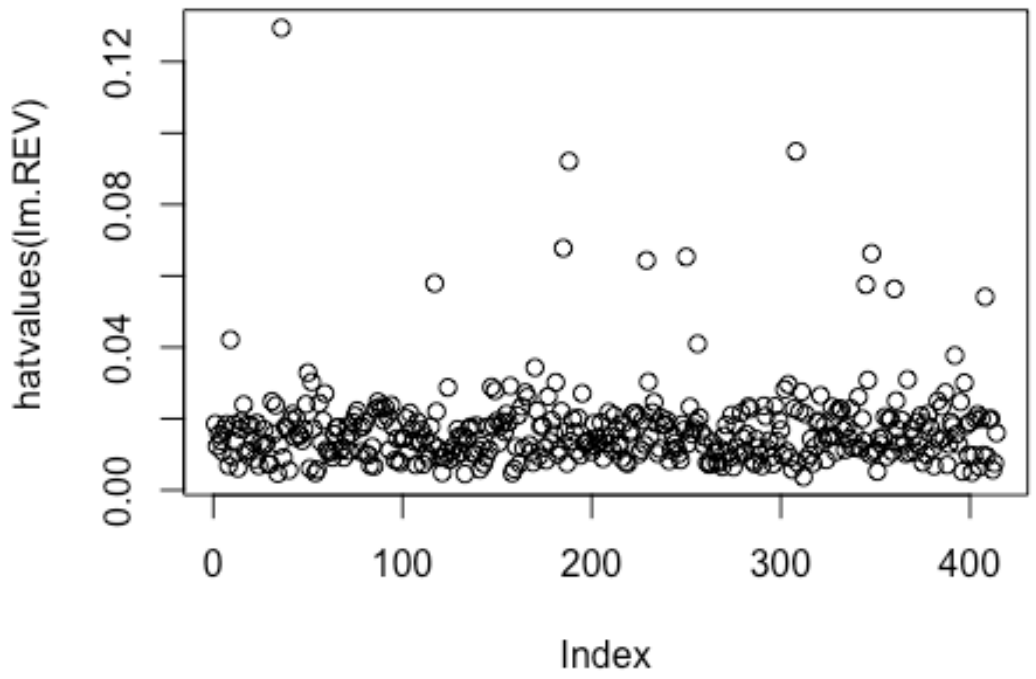
```
#Diagnostic Plots
par(mfrow = c(2, 2))
plot(lm.REV)
```




```
# Outliers & High Leverage Points
index.outlier <- which(rstudent(lm.REV) > 3 | rstudent(lm.REV) < -3)
length(index.outlier) # number of outliers = 7
```

```
## [1] 7
```

```
average_leverage <- 7 / 414
index.highlev <- which(hatvalues(lm.REV) > average_leverage * 3)
length(index.highlev) # number of high leverage points = 11

## [1] 11

par(mfrow = c(1, 1))
plot(hatvalues(lm.REV))
```




```
# Variation Inflation Factor
vif(lm.REV) # House Age has the lowest VIF value of 1.014287 and MRTDist h
as the highest of 4.323019. As none of the VIF scores are above 5 one can
infer that no significant multi-collinearity exists between the variables.
MRTDist is likely to share some collinearity with Latitude and Longitude d
ue to the nature of MRT station distribution around the city, however as s
tated this report doesn't consider collinearity to be a noticeable problem
.

## TransactionDate        HouseAge         MRTDist        StoreNum
Latitude
##        1.014655        1.014287        4.322984        1.617021
1.610225
##       Longitude
##        2.926305

lm.REV.train <-
  lm(HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitu
de + Longitude,
```

```
     data = REV.train)
summary(lm.REV.train)

##
## Call:
## lm(formula = HousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude + Longitude, data = REV.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.286  -5.087  -0.628   4.212  74.789
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.387e+04  8.987e+03  -1.544   0.1238
## TransactionDate  4.596e+00  1.951e+00   2.355   0.0192 *
## HouseAge        -3.021e-01  4.820e-02  -6.268 1.37e-09 ***
## MRTDist         -4.222e-03  9.509e-04  -4.440 1.29e-05 ***
## StoreNum         1.101e+00  2.397e-01   4.595 6.54e-06 ***
## Latitude         2.645e+02  5.560e+01   4.757 3.14e-06 ***
## Longitude       -1.597e+01  6.512e+01  -0.245   0.8065
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.209 on 282 degrees of freedom
## Multiple R-squared:  0.5764, Adjusted R-squared:  0.5674
## F-statistic: 63.95 on 6 and 282 DF,  p-value: < 2.2e-16

lm.pred <- predict(lm.REV.train, REV.test)

mean((lm.pred - REV.test$HousePrice) ^ 2)

## [1] 65.24426

lm.test.r2 <-
  1 - sum((REV.test$HousePrice - lm.pred) ^ 2) / sum((REV.test$HousePrice
- mean(REV.test$HousePrice)) ^2)
lm.test.r2 # 0.5919161

## [1] 0.5919161

# Remove? Introduce during Best Subset #
lm.REV2 <-
  lm(HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitu
de,
     data = RealEstateValuation_Data_)
summary(lm.REV2) # R-sqaured 0.5823, RSE = 8.847

##
## Call:
## lm(formula = HousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude, data = RealEstateValuation_Data_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -35.625  -5.373  -1.020   4.243  75.343
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.596e+04  3.233e+03  -4.938 1.15e-06 ***
## TransactionDate  5.138e+00  1.554e+00   3.305  0.00103 **
## HouseAge        -2.694e-01  3.847e-02  -7.003 1.04e-11 ***
## MRTDist         -4.353e-03  4.899e-04  -8.887  < 2e-16 ***
## StoreNum         1.136e+00  1.876e-01   6.056 3.17e-09 ***
## Latitude         2.269e+02  4.417e+01   5.136 4.35e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.847 on 408 degrees of freedom
## Multiple R-squared:  0.5823, Adjusted R-squared:  0.5772
## F-statistic: 113.8 on 5 and 408 DF,  p-value: < 2.2e-16
```

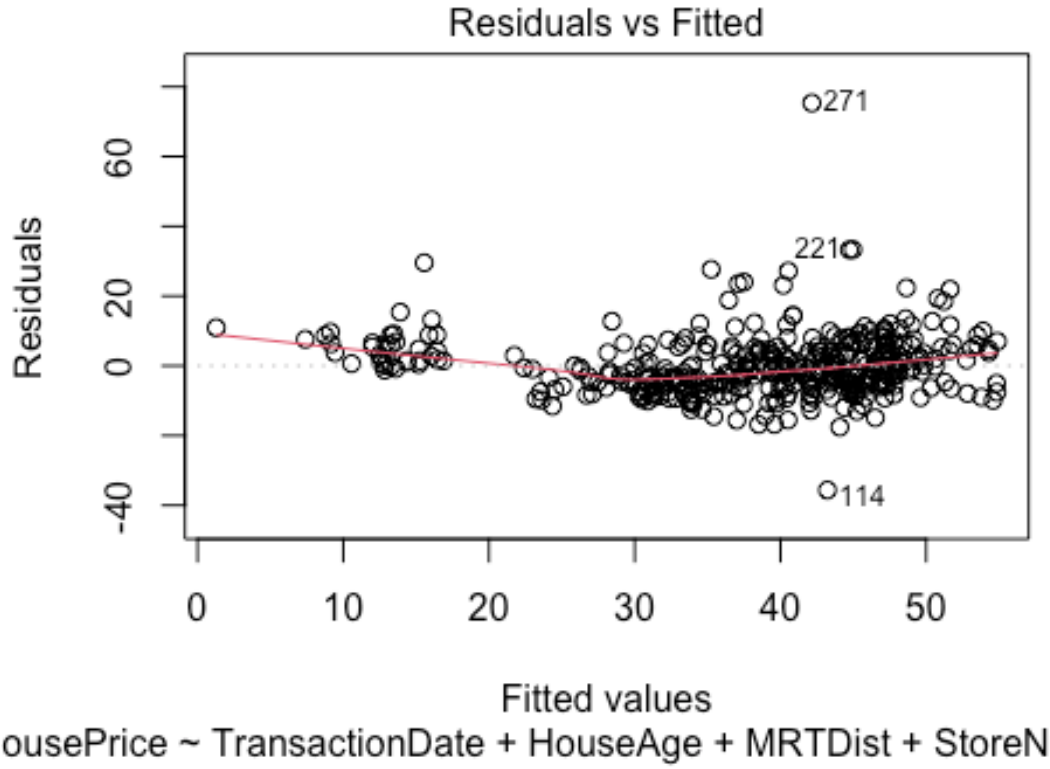
```
mean(lm.REV2$residuals ^ 2) # MSE = 77.14143
```

```
## [1] 77.14143
```

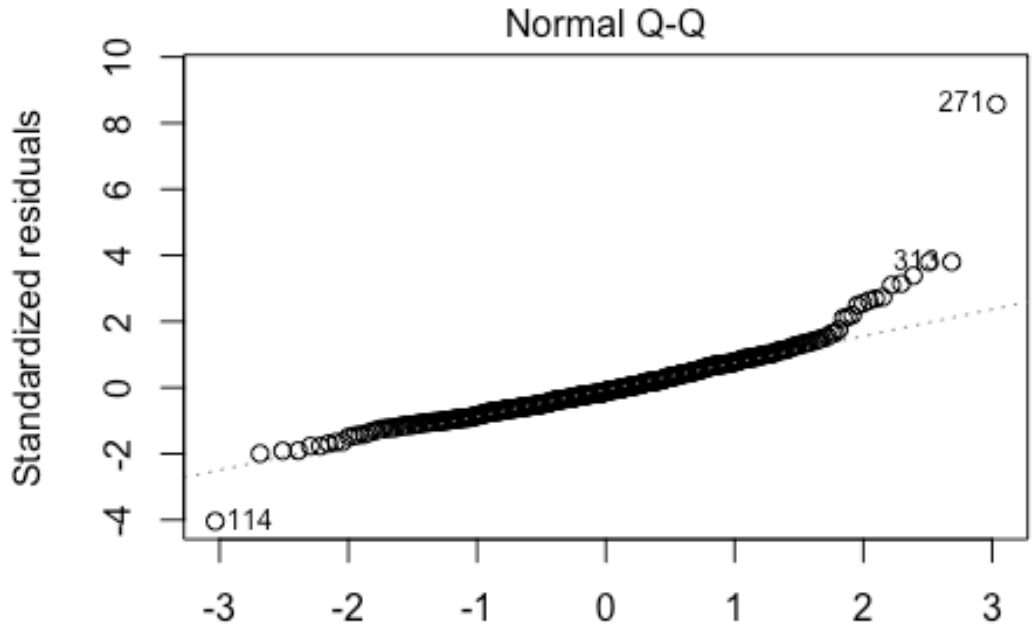
```
anova(lm.REV, lm.REV2) # P-value very high meaning models are very similar
, therefore second model better. One less variable and better adj R-square
d.
```

```
## Analysis of Variance Table
##
## Model 1: HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum +
##     Latitude + Longitude
## Model 2: HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum +
##     Latitude
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1    407 31931
## 2    408 31937 -1   -5.1353 0.0655 0.7982
```

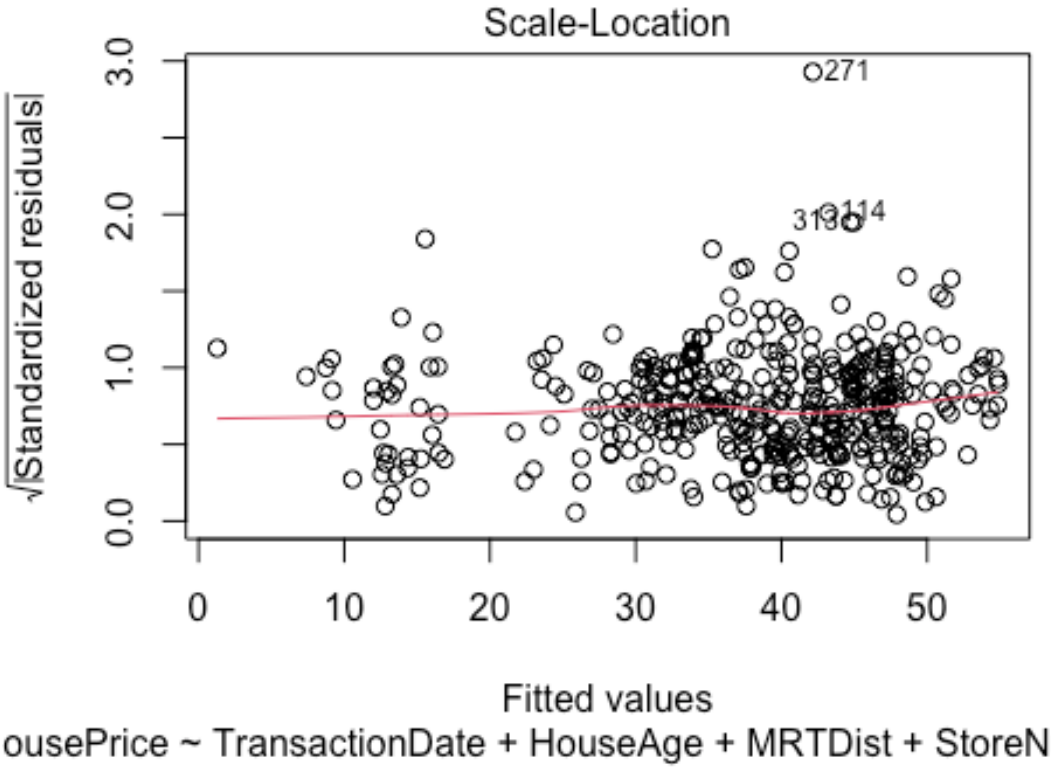
```
plot(lm.REV2)
```

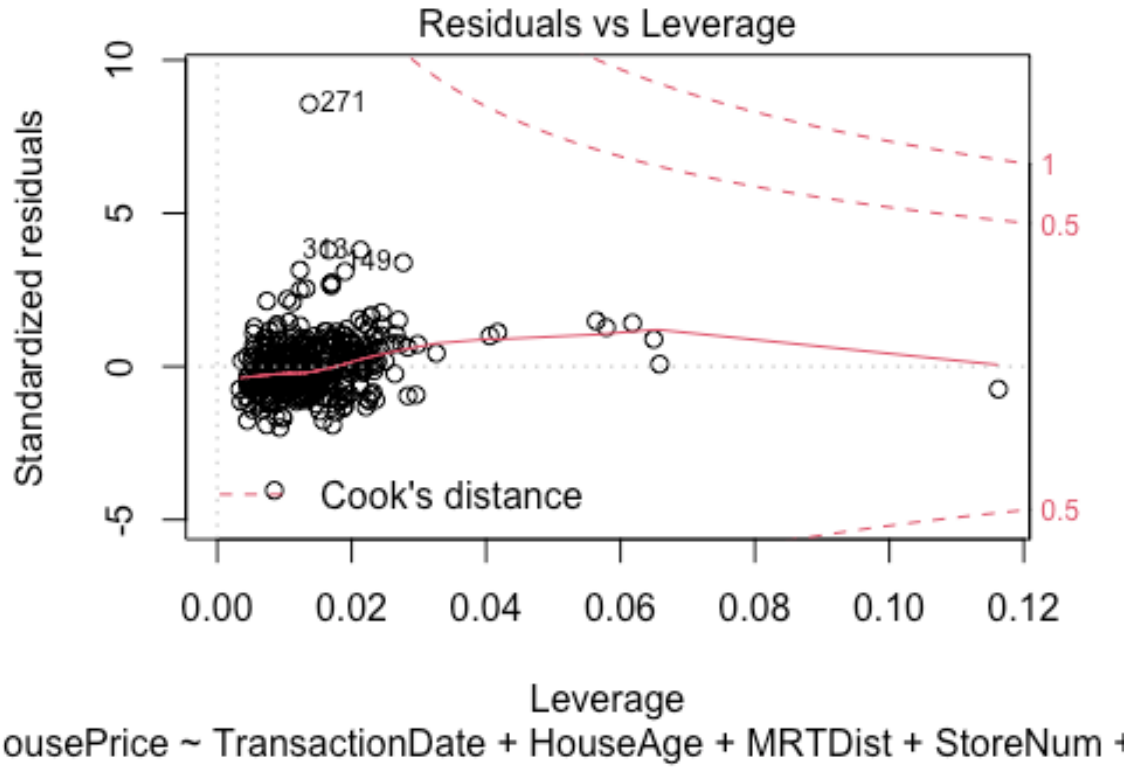


ousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum +



ousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum +

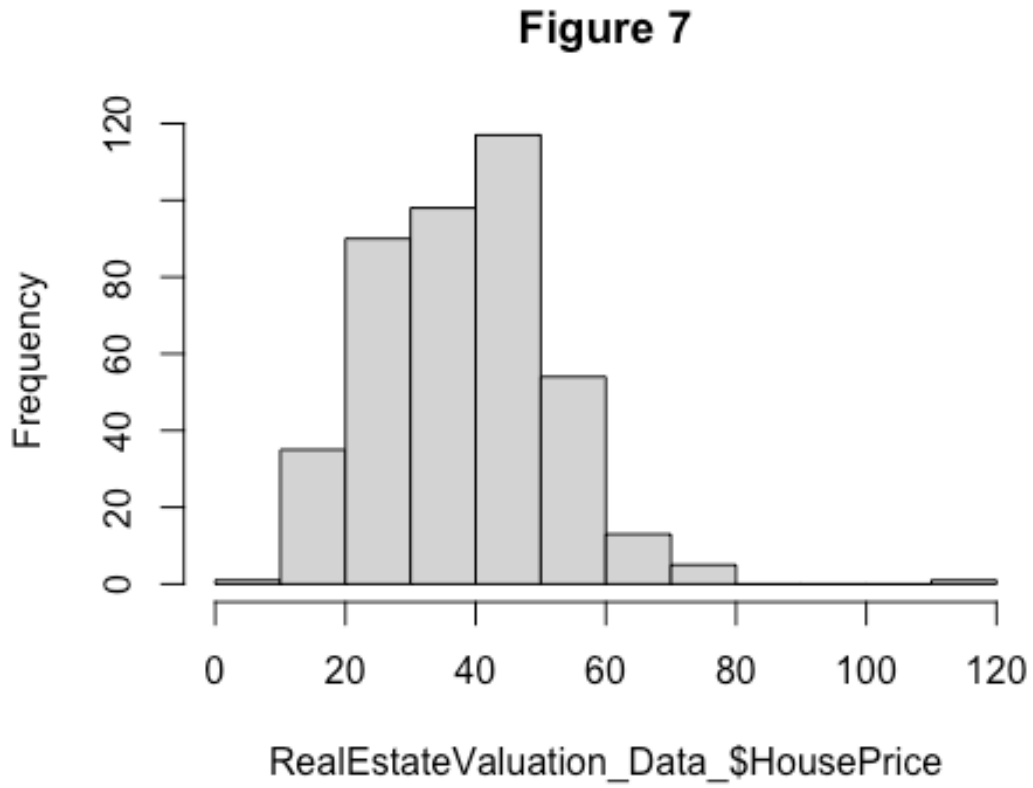

Scale-Location
√|Standardized residuals|
271
3.0
2.0
1.0
0.0
0
10
20
30
40
50
Fitted values
ousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum +


Residuals vs Leverage
Standardized residuals
271
10
5
0
-5
1
0.5
0.5
Cook's distance
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Leverage
ousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum +

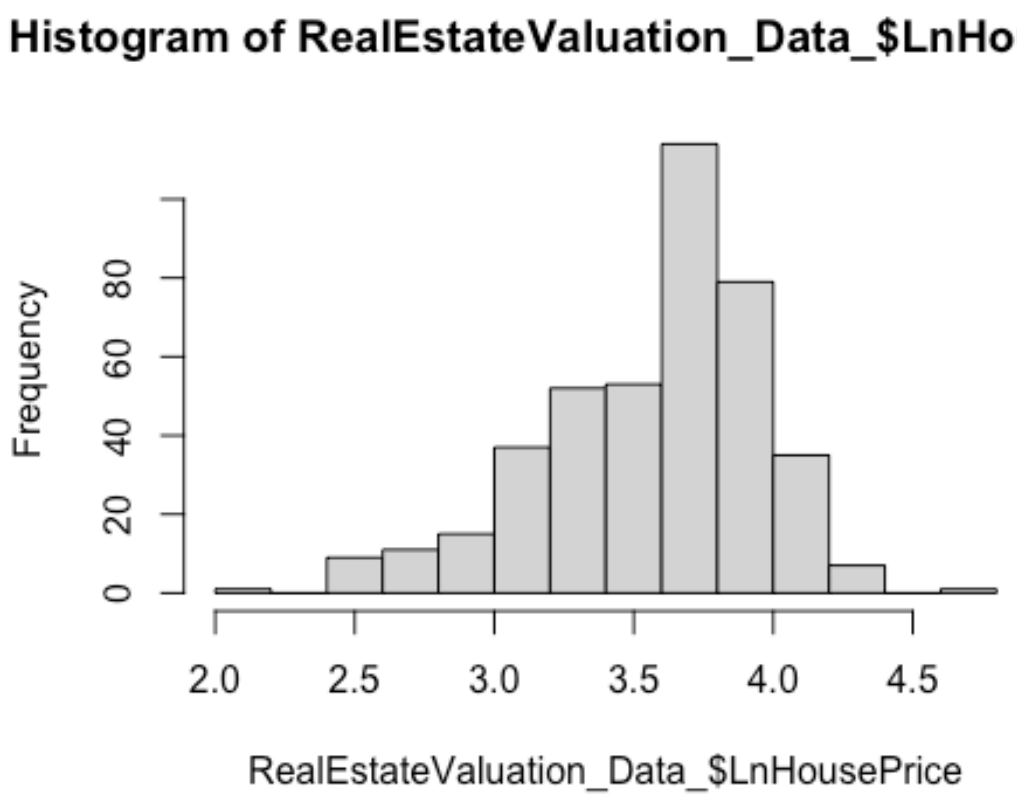
```
lm.REV2.train <-
  lm(HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitu
de,
     data = REV.train)
summary(lm.REV2.train) # R-squared = 0.5826, RSE = 8.957

##
## Call:
## lm(formula = HousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude, data = REV.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.232  -4.997  -0.719   4.399  74.998
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.584e+04  4.011e+03  -3.950 9.88e-05 ***
## TransactionDate  4.590e+00  1.948e+00   2.356   0.0191 *
## HouseAge        -3.017e-01  4.809e-02  -6.274 1.32e-09 ***
## MRTDist         -4.043e-03  6.079e-04  -6.651 1.50e-10 ***
## StoreNum         1.107e+00  2.382e-01   4.647 5.17e-06 ***
## Latitude         2.662e+02  5.510e+01   4.831 2.23e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.193 on 283 degrees of freedom
## Multiple R-squared:  0.5763, Adjusted R-squared:  0.5688
## F-statistic: 76.99 on 5 and 283 DF,  p-value: < 2.2e-16

# Taking the natural logarithm of house price and testing it
set.seed(10)
ln.REV <-
  lm(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude +
      Longitude,
    data = RealEstateValuation_Data_
  )
summary(ln.REV) # R-squared = 0.6857, RSE = 0.2214

##
## Call:
## lm(formula = LnHousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude + Longitude, data = RealEstateValuation_Data_)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.68101 -0.11493 -0.00265  0.11538  1.04844
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -5.117e+02  1.695e+02  -3.018 0.002700 **
## TransactionDate  1.355e-01  3.895e-02   3.478 0.000559 ***
```

```
## HouseAge        -6.967e-03  9.641e-04  -7.227 2.46e-12 ***
## MRTDist         -1.455e-04  1.797e-05  -8.098 6.54e-15 ***
## StoreNum         2.775e-02  4.708e-03   5.894 7.92e-09 ***
## Latitude         7.925e+00  1.115e+00   7.107 5.35e-12 ***
## Longitude        3.687e-01  1.216e+00   0.303 0.761828
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2216 on 407 degrees of freedom
## Multiple R-squared:  0.6858, Adjusted R-squared:  0.6811
## F-statistic:   148 on 6 and 407 DF,  p-value: < 2.2e-16

ln.fit <-
  lm(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude +
      Longitude,
    data = REV.train
  )
summary(ln.fit) # R-sqaured = 0.6984, RSE = 0.2227

##
## Call:
## lm(formula = LnHousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude + Longitude, data = REV.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.67143 -0.10965  0.00671  0.10732  1.04828
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -5.199e+02  2.253e+02  -2.307  0.02176 *
## TransactionDate  1.313e-01  4.892e-02   2.684  0.00772 **
## HouseAge        -7.624e-03  1.208e-03  -6.308 1.09e-09 ***
## MRTDist         -1.363e-04  2.384e-05  -5.718 2.74e-08 ***
## StoreNum         2.855e-02  6.010e-03   4.750 3.24e-06 ***
## Latitude         8.855e+00  1.394e+00   6.352 8.51e-10 ***
## Longitude        3.149e-01  1.633e+00   0.193  0.84721
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2309 on 282 degrees of freedom
## Multiple R-squared:  0.6796, Adjusted R-squared:  0.6727
## F-statistic: 99.67 on 6 and 282 DF,  p-value: < 2.2e-16

ln.pred <- predict(ln.fit, REV.test)
converted.back <- exp(ln.pred)
View(converted.back)
mean((converted.back - REV.test$HousePrice) ^ 2) #MSE = 61.56653

## [1] 61.56653

ln.test.r2 <-
  1 - sum((REV.test$HousePrice - converted.back) ^ 2) / sum((REV.test$Hous
```

```
ePrice - mean(REV.test$HousePrice)) ^
                                                                    2)
ln.test.r2 # 0.6149192

## [1] 0.6149192

par(mfrow = c(1, 1))
hist(RealEstateValuation_Data_$HousePrice, main = paste("Figure 7"))
```



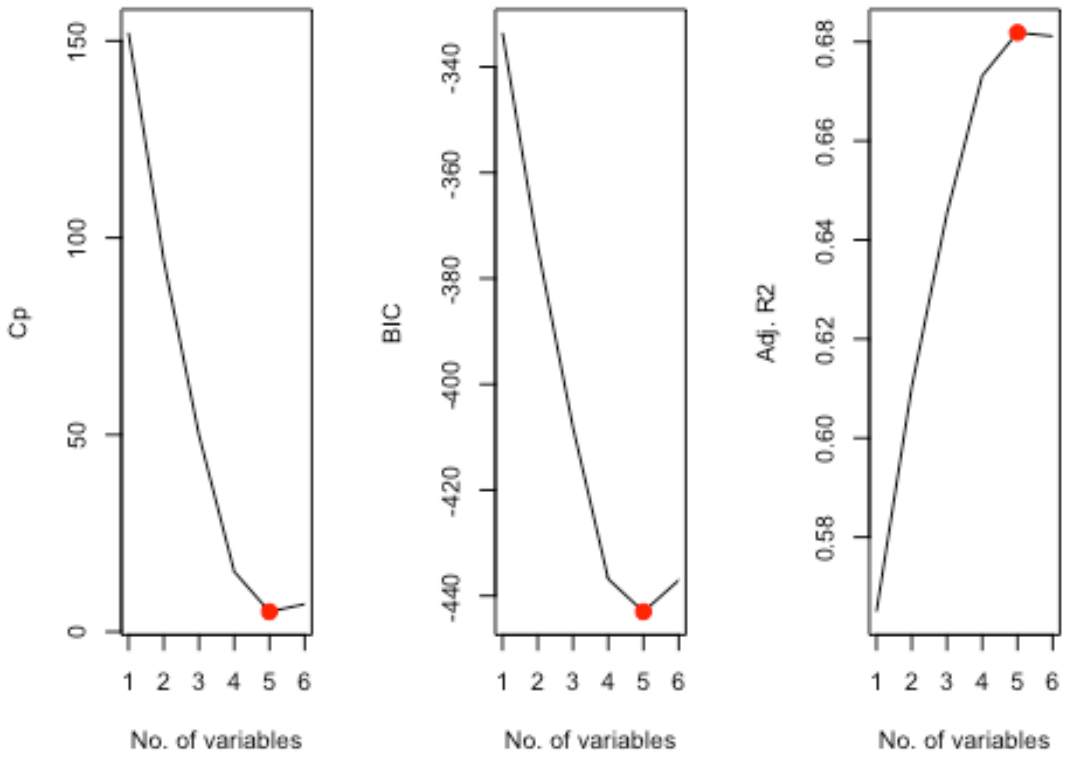

```
hist(RealEstateValuation_Data_$'LnHousePrice')
```




```
##### Subset selection #####

# Best Subset Selection #
reg.full <-
  regsubsets(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
    data = RealEstateValuation_Data_,
    nvmax = 6
  )
reg.summary <- summary(reg.full)
reg.summary

## Subset selection object
## Call: regsubsets.formula(LnHousePrice ~ TransactionDate + HouseAge +
##     MRTDist + StoreNum + Latitude + Longitude, data = RealEstateValuati
on_Data_,
##     nvmax = 6)
## 6 Variables  (and intercept)
##                  Forced in Forced out
## TransactionDate      FALSE      FALSE
## HouseAge             FALSE      FALSE
## MRTDist              FALSE      FALSE
## StoreNum             FALSE      FALSE
## Latitude             FALSE      FALSE
## Longitude            FALSE      FALSE
## 1 subsets of each size up to 6
```

```
## Selection Algorithm: exhaustive
##          TransactionDate HouseAge MRTDist StoreNum Latitude Longitude
## 1  ( 1 ) " "             " "      "*"     " "      " "      " "
## 2  ( 1 ) " "             " "      "*"     " "      "*"      " "
## 3  ( 1 ) " "             "*"      "*"     " "      "*"      " "
## 4  ( 1 ) " "             "*"      "*"     "*"      "*"      " "
## 5  ( 1 ) "*"             "*"      "*"     "*"      "*"      " "
## 6  ( 1 ) "*"             "*"      "*"     "*"      "*"      "*"
```

```r
par(mfrow = c(1, 3))
# Cp = 5 variable model best
plot(reg.summary$cp,
     xlab = "No. of variables",
     ylab = "Cp",
     type = "l")
points(
  which.min(reg.summary$cp),
  reg.summary$cp[which.min(reg.summary$cp)],
  col = "red",
  cex = 2,
  pch = 20
)
# BIC = 5 variable model best
plot(reg.summary$bic,
     xlab = "No. of variables",
     ylab = "BIC",
     type = "l")
points(
  which.min(reg.summary$bic),
  reg.summary$bic[which.min(reg.summary$bic)],
  col = "red",
  cex = 2,
  pch = 20
)
# Adj- R2 = 5 variable model best
plot(reg.summary$adjr2,
     xlab = "No. of variables",
     ylab = "Adj. R2",
     type = "l")
points(
  which.max(reg.summary$adjr2),
  reg.summary$adjr2[which.max(reg.summary$adjr2)],
  col = "red",
  cex = 2,
  pch = 20
)
```




```
coef(reg.full, 5) # Best subset is with 5 variables #
```

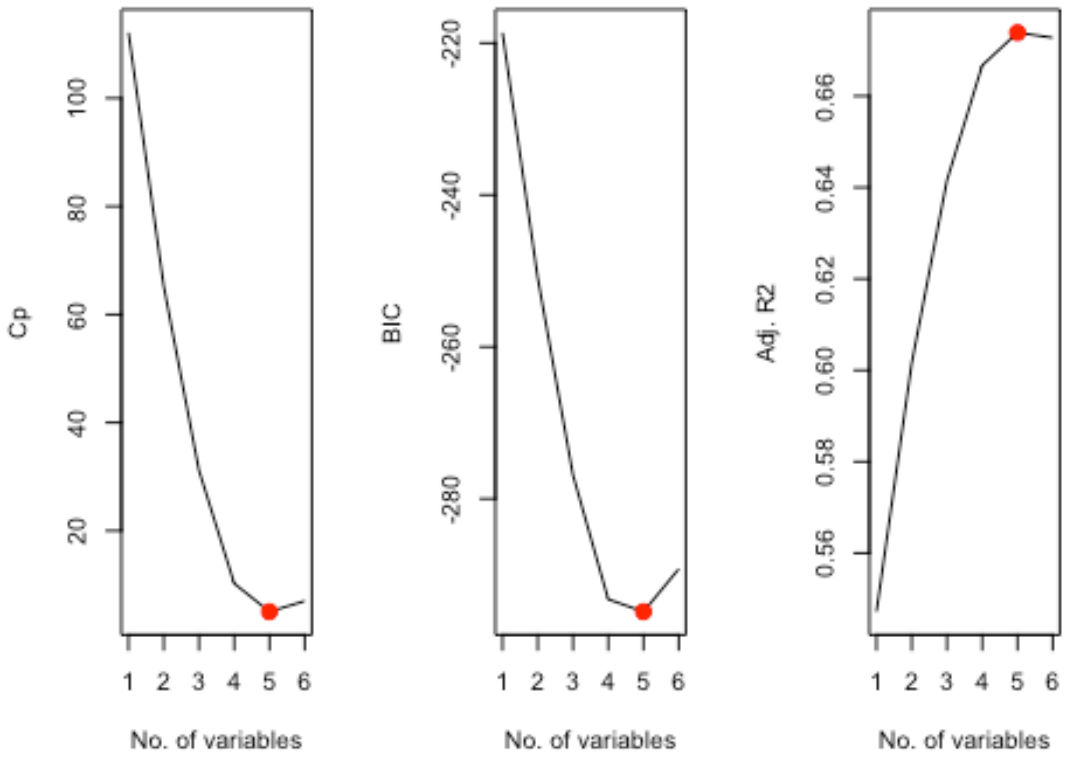
```
##     (Intercept) TransactionDate        HouseAge         MRTDist
StoreNum
##   -4.665529e+02    1.358373e-01   -6.976565e-03   -1.494696e-04    2.76
6358e-02
##        Latitude
##    7.883011e+00
```

```
# Forward Stepwise Selection #
reg.fwd <-
  regsubsets(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
    data = REV.train,
    nvmax = 6,
    method = "forward"
  )
reg.summary.fwd <- summary(reg.fwd)
reg.summary.fwd
```

```
## Subset selection object
## Call: regsubsets.formula(LnHousePrice ~ TransactionDate + HouseAge +
##     MRTDist + StoreNum + Latitude + Longitude, data = REV.train,
##     nvmax = 6, method = "forward")
## 6 Variables  (and intercept)
##                  Forced in Forced out
## TransactionDate     FALSE      FALSE
```

```
## HouseAge            FALSE      FALSE
## MRTDist             FALSE      FALSE
## StoreNum            FALSE      FALSE
## Latitude            FALSE      FALSE
## Longitude           FALSE      FALSE
## 1 subsets of each size up to 6
## Selection Algorithm: forward
##          TransactionDate HouseAge MRTDist StoreNum Latitude Longitude
## 1  ( 1 ) " "             " "      "*"     " "      " "      " "
## 2  ( 1 ) " "             " "      "*"     " "      "*"      " "
## 3  ( 1 ) " "             "*"      "*"     " "      "*"      " "
## 4  ( 1 ) " "             "*"      "*"     "*"      "*"      " "
## 5  ( 1 ) "*"             "*"      "*"     "*"      "*"      " "
## 6  ( 1 ) "*"             "*"      "*"     "*"      "*"      "*"
```

```r
par(mfrow = c(1, 3))
# Cp = 5 variables best
plot(reg.summary.fwd$cp,
     xlab = "No. of variables",
     ylab = "Cp",
     type = "l")
points(
  which.min(reg.summary.fwd$cp),
  reg.summary.fwd$cp[which.min(reg.summary.fwd$cp)],
  col = "red",
  cex = 2,
  pch = 20
)
# BIC = 5 vars best
plot(reg.summary.fwd$bic,
     xlab = "No. of variables",
     ylab = "BIC",
     type = "l")
points(
  which.min(reg.summary.fwd$bic),
  reg.summary.fwd$bic[which.min(reg.summary.fwd$bic)],
  col = "red",
  cex = 2,
  pch = 20
)
# Adj R2 = 5 vars best
plot(
  reg.summary.fwd$adjr2,
  xlab = "No. of variables",
  ylab = "Adj. R2",
  type = "l"
)
points(
  which.max(reg.summary.fwd$adjr2),
  reg.summary.fwd$adjr2[which.max(reg.summary.fwd$adjr2)],
  col = "red",
  cex = 2,
  pch = 20
)
```




```
coef(reg.fwd, 5) # Best subset is with 5 variables #
```

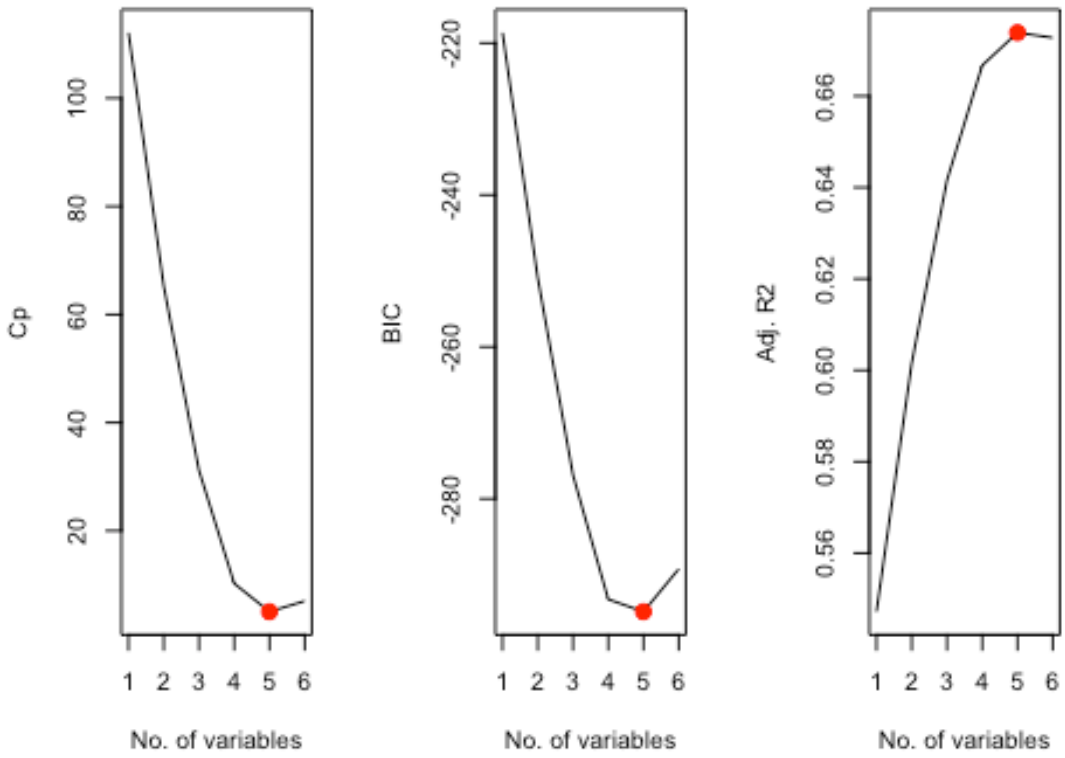
```
##     (Intercept) TransactionDate        HouseAge         MRTDist
StoreNum
##   -4.810637e+02    1.313939e-01   -7.631250e-03   -1.398671e-04    2.84
3698e-02
##        Latitude
##    8.822385e+00
```

```
# Backward stepwise selection #
reg.bwd <-
  regsubsets(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
    data = REV.train,
    nvmax = 6,
    method = "backward"
  )
reg.summary.bwd <- summary(reg.bwd)
reg.summary.bwd
```

```
## Subset selection object
## Call: regsubsets.formula(LnHousePrice ~ TransactionDate + HouseAge +
##     MRTDist + StoreNum + Latitude + Longitude, data = REV.train,
##     nvmax = 6, method = "backward")
## 6 Variables  (and intercept)
##                  Forced in Forced out
## TransactionDate     FALSE      FALSE
```

```
## HouseAge            FALSE      FALSE
## MRTDist             FALSE      FALSE
## StoreNum            FALSE      FALSE
## Latitude            FALSE      FALSE
## Longitude           FALSE      FALSE
## 1 subsets of each size up to 6
## Selection Algorithm: backward
##          TransactionDate HouseAge MRTDist StoreNum Latitude Longitude
## 1  ( 1 ) " "             " "      "*"     " "      " "      " "
## 2  ( 1 ) " "             " "      "*"     " "      "*"      " "
## 3  ( 1 ) " "             "*"      "*"     " "      "*"      " "
## 4  ( 1 ) " "             "*"      "*"     "*"      "*"      " "
## 5  ( 1 ) "*"             "*"      "*"     "*"      "*"      " "
## 6  ( 1 ) "*"             "*"      "*"     "*"      "*"      "*"
```

```r
par(mfrow = c(1, 3))
# Cp = 5 variables best
plot(reg.summary.bwd$cp,
     xlab = "No. of variables",
     ylab = "Cp",
     type = "l")
points(
  which.min(reg.summary.bwd$cp),
  reg.summary.bwd$cp[which.min(reg.summary.bwd$cp)],
  col = "red",
  cex = 2,
  pch = 20
)
# BIC = 5 vars best
plot(reg.summary.bwd$bic,
     xlab = "No. of variables",
     ylab = "BIC",
     type = "l")
points(
  which.min(reg.summary.bwd$bic),
  reg.summary.bwd$bic[which.min(reg.summary.bwd$bic)],
  col = "red",
  cex = 2,
  pch = 20
)
# Adj R2 = 5 vars best
plot(
  reg.summary.bwd$adjr2,
  xlab = "No. of variables",
  ylab = "Adj. R2",
  type = "l"
)
points(
  which.max(reg.summary.bwd$adjr2),
  reg.summary.bwd$adjr2[which.max(reg.summary.bwd$adjr2)],
  col = "red",
  cex = 2,
  pch = 20
)
```




```
coef(reg.bwd, 5) # Best subset is with 5 variables #
```

```
##     (Intercept) TransactionDate        HouseAge         MRTDist
StoreNum
##   -4.810637e+02    1.313939e-01   -7.631250e-03   -1.398671e-04    2.84
3698e-02
##        Latitude
##    8.822385e+00
```

```
# All subset methods highlight a 5 variable model is best. They also agree
that longitude is the variable of least importance and should be dropped #
lm.REV.bestsubset.train <-
  lm(LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Lati
tude,
     data = REV.train)
summary(lm.REV.bestsubset.train) # R-squared = 0.6795, RSE = 0.2305
```

```
##
## Call:
## lm(formula = LnHousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude, data = REV.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6725 -0.1065  0.0060  0.1069  1.0441
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)     -4.811e+02  1.006e+02  -4.784 2.78e-06 ***
## TransactionDate  1.314e-01  4.884e-02   2.691  0.00756 **
## HouseAge        -7.631e-03  1.206e-03  -6.329 9.64e-10 ***
## MRTDist         -1.399e-04  1.524e-05  -9.177  < 2e-16 ***
## StoreNum         2.844e-02  5.972e-03   4.761 3.07e-06 ***
## Latitude         8.822e+00  1.381e+00   6.387 6.94e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2305 on 283 degrees of freedom
## Multiple R-squared:  0.6795, Adjusted R-squared:  0.6739
## F-statistic:   120 on 5 and 283 DF,  p-value: < 2.2e-16
```

```
lm.REV.bestsubset.test <- predict(lm.REV.bestsubset.train, REV.test)
mean((REV.test$HousePrice - (exp(
  lm.REV.bestsubset.test
))) ^ 2) #MSE = 61.45236
```

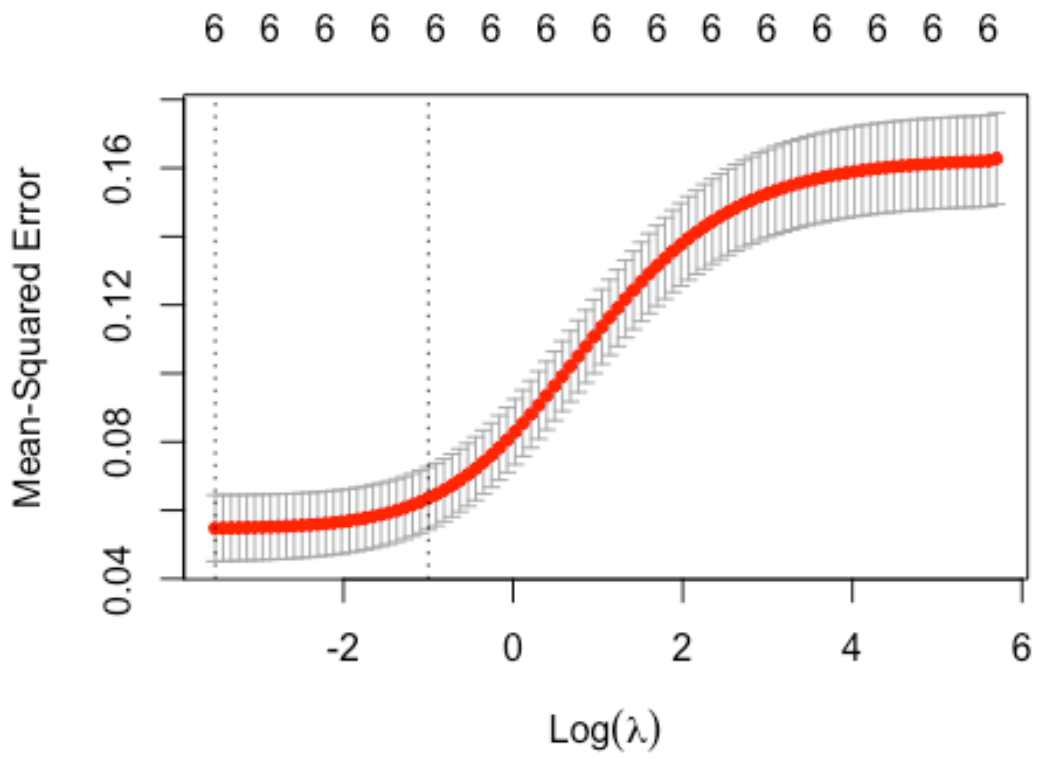
```
## [1] 61.45236
```

```
ln.bestsubset.r2 <-
  1 - sum((REV.test$LnHousePrice - lm.REV.bestsubset.test) ^ 2) / sum((REV
.test$LnHousePrice - mean(REV.test$LnHousePrice)) ^
                                                                      2)
ln.bestsubset.r2 # R-squared = 0.6992486
```

```
## [1] 0.6992486
```

```
##### Ridge and Lasso ##### Tut 9

# Ridge #
set.seed(10)

train.X <-
  model.matrix(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
    data = REV.train
  )[, -1]
test.X <-
  model.matrix(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
    data = REV.test
  )[, -1]
grid <-
  10 ^ seq(5,-3, length = 414) # computes the CV error for the 100 values
for the ridge regression # IS THIS NECESSARY

ridge.mod <-
  glmnet(train.X,
         REV.train$LnHousePrice,
         alpha = 0,
         lambda = grid) # alpha = 0 tells R that we want to do ridge
ridge.cv <-
```

```
  cv.glmnet(train.X,
            REV.train$LnHousePrice,
            alpha = 0,
            lamda = grid)

par(mfrow = c(1, 1))
plot(ridge.cv) # MSE minimised when lambda is very small, so the original
least squared regression good at minimising MSE for this model
```




```
# first dotted line gives minimum MSE and the second gives the smallest MS
E within 1 se of the minimum

ridge.bestlam <- ridge.cv$lambda.min
ridge.bestlam # 0.02984678

## [1] 0.02984678

predict(ridge.mod,
        type = "coefficients",
        s = ridge.bestlam,
        newx = test.X)

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)     -6.820721e+02
## TransactionDate  1.203953e-01
## HouseAge        -7.092077e-03
## MRTDist         -1.121054e-04
```

```
## StoreNum          2.920636e-02
## Latitude          8.785038e+00
## Longitude         1.843442e+00
```

```
summary(lm.REV) # Coefficient Comparison
```

```
##
## Call:
## lm(formula = HousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude + Longitude, data = RealEstateValuation_Data_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.667  -5.412  -0.967   4.217  75.190
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.444e+04  6.775e+03  -2.132  0.03364 *
## TransactionDate  5.149e+00  1.557e+00   3.307  0.00103 **
## HouseAge        -2.697e-01  3.853e-02  -7.000 1.06e-11 ***
## MRTDist         -4.488e-03  7.180e-04  -6.250 1.04e-09 ***
## StoreNum         1.133e+00  1.882e-01   6.023 3.83e-09 ***
## Latitude         2.255e+02  4.457e+01   5.059 6.38e-07 ***
## Longitude       -1.243e+01  4.858e+01  -0.256  0.79820
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.858 on 407 degrees of freedom
## Multiple R-squared:  0.5824, Adjusted R-squared:  0.5762
## F-statistic:  94.6 on 6 and 407 DF,  p-value: < 2.2e-16
```

```
ridge.pred1 <- predict(ridge.mod, s = ridge.bestlam, newx = test.X)
mean((REV.test$HousePrice - exp(ridge.pred1)) ^ 2) # MSE = 62.69749
```

```
## [1] 62.69749
```

```
# within 1 se
ridge.1se <- ridge.cv$lambda.1se
ridge.1se # 0.3679651
```

```
## [1] 0.3679651
```

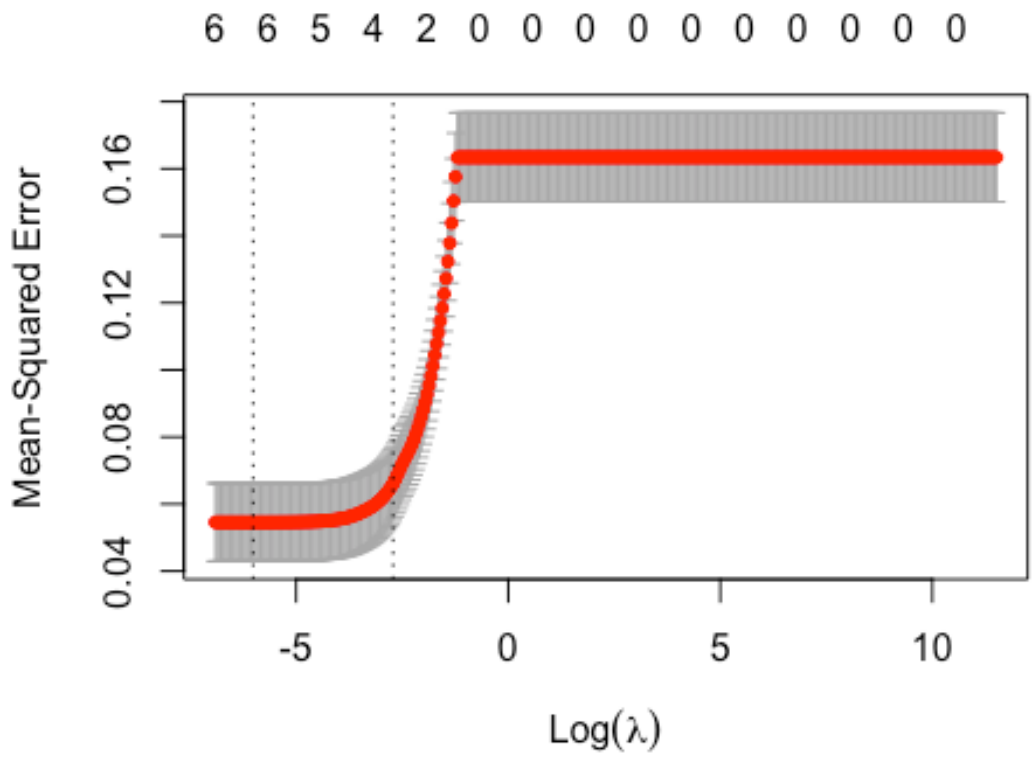
```
ridge.pred2 <- predict(ridge.mod, s = ridge.1se, newx = test.X)
mean((exp(ridge.pred2) - REV.test$HousePrice) ^ 2)  # 71.8387, MSE goes up
quite a bit
```

```
## [1] 71.8387
```

```
rss.ridge <- mean((ridge.pred1 - REV.test$LnHousePrice) ^ 2)
```

```
# Lasso regression #
lasso.mod <-
  glmnet(train.X,
         REV.train$LnHousePrice,
         alpha = 1,
```

```
         lambda = grid) # alpha = 1 for lasso
lasso.cv <-
  cv.glmnet(train.X,
            REV.train$LnHousePrice,
            alpha = 1,
            lambda = grid)
plot(lasso.cv) # if high enough values of lambda chosen, number of predict
or variables drop to 0,
```

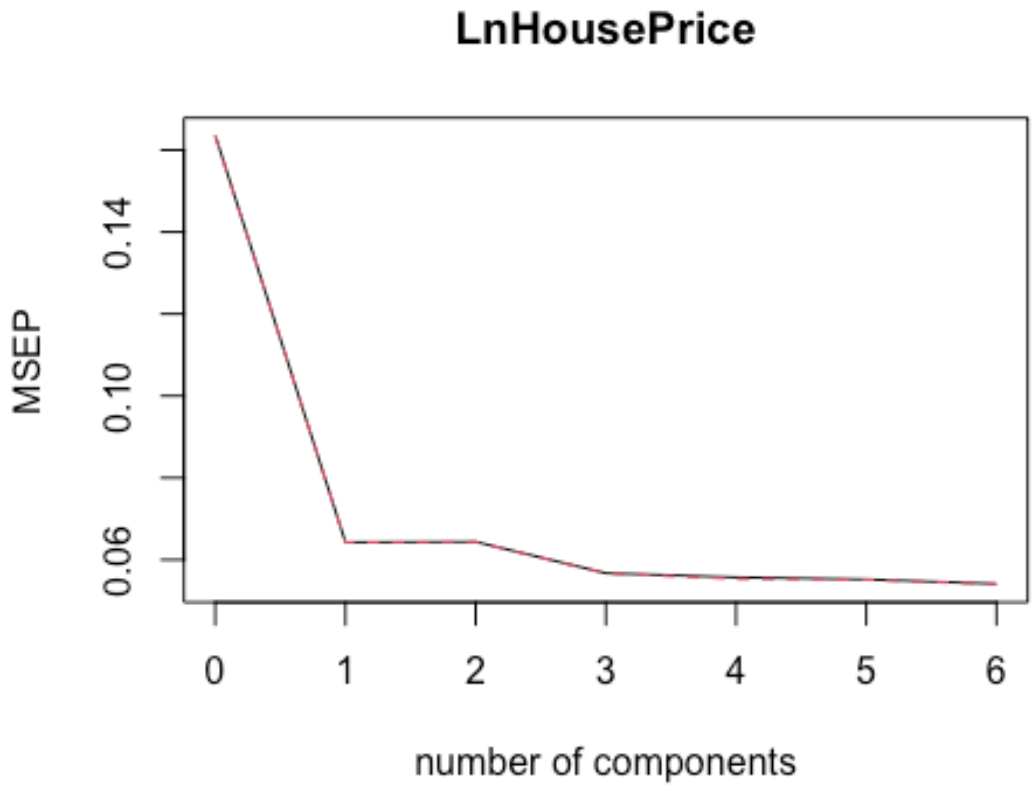



```
lasso.bestlam <- lasso.cv$lambda.min
lasso.bestlam # 0.002440109

## [1] 0.002440109

lasso.pred1 <- predict(lasso.mod, s = lasso.bestlam, newx = test.X)
mean((exp(lasso.pred1) - REV.test$HousePrice) ^ 2) # MSE = 61.44527

## [1] 61.43279

predict(lasso.mod, type = "coefficients", s = lasso.bestlam) # All coeffic
ients used

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept)     -4.849918e+02
## TransactionDate  1.225458e-01
## HouseAge        -7.404231e-03
## MRTDist         -1.371275e-04
```

```
## StoreNum          2.798246e-02
## Latitude          8.743845e+00
## Longitude         1.949798e-01

# Most sparse lasso model with 1 se
lasso.1se <- lasso.cv$lambda.1se
lasso.1se # 2.805058

## [1] 0.0661947

lasso.pred2 <- predict(lasso.mod, s = lasso.1se, newx = test.X)
mean((exp(lasso.pred2) - REV.test$HousePrice) ^ 2) # 71.62681, MSE goes up
quite a bit, is ridge better?

## [1] 72.4778

predict(lasso.mod, type = "coefficients", s = lasso.1se) # only 2 zero coe
fficients on variables.

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)     -1.427030e+02
## TransactionDate  .
## HouseAge        -1.622531e-03
## MRTDist         -1.263240e-04
## StoreNum         1.433893e-02
## Latitude         5.862406e+00
## Longitude        .

ln.test.avg <- mean(RealEstateValuation_Data_$LnHousePrice)
rss.lasso <-
  mean((lasso.pred1 - REV.test$HousePrice) ^ 2) # 1st as MSE is lower and
only uses one more variable
lasso.test.r2 = 1 - mean((REV.test$LnHousePrice - lasso.pred1) ^ 2) / mean
((REV.test$LnHousePrice - ln.test.avg) ^

2)
lasso.test.r2

## [1] 0.7001443

##### PCR and PLS ##### Tut 9

# PCR
set.seed(10)
pcr.mod <-
  pcr(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
    data = REV.train,
    scale = TRUE,
    validation = "CV"
  )
summary(pcr.mod) #
```

```
## Data:    X dimension: 289 6
##  Y dimension: 289 1
## Fit method: svdpc
## Number of components considered: 6
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comp
s
## CV           0.4043   0.2536   0.2538   0.2382   0.2361   0.2350   0.232
7
## adjCV        0.4043   0.2534   0.2536   0.2380   0.2352   0.2348   0.232
4
##
## TRAINING: % variance explained
##               1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X               45.52    62.86    79.38    88.94    97.89   100.00
## LnHousePrice    61.10    61.28    65.74    67.13    67.13    67.96
```

```
validationplot(pcr.mod, val.type = "MSEP")
```




```
pcr.pred <-
  predict(pcr.mod, REV.test, ncomp = 4) # REV.test - should it be 6 compon
ents?
mean((exp(pcr.pred) - REV.test$HousePrice) ^ 2)  # MSE = 67.50837, same as
ridge
```

```
## [1] 67.50837
```

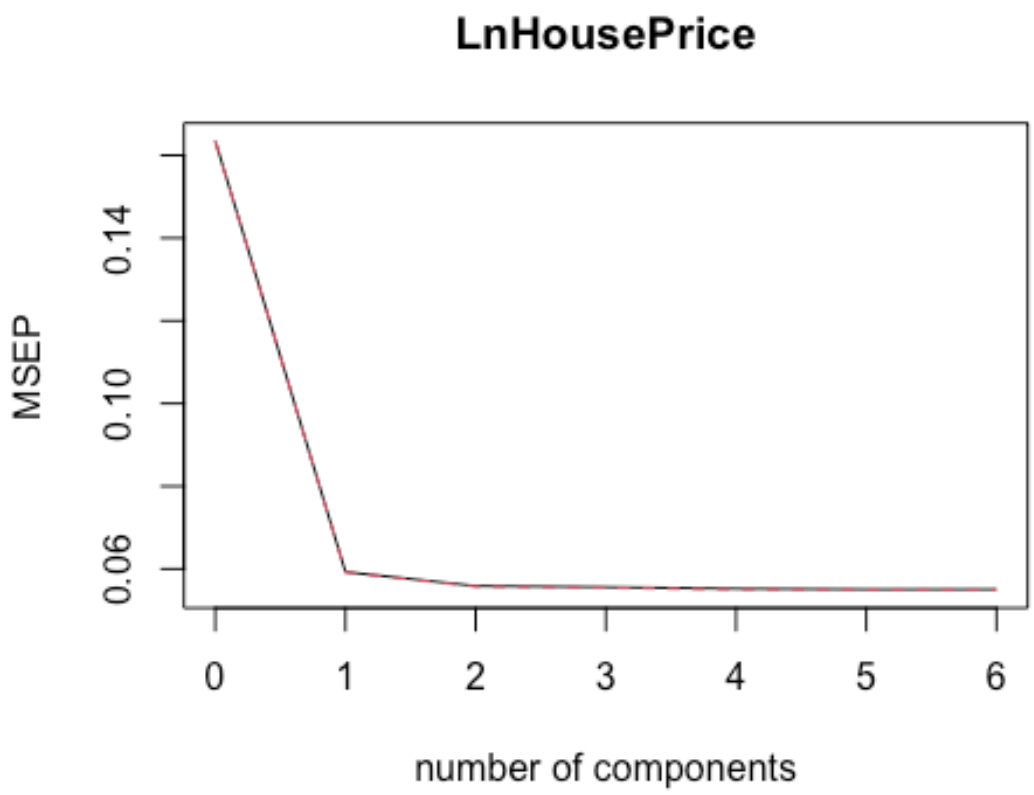
```
ln.test.avg <- mean(RealEstateValuation_Data_$LnHousePrice)
pcr.test.r2 <-
  1 - mean((REV.test$LnHousePrice - pcr.pred) ^ 2) / mean((REV.test$LnHous
ePrice - ln.test.avg) ^
                                                          2)
pcr.test.r2
```

```
## [1] 0.676837
```

```
# PLS
pls.mod <-
  plsr(
    LnHousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
    data = REV.train,
    scale = TRUE,
    validation = "CV"
  )
summary(pls.mod)
```

```
## Data:    X dimension: 289 6
##  Y dimension: 289 1
## Fit method: kernelpls
## Number of components considered: 6
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comp
s
## CV          0.4043   0.2434   0.2364   0.2358   0.2350   0.2347   0.234
6
## adjCV       0.4043   0.2433   0.2361   0.2354   0.2346   0.2343   0.234
3
##
## TRAINING: % variance explained
##               1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X               45.19    61.06    70.45    74.17    91.04   100.00
## LnHousePrice    64.49    67.27    67.66    67.95    67.96    67.96
```

```
validationplot(pls.mod, val.type = "MSEP") # shows ideal M is 13 component
s
```



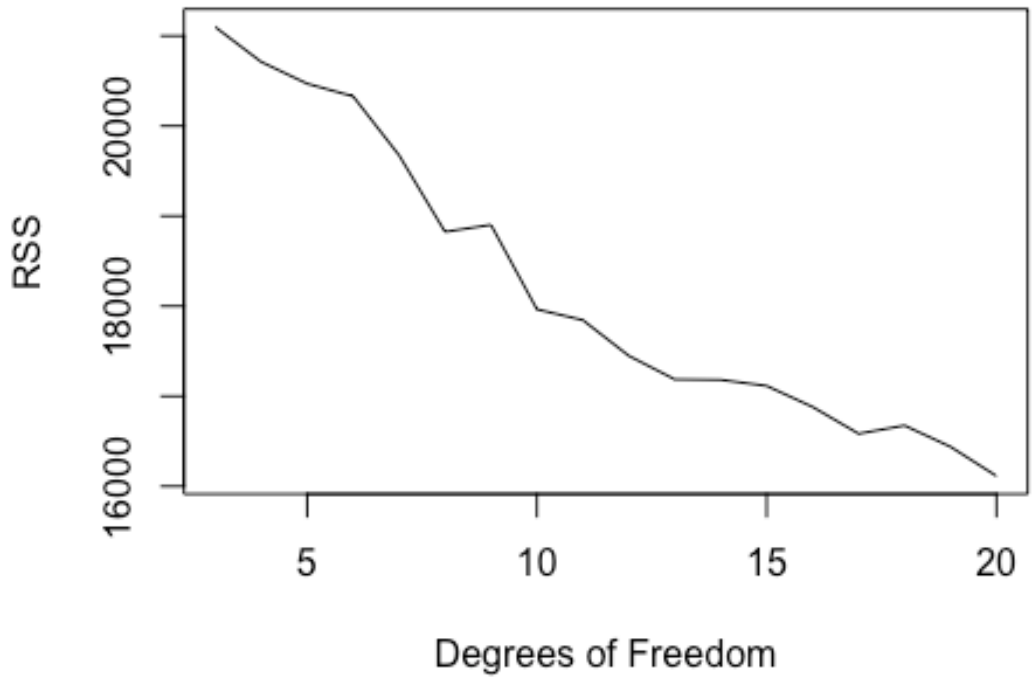

```
pls.pred <- predict(pls.mod, REV.test, ncomp = 4) # REV.test
mean((exp(pls.pred) - REV.test$HousePrice) ^ 2)  # MSE = 61.75218

## [1] 61.75218

ln.test.avg = mean(REV.test$LnHousePrice)
pcr.test.r2 = 1 - mean((REV.test$LnHousePrice - pcr.pred) ^ 2) / mean((REV
.test$LnHousePrice - ln.test.avg) ^
                                                                      2)
pcr.test.r2

## [1] 0.6768346

##### GAM #####
# FIND TOP 3 REGRESSORS AND THEN WORK OUT OPTIMAL SPLITS/NO. LOCAL REGRESS
IONS, check best variables using subset results #
reg.full <-
  regsubsets(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = RealEstateValuation_Data_,
    nvmax = 6
  )
reg.summary <- summary(reg.full)
reg.summary # Top 3 regressors, MRTDist, StoreNum, HouseAge

## Subset selection object
## Call: regsubsets.formula(HousePrice ~ TransactionDate + HouseAge +
```

```
##     MRTDist + StoreNum + Latitude + Longitude, data = RealEstateValuati
on_Data_,
##     nvmax = 6)
## 6 Variables  (and intercept)
##                 Forced in Forced out
## TransactionDate     FALSE      FALSE
## HouseAge            FALSE      FALSE
## MRTDist             FALSE      FALSE
## StoreNum            FALSE      FALSE
## Latitude            FALSE      FALSE
## Longitude           FALSE      FALSE
## 1 subsets of each size up to 6
## Selection Algorithm: exhaustive
##          TransactionDate HouseAge MRTDist StoreNum Latitude Longitude
## 1  ( 1 ) " "             " "      "*"     " "      " "      " "
## 2  ( 1 ) " "             " "      "*"     "*"      " "      " "
## 3  ( 1 ) " "             "*"      "*"     "*"      " "      " "
## 4  ( 1 ) " "             "*"      "*"     "*"      "*"      " "
## 5  ( 1 ) "*"             "*"      "*"     "*"      "*"      " "
## 6  ( 1 ) "*"             "*"      "*"     "*"      "*"      "*"
```

```
set.seed(10)

# Range of DoF, plotted fits and RSS
sp.rss <- rep(0, 18)
for (i in 3:20) {
  sp.fit1 <-
    lm(HousePrice ~ bs(MRTDist, df = i) + bs(StoreNum, df = i) +
                    bs(HouseAge, df = i), data = REV.train)
  sp.rss[i - 2] <- sum(sp.fit1$residuals ^ 2) # storing RSS
}
sp.rss
```

```
##  [1] 21098.41 20711.70 20469.62 20329.66 19674.79 18825.02 18902.29 179
64.62
##  [9] 17844.61 17448.22 17186.96 17181.82 17114.64 16880.74 16583.86 166
74.40
## [17] 16438.63 16113.72
```

```
plot(3:20,
     sp.rss,
     ylab = "RSS",
     xlab = "Degrees of Freedom",
     type = "l")
```




```
min(sp.rss) # 16113.72
```

```
## [1] 16113.72
```

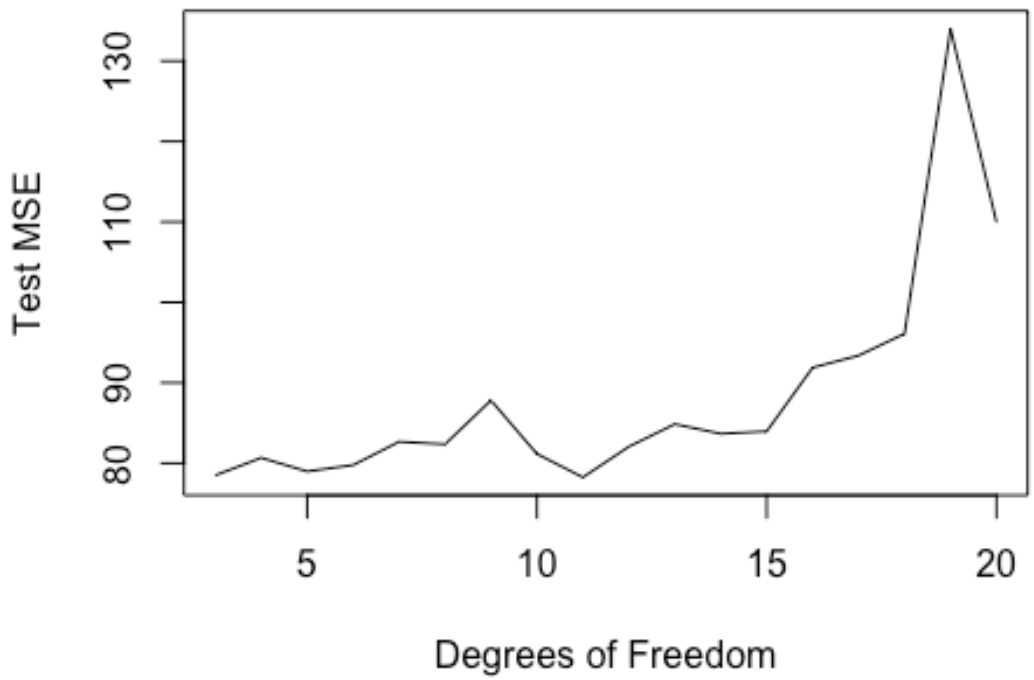
```
# CV to select the best DoF
cv.err <- rep(0, 18)
for (i in 3:20) {
  cv.fit <-
    glm(HousePrice ~ bs(MRTDist, df = i) + bs(StoreNum, df = i) + bs(House
Age, df =
                                                                      i),
        data = REV.train)
  cv.err[i - 2] <-
    cv.glm(REV.train, cv.fit, K = 10)$delta[1] # i-2 because the for loop
starts at df = 3 so has to discount to fit the actual data which starts at
1.
```

```
cv.err
```

```
##  [1]  78.49204  80.66944  78.98475  79.77126  82.67007  82.36099  87.81
094
##  [8]  81.16298  78.25009  82.05895  84.86136  83.66825  83.96373  91.88
695
## [15]  93.38673  96.10243 134.00385 109.94086
```

```
plot(3:20,
     cv.err,
     ylab = "Test MSE",
```

```
     xlab = "Degrees of Freedom",
     type = "l")  # would choose df = 6 model, says 4 but have to add 2 on
```




```
min(cv.err) # DOF = 11 gives lowest cv error of 78.25009

## [1] 78.25009

gam.fit <-
  gam(
    HousePrice ~ bs(MRTDist, df = 11) + bs(StoreNum, df = 11) + bs(HouseAg
e, df = 11) + TransactionDate + Latitude,
    data = REV.train
  ) # REV.train
par(mfrow = c(2, 3))
plot(gam.fit, se = T, col = "green")
summary(gam.fit) # look at Anova for Nonparametric Effects, only one varia
ble is non-linear, can only reject one variables null hypothesis when test
ed for nonparametric effects, shows we need to focus our effect on Expend
and add more knots to it

##
## Call: gam(formula = HousePrice ~ bs(MRTDist, df = 11) + bs(StoreNum,
##     df = 11) + bs(HouseAge, df = 11) + TransactionDate + Latitude,
##     data = REV.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -27.3611  -3.6158  -0.4453   2.9669  60.9494
##
```

```
## (Dispersion Parameter for gaussian family taken to be 61.1485)
##
##     Null Deviance: 56453.21 on 288 degrees of freedom
## Residual Deviance: 15531.72 on 254 degrees of freedom
## AIC: 2043.584
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                          Df Sum Sq Mean Sq F value    Pr(>F)
## bs(MRTDist, df = 11)     11  33922 3083.84 50.4320 < 2.2e-16 ***
## bs(StoreNum, df = 11)    10   1191  119.05  1.9470 0.0396075 *
## bs(HouseAge, df = 11)    11   3496  317.80  5.1972 2.253e-07 ***
## TransactionDate           1    746  745.81 12.1967 0.0005645 ***
## Latitude                  1   1567 1567.08 25.6274 7.959e-07 ***
## Residuals               254  15532   61.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# MRTDist and HouseAge very non linear

gam.pred <- predict(gam.fit, REV.test) # REV.test
```

```
## Warning in bs(MRTDist, degree = 3L, knots = c(`11.11111%` = 185.4296,
## `22.22222%` = 289.3248, : some 'x' values beyond boundary knots may cau
se ill-
## conditioned bases
```

```
## Warning in bs(MRTDist, degree = 3L, knots = c(`11.11111%` = 185.4296,
## `22.22222%` = 289.3248, : prediction from a rank-deficient fit may be m
isleading
```

```
mean((REV.test$HousePrice - gam.pred) ^ 2) # 53.11884 MSE
```

```
## [1] 53.11884
```

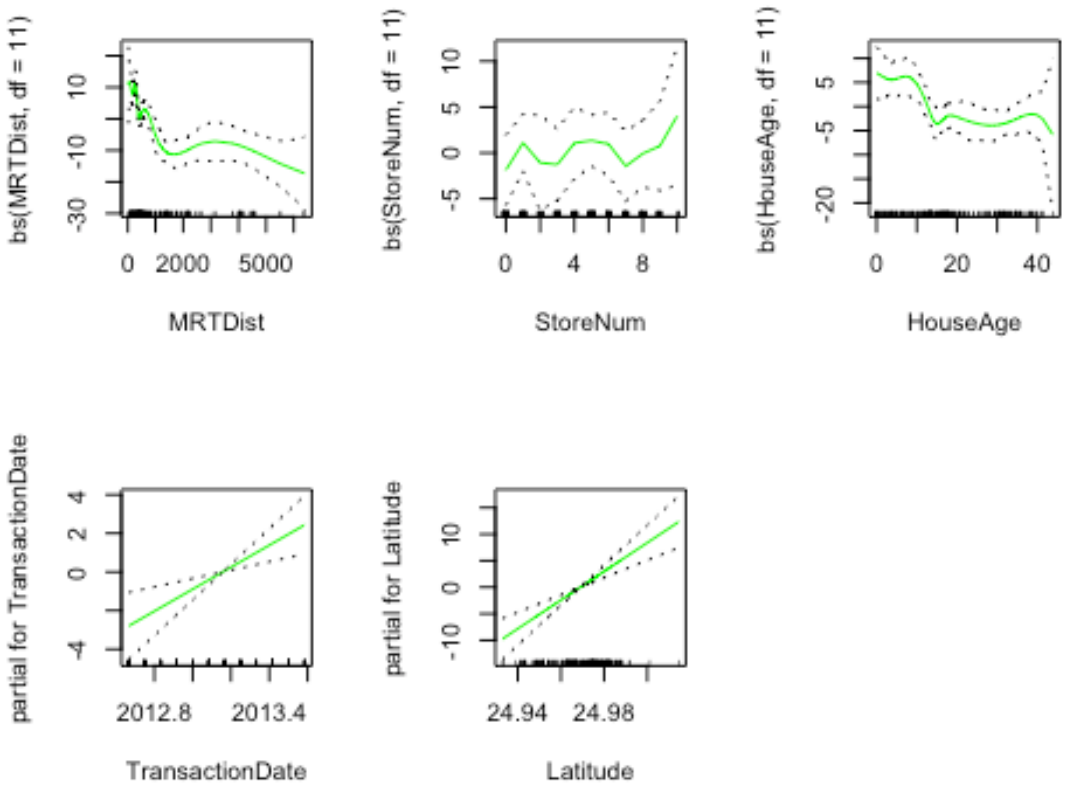
```
# R2 = MSS/TSS = (TSS-RSS)/TSS = 1 - RSS/TSS
gam.test1.r2 <-
  1 - sum((REV.test$HousePrice - gam.pred) ^ 2) / sum((REV.test$HousePrice
- mean(REV.test$HousePrice)) ^
                                                      2)
gam.test1.r2 # = 0.6677571
```

```
## [1] 0.6677571
```

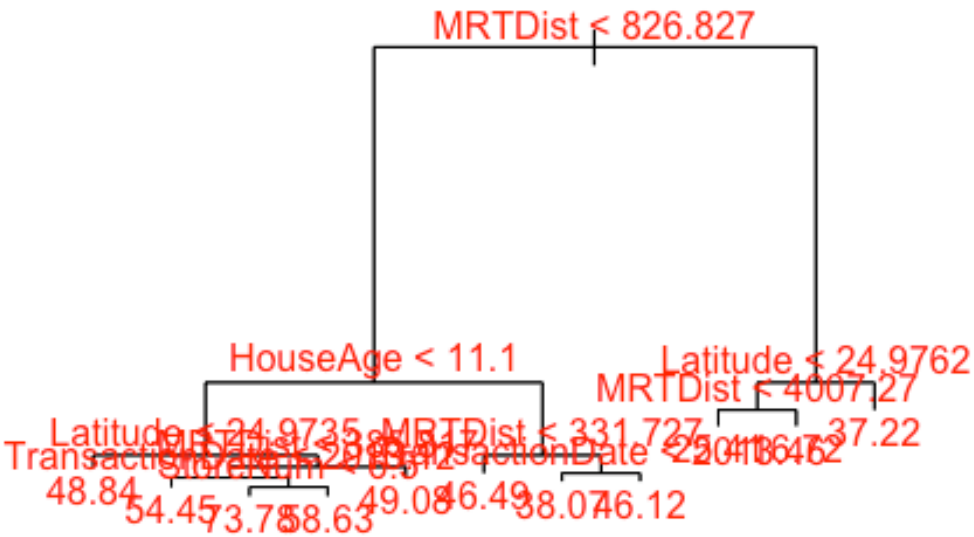
```
##### Regression Tree #####
dim(RealEstateValuation_Data_)
```

```
## [1] 414   9
```

```
set.seed(10)
par(mfrow = c(1, 1))
```



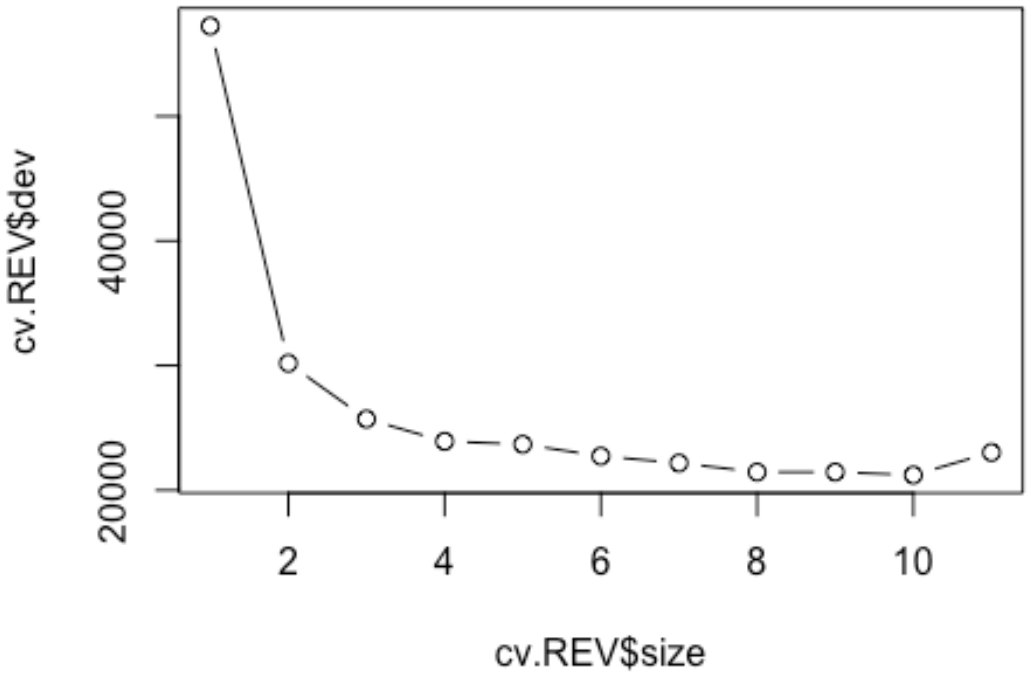

```
regtree.REV  <-
  tree(HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Lati
tude + Longitude,
       data = REV.train) # work out what's going on then add back in + Lat
itude + Longitude
summary(regtree.REV)

##
## Regression tree:
## tree(formula = HousePrice ~ TransactionDate + HouseAge + MRTDist +
##     StoreNum + Latitude + Longitude, data = REV.train)
## Variables actually used in tree construction:
## [1] "MRTDist"         "HouseAge"        "Latitude"        "TransactionD
ate"
## [5] "StoreNum"
## Number of terminal nodes:  11
## Residual mean deviance:  48.86 = 13580 / 278
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -30.4700  -3.8170   0.1625   0.0000   3.3630  43.7200

plot(regtree.REV)
text(regtree.REV, pretty = 0, col = "Red")
```



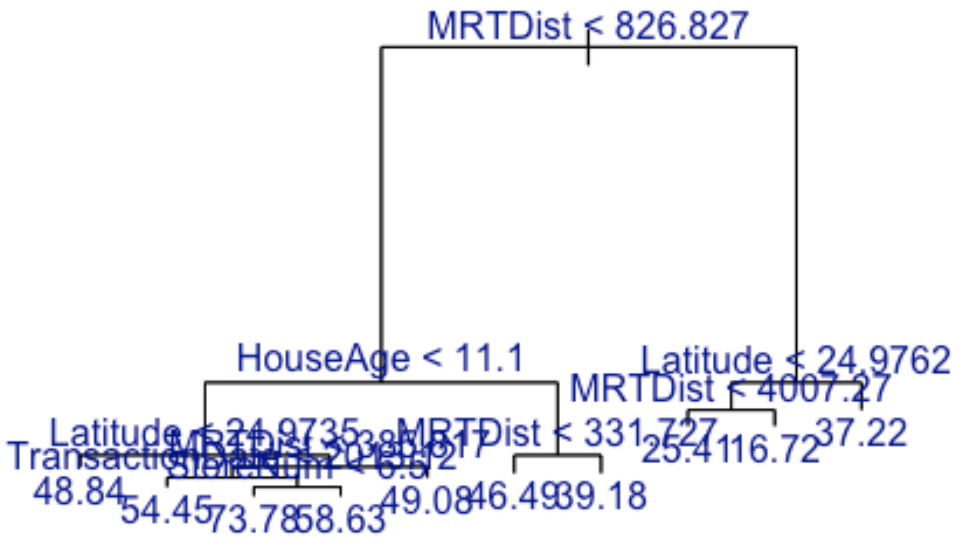

```
regtree.pred <- predict(regtree.REV, newdata = REV.test)
mean((regtree.pred - REV.test$HousePrice) ^ 2) # MSE = 58.07941

## [1] 58.07941

regtree.test1.r2 <-
  1 - sum((REV.test$HousePrice - regtree.pred) ^ 2) / sum((REV.test$HouseP
rice - mean(REV.test$HousePrice)) ^
                                                          2)
regtree.test1.r2# = 0.6367302

## [1] 0.6367302

# CV to determine optimum complexity #
set.seed(10)
cv.REV <- cv.tree(regtree.REV)
cv.REV

## $size
##  [1] 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 23026.54 21215.04 21452.55 21452.55 22177.42 22730.31 23682.26 239
17.92
##  [9] 25707.99 30184.25 57234.89
##
## $k
##  [1]       -Inf   669.0389   742.0005   778.9869   892.8029   963.9144
```

```
##  [7]  1324.1022  1486.6283  2270.1387  6006.1849 27735.2941
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
par(mfrow = c(1, 1))
plot(cv.REV$size, cv.REV$dev, type = "b") #  min occurs at size 10, $dev = RSS
```

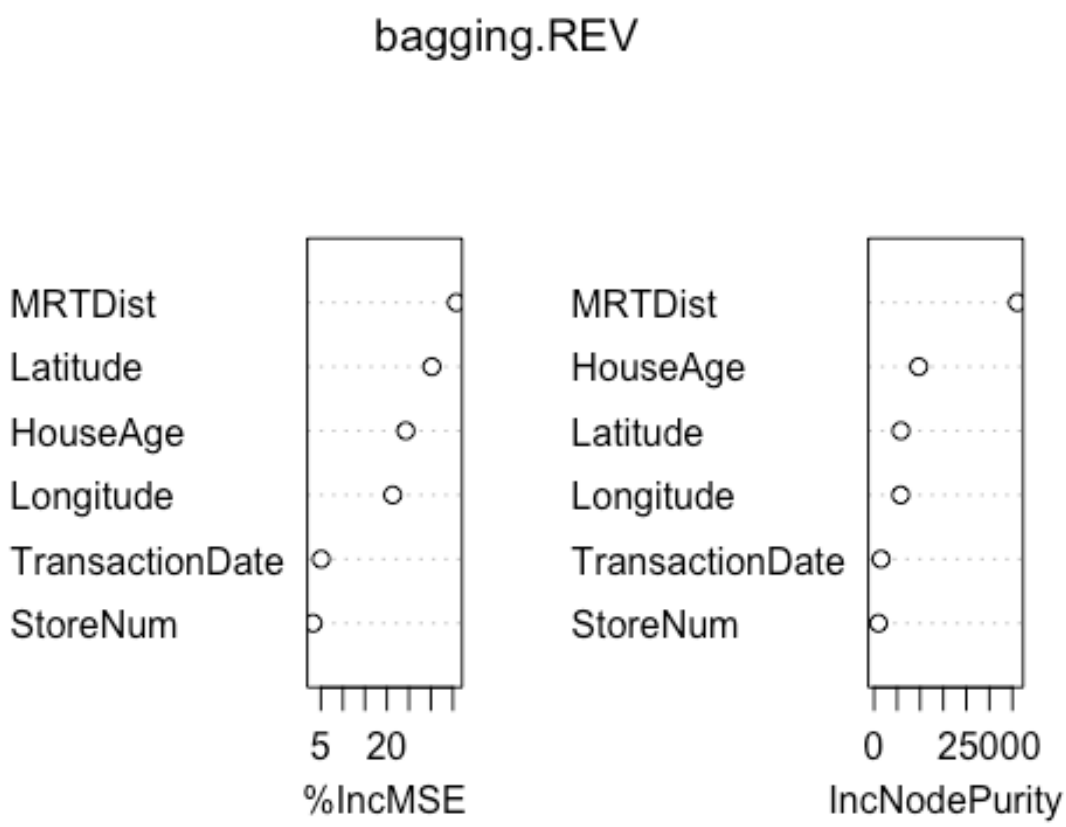



```
# tree-pruning #
prune.REV <-
  prune.tree(regtree.REV, best = 10) # command for weakest link pruning, best = what size tree we want
plot(prune.REV)
text(prune.REV, pretty = 0, col = "Dark Blue")
```




```
regtree.prune.pred <- predict(prune.REV, newdata = REV.test)
mean((REV.test$HousePrice - regtree.prune.pred) ^ 2) # Marginal Improvemen
t in MSE, 56.34469

## [1] 56.34469

regtree.prune.r2 <-
  1 - sum((REV.test$HousePrice - regtree.prune.pred) ^ 2) / sum((REV.test$
HousePrice - mean(REV.test$HousePrice)) ^
                                                                2)
regtree.prune.r2 # = 0.6475803

## [1] 0.6475803

##### Bagging, Random Forests and Boosting #####
# Bagging #
set.seed(10)
bagging.REV <-
  randomForest(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    mtry = 6,
    importance = TRUE
  ) # mtry = how many variables you want to try on each node, equal to 10
because 10 predictor variables in the data
bagging.REV  # % var explained sort of like r-sqaured from linear regressi
on
```

```
##
## Call:
##  randomForest(formula = HousePrice ~ TransactionDate + HouseAge +
MRTDist + StoreNum + Latitude + Longitude, data = REV.train,      mtry = 6
, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 65.37267
##                     % Var explained: 66.53

pred.bagging <- predict(bagging.REV, newdata = REV.test)
mean((REV.test$HousePrice - pred.bagging) ^ 2) # test MSE = 47.46447

## [1] 47.46447

bagging.test1.r2 <-
  1 - sum((REV.test$HousePrice - pred.bagging) ^ 2) / sum((REV.test$HouseP
rice - mean(REV.test$HousePrice)) ^
2)
bagging.test1.r2 # = 0.7031235

## [1] 0.7031235

importance(bagging.REV) # %IncMSE percentage amount MSE will increase on a
verage if you remove the variable

##                   %IncMSE IncNodePurity
## TransactionDate  5.068434     1513.2704
## HouseAge        24.307029     9673.9463
## MRTDist         35.788633    31073.6714
## StoreNum         3.226787      957.8638
## Latitude        30.233011     5820.8623
## Longitude       21.316598     5779.8930

varImpPlot(bagging.REV)
```




```
# Random Forests #
set.seed(10)
rf0.REV <-
  randomForest(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    mtry = 5,
    importance = TRUE
  )
pred.rf0 <- predict(rf0.REV, newdata = REV.test)
mean((pred.rf0  - REV.test$HousePrice) ^ 2) # test MSE = 46.75659

## [1] 45.00267

rf1.REV <-
  randomForest(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    mtry = 4,
    importance = TRUE
  )
pred.rf1 <- predict(rf1.REV, newdata = REV.test)
mean((pred.rf1  - REV.test$HousePrice) ^ 2) # test MSE = 44.79176

## [1] 42.5712
```

```
rf2.REV <-
  randomForest(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    mtry = 3,
    importance = TRUE
  )
pred.rf2 <- predict(rf2.REV, newdata = REV.test)
mean((pred.rf2  - REV.test$HousePrice) ^ 2) # test MSE = 41.35314
```

```
## [1] 40.57722
```

```
rf3.REV <-
  randomForest(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    mtry = 2,
    importance = TRUE
  )
pred.rf3 <- predict(rf3.REV, newdata = REV.test)
mean((pred.rf3  - REV.test$HousePrice) ^ 2) # test MSE = 40.31746 # Use th
is one #
```

```
## [1] 41.00969
```

```
summary(rf3.REV)
```

```
##                 Length Class  Mode
## call              5    -none- call
## type              1    -none- character
## predicted       289    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times       289    -none- numeric
## importance       12    -none- numeric
## importanceSD      6    -none- numeric
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y               289    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

```
rf4.REV <-
  randomForest(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    mtry = 1,
```

```
    importance = TRUE
  )
pred.rf4 <- predict(rf4.REV, newdata = REV.test)
mean((pred.rf4  - REV.test$HousePrice) ^ 2) # test MSE = 41.30107
```

```
## [1] 41.68195
```

```
rf3.REV.test <-
  randomForest(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    mtry = 2,
    importance = TRUE
  )
pred.rf3.test <- predict(rf3.REV, newdata = REV.test)
mean((pred.rf3.test  - REV.test$HousePrice) ^ 2) # test MSE = 40.31746 # U
se this one #
```

```
## [1] 41.00969
```

```
rf3.test.r2 <-
  1 - sum((REV.test$HousePrice - pred.rf3) ^ 2) / sum((REV.test$HousePrice
- mean(REV.test$HousePrice)) ^
                                                      2)
rf3.test.r2 # = 0.7456565
```

```
## [1] 0.7434963
```

```
# test MSE # better performance because the random tree construction is ve
ry different and so variance is reduced because they are uncorrelated with
each other and on average variance is lower
# if we have all 10 variables are available the trees will look very simil
ar whereas with fewer variables available in randomForest it can outperfor
m bagging as variance is reduced by having more different trees
```

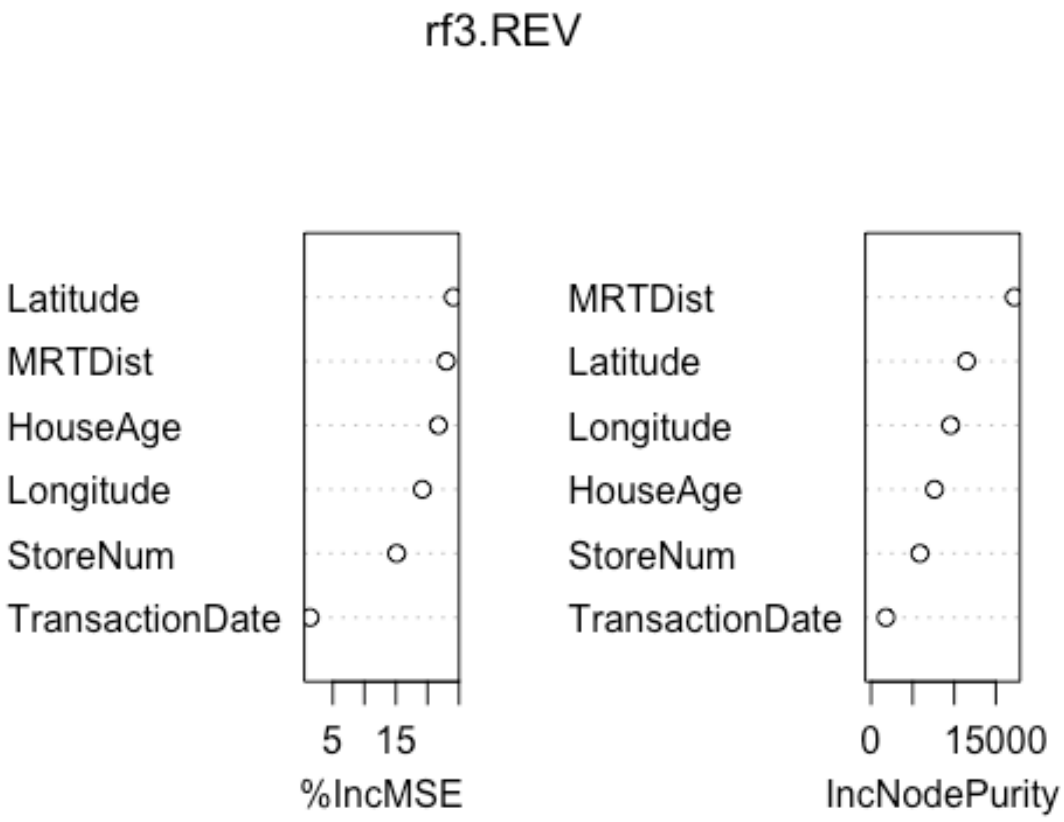
```
rf3.REV
```

```
##
## Call:
##  randomForest(formula = HousePrice ~ TransactionDate + HouseAge +
MRTDist + StoreNum + Latitude + Longitude, data = REV.train,      mtry = 2
, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 62.88434
##                     % Var explained: 67.81
```

```
importance(rf3.REV)
```

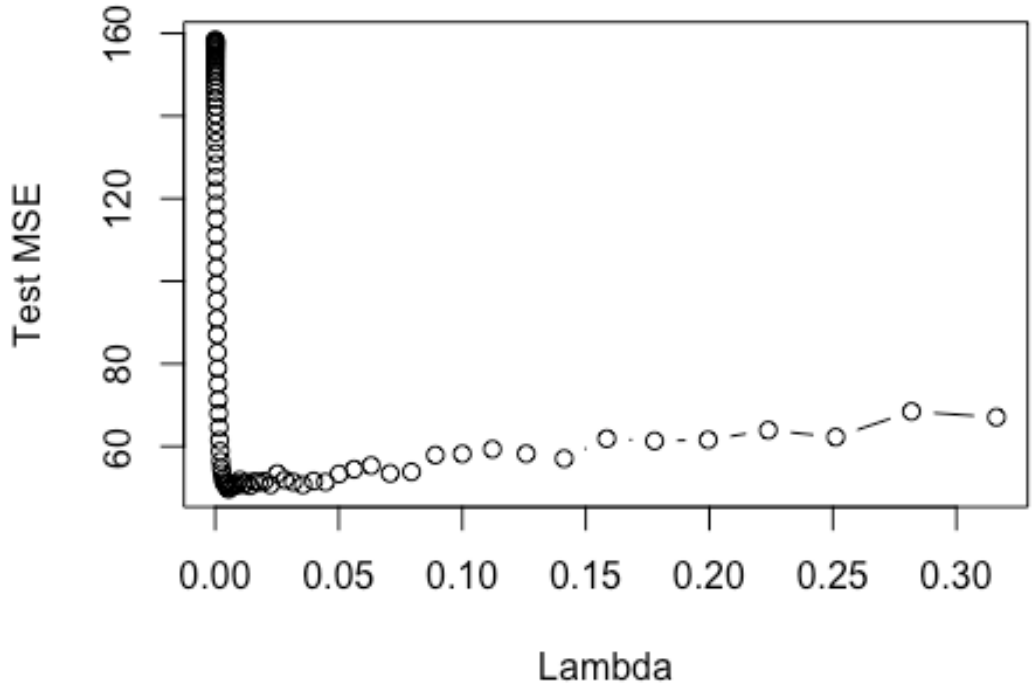
```
##                   %IncMSE IncNodePurity
## TransactionDate  1.395255      1792.482
## HouseAge        21.728589      7622.133
## MRTDist         22.979274     17226.623
```

```
## StoreNum         15.116271      5872.734
## Latitude         24.099782     11483.747
## Longitude        19.186697      9562.770

varImpPlot(rf3.REV)
```




```
# Boosting #
set.seed(10)

lambda <- 10 ^ seq(from = -5, to = -0.5, by = 0.05) # learning rate
test.error <- rep(-1, length(lambda))

for (i in 1:length(lambda)) {
  boosting.REV.train <-
    gbm(
      HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latit
ude + Longitude,
      data = REV.train,
      distribution = "gaussian",
      n.trees = 1000,
      shrinkage = lambda[i]
    )
  pred.boosting <-
    predict(boosting.REV.train, REV.test, n.trees = 1000)
  test.error[i] <- mean((pred.boosting - REV.test$HousePrice) ^ 2)
}
plot(lambda,
```

```
     test.error,
     type = "b",
     xlab = "Lambda",
     ylab = "Test MSE")
```




```
bestlam.boosting <-
  lambda[which.min(test.error)] # shrinkage parameter - lambda - correspon
ding to the minimum
bestlam.boosting # minimum test MSE (Boosting) 4.721133
```

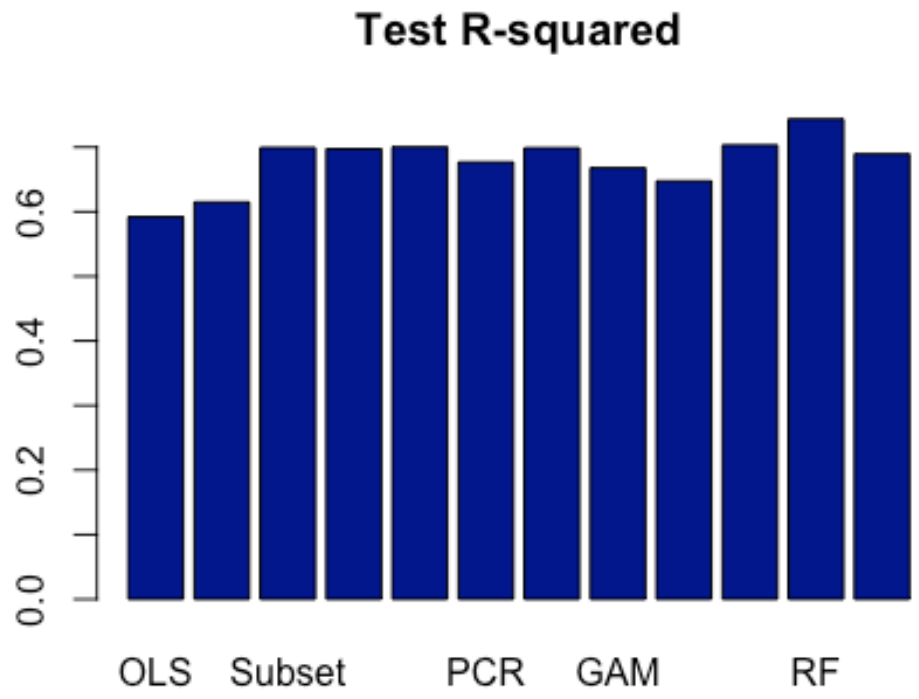
```
## [1] 0.005623413
```

```
boosting.REV <-
  gbm(
    HousePrice ~ TransactionDate + HouseAge + MRTDist + StoreNum + Latitud
e + Longitude,
    data = REV.train,
    distribution = "gaussian",
    n.trees = 1000,
    shrinkage = bestlam.boosting
  )
pred.boosting <- predict(boosting.REV, REV.test, n.trees = 1000)

mean((REV.test$HousePrice - pred.boosting) ^ 2)
```

```
## [1] 49.66527
```

```
test.avg = mean(REV.test$HousePrice)
boosting.test.r2 = 1 - mean((REV.test$HousePrice - pred.boosting) ^ 2) / m
ean((REV.test$HousePrice - test.avg) ^

2)
boosting.test.r2

## [1] 0.6893582

# Method Comparison
# R-squared
par(mfrow = c(1, 1))
test.avg = mean(REV.test$HousePrice)
ln.test.avg = mean(REV.test$LnHousePrice)
lm.test.r2 = 1 - mean((REV.test$HousePrice - lm.pred) ^ 2) / mean((REV.tes
t$HousePrice - test.avg) ^
                                                                    2)
ln.test.r2 <-
  1 - sum((REV.test$HousePrice - converted.back) ^ 2) / sum((REV.test$Hous
ePrice - test.avg) ^
                                                          2)
ln.bestsubset.r2 <-
  1 - sum((REV.test$LnHousePrice - lm.REV.bestsubset.test) ^ 2) / sum((REV
.test$LnHousePrice - ln.test.avg) ^
                                                                      2)
ridge.test.r2 = 1 - mean((REV.test$LnHousePrice - ridge.pred1) ^ 2) / mean
((REV.test$LnHousePrice - ln.test.avg) ^

2)
lasso.test.r2 = 1 - mean((REV.test$LnHousePrice - lasso.pred1) ^ 2) / mean
((REV.test$LnHousePrice - ln.test.avg) ^

2)
pcr.test.r2 = 1 - mean((REV.test$LnHousePrice - pcr.pred) ^ 2) / mean((REV
.test$LnHousePrice - ln.test.avg) ^
                                                                    2)
pls.test.r2 = 1 - mean((REV.test$LnHousePrice - pls.pred) ^ 2) / mean((REV
.test$LnHousePrice - ln.test.avg) ^
                                                                    2)
gam.test.r2 = 1 - mean((REV.test$HousePrice - gam.pred) ^ 2) / mean((REV.t
est$HousePrice - test.avg) ^
                                                                  2)
regtree.test.r2 = 1 - mean((REV.test$HousePrice - regtree.prune.pred ) ^ 2
) / mean((REV.test$HousePrice - test.avg) ^

2)
bagging.test.r2 = 1 - mean((REV.test$HousePrice - pred.bagging) ^ 2) / mea
n((REV.test$HousePrice - test.avg) ^

2)
randomforest.test.r2 = 1 - mean((REV.test$HousePrice - pred.rf3.test) ^ 2)
/ mean((REV.test$HousePrice - test.avg) ^

2)
```

```
boosting.test.r2 = 1 - mean((REV.test$HousePrice - pred.boosting) ^ 2) / m
ean((REV.test$HousePrice - test.avg) ^

2)
R2.plot <-
  barplot(
    c(
      lm.test.r2,
      ln.test.r2,
      ln.bestsubset.r2,
      ridge.test.r2,
      lasso.test.r2,
      pcr.test.r2,
      pls.test.r2,
      gam.test.r2,
      regtree.test.r2,
      bagging.test.r2,
      randomforest.test.r2,
      boosting.test.r2
    ),
    col = "darkblue",
    names.arg = c(
      "OLS",
      "LnOLS",
      "Subset",
      "Ridge",
      "Lasso",
      "PCR",
      "PLS",
      "GAM",
      "RegTree",
      "Bagging",
      "RF",
      "Boosting"
    ),
    main = "Test R-squared"
  )
```




```
# MSE
lm.MSE <- mean((REV.test$HousePrice - lm.pred) ^ 2)
ln.MSE <- mean((REV.test$HousePrice - (exp(ln.pred))) ^ 2)
ln.bestsubset.MSE <-
  mean((REV.test$HousePrice - (exp(
    lm.REV.bestsubset.test
  ))) ^ 2)
ridge.MSE <- mean((REV.test$HousePrice - (exp(ridge.pred1))) ^ 2)
lasso.MSE <- mean((REV.test$HousePrice - (exp(lasso.pred1))) ^ 2)
pcr.MSE <- mean((REV.test$HousePrice - (exp(pcr.pred))) ^ 2)
pls.MSE <- mean((REV.test$HousePrice - (exp(pls.pred))) ^ 2)
gam.MSE <- mean((REV.test$HousePrice - gam.pred) ^ 2)
regtree.MSE <- mean((REV.test$HousePrice - regtree.prune.pred) ^ 2)
bagging.MSE <- mean((REV.test$HousePrice - pred.bagging) ^ 2)
randomforest.MSE <- mean((REV.test$HousePrice - pred.rf3.test) ^ 2)
boosting.MSE <- mean((REV.test$HousePrice - pred.boosting) ^ 2)
MSE.plot <-
  barplot(
    c(
      lm.MSE,
      ln.MSE,
      ln.bestsubset.MSE,
      ridge.MSE,
      lasso.MSE,
      pcr.MSE,
      pls.MSE,
      gam.MSE,
```

```
      regtree.MSE,
      bagging.MSE,
      randomforest.MSE,
      boosting.MSE
    ),
    col = "red",
    names.arg = c(
      "OLS",
      "LnOLS",
      "Subset",
      "Ridge",
      "Lasso",
      "PCR",
      "PLS",
      "GAM",
      "RegTree",
      "Bagging",
      "RF",
      "Boosting"
    ),
    main = "Test MSE"
  )
```