CSC 423 Winter 2017
Kari Palmier
Chicago Rent Project Technical Report
Model Analysis and Conclusion, Exploratory Plots, Code
3/16/2017

1. Model Analysis

1.1 Model Interpretation

The final model is below.

$$\frac{Rent^{\lambda}-1}{\lambda} = 88 + 0.000199 * MedianIncome - 9.03 * WhiteNHProp - 10.01 * AfricanAmerNHProp - 10.2 * RenterProp - 24.8 * Educ8Years + 33.2 * EducBachGrad - 115 * Forecl - 8.26 * Uptown - 8.33 * LincolnPk - 9.92 * ForestGlen - 10.9 * NorthPk - 9.33 * NearWSide - 11.4 * ArmourSq - 12.4 * Douglas - 27.2 * Oakland - 7.16 * GrandBlvd - 7.39 * Kenwood - 11.4 * HydePk - 14.1 * AvalonPk - 26.7 * Riverdale - 7.91 * Bridgeport - 13.9 * Beverly$$

where λ = 0.629374811717281

This model indicates that the Box Cox transformation of rent is linearly dependent on the median family income, proportion of white non-Hispanic residents, proportion of African American non-Hispanic residents, proportion of total renter occupied housing, proportion of people at least 25 years old that have completed between 0 and 8 years of school, proportion of people at least 25 years old that have completed a bachelor or graduate degree, the proportion of foreclosures, and the Uptown, Lincoln Park, Forest Glen, North Park, Near West Side, Armour Square, Douglas, Oakland, Grand Boulevard, Kenwood, Hyde Park, Avalon Park, Riverdale, Bridgeport, and Beverly neighborhoods. This model was achieved using all data in the data set (aside from the incomplete rows or rows with zero values for rent), performing backward selection on the initial model, then eliminating the independent variables with the highest p value (over 0.05), then the highest VIF (over 10). More information will be provided in section 3.6. The overall R-squared coefficient of determination of the model is 0.672, the adjusted R-squared is 0.662, the F statistic is 73.2, and the p value is less than 2.00E-16. The p value being much lower than 0.05 and the F statistic being much higher than 2 indicate that at least one of the independent variables are significant to the model. The model R-square and adjusted R-squared values indicate that approx. 67% of the variation of the rent variable can be explained by the variation of the independent variables. While this value does show that more than half of the variation in rent can be explained by the independent variables, there is still a remaining 33% that is still not explained. This could be due to other variables that are not present in the data set provided (possibly population density of each census tract for example).

Standard Coefficients								
educ160pro	**mdfamy0**	**aanhp**	**whitenhp**	**educ8**	**renter**	**oaklandFlag**	**near_w_sideFlag**	**hyde_pkFlag**
0.4503	0.3257	-0.2671	-0.1736	-0.1659	-0.1372	-0.1156	-0.1136	-0.0905
forecl	**douglasFlag**	**lincoln_pkFlag**	**beverlyFlag**	**bridgeportFlag**	**uptownFlag**	**gd_blvdFlag**	**riverdaleFlag**	**armour_sqFlag**
0.0846	-0.0831	-0.0784	-0.078	-0.0626	-0.0606	-0.0605	-0.0569	-0.054
avalon_pkFlag	**north_pkFlag**	**forest_glenFlag**	**kenwoodFlag**					
-0.0522	-0.052	-0.0422	-0.0415					

VIF Values								
educ160pro	**mdfamy0**	**aanhp**	**whitenhp**	**educ8**	**renter**	**oaklandFlag**	**near_w_sideFlag**	**hyde_pkFlag**
6.31	6.03	9.34	8.78	5.28	2.1	1.03	1.07	1.18
forecl	**douglasFlag**	**lincoln_pkFlag**	**beverlyFlag**	**bridgeportFlag**	**uptownFlag**	**gd_blvdFlag**	**riverdaleFlag**	**armour_sqFlag**
1.29	1.04	1.64	1.06	1.05	1.05	1.1	1.01	1.02
avalon_pkFlag	**north_pkFlag**	**forest_glenFlag**	**kenwoodFlag**					
1.03	1.02	1.03	1.04					

R Model Coefficient Output					
	Estimate	**Std. Error**	**t value**	**Pr(>|t|)**	
(Intercept)	88.000000	3.140000	28.06	< 2 E -16	***
mdfamy0	0.000199	0.000031	6.49	0.000000	***
whitenhp	-9.030000	3.150000	-2.87	0.004250	**
aanhp	-10.100000	2.360000	-4.28	0.000021	***
renter	-10.200000	2.190000	-4.64	0.000004	***
educ8	-24.800000	7.030000	-3.53	0.000430	***
educ160pro	33.200000	3.780000	8.78	< 2 E -16	***
forecl	115.000000	0.316000	3.64	0.000290	***
uptownFlag	-8.260000	2.850000	-2.9	0.003880	**
lincoln_pkFlag	-8.330000	2.770000	-3	0.002770	**
forest_glenFlag	-9.920000	4.880000	-2.03	0.042310	*
north_pkFlag	-10.900000	4.340000	-2.52	0.011990	*
near_w_sideFlag	-9.330000	1.730000	-5.39	0.000000	***
armour_sqFlag	-11.400000	4.340000	-2.62	0.009050	**
douglasFlag	-12.400000	3.110000	-3.99	0.000072	***
oaklandFlag	-27.200000	4.870000	-5.58	0.000000	***
gd_blvdFlag	-7.160000	2.540000	-2.83	0.004830	**
kenwoodFlag	-7.390000	3.700000	-1.99	0.046490	*
hyde_pkFlag	-11.400000	2.800000	-4.08	0.000049	***
avalon_pkFlag	-14.100000	5.610000	-2.52	0.011860	*
riverdaleFlag	-26.700000	9.620000	-2.77	0.005680	**
bridgeportFlag	-7.910000	2.640000	-3	0.002820	**
beverlyFlag	-13.900000	3.750000	-3.7	0.000230	***

All of the independent variable coefficients have a p value less than 0.05 and t statistics above approximately 2, so they are all significant to the final model. Based on the standardized coefficients, the most significant independent variable on determining rent is the proportion of people over 25 years that have completed a bachelor or graduate degree. The second most significant is the median family income, followed by the proportion of African American non-Hispanic residents, proportion of white non-Hispanic residents, proportion of total renter occupied housing, the Oakland neighborhood, then the Near West Side neighborhood. The remaining independent variables have standardized coefficients less than 0.1, so while they are significant in the model due to their p values being less than 0.05, they are less significant as those listed. The VIF values of all of the independent variables of the model are less than 10, which indicates that there are no multicollinearity issues in the model.

Rent Changes In Dollars Per Unit Increase of Independent Variables (Only Independent Variable of Row of Table Below Changed - All Others Held Constant)									
		Initial Value (Mean for Non-Dummy, 0 for Dummy)	Unit Increment Value	BC(Initial Y)	BC(Final Y)	BC Lambda	Initial Y (Inverted BC Result)	Final Y Inverted BC Result)	Rent Change (In Dollars) Per Unit Increase of Coeff Variable
(Intercept)	88.000000								
mdfamy0	0.000199	46345.26	1	89.918027	89.918226	0.629375	626.618167	626.620332	0.002165
whitenhp	-9.030000	0.296	0.01	89.918027	89.827727	0.629375	626.618167	625.635964	-0.982203
aanhp	-10.100000	0.429	0.01	89.918027	89.817027	0.629375	626.618167	625.519616	-1.098550
renter	-10.200000	0.576	0.01	89.918027	89.816027	0.629375	626.618167	625.508743	-1.109423
educ8	-24.800000	0.129	0.01	89.918027	89.670027	0.629375	626.618167	623.922013	-2.696153
educ160pro	33.200000	0.231	0.01	89.918027	90.250027	0.629375	626.618167	630.234274	3.616107
forecl	115.000000	0.00962	0.01	89.918027	91.068027	0.629375	626.618167	639.176700	12.558533
uptownFlag	-8.260000	0	1	89.918027	81.658027	0.629375	626.618167	539.166240	-87.451927
lincoln_pkFlag	-8.330000	0	1	89.918027	81.588027	0.629375	626.618167	538.446069	-88.172098
forest_glenFlag	-9.920000	0	1	89.918027	79.998027	0.629375	626.618167	522.184239	-104.433928
north_pkFlag	-10.900000	0	1	89.918027	79.018027	0.629375	626.618167	512.253564	-114.364603
near_w_sideFlag	-9.330000	0	1	89.918027	80.588027	0.629375	626.618167	528.196928	-98.421238
armour_sqFlag	-11.400000	0	1	89.918027	78.518027	0.629375	626.618167	507.214164	-119.404002
douglasFlag	-12.400000	0	1	89.918027	77.518027	0.629375	626.618167	497.190879	-129.427287
oaklandFlag	-27.200000	0	1	89.918027	62.718027	0.629375	626.618167	357.760811	-268.857356
gd_blvdFlag	-7.160000	0	1	89.918027	82.758027	0.629375	626.618167	550.529980	-76.088186
kenwoodFlag	-7.390000	0	1	89.918027	82.528027	0.629375	626.618167	548.146666	-78.471501
hyde_pkFlag	-11.400000	0	1	89.918027	78.518027	0.629375	626.618167	507.214164	-119.404002
avalon_pkFlag	-14.100000	0	1	89.918027	75.818027	0.629375	626.618167	480.322110	-146.296057
riverdaleFlag	-26.700000	0	1	89.918027	63.218027	0.629375	626.618167	362.190649	-264.427518
bridgeportFlag	-7.910000	0	1	89.918027	82.008027	0.629375	626.618167	542.772444	-83.845723
beverlyFlag	-13.900000	0	1	89.918027	76.018027	0.629375	626.618167	482.295458	-144.322709

The inverse Box Cox transformation is below.

$$y = ((x * \lambda) + 1)^{\frac{1}{\lambda}}$$

Because the dependent variable rent had a Box Cox transformation applied to it, it was not straight forward to analyze the effect of a unit change of each independent variable on the rent. To achieve this, the Box Cox of the predicted value of rent was calculated at two different points, one with baseline values of all independent variables, and another with baseline values of all but the one independent variable in question. This independent variable had its baseline increase by its unit value. The baseline values of the independent values were chosen to be their mean values for all non-dummy variables and zero for all dummy variables (all neighborhoods not contributing). The unit increase was different for each type of variable. It was $1 for median income, 1% for the proportion independent variables, and 1 for the neighborhood dummy variables (the dummy variables represent the change in rent with or without them included, not a unit change). Once the predicted baseline (or initial) rent value and the unit increased (final) rent value were calculated, the inverse Box Cox transformation was applied to each separately. This yielded the actual baseline (initial) and unit increased (final) rent values. The difference of these points was then taken (final/unit increased rent minus the baseline/initial rent). This difference is the amount that rent changes due to the unit increase of the independent variable. A unit increase of $1 of median family income increases rent by $.002. More meaningfully, an increase of $1000 of median family income increases rent by $2.12. A 1% increase of white non-Hispanic leads to a decrease in rent of -$0.98. Likewise, a 1% increase in African American non-Hispanic decreases rent by -$1.10, a 1% increase in renter occupied housing decreases rent by -$1.11, and a 1% increase in people over 25 that completed education between 0 and 8 years decreases rent by $2.70. A 1% increase in people over 25 that completed a bachelor or graduate degree increases rent by $3.62 and a 1% increase in foreclosures increases rent by $12.56. Including the Uptown neighborhood decreases rent by -$87.45. Likewise, Lincoln Park decreases rent by -$88.17, Forest Glen decreases rent by -$104.43, North Park decreases rent by -$114.36, Near West Side decreases rent by $98.42, Armour Square decreases rent by -$119.40, Douglas decreases rent by -$129.43, Oakland decreases rent by -$268.86, Grand Boulevard decreases rent by -$76.09, Kenwood decreases rent by -$78.47, Hyde

Park decreases rent by -$119.40, Avalon Park decreases rent by -$146.30, Riverdale decreases rent by -$264.43, Bridgeport decreases rent by -$83.85, and Beverly decreases rent by -$144.32.

During the modeling process, two influential points were found and eliminated from the model. These points both had low rent and high median family incomes, which do not fit with the trend of the majority of the data. Rent was positively linearly correlated to median family income, and the coefficient of median family income in the model is positive, which means that low rent is associated with low income and high rent with high income. The points were found to be influential because their deleted studentized residual values were outside of the range of -3 to 3 and their hat values were above 0.5 (have high leverage). Cook's distance values were also used to access influential points, but no points had a Cook's distance over 1. Because these points were uncharacteristic and were influencing the model, they were removed from the model. Both influential points removed were in the Near South Side neighborhood. They were the only data points from that neighborhood, so their removal eliminated this neighborhood from the model. The modeling process also revealed one extreme outlier that had a standardized residual value above 10. Upon examination, it was found that this point corresponded to a census tract in the Near West Side neighborhood with very low rent and very low median family income, lower than the mean – standard deviation of rent and income of the Near West Side. Since the Near West Side is a neighborhood that is in the model, this point could possibly change the model by skewing the Near West Side data. Because of this reasons, this point was removed from the model.

1.2 Residual Analysis

The standardized residuals of the model plotted against the predicted values resulted in a scatterplot that had a large clustering of points between Box Cox predicted values of 75 and 100 (rent values of $472.28 and $739.82), and between standardized residual values of -1 and 1. The remaining points are scattered mostly randomly around this cluster of points. The behavior of this plot was mostly unbiased and homoscedastic (except for the presence of the cluster). There were four outlier present above the standardized residual value of 3 and twelve below -3. Even though these outliers are present, they have been left in the model because they are not extreme enough to warrant removal. When the data points corresponding to the outliers were analyzed, no results could be found for to warrant their removal (results fell within mean and standard deviations of most of the independent variables and those that did not were not grossly different). To see the impact of these outliers, they were removed during a separate modeling process. When the outliers outside of the range -3 and 3 were removed, more appeared in the residual plot of the subsequent model. When the outliers were removed from this next model, more appeared in the following model. When all outliers that appeared were removed during each modeling step, it resulted in over sixty points being removed from the data set. This was too many points to remove, especially since some neighborhoods only contain a few points. Removing sixty points resulted in the data for some neighborhoods being completely removed or mostly removed. It also caused some of the high and low value of proportion in dependent variables to be excluded. The lack of neighborhood and proportion data points caused a very different model outcome, one that was found to be incorrect due to the lack of data. These are the reasons that the rest of the outliers were left in the data set during the creation of the final model.

The normal plot of the standardized residuals showed very normal behavior between sample quantities of approximately -1 and 1. The plot strayed away from the normal line both below -1 and

above 1, indicative of a data distribution with thick tails. This behavior matches the behavior of the normal plot of the dependent rent variable itself. Different variables were added in an attempt to reduce these tails (these results are discussed in section 3.3). None of the variables added had a normalizing effect on the tails without negatively impacting the standardized residual versus predicted values scatterplot. Other rent transformations were also attempted (square root of rent, natural log of rent, inverse of rent, rent squared, and rent cubed). None of these transformations successfully reduced the tails of the residual normal plot (most exaggerated the tails). Since this abnormal behavior was present in the normal plot of the rent variable itself, the abnormal behavior of the residual normal plot is most likely due to the behavior of the actual rent variable itself.

The deleted studentized residuals versus hat value plot shows that after the influential points were removed from the model, no more influential points appeared. The plot still shows the sixteen outlier points. The hat value for all of these outliers are less than 0.5, so they do not have enough leverage to cause negative effects on the model itself.

The plot of the standardized residuals versus the proportion of people over 25 that completed a bachelor or graduate degree is unbiased and to have fairly consistent variance throughout the plot (homoscedastic). There is a cluster of data between proportions of 0 and 0.2, which implies that most census tracts have low proportions of people who have completed bachelor and graduate degrees. The plot of the standardized residuals versus the proportion of white non-Hispanic also is unbiased and homoscedastic. There is a higher concentration of points near the 0 proportion value, which implies that that most census tracts have low proportions of white non-Hispanic residents. The plot of the standardized residual versus the proportion of total renter occupied housing is unbiased and homoscedastic, with a cluster of points between proportions of 0.6 and 0.8. This means that the most common percentage of total renter occupied housing is 60% to 80% in the census tracts. The plot of standardized residuals versus the proportion of African American non-Hispanic is also homoscedastic. There are two groupings of data present in the plot, one between proportions 0 to 0.2 and one between proportions 0.9 and 1. This indicates that census tracts are divided in terms of the proportion of African Americans present. The proportion is either low or high. There are not many census tracts that have an equal amount of African American non-Hispanic residents as there are residents of other races (there is not a diverse spread of the African American non-Hispanic residents in most census tracts). The plot of the standardized residuals versus the proportion of foreclosures is unbiased and homoscedastic, but the vast majority of points are contained between the proportions of 0 and 0.35. This implies that there are low numbers of foreclosures throughout the census tracts. The plot of the standardized residuals versus the proportion of people who completed 0 to 8 years of school is unbiased and contains a cluster of points between the proportions 0 and 0.2. The points above 0.2 appear closer to the zero line than the points in the cluster, but this could be an artifact of the data set itself (there are less data points above 0.2), which may not imply an actual issue with heteroscedasticity. The plot of the standardized residuals versus the median family income shows very similar behavior to the plot of the standardized residuals versus predicted values (is unbiased and homoscedastic with a cluster between incomes of $25,000 and $75,000). This could indicate that the median family income has a large enough influence on the model that it shapes the residual versus predicted plot behavior. The median family income is the second most significant independent variable in the model, so this could explain the similarity in these results.

1.3 Residual Normalization

In order to attempt to normalize the standardized residual normal plot tails, three different types of variables were added to the models. Each variable type was added one at a time so the effect of the variables could be easily distinguished. The first type of variables added were dummy variables, one for the proportion of non-African American non-Hispanic residents and one for non-white non-Hispanic residents. The full modeling process was done with these variables included. The initial model (with all variables) assigned a value of NA for the coefficients of these new race dummy variables. In R, this means that these variables are redundant and should be removed. They were then removed during both the backward and stepwise selection processes. Due to this, it was determined that these variables did not influence the model and were removed from the final model. The second type of variables added were dummy variables corresponding to points that have rent values above $720 and below $450. These rent cut-off points were chosen because they are the rent values at which the rent normal plot begins to stray from the normal line. The full modeling procedure was also performed with these variables. These variables (referred to as low rent and high rent variables) did successfully reduce the tails of the standardized residual normal plot (tails were still present), but plot of the standardized residuals versus predicted values was negatively affected. This plot went from being mostly unbiased and homoscedastic with one cluster of points to having three distinct clusters, one at low predicted rent values, on at high predicted rent values, and the densest cluster in the middle. Because these variables caused the standardized residual versus predicted value plot to become biased and heteroscedastic, these variables were removed. The third variable added was a rent affordability index obtained by dividing the median family income index (income values divided by the mean income value) by the rent index (rent values divided by the mean rent value). This variable improved the model statistics (R-squared and F statistic) but created a slightly inverse quadratic bias in the standardized residuals versus predicted values. More, and larger, outliers were also present. The tail of the standardized residual normal plot actually increased with this variable. Because the standardized residual versus predicted values plot shows an issue with bias, it was removed from the model. Another issue with the rent affordability index variable and the low rent and high rent dummy variables is they all depend on knowing the dependent variable rent to be calculated. This makes them also dependent variables, which would not make sense to include as independent variables in the model.

1.4 Model Validation

Training and Testing Data Set Validation	
Model R-Sq	0.67
Val Train RMSE	9.46
Val Train MAE	6.62
Val Train MAPE	8.45
Val Test RMSE	9.6
Val Test MAE	6.84
Val Test MAPE	8.91
Val Test R-Sq	0.664
Val RMSE Diff	-0.14
Val MAE Diff	-0.22
Val MAPE Diff	-0.46
Train R-Sq - Test R-Sq	0.006

All Data Set Cross Validation	
Model R-Sq	0.672
CV Model RMSE	9.44
CV Model MAE	6.62
CV Model MAPE	8.46
CV Test RMSE	9.92
CV Test MAE	6.91
CV Test MAPE	8.85
CV Test R-Sq	0.638
CV RMSE Diff	-0.48
CV MAE Diff	-0.29
CV MAPE Diff	-0.39
Model R-Sq - CV R-Sq	0.034

The final model was obtained by using all of the data points remaining in the data set after the incomplete rows and rows containing zero values for rent. The data was then split randomly into a training data set containing 75% of the data and a testing set containing 25% of the data. A model

was then generated using the training set and the same formula (dependent and independent variables) as the final model. The model generated from the training data was then used to predict data in the testing set. The root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and R-squared were computed for both the model with the training set and the prediction data with the testing set. The difference of these statistics was then taken (training minus testing) to access how well the model predicted the data. The difference in the RMSE was -0.14, in the MAE was -0.22, in MAPE was -0.46, and in R-squared was 0.006. Because all of these values were very small (much lower than 1), the model was proven to be good to use for predictions. Note that these difference values may be better than should be expected with new data because all of the data points were used in generating the final model, which chose the independent variables to use (even though the training set model was the one used for predictions of the testing data). There could be some bias present that is generating better than expected predictions.

A 5-fold cross validation was also performed on the final model which used all data points (after eliminating incomplete rows and rows with zero rent). Each fold of the cross validated data predicted rent values for 1/5 of the data and there were no repeated points so the result was a complete set of predicted data the same size as the actual data set. The RMSE, MAE, MAPE, and R-squared were calculated for the original model data and for the cross validation predicted data. The difference between these statistics (model minus cross validated) was then taken to access how well the model predicted data. The difference in RMSE was -0.48, in MAE was -0.29, in MAPE was -0.39, and in R-squared was 0.034. Again, because these difference values were all much less than 1, the model was proven to be good to use for predictions.

1.5 Model Prediction

	Predicted Value	Pred Int LL	Pred Int UL	Conf Int LL	Conf Int UL	Actual Value	Actual - Pred	Diff % of Actual
All Data Model	$762.00	$553.00	$994.00	$738.00	$786.00	$728.00	-$34.00	-4.67
	$662.00	$465.00	$884.00	$640.00	$685.00	$545.00	-$117.00	-21.47
	$746.00	$539.00	$978.00	$723.00	$771.00	$925.00	$179.00	19.35
	$689.00	$489.00	$915.00	$667.00	$713.00	$1,033.00	$344.00	33.30

The final model was used to predict the rent for four of the data points that were eliminated from the data set due to being incomplete. Any missing data was replaced with the mean value of the data in the column corresponding to the missing entry. These data points were not used in the creation of the model (were completely new points to the model). The first two data points were only missing the proportion of foreclosures in one (replaced by zero) and total crime in the other (also replaced by zero) (the first two were only missing one entry each). The third and fourth points were missing the proportion of people over 25 who completed a bachelor or graduate degree, total crime, and personal crime. These were replaced with 0.2309, 0.0106, and 0.0235, respectively. The predicted result for the first data point predicted was $762 with a prediction interval of ($553, $994) and a confidence interval of ($738, $786). The second data point predicted was $662 with a prediction interval of ($465, $884) and a confidence interval of ($640, $685). The third data point predicted was $746 with a prediction interval of ($539, $978) and a confidence interval of ($723, $771). The fourth data point predicted was $689 with a prediction interval of ($489, $915) and a confidence interval of ($667, $713). The difference of the first predicted point to its actual value was -$34. The difference of the second was -$117. The difference of the third was $179. The difference of the fourth was $344. The prediction difference was close for the first data point and relatively close for the second. The third and the fourth differed enough for concern. It is possible

that the values used to populate the missing data of the third and fourth points were too far off of what they actually would have been for the neighborhoods and other independent variables. For example, the median family income of the third point was $102,198 and the proportion of people over 25 that completed bachelor or graduate degrees was replaced with 0.2309. From scatterplot of median family income versus proportion of people over 25 that completed bachelor or graduate degrees, the proportion values corresponding to an income of $100,000 were between approximately 0.5 and 1. This means the substituted value was much lower than it realistically would have been. The same is true about the proportion of people over 25 that completed bachelor or graduate degrees of the fourth point as well. The median family income was $85,489, corresponding to a proportion also between approximately 0.5 and 1. The replacement value of 0.2309 was also much lower than would be expected for this point also.

Because the dependent variable was the Box Cox transform of the rent variable, the inverse Box Cox transform had to be performed on prediction values, prediction intervals, and confidence intervals (see section 3.1 for the formula). All results provided in this section were after the inverse transform had been applied.

1.6 Modeling Process

The final model was created after a lengthy process involving several intermediate modeling steps. Initially, the data set was divided into the training and testing sets before any modeling was done. The data sets were chosen randomly, so there were cases where all of the data of one neighborhood ended up in the testing set. In these cases, those data points were moved from the testing to the training set because trying to predict data for neighborhoods not present in the model data would amount to extrapolation (which is unacceptable and would throw off predicted results). The code used to divide the data set and moel the data was repeated several times. It was noted that the models generated were vastly different each time (some had adjusted R-squared values of 0.5, others of 0.8, and the independent variables chosen by model selection were very different each time). It was concluded that this variability was due to different training data being chosen randomly during the data set division. The data set provided did not have enough points at each neighborhood to allow for a stable model using data set division from the start.

The first step of the final modeling process was to generate a model with all of the original independent variables. The output of this was analyzed and it was found that there were influential points present. These points were analyzed then removed, and another model was generated. This model did not have any influential points, but still had a very large residual outlier. This outlier was analyzed then removed, and another model was created. This model did not have any influential points or residual outliers greater than 5 or less than -5. Backward and stepwise model selection was performed on this model output to determine the best set of independent variables to use for the original rent variable. Next the Box Cox transform was applied to the rent variable and another model was generated (using data after outlier removal). Backward and stepwise model selection was performed on this model output to determine the best set of independent variables to use for the Box Cox transformed rent variable. At this point, four modeling loops were performed to eliminate any coefficients with large p values (above 0.05) and with VIF values above 10. One loop was performed using the original rent as the dependent variable and the independent variables recommended from the backward selection during the previous rent model. Another loop was performed using the original rent as the dependent variable and the independent variables

recommended from the stepwise selection during the previous rent model. The third loop was performed using the Box Cox transformed rent as the dependent variable and the independent variables recommended from the backward selection during the previous Box Cox rent model. The last loop was performed using the Box Cox transformed rent as the dependent variable and the independent variables recommended from the stepwise selection during the previous Box Cox rent model. Each of these loops modeled data then eliminated one variable at a time, eliminating the coefficient with the largest p value above 0.05 first, and if no coefficients had p values above 0.05, then by the largest VIF value above 10. This was continued until all of the coefficients had p values under 0.05 and VIF values under 10. Note that these loops were necessary because the models recommended by all model selection methods contained several coefficients that had p values higher than 0.05 and VIF values over 10. Once the four final models were achieved (rent backwards, rent stepwise, Box Cox rent backward, and Box Cox rent stepwise), the data set was divided into training and testing sets randomly (75% training, 25% testing). The training set was used to create four models, one using each set of independent variables from the four final models. Lastly, the four final models were used to create predictions of data that was not part of any modeling process. This entire modeling process was repeated with the method of eliminating data within the loops reversed (eliminating the largest VIF over 10 first, then the largest p value over 0.05). The entire original process (eliminating largest p value over 0.05 first then largest VIF over 10) was repeated again with including the three types of variables discussed in the section 3.3 to see the impact of each variable type (the prediction with new data step was not performed with these). Result analysis concluded these three additional types of variables should not be in the final model. Next, the models with the original rent dependent variable were excluded due to the non-normality of the rent variable histogram. After that, the models generated using the stepwise selection process were eliminated because the standardized residual versus predicted value plots were less homoscedastic and random than those of the backward selection, and the tails of the standardized residual normal plot strayed farther from the normal line than those of the backward selection. Finally, the model that was generated eliminating the largest VIF value above 10 first, then the largest p value above 0.05 was excluded because the p value significance levels of the coefficients generated were less significant than those generated using the largest p first, then largest VIF method (the largest p first method had more coefficients with lower p values than the largest VIF first). Most of the independent variables between the last two models were the same. The two that differed in the max VIF then max p value model were less correlated to the rent dependent variable. The residual plots of the two models were also almost identical, as were the training/testing validation and cross-validation differences in RMSE, MAE, MAPE, and R-squared (models were within tenths to hundredths of other). These are the reasons the significance p values of the coefficients were used to determine which model to use. The final model was generated using the Box Cox transform of the rent variable, the original variables in the data set with the addition of the neighborhood dummy variables, and backward model selection.

2. Conclusion

The final model obtained showed tha median family income, proportion of people over 25 that completed a bachelor or graduate degree, and the proportion of foreclosures have positive impacts on rent. The proportion of white non-Hispanic residents, proportion of African American non-Hispanic residents, proportion of total renter occupied housing, and proportion of people over 25 that completed 0 to 8 years of school have negative impacts on rent. The neighborhoods of Uptown, Lincoln Park, Forest Glen, North Park, Near West Side, Armour Square, Douglas, Oakland,

Grand Boulevard, Kenwood, Hyde Park, Avalon Park, Riverdale, Bridgeport, and Beverly all have negative impacts on rent as well. None of the crime related independent variables (total crime, personal crimes, property crimes, robbery, and murder) were present in the final model, implying rent is not dependent on crime. The neighborhoods in the model were evaluated to find if there were any commonalities that would cause them to be chosen in the modeling process, but none could be found. They are in different areas of the city, have different mean and standard deviation values of the independent variables, and are not unique with their mean, standard deviation, number of data points, or location in the city as compared to other neighborhoods not chosen. The model validation and prediction analysis showed that the model did a fairly good job predicting new rent values, but other factors present make the model not recommended for usage. There were several inconsistencies found between the model coefficient values and the expected behavior based on correlation and explanatory scatterplots. The correlation coefficient of rent and the proportion of white non-Hispanic was positive and high enough to imply that the proportion of white non-Hispanic should positively affect rent (the scatterplot of these variables confirmed this behavior) in the model, but the model indicates the opposite. This is also the case for the proportion of foreclosures. The correlation coefficient shows a negative relationship with rent (the scatterplot of these variables also confirmed this behavior), but the model indicates the relationship is positive. The boxplots of Lincoln Park and North Park indicate that the rent is higher in these areas, which implies that these neighborhoods should increase rent. The model, however, has these neighborhoods decreasing rent. There also appear to be possible independent variable interactions that were not taken into account. The scatterplot of median income versus the proportion of people over 25 that completed a bachelor or graduate degree and the proportion of white non-Hispanic residents show strong linear relationships. Even though these variables are linearly related, they are all present in the model and all have VIF values below 10. There are also other independent variables that show possible interaction that were not chosen in the final model. These inconsistencies may be brought about by other issues present in the model, mainly the distribution of the residuals not being normal. The residual normal plot shows that the residual distribution has large tails at high and low rent values. These tails were found to be a result of the distribution of the rent variable itself. After several attempts to normalize the residuals through the addition of different types of variables, it was determined that the rent variable actually represents three different distributions, one with the low rent values, one with the middle rent values, and one with the high rent values. A recommendation to improve the model would be to actually break apart the rent variable into these three different distributions, then generate separate models for each. Another recommendation would be to generate the model using interaction terms for the independent variables whose scatterplots showed linear relationshipts (the median income and the proportion of people over 25 that completed a bachelor and graduate degree, the median income and the proportion of white non-Hispanic residents, the proportion of total crime and the proportion of property crime, the proportion of robbery and the proportion of personal crime, the proportion of white non-Hispanic residents and the proportion of African American non-Hispanic residents, the proportion of white non-Hispanic residents and the proportion of people over 25 that completed a bachelor or graduate degree, and finally the proportion of Hispanic residents and the proportion of people over 25 that completed between 0 and 8 years of school). These interaction terms are recommended due to the linear relationships seen between these variables in their scatterplots, as well as large correlation coefficients between them in the correlation matrix.

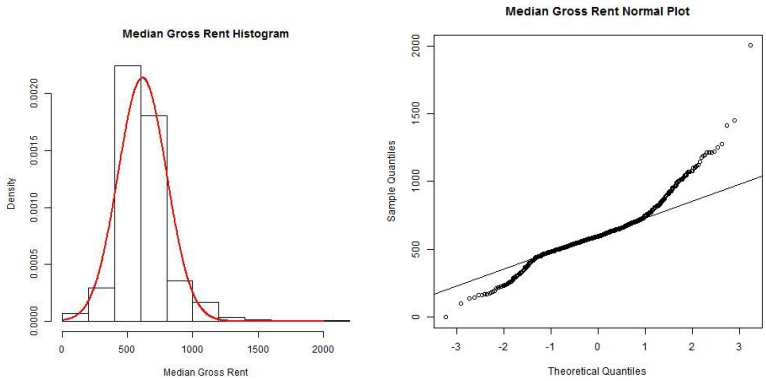
3. Appendix

3.1 Exploratory Analysis

3.1.1 Dependent Rent Variable Analysis

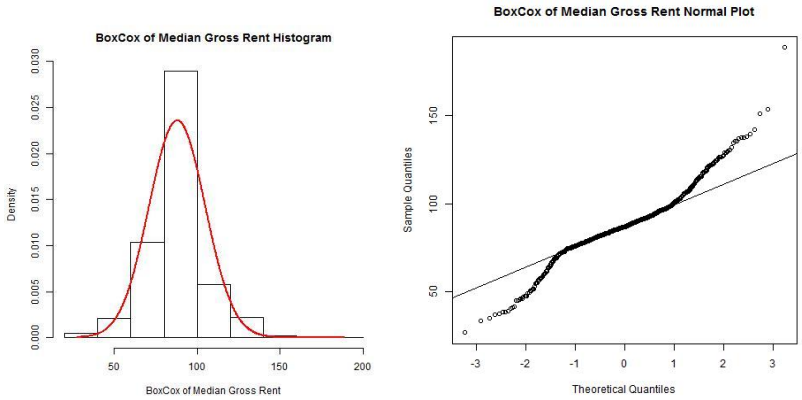
3.1.1.1 Rent Histogram and Normal Plot Analysis

Histogram and Normal Plot of Rent Dependent Variable Before Transforms:



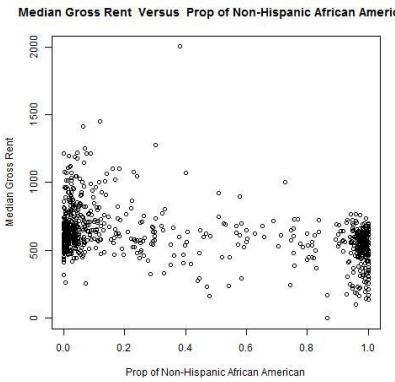
3.1.1.2 Rent Transformation Information

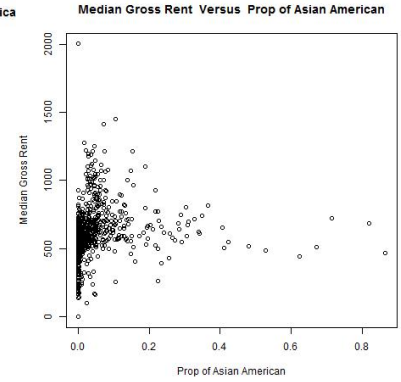
Histogram and Normal Plot of Rent Dependent Variable After Box Cox Transform:

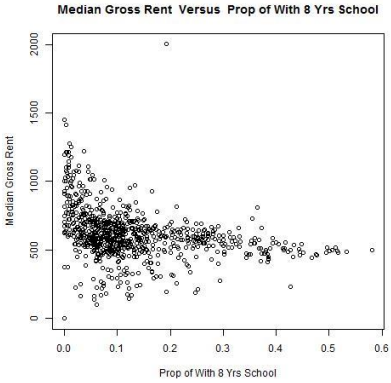


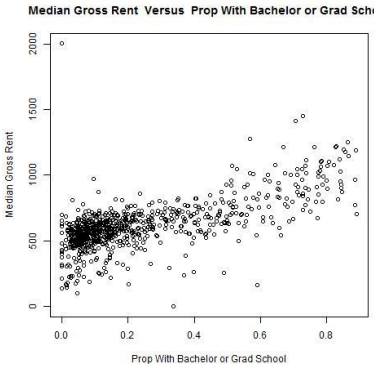
3.1.1.3 Rent and Independent Variable Associations

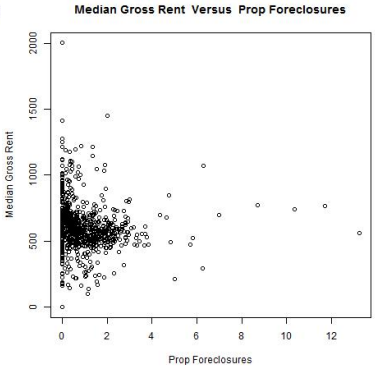
Plots of Dependent Variabe Rent Versus Non-Dummy Independent Variables:

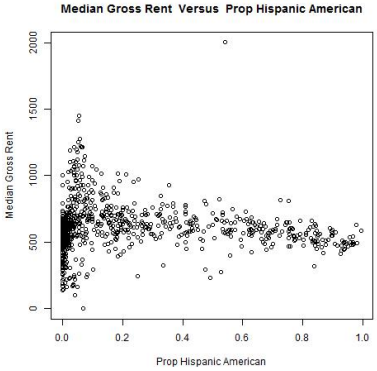

Median Gross Rent Versus Prop of Non-Hispanic African America
Median Gross Rent
Prop of Non-Hispanic African American

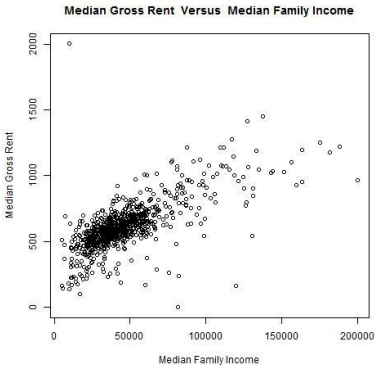

Median Gross Rent Versus Prop of Asian American
Median Gross Rent
Prop of Asian American

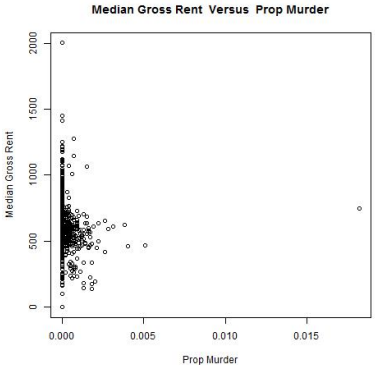

Median Gross Rent Versus Prop of With 8 Yrs School
Median Gross Rent
Prop of With 8 Yrs School

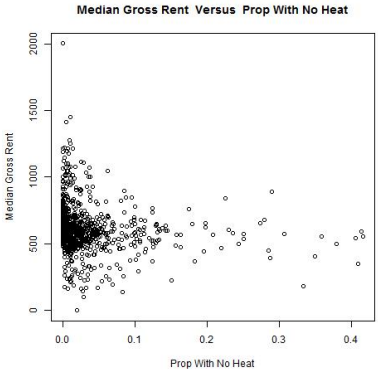

Median Gross Rent Versus Prop With Bachelor or Grad School
Median Gross Rent
Prop With Bachelor or Grad School

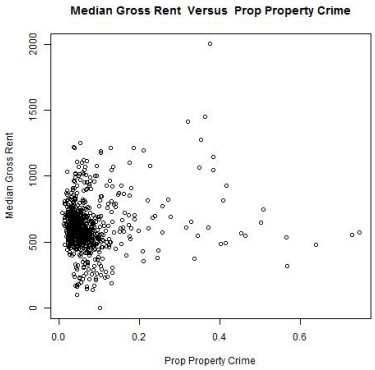

Median Gross Rent Versus Prop Foreclosures
Median Gross Rent
Prop Foreclosures

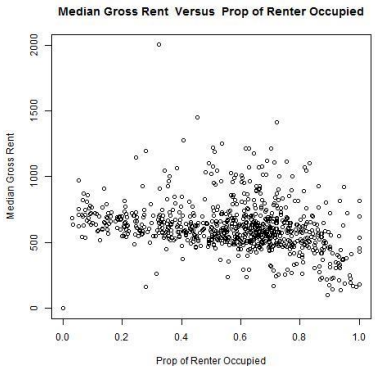

Median Gross Rent Versus Prop Hispanic American
Median Gross Rent
Prop Hispanic American

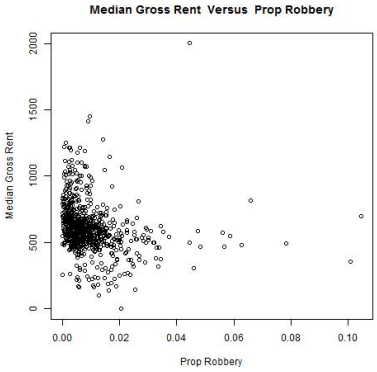

Median Gross Rent Versus Median Family Income
Median Gross Rent
Median Family Income

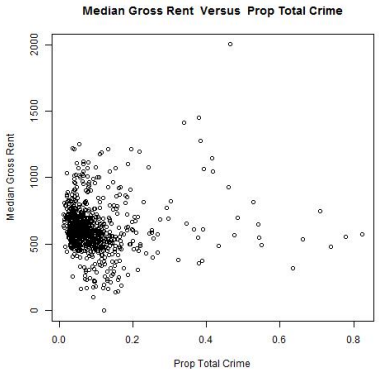

Median Gross Rent Versus Prop Murder
Median Gross Rent
Prop Murder

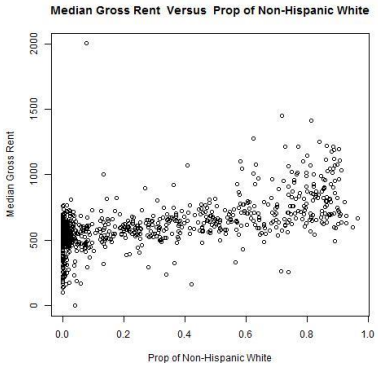

Median Gross Rent Versus Prop With No Heat
Median Gross Rent
Prop With No Heat


Median Gross Rent Versus Prop Property Crime
Median Gross Rent
Prop Property Crime

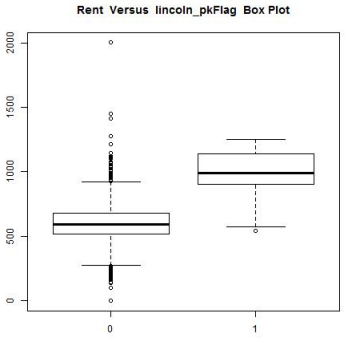

Median Gross Rent Versus Prop of Renter Occupied
Median Gross Rent
Prop of Renter Occupied

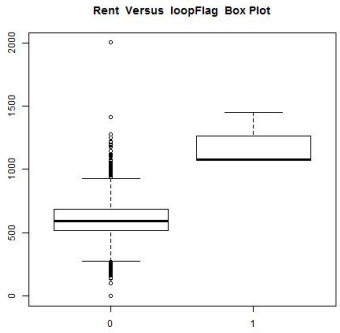

Median Gross Rent Versus Prop Robbery
Median Gross Rent
Prop Robbery

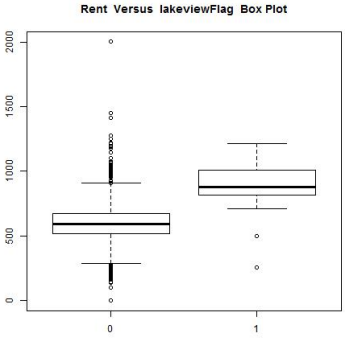


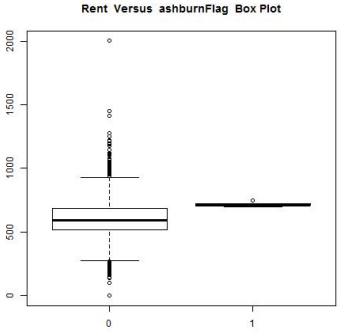


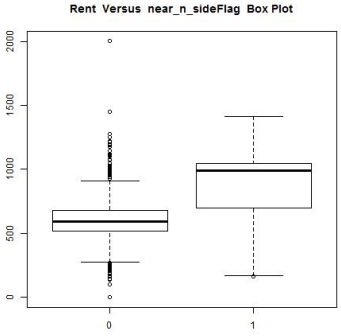
Box Plots of Dependent Variabe Rent Versus Dummy Neighborhood Independent Variables with High Rent Values:

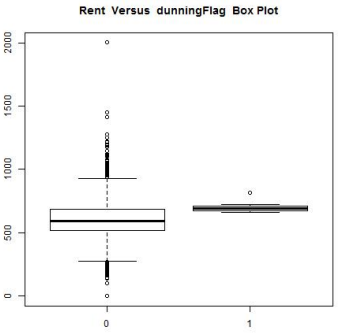


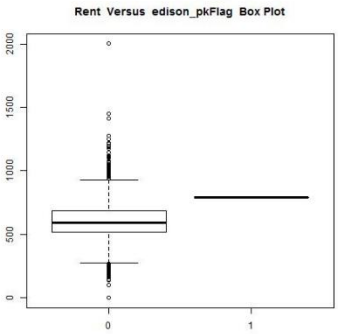


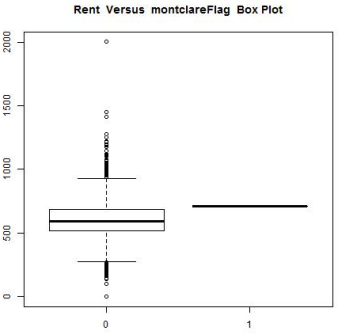


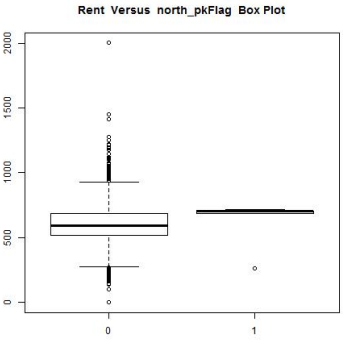


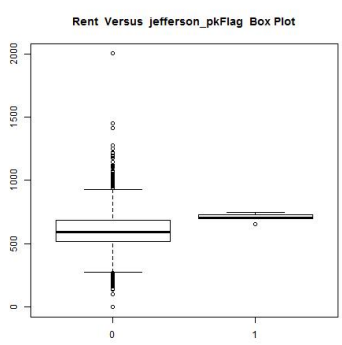


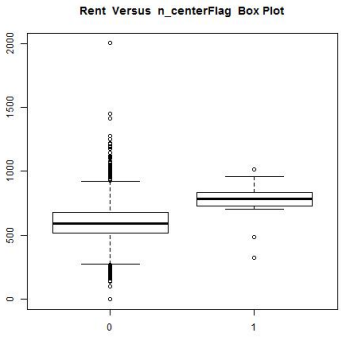


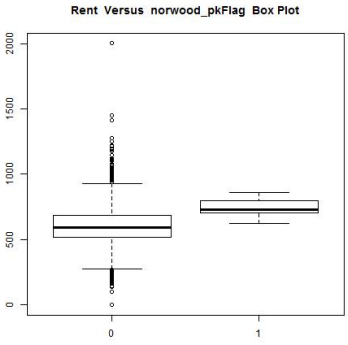


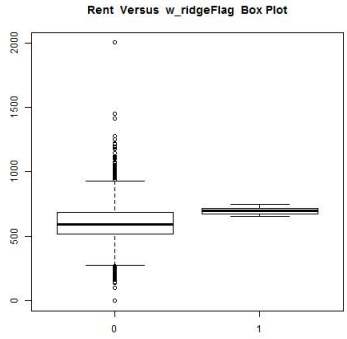




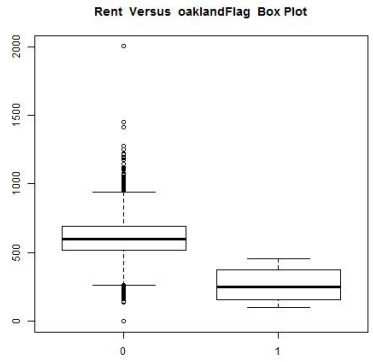


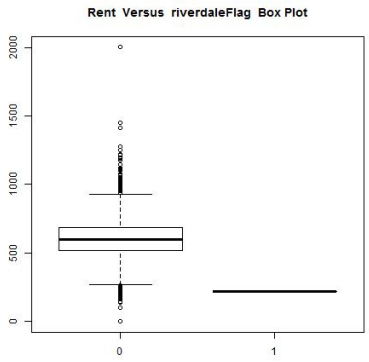


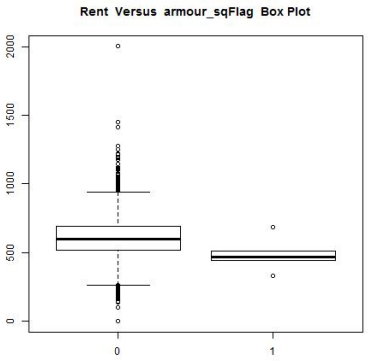


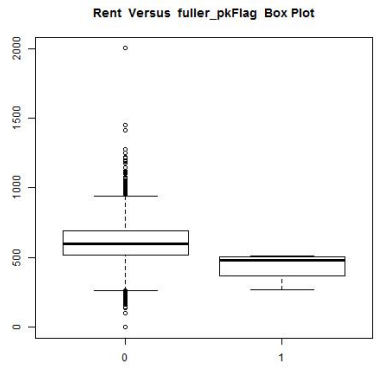


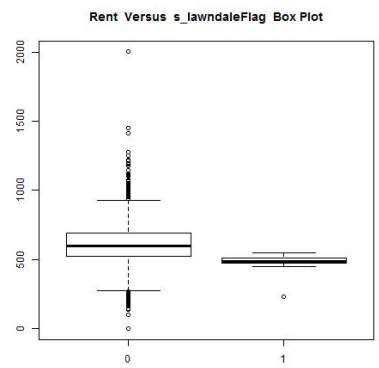
Box Plots of Dependent Variabe Rent Versus Dummy Neighborhood Independent Variables with Low Rent Values:

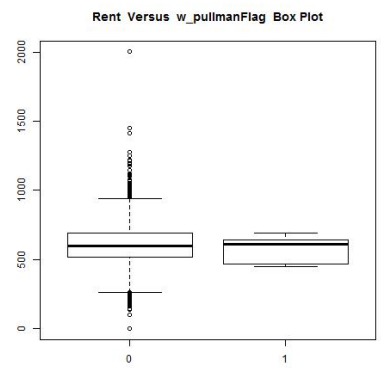


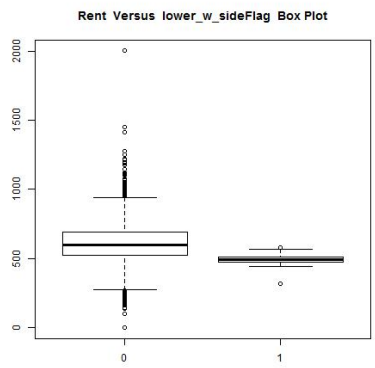


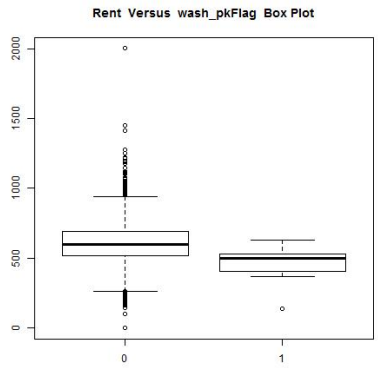


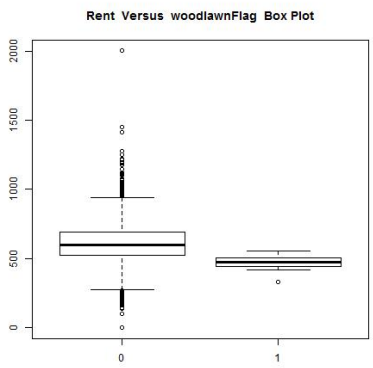






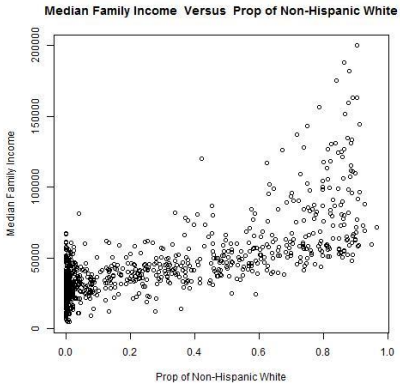


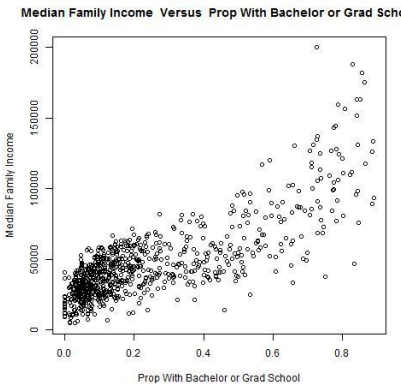


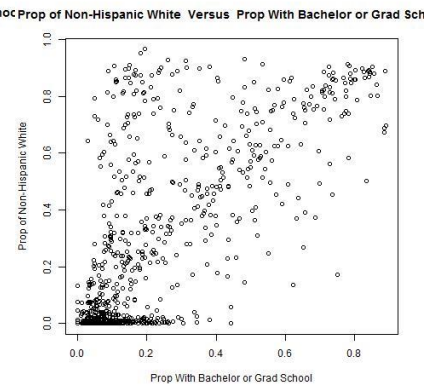


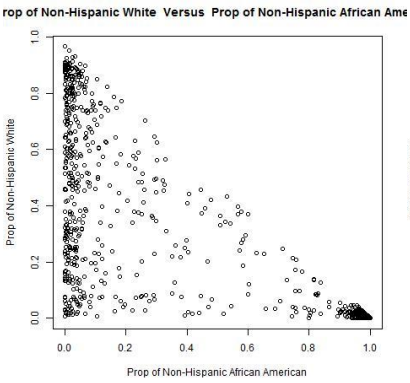
3.1.2 Independent Multicollinearity Analysis

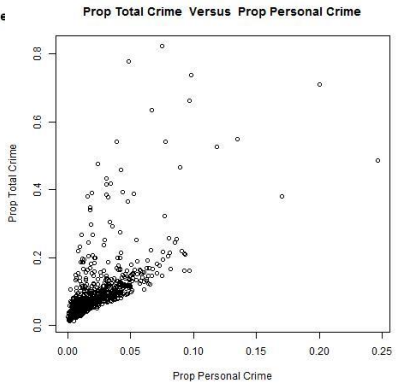
Scatterplots of Independent Variables with Possible Multicollinearity Issues

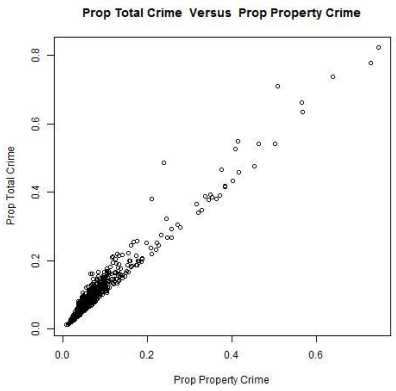


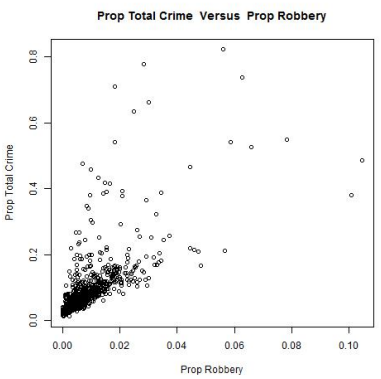


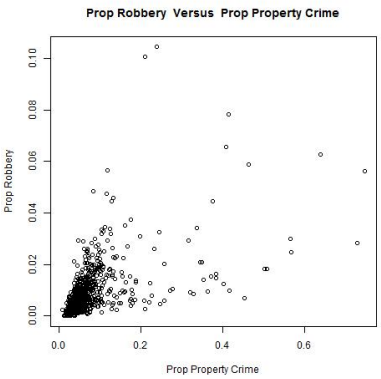


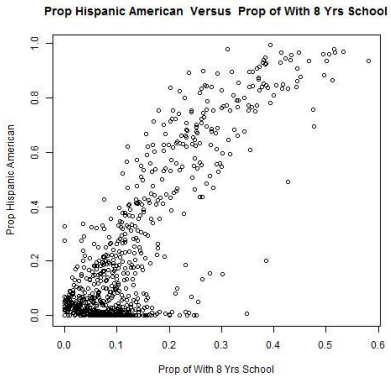


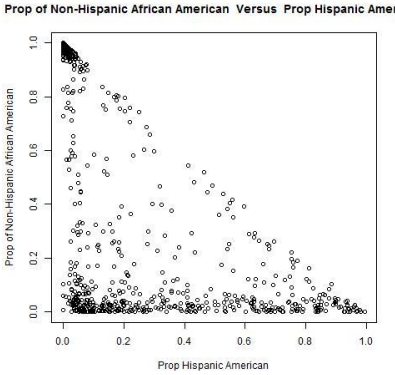








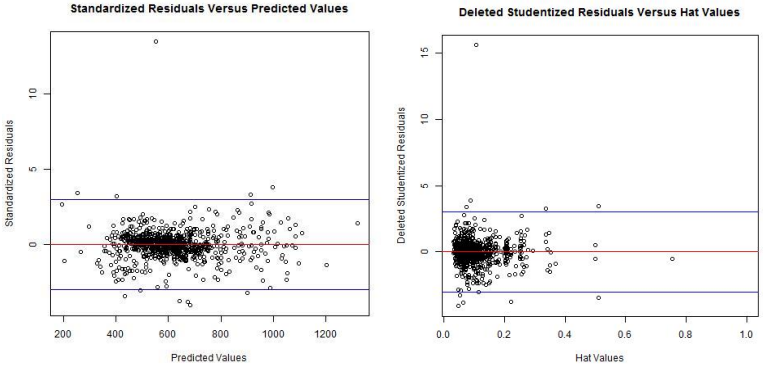




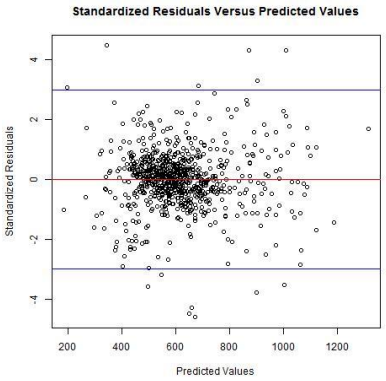
3.2 Model Analysis

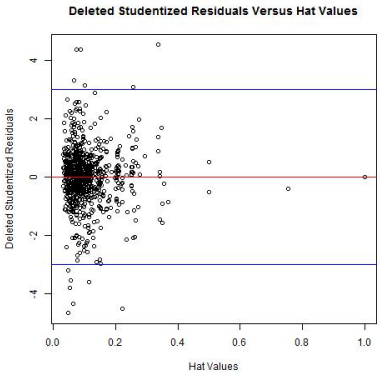
3.2.1 Model Interpretation

Standardized Residuals Versus Predicted Values and Deleted Studentized Residuals Versus Hat Values with Influential and Outlier Points Present:



Standardized Residuals Versus Predicted Values and Deleted Studentized Residuals Versus Hat Values without Influential and Outlier Points Present:





R Model Anaysis Output:

[1] Model Formula:
[1]
bcRent ~ mdfamy0 + whitenhp + aanhp + renter + educ8 + educ160pro +
  forecl + uptownFlag + lincoln_pkFlag + forest_glenFlag +
  north_pkFlag + near_w_sideFlag + armour_sqFlag + douglasFlag +
  oaklandFlag + gd_blvdFlag + kenwoodFlag + hyde_pkFlag + avalon_pkFlag +
  riverdaleFlag + bridgeportFlag + beverlyFlag
<environment: 0x000000001ab37b58>
[1]
[1] Model Summary:
[1]

Call:
lm(formula = modelForm, data = inData)

Residuals:
  Min 1Q Median 3Q Max
-42.41 -4.22 0.33 4.82 33.97

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	8.80e+01	3.14e+00	28.06	< 2e-16	***
mdfamy0	1.99e-04	3.07e-05	6.49	1.5e-10	***
whitenhp	-9.03e+00	3.15e+00	-2.87	0.00425	**
aanhp	-1.01e+01	2.36e+00	-4.28	2.1e-05	***
renter	-1.02e+01	2.19e+00	-4.64	4.1e-06	***
educ8	-2.48e+01	7.03e+00	-3.53	0.00043	***
educ160pro	3.32e+01	3.78e+00	8.78	< 2e-16	***
forecl	1.15e+02	3.16e+01	3.64	0.00029	***
uptownFlag	-8.26e+00	2.85e+00	-2.90	0.00388	**
lincoln_pkFlag	-8.33e+00	2.77e+00	-3.00	0.00277	**
forest_glenFlag	-9.92e+00	4.88e+00	-2.03	0.04231	*
north_pkFlag	-1.09e+01	4.34e+00	-2.52	0.01199	*
near_w_sideFlag	-9.33e+00	1.73e+00	-5.39	9.4e-08	***
armour_sqFlag	-1.14e+01	4.34e+00	-2.62	0.00905	**
douglasFlag	-1.24e+01	3.11e+00	-3.99	7.2e-05	***
oaklandFlag	-2.72e+01	4.87e+00	-5.58	3.3e-08	***
gd_blvdFlag	-7.16e+00	2.54e+00	-2.83	0.00483	**
kenwoodFlag	-7.39e+00	3.70e+00	-1.99	0.04649	*
hyde_pkFlag	-1.14e+01	2.80e+00	-4.08	4.9e-05	***
avalon_pkFlag	-1.41e+01	5.61e+00	-2.52	0.01186	*
riverdaleFlag	-2.67e+01	9.62e+00	-2.77	0.00568	**
bridgeportFlag	-7.91e+00	2.64e+00	-3.00	0.00282	**
beverlyFlag	-1.39e+01	3.75e+00	-3.70	0.00023	***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

```
Residual standard error: 9.58 on 787 degrees of freedom
Multiple R-squared:  0.672, Adjusted R-squared:  0.662
F-statistic: 73.2 on 22 and 787 DF,  p-value: <2e-16

[1]
[1] Model ANOVA Analysis:
[1]
Analysis of Variance Table

Response: bcRent
               Df Sum Sq Mean Sq F value  Pr(>F)
mdfamy0         1 116864  116864 1274.55 < 2e-16 ***
whitenhp        1   4176    4176   45.55 2.9e-11 ***
aanhp           1    181     181    1.98 0.16013
renter          1    333     333    3.64 0.05689 .
educ8           1   5785    5785   63.10 6.8e-15 ***
educ160pro      1   5641    5641   61.52 1.4e-14 ***
forecl          1   1590    1590   17.34 3.5e-05 ***
uptownFlag      1    365     365    3.98 0.04637 *
lincoln_pkFlag  1    290     290    3.17 0.07554 .
forest_glenFlag 1    328     328    3.58 0.05897 .
north_pkFlag    1    448     448    4.88 0.02744 *
near_w_sideFlag 1   1680    1680   18.32 2.1e-05 ***
armour_sqFlag   1    527     527    5.75 0.01673 *
douglasFlag     1   1039    1039   11.33 0.00080 ***
oaklandFlag     1   2700    2700   29.44 7.7e-08 ***
gd_blvdFlag     1    692     692    7.55 0.00615 **
kenwoodFlag     1    241     241    2.63 0.10511
hyde_pkFlag     1   1368    1368   14.92 0.00012 ***
avalon_pkFlag   1    553     553    6.03 0.01428 *
riverdaleFlag   1    717     717    7.82 0.00529 **
bridgeportFlag  1    827     827    9.02 0.00275 **
beverlyFlag     1   1256    1256   13.69 0.00023 ***
Residuals     787  72160      92
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1]
[1]
[1] Standardized Coefficients:
[1]
     mdfamy0      whitenhp         aanhp        renter         educ8    educ160pro        forecl     uptownFlag  lincoln_pkFlag
      0.3257       -0.1736       -0.2671       -0.1372       -0.1659        0.4503        0.0846       -0.0606       -0.0784
forest_glenFlag   north_pkFlag near_w_sideFlag   armour_sqFlag   douglasFlag    oaklandFlag    gd_blvdFlag    kenwoodFlag
hyde_pkFlag
     -0.0422       -0.0520       -0.1136       -0.0540       -0.0831       -0.1156       -0.0605       -0.0415       -0.0905
 avalon_pkFlag   riverdaleFlag  bridgeportFlag     beverlyFlag
     -0.0522       -0.0569       -0.0626       -0.0780
[1]
[1] 95% Confidence Intervals:
[1]
                  2.5 %    97.5 %
(Intercept)     8.19e+01  9.42e+01
mdfamy0         1.39e-04  2.59e-04
whitenhp       -1.52e+01 -2.85e+00
aanhp          -1.47e+01 -5.47e+00
renter         -1.45e+01 -5.86e+00
educ8          -3.86e+01 -1.10e+01
educ160pro      2.58e+01  4.06e+01
forecl          5.31e+01  1.77e+02
uptownFlag     -1.39e+01 -2.66e+00
lincoln_pkFlag -1.38e+01 -2.88e+00
forest_glenFlag -1.95e+01 -3.45e-01
north_pkFlag    -1.94e+01 -2.41e+00
near_w_sideFlag -1.27e+01 -5.93e+00
armour_sqFlag   -1.99e+01 -2.84e+00
```

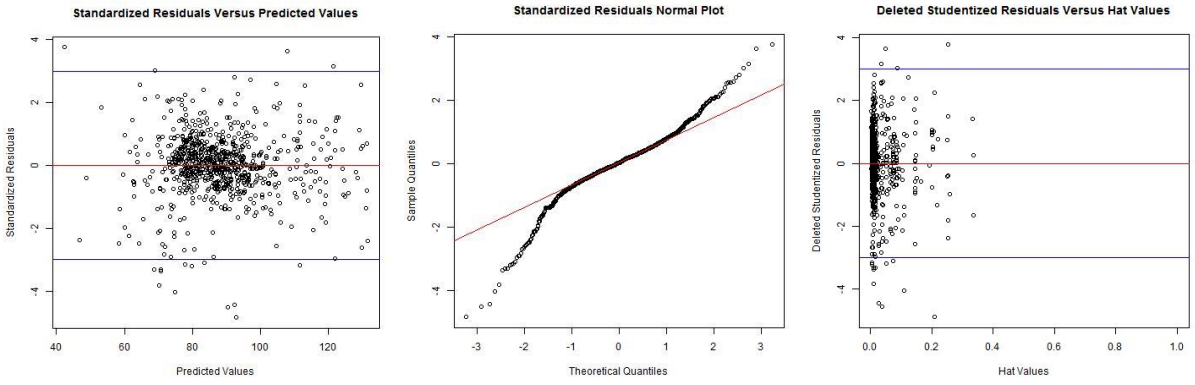
```
douglasFlag    -1.85e+01 -6.30e+00
oaklandFlag    -3.67e+01 -1.76e+01
gd_blvdFlag    -1.21e+01 -2.19e+00
kenwoodFlag    -1.47e+01 -1.15e-01
hyde_pkFlag    -1.69e+01 -5.94e+00
avalon_pkFlag  -2.52e+01 -3.14e+00
riverdaleFlag  -4.56e+01 -7.79e+00
bridgeportFlag -1.31e+01 -2.73e+00
beverlyFlag    -2.12e+01 -6.51e+00
[1]
[1]
[1] Model VIF Values:
[1]
      mdfamy0      whitenhp         aanhp         renter         educ8     educ160pro         forecl     uptownFlag  lincoln_pkFlag
         6.03          8.78          9.34          2.10          5.28          6.31          1.29          1.05          1.64
forest_glenFlag   north_pkFlag near_w_sideFlag   armour_sqFlag   douglasFlag    oaklandFlag    gd_blvdFlag    kenwoodFlag
hyde_pkFlag
         1.03          1.02          1.07          1.02          1.04          1.03          1.10          1.04          1.18
 avalon_pkFlag  riverdaleFlag bridgeportFlag    beverlyFlag
         1.03          1.01          1.05          1.06
[1]
[1]
[1] Potential Influential Points:
Potentially influential observations of
        lm(formula = modelForm, data = inData) :

    dfb.1_ dfb.mdf0 dfb.whtn dfb.anhp dfb.rntr dfb.edc8 dfb.e160 dfb.frcl dfb.uptF dfb.ln_F dfb.fr_F dfb.nr_F dfb.n__F dfb.ar_F dfb.dglF
28   0.01   0.00   -0.01   -0.01   0.00   -0.01   0.00   0.00   0.09   0.00   0.00   0.00   0.00   0.00   0.00
29   0.00   0.01   0.00   0.00   0.01   0.00   -0.01   -0.01   0.17   -0.01   0.00   0.00   0.00   0.00   0.00
30   0.01   0.00   0.00   0.00   -0.01   0.00   0.01   0.00   -0.05   0.00   0.00   0.00   0.00   0.00   0.00
31   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
32   0.00   0.00   0.01   0.01   -0.02   0.01   0.02   0.02   0.13   -0.01   -0.01   0.00   0.00   0.00   0.00
33   0.01   0.00   -0.01   0.00   -0.01   0.00   0.01   -0.01   -0.14   -0.01   0.00   0.00   0.00   0.00   0.00
34   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.03   0.00   0.00   0.00   0.00   0.00   0.00
35   0.00   0.01   0.00   0.00   0.00   0.00   -0.01   0.00   -0.10   0.00   0.00   0.00   0.00   0.00   0.00
36   0.00   -0.03   0.01   0.01   -0.01   0.01   0.02   0.01   -0.20   0.01   0.00   0.00   0.00   0.00   0.00
37   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.05   0.00   0.00   0.00   0.00   0.00   0.00
38   -0.01   -0.02   0.03   0.02   -0.01   0.02   0.01   0.01   -0.14   0.01   0.00   0.00   0.00   0.00   0.00
39   0.00   -0.02   0.02   0.01   0.00   0.01   0.02   0.00   0.15   0.01   0.00   0.00   0.00   0.00   0.00
63   -0.07   0.13   0.00   0.02   0.01   0.03   -0.08   0.06   0.05   -0.04   0.01   0.02   0.03   0.02   0.02
69   0.37   0.37   -0.54   -0.50   0.10   -0.55   -0.43   0.01   0.10   0.01   0.04   -0.02   0.10   -0.01   0.03
102   0.04   -0.05   -0.04   -0.01   -0.08   0.00   0.10   0.03   -0.01   -0.57   -0.01   0.00   -0.01   0.00   -0.01
105   -0.02   0.03   -0.01   0.00   0.01   0.00   -0.01   0.00   0.00   -0.05   0.00   0.00   0.00   0.00   0.00
106   -0.02   0.05   -0.01   0.00   0.02   0.00   -0.02   0.00   0.00   -0.07   0.00   0.00   0.00   0.00   0.00
109   -0.03   0.07   -0.01   0.00   0.04   -0.01   -0.04   -0.01   0.01   0.06   0.00   0.00   0.00   0.00   0.00
114   -0.04   0.10   -0.01   0.00   0.05   0.00   -0.05   0.00   0.01   0.07   0.00   0.00   0.00   0.00   0.00
120   0.16   -0.20   -0.03   -0.06   -0.03   -0.07   -0.05   -0.19   0.01   0.22   0.04   0.01   0.05   -0.01   0.03
123   0.07   -0.02   -0.06   -0.08   -0.17   0.00   0.08   0.16   0.03   0.00   -0.02   0.00   0.06   0.01   0.04
126   -0.09   0.00   0.05   0.08   -0.24   0.21   0.16   0.16   0.04   -0.05   -0.03   0.02   0.05   0.03   0.04
127   -0.05   -0.08   0.03   0.06   0.06   0.04   0.14   -0.07   -0.05   -0.03   0.00   -0.01   -0.06   -0.02   -0.04
131   0.14   -0.24   0.00   0.01   -0.07   -0.02   0.03   -0.31   0.00   0.21   0.03   0.01   0.04   -0.02   0.02
133   -0.25   0.48   -0.04   0.01   0.34   -0.05   -0.18   -0.10   -0.02   -0.36   -0.02   -0.03   -0.08   0.00   -0.04
134   -0.05   0.05   0.01   0.02   0.06   0.00   0.07   -0.05   -0.04   -0.12   -0.02   -0.02   -0.05   -0.01   -0.03
137   0.03   -0.03   -0.03   0.01   -0.25   0.11   0.15   -0.01   0.03   -0.02   -0.03   0.01   0.05   0.01   0.05
153   0.05   -0.02   -0.03   -0.06   0.03   -0.06   -0.04   0.00   0.00   0.03   -1.03_*   0.00   0.01   -0.01   0.00
154   0.00   -0.02   0.01   0.01   0.00   0.00   0.00   0.01   0.00   0.01   0.42   0.00   0.00   0.00   0.00
155   0.00   0.00   0.01   0.00   0.01   0.00   -0.01   0.00   0.00   0.00   -0.24   0.00   0.00   0.00   0.00
156   0.03   0.03   -0.03   -0.05   0.05   -0.06   -0.07   0.00   0.00   0.01   0.87   0.00   0.01   0.00   0.00
157   0.34   -0.03   -0.34   -0.39   0.15   -0.42   -0.12   -0.03   -0.03   0.10   0.04   -2.47_*   0.01   -0.04   -0.02
158   0.00   -0.01   0.00   0.01   -0.02   0.01   0.01   0.00   0.00   0.00   0.00   0.26   0.00   0.00   0.00
159   0.02   0.01   -0.03   -0.03   0.01   -0.02   -0.01   0.00   0.00   0.00   0.00   0.53   0.00   0.00   0.00
160   0.08   0.05   -0.10   -0.15   0.18   -0.19   -0.14   -0.03   -0.01   0.02   0.03   1.10_*   0.00   -0.02   -0.01
161   0.01   -0.03   0.00   0.00   -0.01   0.01   0.03   0.01   -0.01   0.01   0.00   0.48   0.00   0.00   0.00
288   0.03   0.03   -0.02   -0.09   0.15   -0.08   -0.23   0.05   0.04   0.13   0.05   0.02   0.06   0.00   0.02
    dfb.oklF dfb.gd_F dfb.knwF dfb.hy_F dfb.av_F dfb.rvrF dfb.brdF dfb.bvrF dffit   cov.r   cook.d hat
28   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.10   1.12_*   0.00   0.08
```

29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18	1.11_*	0.00	0.08
30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.05	1.13_*	0.00	0.09_*
31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.12_*	0.00	0.08
32	0.00	0.00	0.00	-0.01	0.00	0.00	0.00	-0.01	0.14	1.13_*	0.00	0.09_*
33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.14	1.12_*	0.00	0.09_*
34	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	1.12_*	0.00	0.08
35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.10	1.12_*	0.00	0.08
36	0.00	0.00	0.00	-0.01	0.00	0.00	0.00	0.00	-0.21	1.11_*	0.00	0.09_*
37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	1.12_*	0.00	0.08
38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.15	1.12_*	0.00	0.09_*
39	0.00	0.00	0.00	-0.01	0.00	0.00	0.00	0.00	0.16	1.12_*	0.00	0.09_*
63	0.01	0.01	0.02	0.05	0.00	0.01	0.02	0.01	-0.21	0.81_*	0.00	0.01
69	0.00	0.02	0.05	0.12	-0.04	-0.01	-0.02	0.06	-0.74_*	0.59_*	0.02	0.03
102	0.01	0.01	-0.01	-0.04	-0.01	0.01	0.00	-0.02	-0.74_*	0.81_*	0.02	0.05
105	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.05	1.13_*	0.00	0.09_*
106	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.08	1.15_*	0.00	0.10_*
109	0.00	-0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.14	1.10_*	0.00	0.07
114	0.00	-0.01	0.00	0.01	-0.01	0.00	0.00	0.00	0.19	1.09_*	0.00	0.07
120	-0.01	0.02	0.03	0.07	0.02	-0.02	0.01	0.06	-0.44	0.86_*	0.01	0.03
123	0.05	0.07	0.03	-0.01	0.00	0.03	0.00	-0.03	-0.33	0.75_*	0.00	0.01
126	0.06	0.08	0.02	-0.01	0.00	0.04	0.02	-0.05	-0.44	0.76_*	0.01	0.02
127	-0.02	-0.03	-0.04	-0.08	0.00	-0.01	0.01	-0.01	0.25	0.90_*	0.00	0.01
131	-0.01	0.04	0.01	0.04	0.01	-0.02	0.03	0.04	-0.49	0.91_*	0.01	0.04
133	-0.05	-0.05	-0.03	-0.02	-0.02	0.00	-0.01	-0.02	0.59_*	0.80_*	0.02	0.03
134	-0.01	-0.01	-0.03	-0.07	-0.01	0.00	0.00	-0.04	0.26	0.86_*	0.00	0.01
137	0.05	0.10	0.02	-0.01	0.00	0.03	0.02	-0.05	-0.36	0.75_*	0.01	0.01
153	0.00	0.00	0.01	0.01	0.00	0.00	-0.01	0.02	-1.06_*	1.25_*	0.05	0.25_*
154	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.43	1.35_*	0.01	0.25_*
155	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.25	1.37_*	0.00	0.25_*
156	0.00	-0.01	0.01	0.02	0.00	0.00	-0.01	0.02	0.88_*	1.29_*	0.03	0.25_*
157	-0.02	-0.02	0.00	-0.01	0.00	-0.02	-0.04	0.06	-2.51_*	0.65_*	0.27	0.21_*
158	0.00	0.01	0.00	0.00	0.00	0.00	0.00	-0.01	0.27	1.28_*	0.00	0.20_*
159	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.54_*	1.25_*	0.01	0.20_*
160	-0.02	-0.03	0.00	0.02	0.01	-0.02	-0.02	0.04	1.16_*	1.12_*	0.06	0.21_*
161	0.00	0.00	0.00	-0.01	0.00	0.00	0.00	0.00	0.49	1.25_*	0.01	0.20_*
288	-0.01	-0.03	0.05	0.12	0.02	-0.01	-0.01	0.09	-0.38	0.81_*	0.01	0.02

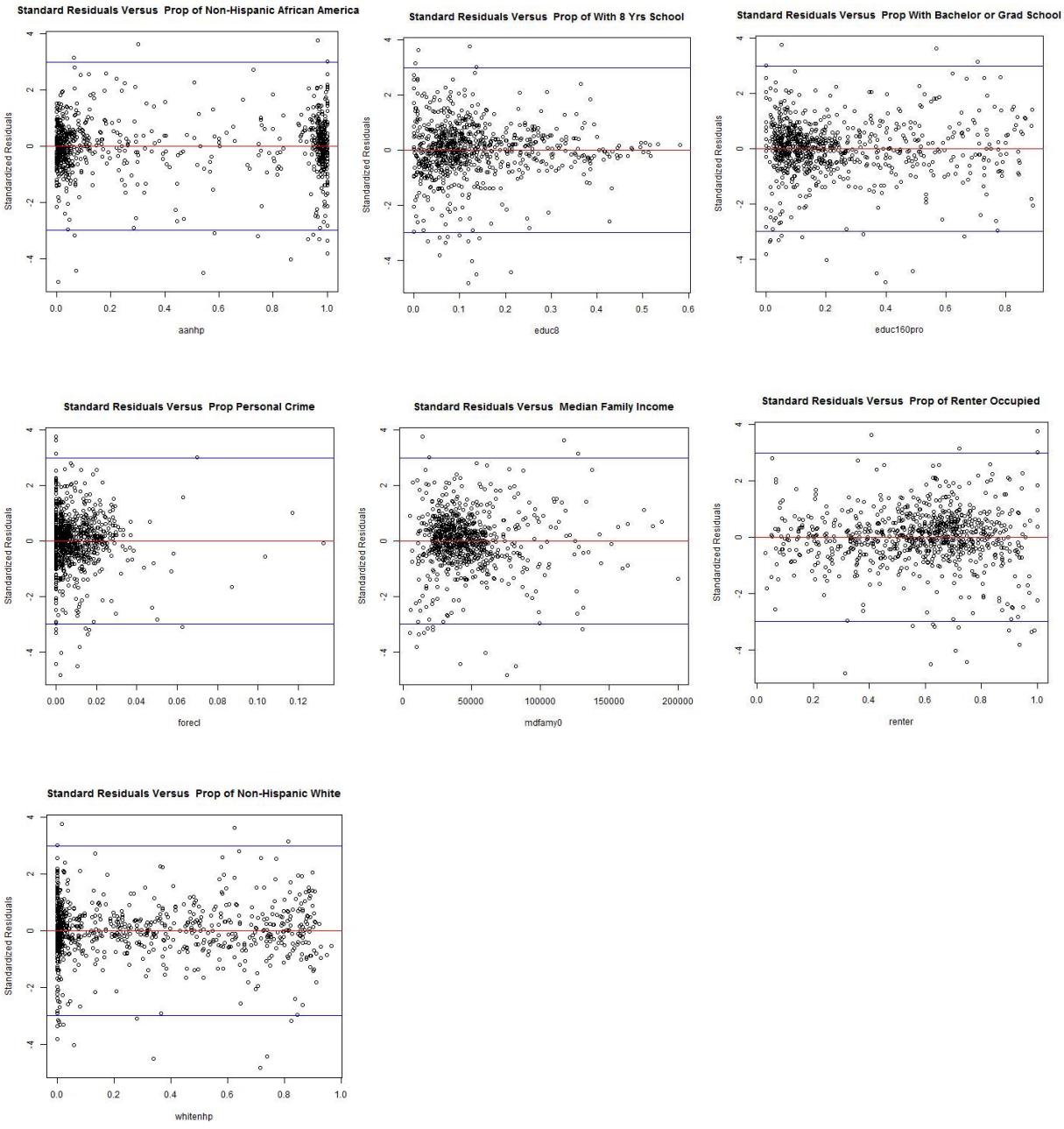
[reached getOption("max.print") -- omitted 101 rows]

3.2.2 Residual Analysis

Standardized Residual Versus Predicted Values, Normal Plot, and Deleted Studentized Residual Versus Hat Value Plots:

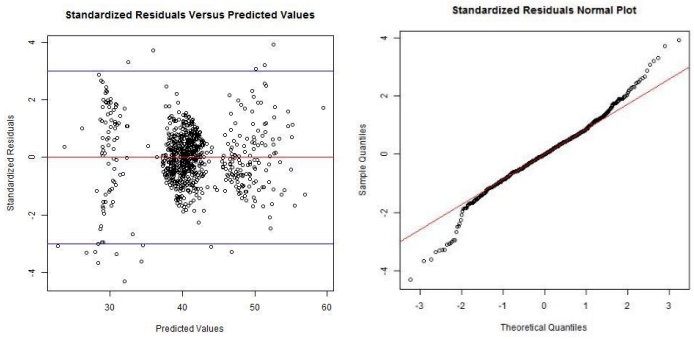


Standardized Residual Versus Each Independent Variable Plots:

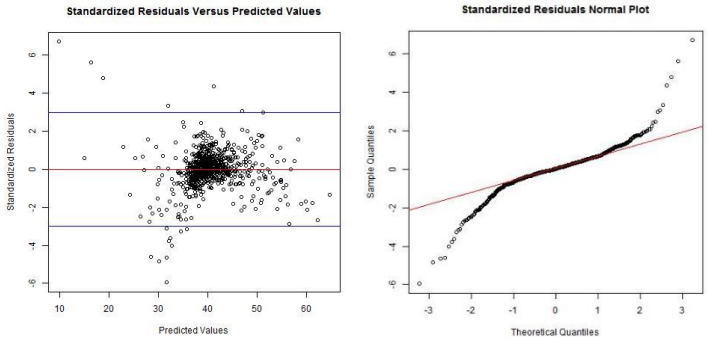


3.2.3 Residual Normalization

Residual plots of model with the high rent and low rent dummy variables:



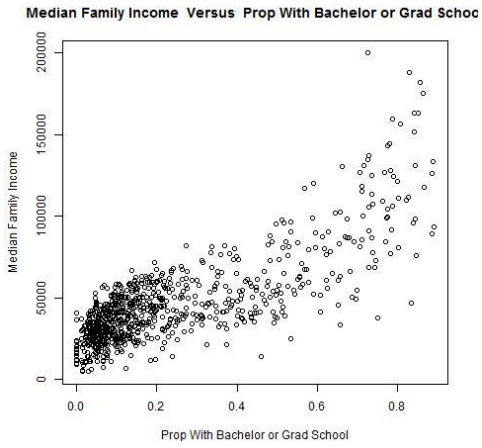
Residual plots of model with rent affordability index variable:



3.2.4 Model Validation

3.2.5 Model Predictions

Median Family Income Versus Proportion of People Over 25 With Completed Bachelor or Graduate Degree Plot:



3.2.6 Modeling Process

R Model Selection Output For Box Cox Rent After Outlier Removal:

[1] Too many variables for Cp and Adjusted R-Squared Model Selection
[1]
[1]
[1] Output of Backward Model Selection:
[1]

Call:
lm(formula = bcRent ~ mdfamy0 + asianp + whitenhp + aanhp + hisp +
    renter + educ8 + educ160pro + Noheat + totcrime + personal +
    property + forecl + w_ridgeFlag + edgewaterFlag + uptownFlag +
    lincoln_sqFlag + n_centerFlag + lakeviewFlag + lincoln_pkFlag +
    near_n_sideFlag + forest_glenFlag + north_pkFlag + albany_pkFlag +
    portage_pkFlag + irving_pkFlag + w_townFlag + near_w_sideFlag +
    s_lawndaleFlag + lower_w_sideFlag + armour_sqFlag + douglasFlag +
    oaklandFlag + fuller_pkFlag + gd_blvdFlag + kenwoodFlag +
    hyde_pkFlag + woodlawnFlag + chathamFlag + avalon_pkFlag +
    s_chgoFlag + calumet_htsFlag + pullmanFlag + s_deeringFlag +
    e_sideFlag + riverdaleFlag + hegewischFlag + mckinley_pkFlag +
    bridgeportFlag + new_cityFlag + grtr_grnd_crsngFlag + beverlyFlag +
    mt_greenwoodFlag + morgan_pkFlag, data = inData)

Coefficients:
(Intercept)	mdfamy0	asianp	whitenhp	aanhp	hisp
1.75e+02	2.03e-04	-7.78e+01	-9.54e+01	-9.67e+01	-8.75e+01
renter	educ8	educ160pro	Noheat	totcrime	personal
-6.11e+00	-2.33e+01	3.77e+01	-1.35e+01	1.46e+04	-1.46e+04
property	forecl	w_ridgeFlag	edgewaterFlag	uptownFlag	lincoln_sqFlag
-1.46e+04	1.06e+02	-5.79e+00	-9.26e+00	-1.32e+01	-9.38e+00
n_centerFlag	lakeviewFlag	lincoln_pkFlag	near_n_sideFlag	forest_glenFlag	north_pkFlag
-6.25e+00	-7.70e+00	-1.54e+01	-9.49e+00	-1.49e+01	-1.73e+01
albany_pkFlag	portage_pkFlag	irving_pkFlag	w_townFlag	near_w_sideFlag	s_lawndaleFlag
-4.89e+00	-4.92e+00	-5.90e+00	-3.34e+00	-1.40e+01	-4.44e+00
lower_w_sideFlag	armour_sqFlag	douglasFlag	oaklandFlag	fuller_pkFlag	gd_blvdFlag
-8.19e+00	-2.09e+01	-1.67e+01	-3.00e+01	-8.87e+00	-8.47e+00
kenwoodFlag	hyde_pkFlag	woodlawnFlag	chathamFlag	avalon_pkFlag	s_chgoFlag
-1.07e+01	-1.76e+01	-6.52e+00	-4.71e+00	-1.60e+01	-6.42e+00
calumet_htsFlag	pullmanFlag	s_deeringFlag	e_sideFlag	riverdaleFlag	hegewischFlag
-8.27e+00	-8.49e+00	-6.75e+00	-8.91e+00	-3.00e+01	-1.24e+01
mckinley_pkFlag	bridgeportFlag	new_cityFlag	grtr_grnd_crsngFlag	beverlyFlag	mt_greenwoodFlag
-5.71e+00	-1.38e+01	-4.35e+00	-5.30e+00	-1.74e+01	-7.39e+00
morgan_pkFlag					
-8.70e+00					

[1]
[1]
[1] Output of Stepwise Model Selection:
[1]

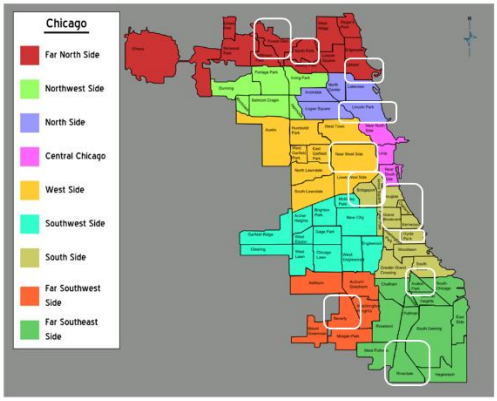
Call:
lm(formula = bcRent ~ mdfamy0 + educ160pro + renter + oaklandFlag +
    near_w_sideFlag + loopFlag + logan_sqFlag + beverlyFlag +
    douglasFlag + hyde_pkFlag + austinFlag + humboldt_pkFlag +
    w_englewoodFlag + lincoln_pkFlag + riverdaleFlag + avalon_pkFlag +
    uptownFlag + w_garfield_pkFlag + forest_glenFlag + aanhp +
    forecl + educ8 + hisp + north_pkFlag + bridgeportFlag + roselandFlag +
    armour_sqFlag + whitenhp + n_lawndaleFlag + archer_htsFlag +
    w_townFlag + s_shoreFlag + asianp + e_garfield_pkFlag + hegewischFlag +
    englewoodFlag + e_sideFlag + ashburn_greshFlag, data = inData)

Coefficients:
(Intercept)	mdfamy0	educ160pro	renter	oaklandFlag	near_w_sideFlag	loopFlag
1.80e+02	1.96e-04	3.13e+01	-1.25e+01	-2.21e+01	-6.21e+00	1.87e+01
logan_sqFlag	beverlyFlag	douglasFlag	hyde_pkFlag	austinFlag	humboldt_pkFlag	w_englewoodFlag
6.03e+00	-1.26e+01	-8.03e+00	-9.26e+00	8.88e+00	8.31e+00	9.50e+00

```
  lincoln_pkFlag     riverdaleFlag     avalon_pkFlag       uptownFlag  w_garfield_pkFlag    forest_glenFlag           aanhp
       -6.49e+00         -2.26e+01         -1.04e+01         -7.03e+00          1.05e+01         -1.05e+01         -1.04e+02
           forecl            educ8             hisp      north_pkFlag    bridgeportFlag     roselandFlag     armour_sqFlag
        7.94e+01         -2.50e+01         -9.12e+01         -1.22e+01         -8.80e+00          6.03e+00         -1.43e+01
       whitenhp    n_lawndaleFlag     archer_htsFlag        w_townFlag       s_shoreFlag           asianp  e_garfield_pkFlag
       -9.93e+01          5.84e+00          6.77e+00          2.97e+00          6.09e+00         -8.35e+01          5.21e+00
  hegewischFlag     englewoodFlag        e_sideFlag  ashburn_greshFlag
       -1.11e+01          5.03e+00         -7.02e+00          3.81e+00

[1]
[1]
[1] Output of Forward Model Selection:
[1]

Call:
lm(formula = bcRent ~ mdfamy0 + educ160pro + renter + oaklandFlag +
    near_w_sideFlag + loopFlag + logan_sqFlag + beverlyFlag +
    douglasFlag + hyde_pkFlag + austinFlag + humboldt_pkFlag +
    w_englewoodFlag + lincoln_pkFlag + riverdaleFlag + avalon_pkFlag +
    uptownFlag + w_garfield_pkFlag + forest_glenFlag + aanhp +
    forecl + educ8 + hisp + north_pkFlag + bridgeportFlag + gd_blvdFlag +
    roselandFlag + armour_sqFlag + whitenhp + n_lawndaleFlag +
    archer_htsFlag + w_townFlag + s_shoreFlag + asianp + e_garfield_pkFlag +
    hegewischFlag + englewoodFlag + e_sideFlag + ashburn_greshFlag,
    data = inData)

Coefficients:
     (Intercept)          mdfamy0        educ160pro           renter       oaklandFlag    near_w_sideFlag          loopFlag
        1.78e+02          1.99e-04          3.09e+01         -1.21e+01         -2.25e+01         -6.44e+00          1.85e+01
    logan_sqFlag      beverlyFlag      douglasFlag       hyde_pkFlag      austinFlag   humboldt_pkFlag   w_englewoodFlag
        6.02e+00         -1.25e+01         -8.33e+00         -9.33e+00          8.61e+00          8.18e+00          9.23e+00
  lincoln_pkFlag     riverdaleFlag     avalon_pkFlag       uptownFlag  w_garfield_pkFlag    forest_glenFlag           aanhp
       -6.58e+00         -2.29e+01         -1.06e+01         -7.12e+00          1.02e+01         -1.04e+01         -1.02e+02
           forecl            educ8             hisp      north_pkFlag    bridgeportFlag      gd_blvdFlag     roselandFlag
        8.30e+01         -2.48e+01         -9.01e+01         -1.22e+01         -8.90e+00         -2.32e+00          5.87e+00
    armour_sqFlag          whitenhp     n_lawndaleFlag     archer_htsFlag        w_townFlag       s_shoreFlag           asianp
       -1.45e+01         -9.79e+01          5.51e+00          6.78e+00          2.95e+00          5.79e+00         -8.21e+01
e_garfield_pkFlag     hegewischFlag     englewoodFlag        e_sideFlag  ashburn_greshFlag
        4.89e+00         -1.10e+01          4.71e+00         -6.93e+00          3.57e+00
```

3.3 Conclusion

Neighborhoods included in the model (circled in white):



Correlation of independent variables with dependent rent and Box Cox of rent:

Correlation of Rent and Box Cox of Rent to Independent Variables (Only > 0.1 Shown)					
	Rent	Box Cox of Rent		Rent	Box Cox of Rent
mdfamy0	0.703704412	0.6612591	property	0.116441188	0.07602817
asianp	0.132488176	0.1391911	forecl	-0.117521105	-0.10395209
whitenhp	0.603970093	0.5875034	n_centerFlag	0.10811778	0.10451594
aanhp	-0.409192145	-0.4216595	lakeviewFlag	0.29245562	0.26512215
renter	-0.282186323	-0.3245187	lincoln_pkFlag	0.30330934	0.26513777
educ8	-0.3340027	-0.3069567	near_n_sideFlag	0.17151754	0.11475718
educ160pro	0.67587045	0.6332159	n_lawndaleFlag	-0.11283053	-0.1150703
Noheat	-0.169565777	-0.16642686	s_lawndaleFlag	-0.11511727	-0.1123547
murder	-0.117009175	-0.12239347	loopFlag	0.19101863	0.1561445
robbery	-0.219150652	-0.24447906	oaklandFlag	-0.13316763	-0.17695757
personal	-0.318707323	-0.35228226	gd_blvdFlag	-0.13566259	-0.15106625

Correlation of independent variables to each other (multicollinearity analysis):

Independent Variable Multicollinearity Summary (Only > -.6 Shown)					
Income Multicollinearity Correlations			White Proportion Multicollinearity Correlations		
mdfamy0	whitenhp	0.723461691	whitenhp	mdfamy0	0.723461691
mdfamy0	educ160pro	0.788419205	whitenhp	aanhp	-0.732528761
			whitenhp	educ160pro	0.700232204
Total Crime Multicollinearity Correlations					
totcrime	robbery	0.680450686	Hispanic Proportion Multicollinearity Correlations		
totcrime	personal	0.660623077	hisp	educ8	0.83948022
totcrime	property	0.9776299	hisp	aanhp	-0.603465283
Robbery Multicollinearity Correlations			African Amerian Proportion Multicollinearity Correlations		
robbery	personal	0.8858553	aanhp	hisp	-0.603465283
robbery	totcrime	0.680450686	aanhp	whitenhp	-0.732528761

Neighborhood model descriptive statistics:

	Num	Rent		bcRent		mdfamy0		whitenhp		aanhp		renter		educ8		educ160pro		forecl	
canam77	Pts	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
ARMOUR-SQ	5	486.4	128.5508	75.936	12.83434	35171.2	9118.276	0.23858	0.205033	0.10084	0.130881	0.68716	0.074785	0.20666	0.099636	0.22256	0.127703	0.001497	0.001424
AVALON-PK	3	513.6667	124.8092	78.76438	12.63698	56131	4117.187	0.008067	0.008709	0.980533	0.016977	0.308967	0.103136	0.034633	0.025297	0.192967	0.044286	0.011292	0.003448
BEVERLY	7	704	128.5729	96.55332	11.22815	81472.57	13937.52	0.680729	0.227059	0.279786	0.227736	0.155886	0.156685	0.013357	0.009558	0.4935	0.074823	0.006041	0.005908
BRIDGEPORT	14	545.6429	73.06226	82.13222	6.90105	44047.64	12596.56	0.440771	0.116768	0.007993	0.014737	0.552793	0.06891	0.176057	0.07621	0.180307	0.061042	0.011493	0.02804
DOUGLAS	10	498.3	272.8272	75.08011	27.30241	37831.8	20653.72	0.06366	0.111238	0.8619	0.216973	0.78073	0.221035	0.07583	0.054329	0.27296	0.174574	0.017148	0.035707
FOREST-GLEN	4	721.5	62.44731	98.34219	5.503154	80270.5	12727.13	0.81505	0.093656	0.00585	0.001737	0.117	0.074029	0.0485	0.018289	0.419025	0.117281	0.001469	0.000764
GD-BLVD	15	514.6667	89.97275	78.90859	9.860825	31454.47	12844.54	0.006993	0.009363	0.981973	0.014058	0.628153	0.145266	0.06688	0.030811	0.11064	0.060271	0.022116	0.017785
HYDE-PK	13	634.5385	41.41581	90.60114	3.818456	49103.85	7455.712	0.521046	0.165586	0.023385	0.012765	0.558977	0.096344	0.118554	0.060474	0.236231	0.072672	0.002679	0.001385
KENWOOD	32	884.125	184.0708	111.3696	16.16204	84486.25	22918.47	0.800884	0.082378	0.044738	0.047753	0.699903	0.083127	0.035916	0.038248	0.700344	0.090866	0.002874	0.005974
LINCOLN-PK	10	667.7	46.03875	93.60319	4.081442	50718.4	5374.666	0.58012	0.069168	0.04546	0.025924	0.69932	0.137102	0.10588	0.02231	0.43224	0.099744	0.003098	0.003591
NEAR-W-SIDE	35	594.3714	373.3429	83.5608	32.34475	44435.03	30123.3	0.206114	0.238732	0.622151	0.362896	0.697623	0.229992	0.111166	0.088785	0.283403	0.27748	0.010271	0.017863
NORTH-PK	5	615.8	196.4528	87.79279	20.23137	57271.2	15062.69	0.54006	0.137961	0.06456	0.105382	0.50218	0.236454	0.0674	0.044713	0.34424	0.065053	0.001569	0.001459
OAKLAND	4	265.5	149.7765	50.12041	19.30686	29927.25	25363.44	0.00475	0.006936	0.973325	0.01819	0.87375	0.120312	0.089475	0.037263	0.11125	0.074565	0.005278	0.004751
RIVERDALE	1	218	0	45.49327	0	13979	0	0.0049	0	0.949	0	0.8672	0	0.0709	0	0.0265	0	0.006092	0
UPTOWN	12	590.0833	131.0465	86.05068	12.37577	43763.17	18176.55	0.449458	0.149404	0.187733	0.127858	0.757992	0.142869	0.114508	0.058992	0.40955	0.148173	0.004219	0.0055

Highlighting Legend			
	= Model P Value Significance <= 0.001		= Value greater than or equal to overall variable 3rd quartile
	= Model P Value Significance <= 0.01		= Value less than or equal to overall variable 1st quartile
	= Model P Value Significance <= 0.05		

4. Code

4.1 Kari Palmier R Code

This code was written in a modular fashion. All R outputs were written to a series of text files that were automatically saved in a given folder structure, so they could be accessed anytime for analysis. The plots generated in R were also automatically saved into this same folder structure for analysis.

A model_data. function creates the model and calls several other functions, one that generates the R outputs and plots required for exploratory analysis (descriptive statistics, dependent variable versus independent variable plots, dependent variable versus independent dummy variable box plots, all variable box plots, independent variable versus each of the other independent variable plots, dependent variable histogram, dependent variable normal plot, and correlation outputs), one that generates the R outputs and plots for model analysis (model summary, ANOVA, residual plots, influential points, VIF value, standardized coefficients), one that generates the model selection R outputs for adjusted R-squared, Cp, backward, stepwise, and forward model selection, one that performs cross validation on the model, and one that performs validation using the training and testing sets. Upper level files then call the model_data function with different sets of data and different independent variables specified. These upper level files contain the logic to eliminate rows with missing data or dependent variables with bad values, convert the foreclosure variable from percentage to proportion (was found to be actually given in percentages instead of proportions in the data text file), eliminate influential and outlier points, perform the modeling loops that eliminate independent variables one a time (either by eliminating largest coefficient p value first then largest VIF, or the opposite), divides the dataset into training and testing sets, and calls function used to generate predicted data from new data points.

Final Main Upper Level File:

```
#############################################################################################################################
#############
#
#  Project Final.R
#
#  This file contains the high level code used in creating the final models.  The first model created uses all the indep vars.
#  The second model removes influential points.  The third model removes extreme outliers.  Next models are made starting at the
#  backward and stepwise model selection recommendations with the regular Rent variable.  After that, the Box Cox of Rent is
#  applied to the data after the extreme outliers were removed.  After that, models starting with the backward and stepwise
#  model selection recommendations with the Box Cox of the Rent variable.  During the creation of these models, first the coeffs
#  with the max p value above 0.05 are eliminated, then the coeffs with the max VIF over 10.  After the backward and stepwise
#  models of both Rent and Box Cox Rent are generated, the data set is divided into training and testing sets and a new model is
#  generated for each combo (Rent backward, Rent stepwise, Box Cox Rent backward, Box Cox Rent stepwise) using the training set.
#  Validatio is then done using the testing set.  Finally, new data points are imported and used to generate predictions for each
#  model combination.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#############################################################################################################################
#############

library(psych)
library(QuantPsyc)
library(car)
library(leaps)
library(DAAG)

# Assign output file directory based on relative path of this R script
mainPath = getSrcDirectory(function(x) {x})
mainPath = gsub("/", "\\\\", mainPath)
mainPath = paste(mainPath, '\\', sep="")
dir.create(file.path(mainPath, "R Anlys Final"), showWarnings = FALSE)
anlysPath = paste(mainPath, "R Anlys Final\\", sep="")

# Get subfunctions required
source(paste(mainPath, "Common R Functions\\analyze_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\explore_data.R", sep=""))
```

```
source(paste(mainPath, "Common R Functions\\select_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\validate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\crossvalidate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\get_outliers.R", sep=""))
source(paste(mainPath, "Common R Functions\\model_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\remove_inf_pts.R", sep=""))
source(paste(mainPath, "Common R Functions\\predict_values.R", sep=""))

# Import Dataset
rentData = read.table(paste(mainPath, "\\Chicago_rent\\ChicagoProject.txt", sep=""),
                header=T)

##################################### Variable Initialization
#####################################################################

varInfo.xVarNames = c("mdfamy0", "asianp", "whitenhp", "aanhp", "hisp", "renter", "educ8",
                "educ160pro", "Noheat", "totcrime", "murder", "robbery", "personal", "property", "forecl")
varInfo.xVarPlotNames = c("Median Family Income", "Prop of Asian American",
                  "Prop of Non-Hispanic White", "Prop of Non-Hispanic African American",
                  "Prop Hispanic American", "Prop of Renter Occupied", "Prop of With 8 Yrs School",
                  "Prop With Bachelor or Grad School", "Prop With No Heat", "Prop Total Crime",
                  "Prop Murder", "Prop Robbery", "Prop Personal Crime", "Prop Property Crime",
                  "Prop Foreclosures")
varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

percTestPts = 25
percCrossValPts = 20
highRentLimit =  720
lowRentLimit = 450
bcHighRentLimit = 98
bcLowRentLimit = 73

outlierLevel = 5

##################################### Dataset Conversion
#############################################################################

dir.create(file.path(anlysPath, "Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "Init\\", sep="")

# Change the data frame columns from factor type to character type
i <- sapply(rentData, is.factor)
rentData[i] <- lapply(rentData[i], as.character)

# Remove any missing points from the dataset
outData = get_outliers(rentData, "-", "EQ", varInfo.xVarNames, basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}


# Change all data frame columns from character to numeric (except for the column of neighborhood names)
totVars = append(varInfo.yVarName, varInfo.xVarNames)
numVars = length(totVars)
for (i in 1:numVars){
  var = totVars[i]
  if (is.character(var)){
    rentData[var] = as.numeric(rentData[,var])
  }
}

################################################  Outlier Removal
#####################################################################
```

```
# Remove any 0 Rent entries since 0 rent doesn't make logical sense and can sway the data
outData = get_outliers(rentData, 0, "LTE", c("Rent"), basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}


################################################  Convert Foreclosures To Proportions
################################################

rentData$forecl = rentData$forecl / 100

######################################### Neighborhood Dummy Variable Creation
#########################################################

allVarInfo.xVarNames = varInfo.xVarNames
allVarInfo.xVarPlotNames = varInfo.xVarPlotNames

uniqAreas = unique(rentData$canam77)
numAreas = length(uniqAreas)
flagNames = c()
flagDescs = c()
for(i in 1:(numAreas-1)){
  val = uniqAreas[i]
  lowVal = tolower(val)
  matchNdx = which(rentData$canam77 %in% val)

  if(length(matchNdx) > 0){
    temp = gsub("-", "_", lowVal)
    temp2 = gsub("'", "", temp)
    newVal = paste(temp2, "Flag", sep="")

    tempFlag = (rentData$canam77 == val)*1
    rentData[newVal] = tempFlag

    flagNames = append(flagNames, newVal)
    flagDescs = append(flagDescs, val)
  }
}

numFlagVars = length(flagNames)
for(i in 1:numFlagVars){
  allVarInfo.xVarNames = append(allVarInfo.xVarNames, flagNames[i])
  allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, flagDescs[i])
}

########################################### Assign x variables for functions
#############################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

########################################### Create Box Cox of Rent
#########################################################################

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
    xformStr = var
  }else{
    xformStr = paste(xformStr, "+", var, sep="")
  }
```

```
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda = transInfo$result[1]

if(yLambda > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda) - 1)/yLambda
}else{
  rentData$bcRent = log(rentData$Rent)
}


########################################## Write dataset to a file for other group members
##############################################

write.table(rentData, paste(mainPath, "ChicagoProject_WithDummyNoBadPts.txt", sep=""), sep="\t", row.names=FALSE)

########################################## All Var Modeling
#############################################################################

dir.create(file.path(anlysPath, "M1 All Vars"), showWarnings = FALSE)
basePath = paste(anlysPath, "M1 All Vars\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
baseData = data.frame()

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")


########################################## Remove Influential Pts
#############################################################################

dir.create(file.path(anlysPath, "M2 All Vars No Inf Pts"), showWarnings = FALSE)
basePath = paste(anlysPath, "M2 All Vars No Inf Pts\\", sep="")

infPtsNdx = modelOutParams[[1]]
if(length(infPtsNdx) > 0){

  outData = remove_inf_pts(rentData, infPtsNdx, basePath, "InfPts")
  rentData = outData
}

########################################## All Var Modeling w/o Inf Pts
#############################################################

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
```

```
modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

########################################## Remove Extreme Outlier Pts
#####################################################################

dir.create(file.path(anlysPath, "M3 All Vars No Out"), showWarnings = FALSE)
basePath = paste(anlysPath, "M3 All Vars No Out\\", sep="")

infPtsNdx = modelOutParams[[2]]
if(length(infPtsNdx) > 0){

  outData = remove_inf_pts(rentData, infPtsNdx, basePath, "ExtremeOut")
  rentData = outData
}


########################################## All Var Modeling w/o Outliers
#################################################################

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

selNames = modelOutParams[[9]]
backTempNames = selNames[[1]]
stepTempNames = selNames[[2]]

########################################## Backward Sel Modeling
#############################################################################

tempLength = length(backTempNames)
backVarInfo.xVarNames = backTempNames[2:tempLength]
numBackVars = length(backVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numBackVars){
  var = backVarInfo.xVarNames[i]
  matchNdx = which(backVarInfo.xVarNames %in% var)
  tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

backVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M4 Back Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 Back Init\\", sep="")

varInfo.xVarNames = backVarInfo.xVarNames
varInfo.xVarPlotNames = backVarInfo.xVarPlotNames

maxPVal = 1
maxVIF = 20
modelNdx = 5
```

```
backCoeffsRmvd = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

 if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  backCoeffsRmvd = append(backCoeffsRmvd, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }else if (maxVIF > 10){
  tempVals = vifVals[,1]
  maxVifNdx = which(tempVals %in% maxVIF)
  vifNames = rownames(vifVals)
  maxVifName = vifNames[maxVifNdx]
  backCoeffsRmvd = append(backCoeffsRmvd, maxVifName)

  matchNdx = which(varInfo.xVarNames %in% maxVifName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1

 }
}

finalBackVarInfo.xVarNames = varInfo.xVarNames
finalBackVarInfo.xVarPlotNames = varInfo.xVarPlotNames

########################################## Stepwise Sel Modeling
#########################################################################

tempLength = length(stepTempNames)
stepVarInfo.xVarNames = stepTempNames[2:tempLength]
numStepVars = length(stepVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numStepVars){
 var = stepVarInfo.xVarNames[i]
 matchNdx = which(stepVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

stepVarInfo.xVarPlotNames = tempPlotVars
```

```
# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M4 Step Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 Step Init\\", sep="")

varInfo.xVarNames = stepVarInfo.xVarNames
varInfo.xVarPlotNames = stepVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 5
maxVIF = 20
stepCoeffsRmvd = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

  if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  stepCoeffsRmvd = append(stepCoeffsRmvd, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
  }else if (maxVIF > 10){
   tempVals = vifVals[,1]
   maxVifNdx = which(tempVals %in% maxVIF)
   vifNames = rownames(vifVals)
   maxVifName = vifNames[maxVifNdx]
   stepCoeffsRmvd = append(stepCoeffsRmvd, maxVifName)

   matchNdx = which(varInfo.xVarNames %in% maxVifName)
   varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
   varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

   folderName = paste('M', modelNdx, " Step", sep="")
   dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
   basePath = paste(anlysPath, folderName, "\\", sep="")

   modelNdx = modelNdx + 1

  }
}

finalStepVarInfo.xVarNames = varInfo.xVarNames
finalStepVarInfo.xVarPlotNames = varInfo.xVarPlotNames
```

```
######################################### Box Cox All Var Modeling
#########################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

dir.create(file.path(anlysPath, "M4 All Vars BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 All Vars BC\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

selNames = modelOutParams[[9]]
backTempNames = selNames[[1]]
stepTempNames = selNames[[2]]

######################################### Backward Sel Modeling
#########################################################################

tempLength = length(backTempNames)
backVarInfo.xVarNames = backTempNames[2:tempLength]
numBackVars = length(backVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numBackVars){
 var = backVarInfo.xVarNames[i]
 matchNdx = which(backVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

backVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M5 Back BC Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M5 Back BC Init\\", sep="")

varInfo.xVarNames = backVarInfo.xVarNames
varInfo.xVarPlotNames = backVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 6
maxVIF = 20
backCoeffsRmvdBc = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
```

```
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

 if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  backCoeffsRmvdBc = append(backCoeffsRmvdBc, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }else if (maxVIF > 10){
  tempVals = vifVals[,1]
  maxVifNdx = which(tempVals %in% maxVIF)
  vifNames = rownames(vifVals)
  maxVifName = vifNames[maxVifNdx]
  backCoeffsRmvdBc = append(backCoeffsRmvdBc, maxVifName)

  matchNdx = which(varInfo.xVarNames %in% maxVifName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1

 }
}

finalBackBcVarInfo.xVarNames = varInfo.xVarNames
finalBackBcVarInfo.xVarPlotNames = varInfo.xVarPlotNames

########################################## Stepwise Sel Modeling
#########################################################################

tempLength = length(stepTempNames)
stepVarInfo.xVarNames = stepTempNames[2:tempLength]
numStepVars = length(stepVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numStepVars){
 var = stepVarInfo.xVarNames[i]
 matchNdx = which(stepVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

stepVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
```

```
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M5 Step BC Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M5 Step BC Init\\", sep="")

varInfo.xVarNames = stepVarInfo.xVarNames
varInfo.xVarPlotNames = stepVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 6
maxVIF = 20
stepCoeffsRmvdBc = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

 if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  stepCoeffsRmvdBc = append(stepCoeffsRmvdBc, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }else if (maxVIF > 10){
  tempVals = vifVals[,1]
  maxVifNdx = which(tempVals %in% maxVIF)
  vifNames = rownames(vifVals)
  maxVifName = vifNames[maxVifNdx]
  stepCoeffsRmvdBc = append(stepCoeffsRmvdBc, maxVifName)

  matchNdx = which(varInfo.xVarNames %in% maxVifName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1

 }
}

finalStepBcVarInfo.xVarNames = varInfo.xVarNames
finalStepBcVarInfo.xVarPlotNames = varInfo.xVarPlotNames

######################################### Create Training and Testing Datasets
#########################################################

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"
```

```
# Create sample indices
selectNdx = sample(1:nrow(rentData), (1-(percTestPts/100))*nrow(rentData))

# Select the training dataset
trainData = rentData[selectNdx, ]

# Select the testing dataset
testData = rentData[-selectNdx, ]

testSetDiff = setdiff(testData$canam77, trainData$canam77)
numDiff = length(testSetDiff)
numOrigRows = length(rentData$canam77)
allMatchNdx = c()
if(numDiff >0){
 for(i in 1:numDiff){
  val = testSetDiff[i]

  matchNdx = which(testData$canam77 %in% val)

  numMatches = length(matchNdx)
  if(matchNdx > 0){
   allMatchNdx = append(allMatchNdx, matchNdx)
  }
 }

 numAllMatches = length(allMatchNdx)
 percMatches = (numAllMatches / numOrigRows) * 100

 if(percMatches < 5){
  tempData = testData[allMatchNdx,]
  trainData = rbind(trainData, setNames(tempData, names(trainData)))
  testData = testData[-allMatchNdx,]
 }
}


testSetDiff = setdiff(testData$canam77, trainData$canam77)
trainSetDiff = setdiff(trainData$canam77, testData$canam77)
trainDataPerc = (length(trainData$canam77)/numOrigRows)*100
testDataPerc = (length(testData$canam77)/numOrigRows)*100


outFileName = paste(anlysPath, "DataSet_Split_Information.txt", sep="")
outFile = file(outFileName, open="wt")

# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Neighborhoods in Testing data that are not in Training data:", quote=FALSE)
print("", quote=FALSE)
print(testSetDiff, quote=FALSE)
print("", quote=FALSE)
print("Neighborhoods in Training data that are not in Testing data:", quote=FALSE)
print("", quote=FALSE)
print(trainSetDiff, quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
print("Percentage of Original data used for Training data:", quote=FALSE)
print("", quote=FALSE)
print(round(trainDataPerc, digits=2), quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
print("Percentage of Original data used for Training data:", quote=FALSE)
print("", quote=FALSE)
print(round(testDataPerc, digits=2), quote=FALSE)
```

```
print("", quote=FALSE)
print("", quote=FALSE)
sink()

close(outFile)
closeAllConnections()


########################################## Backward Train/Test Modeling
################################################################

dir.create(file.path(anlysPath, "M Last Back"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back\\", sep="")

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

varInfo.xVarNames = finalBackVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Stepwise Train/Test Modeling
################################################################

dir.create(file.path(anlysPath, "M Last Step"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step\\", sep="")

varInfo.xVarNames = finalStepVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Backward BC Train/Test Modeling
############################################################

dir.create(file.path(anlysPath, "M Last Back BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back BC\\", sep="")

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"

varInfo.xVarNames = finalBackBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackBcVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
```

```
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Stepwise BC Train/Test Modeling
#########################################################

dir.create(file.path(anlysPath, "M Last Step BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step BC\\", sep="")

varInfo.xVarNames = finalStepBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepBcVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Create Prediction Points
#####################################################################

predictData = read.table(paste(mainPath, "Rent_PredictionData.txt", sep=""), header=T)

# Change the data frame columns from factor type to character type
i <- sapply(predictData, is.factor)
predictData[i] <- lapply(predictData[i], as.character)

# Change all data frame columns from character to numeric (except for the column of neighborhood names)
totVars = length(colnames(predictData))
numVars = length(totVars)
for (i in 1:numVars){
 var = totVars[i]
 if (is.character(var)){
  predictData[var] = as.numeric(predictData[,var])
 }
}

uniqAreas = unique(rentData$canam77)
numAreas = length(uniqAreas)
for(i in 1:(numAreas-1)){
 val = uniqAreas[i]
 lowVal = tolower(val)

 temp = gsub("-", "_", lowVal)
 temp2 = gsub("'", "", temp)
 newVal = paste(temp2, "Flag", sep="")

 tempFlag = (predictData$canam77 == val)*1
 predictData[newVal] = tempFlag
}


predictData["Rent"] = NULL

########################################## Predict Points With Backward Model
#########################################################

dir.create(file.path(anlysPath, "M Last Back"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back\\", sep="")
```

```
varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

varInfo.xVarNames = finalBackVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
    xformStr = var
  }else{
    xformStr = paste(xformStr, "+", var, sep="")
  }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

backModel = lm(modelForm, data=rentData)

passFlag = predict_values(backModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################### Predict Points With Stepwise Model
#############################################################

dir.create(file.path(anlysPath, "M Last Step"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step\\", sep="")

varInfo.xVarNames = finalStepVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
    xformStr = var
  }else{
    xformStr = paste(xformStr, "+", var, sep="")
  }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

stepModel = lm(modelForm, data=rentData)

passFlag = predict_values(stepModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################### Backward BC Train/Test Modeling
#############################################################

dir.create(file.path(anlysPath, "M Last Back BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back BC\\", sep="")

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"

varInfo.xVarNames = finalBackBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackBcVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]
```

```
 if(j == 1){
   xformStr = var
 }else{
   xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

backBcModel = lm(modelForm, data=rentData)

passFlag = predict_values(backBcModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################## Stepwise BC Train/Test Modeling
#########################################################

dir.create(file.path(anlysPath, "M Last Step BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step BC\\", sep="")

varInfo.xVarNames = finalStepBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepBcVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
   xformStr = var
 }else{
   xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

stepBcModel = lm(modelForm, data=rentData)

passFlag = predict_values(stepBcModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################## Create Output Model Sel Info File
#########################################################

# Create output file
outFileName = paste(anlysPath, "Model_Sel_Info_R_Output.txt", sep="")
outFile = file(outFileName, open="wt")

# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Variables removed during backward selection of Rent (in order or removal):", quote=FALSE)
print("", quote=FALSE)
print(backCoeffsRmvd, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during stepwise selection of Rent (in order or removal):", quote=FALSE)
print(stepCoeffsRmvd, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during backward selection of Box Cox Rent (in order or removal):", quote=FALSE)
print("", quote=FALSE)
print(backCoeffsRmvdBc, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during stepwise selection of Box Cox Rent (in order or removal):", quote=FALSE)
print(stepCoeffsRmvdBc, quote=FALSE)
print("", quote=FALSE)
```

```
sink()

close(outFile)
closeAllConnections()
```

model_data.R File:

```
#########################################################################################################################################
#############
#
#  model_data.R
#
#  This file calls a function to generate all of the exploratory plots and R outputs, creates the model using the data passed in,
#  then calls functions that analyze the model, perform cross validation, perform training/testing validation, and perform
#  predictions.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#########################################################################################################################################
#############

model_data <- function(inData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, outPath, dataDesc){

 numXVars = length(varInfo.xVarNames)
 for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
   xformStr = var
  }else{
   xformStr = paste(xformStr, "+", var, sep="")
  }
 }

 yOutName = paste(toupper(substr(varInfo.yVarName, 1, 1)), substr(varInfo.yVarName, 2, nchar(varInfo.yVarName)), sep="")

 dir.create(file.path(outPath, yOutName), showWarnings = FALSE)
 basePath = paste(outPath, yOutName, "\\", sep="")

 ################################################### Exploratory Step
#############################################################

 if(modelFlags.performExp == TRUE){

  # Perform Exploratory Data Analysis after outlier removal
  exploreStr = paste("Explore_", dataDesc, "_", yOutName, sep="")

  if(i == 1){
   explorePath = explore_data(inData, varInfo.yVarName, varInfo.yVarPlotName, varInfo, modelFlags.makeExpPlots,
                 modelFlags.makeExpXXPlots, basePath, exploreStr)
  }else {
   explorePath = explore_data(inData, varInfo.yVarName, varInfo.yVarPlotName, varInfo, modelFlags.makeExpPlots,
                 FALSE, basePath, exploreStr)
  }

  if(numXVars <= 20){
   if(modelFlags.makeExpPlots == TRUE){

    # Plot Matrix of Scatterplots
    exploreBasePath = paste(basePath, exploreStr, "\\", sep="")
    fileName = paste(exploreBasePath, exploreStr, "_ScatterPlotMatrix.jpeg", sep="")
    plotTitle = "Scatterplot Matrix For All Variables"
```

```
      jpeg(file=fileName)

      formStr = paste("~", varInfo.yVarName, "+", xformStr, sep="")
      pairForm = as.formula(formStr)
      pairs(pairForm, data=inData, main=plotTitle)
      dev.off()
    }
   }

  }


  ################################################# Data Modeling Step
############################################################

  if(modelFlags.performModel == TRUE){
   modelStr = paste("Model_", dataDesc, "_", yOutName, sep="")

   dir.create(file.path(basePath, modelStr), showWarnings = FALSE)
   modelPath = paste(basePath, modelStr, "\\", sep="")

   # Create model with all x variables and data and perform model analysis
   if(modelFlags.quadXModel == TRUE){
    formStr = paste(varInfo.yVarName, "~(", xformStr, ")^2", sep="")
   }else{
    formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
   }

   modelForm = as.formula(formStr)

   inModel = lm(modelForm, data=inData)

   fitSummary = summary(inModel)

   modelOutParams = vector("list", 11)
   modelOutParams[[1]] = analyze_model(inModel, inData, modelForm, varInfo, modelPath, modelStr)

   stdResiduals = rstandard(inModel)
   outlierVal = abs(stdResiduals) > outlierLevel
   outlierNdx = which(outlierVal %in% TRUE)
   modelOutParams[[2]] = outlierNdx

   fStatInfo = fitSummary$fstatistic
   modelOutParams[[3]] = fitSummary$r.squared
   modelOutParams[[4]] = fitSummary$adj.r.squared
   modelOutParams[[5]] = fStatInfo[1]
   modelOutParams[[6]] = pf(fStatInfo[1],fStatInfo[2],fStatInfo[3],lower.tail=F)
   modelOutParams[[7]] = fitSummary$coefficients
   modelOutParams[[8]] = vif(inModel)


   baseFormStr = paste(varInfo.yVarName, "~1", sep="")
   baseForm = as.formula(baseFormStr)

   baseModel = lm(baseForm, data=inData)
   modelOutParams[[9]] = select_model(inModel, inData, varInfo.yVarName, baseModel, varInfo, modelPath, modelStr)

   modelOutParams[[10]] = crossvalidate_data(inModel, inData, percCrossValPts, varInfo.yVarName, varInfo, modelPath, modelStr)

   modelOutParams[[11]] = validate_data(inModel, inData, testData, varInfo.yVarName, percTestPts, modelFlags.validateTest,
                varInfo, modelPath, modelStr)

  }

  return(modelOutParams)
 }
```

explore_data.R File:

```
#########################################################################################################################################
#############
#
#  explore_data.R
#
#  This file generates the descriptive statistic R outputs, the dependent vs independent var plots, the dependent vs. independent
#  dummy variable box plots, the individial var box plots, the independent vs. other independent var plots, the dependent
#  histogram and normal plot, and the full correlation of all vars to each other.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#########################################################################################################################################
#############

explore_data <- function(inData, currentY, currentPlotY, varInfo, makeAllPlots, makeXXPlots, outPath, dataDesc){

 # Create output file
 dir.create(file.path(outPath, dataDesc), showWarnings = FALSE)
 explorePath = paste(outPath, dataDesc, "\\", sep="")
 outFileName = paste(explorePath, dataDesc, "_R_Output.txt", sep="")
 outFile = file(outFileName, open="wt")

 sumData = summary(inData)
 descData = describe(inData)

 allNames = append(currentY, varInfo.xVarNames)
 allPlotNames = append(currentPlotY, varInfo.xVarPlotNames)


 # Print descriptive statistics
 sink(file=outFile, append=TRUE)
 print("Data Set Descriptive Statistics Summary:", quote=FALSE)
 print("", quote=FALSE)
 print(sumData, quote=FALSE)
 print("", quote=FALSE)
 print("Data Set Descriptive Statistics Describe Values:", quote=FALSE)
 print("", quote=FALSE)
 print(descData, quote=FALSE)
 print("", quote=FALSE)
 sink()

 if(makeAllPlots == TRUE){
  # Plot Y Histogram
  fileName = paste(explorePath, dataDesc, "_", currentY, "Hist.jpeg", sep="")
  plotTitle = paste(currentPlotY, "Histogram")
  jpeg(file=fileName)
  yData = c(t(inData[currentY]))
  hist(yData, prob=TRUE, main=plotTitle, xlab=currentPlotY)
  xFit = seq(min(yData), max(yData), length=length(yData))
  yFit = dnorm(xFit, mean=mean(yData), sd=sd(yData))
  lines(xFit, yFit, col="red", lwd=2)
  dev.off()

  # Plot Y Normal Plot
  fileName = paste(explorePath, dataDesc, "_", currentY, "NormPlot.jpeg", sep="")
  plotTitle = paste(currentPlotY, "Normal Plot")
  jpeg(file=fileName)
  qqnorm(yData, main=plotTitle)
  qqline(yData)
```

```
dev.off()

# Boxplot Of All Variables
numVars = length(allNames)
dir.create(file.path(explorePath, "Box Plots"), showWarnings = FALSE)
newPath = paste(explorePath, "Box Plots\\", sep="")
for (i in 1:numVars){
 var = allNames[i]
 fileName = paste(newPath, dataDesc, "_", allNames[i], "_BoxPlot.jpeg", sep="")
 plotTitle = paste(allPlotNames[i], " Box Plot")
 jpeg(file=fileName)
 plotData = c(t(inData[var]))
 boxplot(plotData, main=plotTitle)
 dev.off()
}

# Boxplot Of Dummy Variables vs Y Variable
numXVars = length(varInfo.xVarNames)
dir.create(file.path(newPath, currentPlotY), showWarnings = FALSE)
newBoxPath = paste(newPath, currentPlotY, "\\", sep="")
for (i in 1:numXVars){

 var = varInfo.xVarNames[i]

 uniqXVar = unique(inData[,var])

 if (length(uniqXVar) <= 2){
  fileName = paste(newBoxPath, dataDesc, "_", currentPlotY, "_", varInfo.xVarNames[i], "_BoxPlot.jpeg", sep="")
  plotTitle = paste(currentY, " Versus ", varInfo.xVarNames[i], " Box Plot")
  jpeg(file=fileName)
  boxplot(inData[,currentY]~inData[,var], main=plotTitle)
  dev.off()

 }
}

fileName = paste(newPath, dataDesc, "_AllXBoxPlot.jpeg", sep="")
plotTitle = paste("All X Variable Box Plot")
jpeg(file=fileName)
boxplot(inData[varInfo.xVarNames], main=plotTitle)
dev.off()

fileName = paste(newPath, dataDesc, "_AllVarBoxPlot.jpeg", sep="")
plotTitle = paste("All Variable Box Plot")
jpeg(file=fileName)
boxplot(inData[allNames], main=plotTitle)
dev.off()

# Create scatterplots of all x values versus y value
numXVars = length(varInfo.xVarNames)
newFolder = paste(currentY, " Scatter Plots", sep="")
dir.create(file.path(explorePath, newFolder), showWarnings = FALSE)
newPath = paste(explorePath, newFolder, "\\", sep="")
for (i in 1:numXVars){
 var = varInfo.xVarNames[i]
 fileName = paste(newPath, dataDesc, "_", var, "Vs", currentY, ".jpeg", sep="")
 plotTitle = paste(currentPlotY, " Versus ", varInfo.xVarPlotNames[i])
 plotData = data.frame(inData[var], inData[currentY])
 jpeg(file=fileName)
 plot(plotData, main=plotTitle,
     xlab=varInfo.xVarPlotNames[i], ylab=currentPlotY)
 dev.off()
}

if(makeXXPlots == TRUE){
 # Create scatterplots of all x values versus x value
```

```
    numXVars = length(varInfo.xVarNames)
    otherVars = varInfo.xVarNames
    for (i in 1:numXVars){

      var = varInfo.xVarNames[i]
      otherVars = setdiff(otherVars, var)
      numOtherVars = length(otherVars)

      if(numOtherVars > 0){
        newFolder = paste(var, " Scatter Plots", sep="")
        dir.create(file.path(explorePath, newFolder), showWarnings = FALSE)
        newPath = paste(explorePath, newFolder, "\\", sep="")


        for (j in 1:numOtherVars){
          otherVar = otherVars[j]
          otherNdx = match(otherVar, varInfo.xVarNames)
          fileName = paste(newPath, dataDesc, "_", var, "Vs", otherVar, ".jpeg", sep="")
          plotTitle = paste(varInfo.xVarPlotNames[i], " Versus ", varInfo.xVarPlotNames[otherNdx])
          plotData = data.frame(inData[otherVar], inData[var])
          jpeg(file=fileName)
          plot(plotData, main=plotTitle,
              xlab=varInfo.xVarPlotNames[otherNdx], ylab=varInfo.xVarPlotNames[i])
          dev.off()

        }
      }
    }
  }
}


  # Create Correlation Values for all data
  numAllVars = length(allNames)
  sink(file=outFile, append=TRUE)
  print("", quote=FALSE)

  for(i in 1:numAllVars){
    currentCor = cor(inData[allNames[i]], inData[allNames])

    assign("last.warning", NULL, envir = baseenv())

    print(paste("Correlation of ", allNames[i], " and rest of variables:", sep=""), quote=FALSE)
    print("", quote=FALSE)
    print(currentCor, quote=FALSE)
    print("", quote=FALSE)
    print("", quote=FALSE)

    assign("last.warning", NULL, envir = baseenv())

  }

  sink()

  close(outFile)
  closeAllConnections()

  return(explorePath)
}
```

analyze_model.R File:

```
#########################################################################################################################################
#############
#
#  analyze_model.R
#
#  This file generates the model summary, ANOVA, standardized coeff, VIF, and influential point R outputs and the std residual vs
#  predicted, std residual normal plot, and std residual vs independent var plots.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#########################################################################################################################################
#############

analyze_model <- function(model, inData, modelForm, varInfo, basePath, modelDesc){

 # Create output file
 outFileName = paste(basePath, modelDesc, "_Analysis_R_Output.txt", sep="")
 outFile = file(outFileName, open="wt")

 # Create plot path
 dir.create(file.path(basePath, "Analysis Plots"), showWarnings = FALSE)
 plotPath = paste(basePath, "Analysis Plots\\", sep="")


 modelSummary = summary(model)
 modelAnova = anova(model)

 # Print model summary values to output file
 sink(file=outFile, append=TRUE)
 print("Model Formula:", quote=FALSE)
 print("", quote=FALSE)
 print(modelForm)
 print("", quote=FALSE)
 print("Model Summary:", quote=FALSE)
 print("", quote=FALSE)
 print(modelSummary)
 print("", quote=FALSE)
 print("Model ANOVA Analysis:", quote=FALSE)
 print("", quote=FALSE)
 print(modelAnova)
 print("", quote=FALSE)
 sink()

 # Calculate Standardized Coefficients and display them and confidence levels
 stdCoeffs = lm.beta(model)
 modelCIs = confint(model, level=0.95)

 # Print model stnd coeffs conf int values to output file
 sink(file=outFile, append=TRUE)
 print("", quote=FALSE)
 print("Standardized Coefficients:", quote=FALSE)
 print("", quote=FALSE)
 print(stdCoeffs)
 print("", quote=FALSE)
 print("95% Confidence Intervals:", quote=FALSE)
 print("", quote=FALSE)
 print(modelCIs)
 print("", quote=FALSE)
 sink()


 # Plot Standardized Residuals vs Predicted Values
 fileName = paste(plotPath, modelDesc, "_StndResVsPredVals.jpeg", sep="")
 plotTitle = "Standardized Residuals Versus Predicted Values"
 jpeg(file=fileName)
```

```
plot(fitted(model), rstandard(model), main=plotTitle,
    xlab="Predicted Values", ylab="Standardized Residuals")
abline(a=0, b=0, col="red")
abline(a=3, b=0, col="blue")
abline(a=-3, b=0, col="blue")
dev.off()

# Plot Standardized Residuals Normal Plot
fileName = paste(plotPath, modelDesc, "_StndResNormalPlot.jpeg", sep="")
plotTitle = "Standardized Residuals Normal Plot"
jpeg(file=fileName)
qqnorm(rstandard(model), main=plotTitle)
qqline(rstandard(model), col=2)
dev.off()

# Plot Standardized Residuals vs x variables
numXVars = length(varInfo.xVarNames)
for (i in 1:numXVars){
  var = varInfo.xVarNames[i]
  fileName = paste(plotPath, modelDesc, "_StndResVs", var, ".jpeg", sep="")
  plotTitle = paste("Standard Residuals Versus ", varInfo.xVarPlotNames[i])
  plotData = data.frame(inData[var], rstandard(model))
  jpeg(file=fileName)
  plot(plotData, main=plotTitle,
      xlab=var, ylab="Standardized Residuals")
  abline(a=0, b=0, col="red")
  abline(a=3, b=0, col="blue")
  abline(a=-3, b=0, col="blue")
  dev.off()
}


# Compute the model VIF values
modelVIF = vif(model)

# Print model VIF values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Model VIF Values:", quote=FALSE)
print("", quote=FALSE)
print(modelVIF, quote=FALSE)
print("", quote=FALSE)
sink()

# Get influence point information
inflMeas = influence.measures(model)
sumInflMeas = summary(inflMeas);
dfBetaVal = dfbeta(model)
covRatVal = covratio(model)
dfFitVal = dffits(model)
cookDVal = cooks.distance(model)
hatVals = hatvalues(model)
delStudRes = rstudent(model)

hatInfPts = (hatVals >= 0.5) & (abs(delStudRes) > 3)
hatInfNdx = which(hatInfPts %in% TRUE)

cookDPts = (cookDVal >= 1)
cookDNdx = which(cookDPts %in% TRUE)

combInfNdx = append(hatInfNdx, cookDNdx)
uniqInfNdx = unique(combInfNdx)

# Print model influence point values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
```

```
 print("Potential Influential Points:", quote=FALSE)
 summary(inflMeas)
 print("", quote=FALSE)
 sink()

 # Plot Studentized Residuals vs Hat Values
 fileName = paste(plotPath, modelDesc, "_DelStudResVsHatVals.jpeg", sep="")
 plotTitle = "Deleted Studentized Residuals Versus Hat Values"
 jpeg(file=fileName)
 plot(delStudRes~hatVals, main=plotTitle,
    xlab="Hat Values", ylab="Deleted Studentized Residuals")
 abline(a=0, b=0, col="red")
 abline(a=3, b=0, col="blue")
 abline(a=-3, b=0, col="blue")
 dev.off()


 close(outFile)
 closeAllConnections()

 return(uniqInfNdx)

}
```

select_model.R File:

```
#############################################################################################################################
#############
#
#  select_model.R
#
#  This file performs model selection on the model and input data passed in.  It generates the R output for the R-squared, Cp,
#  backward, stepwise, and forward model selection methods.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#############################################################################################################################
#############

select_model <- function(inModel, inData, currentY, baseModel, varInfo, basePath, modelDesc){

 # Create output file
 outFileName = paste(basePath, modelDesc, "_Selection_R_Output.txt", sep="")
 outFile = file(outFileName, open="wt")


 # Create matrix of X variables and y variable to prepare for model selection
 xData = inData[varInfo.xVarNames]
 yData = c(t(inData[currentY]))

 # Perform CP model selection
 allVarNames = cbind(currentY, varInfo.xVarNames)
 numAllVars = length(allVarNames)

 if(numAllVars <= 31){
  leapCp = leaps(x=xData, y=yData, names=varInfo.xVarNames, method="Cp")
  cpMatrix = cbind(leapCp$size, leapCp$which, leapCp$Cp)

  # Display results in increasing order of Cp
  cpSubset = head(cpMatrix[order(cpMatrix[, dim(cpMatrix)[2]]),])

  # Print cp model selection values to output file
  sink(file=outFile, append=TRUE)
  print("", quote=FALSE)
```

```
 print("Output of Cp Model Selection:", quote=FALSE)
 print("", quote=FALSE)
 print(leapCp, quote=FALSE)
 print("", quote=FALSE)
 print("Cp Model Subsets of Interest:", quote=FALSE)
 print(cpSubset, quote=FALSE)
 print("", quote=FALSE)
 sink()

 # Perform adj-R2 model selection
 leapAdjR2 = leaps(x=xData, y=yData, names=varInfo.xVarNames, method="adjr2")
 adjR2Matrix = cbind(leapAdjR2$size, leapAdjR2$which, leapAdjR2$adjr2)

 # Display results in increasing order of Cp
 adjR2Subset = head(adjR2Matrix[order(adjR2Matrix[, dim(adjR2Matrix)[2]], decreasing=T),])

 # Print adjusted r2 model selection to output file
 sink(file=outFile, append=TRUE)
 print("", quote=FALSE)
 print("Output of Adusted R-Squared Model Selection:", quote=FALSE)
 print("", quote=FALSE)
 print(leapAdjR2, quote=FALSE)
 print("", quote=FALSE)
 print("Adusted R-Squared Model Subsets of Interest:", quote=FALSE)
 print(adjR2Subset, quote=FALSE)
 print("", quote=FALSE)
 sink()
}else{
 sink(file=outFile, append=TRUE)
 print("", quote=FALSE)
 print("Too many variables for Cp and Adjusted R-Squared Model Selection", quote=FALSE)
 print("", quote=FALSE)
 sink()

}

# Perform Backward model selection
backSel = step(inModel, direction="backward", trace=FALSE)

# Print backward selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Output of Backward Model Selection:", quote=FALSE)
print("", quote=FALSE)
print(backSel, quote=FALSE)
print("", quote=FALSE)
sink()

# Create a vector of names for forward sel
tempVar = backSel$coefficients
tempDf = data.frame(tempVar)
backNames = rownames(tempDf)

# Perform Stepwise model selection and print to output file
stepSel = step(baseModel, scope=list(upper=inModel, lower=~1), direction="both", trace=FALSE)

# Print stepwise selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Output of Stepwise Model Selection:", quote=FALSE)
print("", quote=FALSE)
print(stepSel, quote=FALSE)
print("", quote=FALSE)
sink()

# Create a vector of names for forward sel
```

```
 tempVar = stepSel$coefficients
 tempDf = data.frame(tempVar)
 stepNames = rownames(tempDf)

 # Perform Forward model selection
 forwSel = step(baseModel, scope=list(upper=inModel, lower=~1), direction="forward", trace=FALSE)

 # Print forward model selection values to output file
 sink(file=outFile, append=TRUE)
 print("", quote=FALSE)
 print("Output of Forward Model Selection:", quote=FALSE)
 print("", quote=FALSE)
 print(forwSel, quote=FALSE)
 print("", quote=FALSE)
 sink()

 # Create a vector of names for forward sel
 tempVar = forwSel$coefficients
 tempDf = data.frame(tempVar)
 forwNames = rownames(tempDf)

 close(outFile)
 closeAllConnections()

 selNames = vector("list", 3)
 selNames[[1]] = backNames
 selNames[[2]] = stepNames
 selNames[[3]] = forwNames

 return(selNames)

}
```

crossvalidate_data.R File:

```
#############################################################################################################################
#############
#
#  crossvalidate_data.R
#
#  This file performs cross validation on the model and data passed in then generates the RMSE, MAPE, MAE, and R2 difference
#  R outputs.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#############################################################################################################################
#############

crossvalidate_data <- function(inModel, inData, percentTest, currentY, varInfo, basePath, modelDesc){

 # Create output file
 outFileName = paste(basePath, modelDesc, "_Cross_Validation_R_Output.txt", sep="")
 outFile = file(outFileName, open="wt")


 # Calculate the number of points for cross validation
 numFolds = ceiling(100/percentTest)


 fileName = paste(basePath, modelDesc, "_CrossValObsPred.jpeg", sep="")
 jpeg(file=fileName)

 # Print cp model selection values to output file
 sink(file=outFile, append=TRUE)
```

```
print("", quote=FALSE)
print("Output of Cross Validation:", quote=FALSE)
print("", quote=FALSE)

# Perform Cross validation
crossVal = cv.lm(data=inData, inModel, m=numFolds)

print("", quote=FALSE)
print(crossVal, quote=FALSE)
print("", quote=FALSE)
sink()

dev.off()

yPredCv = crossVal$cvpred
yPredModel = crossVal$Predicted
yObs = crossVal[,currentY]

# Calculate the RMSE of CV prediction error
rmseCv = sqrt(sum((yObs - yPredCv)^2)/nrow(inData))

# Calculate the CV Mean Absolute Error
maeCv = sum(abs(yObs-yPredCv))/nrow(inData)

# Calculate the CV Mean Percentage Absolute Error
mapeCv = (100/nrow(inData))*sum((abs(yObs - yPredCv))/yObs)

# Calculate the modelRMSE of prediction error
rmseModel = sqrt(sum((yObs - yPredModel)^2)/nrow(inData))

# Calculate the model Mean Absolute Error
maeModel = sum(abs(yObs-yPredModel))/nrow(inData)

# Calculate the model Mean Percentage Absolute Error
mapeModel = (100/nrow(inData))*sum((abs(yObs - yPredModel))/yObs)

r2Cv = cor(cbind(yObs, yPredCv))**2
r2Cv = r2Cv[1,2]
r2Model = summary(inModel)$r.squared
r2Diff = abs(r2Cv - r2Model)

cvParams = vector("list", 8)
cvParams[[1]] = rmseCv
cvParams[[2]] = maeCv
cvParams[[3]] = mapeCv
cvParams[[4]] = rmseModel
cvParams[[5]] = maeModel
cvParams[[6]] = mapeModel
cvParams[[7]] = r2Cv
cvParams[[8]] = r2Diff

# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("", quote=FALSE)
print(paste("Number of folds performed: ", numFolds, sep=""), quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
print("Original Model Root Mean Square of Prediction Data (RMSE):", quote=FALSE)
print("", quote=FALSE)
print(rmseModel, quote=FALSE)
print("", quote=FALSE)
print("Original Model Mean Absolute Error of Predicted Data (MAE):", quote=FALSE)
print("", quote=FALSE)
print(maeModel, quote=FALSE)
print("", quote=FALSE)
```

```
 print("Original Model Mean Percentage Absolute Error of Predicted Data (MAPE):", quote=FALSE)
 print("", quote=FALSE)
 print(mapeModel, quote=FALSE)
 print("", quote=FALSE)
 print("", quote=FALSE)
 print("Cross Validation Root Mean Square of Prediction Data (RMSE):", quote=FALSE)
 print("", quote=FALSE)
 print(rmseCv, quote=FALSE)
 print("", quote=FALSE)
 print("Cross Validation Mean Absolute Error of Predicted Data (MAE):", quote=FALSE)
 print("", quote=FALSE)
 print(maeCv, quote=FALSE)
 print("", quote=FALSE)
 print("Cross Validation Mean Percentage Absolute Error of Predicted Data (MAPE):", quote=FALSE)
 print("", quote=FALSE)
 print(mapeCv, quote=FALSE)
 print("", quote=FALSE)
 print("", quote=FALSE)
 print("Cross Validation R-Squared:", quote=FALSE)
 print("", quote=FALSE)
 print(r2Cv, quote=FALSE)
 print("", quote=FALSE)
 print("Model R-Squared:", quote=FALSE)
 print("", quote=FALSE)
 print(r2Model, quote=FALSE)
 print("", quote=FALSE)
 print("Model R-Squared and Cross Validation R-Squared Difference:", quote=FALSE)
 print("", quote=FALSE)
 print(r2Diff, quote=FALSE)
 print("", quote=FALSE)
 sink()

 crossVal$cvresidual = yObs - yPredCv

 fileName = paste(basePath, modelDesc, "_CrossValResVsPred.jpeg", sep="")
 plotTitle = paste("Cross Validation Residuals Vs Predicted")
 jpeg(file=fileName)
 plot(crossVal$Predicted, crossVal$cvresidual, main=plotTitle, xlab="Model Predicted Values",
    ylab="Cross Validation Prediction Residuals")
 dev.off()

 close(outFile)
 closeAllConnections()

 return(cvParams)

}
```

validate_data.R File:

```
#########################################################################################################################
#############
#
#  validate_data.R
#
#  This file performs validation on the model and training and testing data passed in then generates the RMSE, MAPE, MAE, and R2
#  difference R outputs.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#########################################################################################################################
#############

validate_data <- function(inModel, trainData, testData, currentY, percentTest, validateTest, varInfo, basePath, modelDesc){
```

```
# Create output file
outFileName = paste(basePath, modelDesc, "_Validation_R_Output.txt", sep="")
outFile = file(outFileName, open="wt")


 # Get Training Set Predicted Data
yPredTrain = fitted(inModel)
yObsTrain = trainData[, currentY]

# Calculate the Training Set RMSE of prediction error
rmseTrain = sqrt(sum((yObsTrain - yPredTrain)^2)/nrow(trainData))

# Calculate Training Set Mean Absolute Error
maeTrain = sum(abs(yObsTrain-yPredTrain))/nrow(trainData)

# Calculate Training Set Mean Percentage Absolute ERror
mapeTrain = (100/nrow(trainData))*sum((abs(yObsTrain - yPredTrain))/yObsTrain)


valParams = vector("list", 8)
valParams[[1]] = rmseTrain
valParams[[2]] = maeTrain
valParams[[3]] = mapeTrain

if(validateTest == TRUE){
  # Get Testing Set Predicted Data
  yPredTest = predict(inModel, testData)
  yObsTest = testData[, currentY]

  # Calculate the Testing Set RMSE of prediction error
  rmseTest = sqrt(sum((yObsTest - yPredTest)^2)/nrow(testData))

  # Calculate Testing Set Mean Absolute Error
  maeTest = sum(abs(yObsTest-yPredTest))/nrow(testData)

  # Calculate Testing Set Mean Percentage Absolute ERror
  mapeTest = (100/nrow(testData))*sum((abs(yObsTest - yPredTest))/yObsTest)

  r2Test = cor(cbind(yObsTest, yPredTest))**2
  r2Test = r2Test[1,2]
  r2Train = summary(inModel)$r.squared
  r2Diff = abs(r2Test - r2Train)

  valParams[[4]] = rmseTest
  valParams[[5]] = maeTest
  valParams[[6]] = mapeTest

  valParams[[7]] = r2Test
  valParams[[8]] = r2Diff

}else{
  valParams[[4]] = NA
  valParams[[5]] = NA
  valParams[[6]] = NA

  valParams[[7]] = NA
  valParams[[8]] = NA

}


# Calculate Testing Set validated R^2 and Testing Set validated/training R^2 difference
if(validateTest == TRUE){
}
```

```
 # Print cp model selection values to output file
 sink(file=outFile, append=TRUE)
 print(paste("Training Set Percentage Of Original Data: ", (100-percentTest), " %, Testing Set Percentage of Original Data: ", percentTest,
         " %", sep=""), quote=FALSE)
 print("", quote=FALSE)
 print("", quote=FALSE)
 print("Training Set Root Mean Square of Prediction Data (RMSE):", quote=FALSE)
 print("", quote=FALSE)
 print(rmseTrain, quote=FALSE)
 print("", quote=FALSE)
 print("Training Set Mean Absolute Error of Predicted Data (MAE):", quote=FALSE)
 print("", quote=FALSE)
 print(maeTrain, quote=FALSE)
 print("", quote=FALSE)
 print("Training Set Mean Percentage Absolute Error of Predicted Data (MAPE):", quote=FALSE)
 print("", quote=FALSE)
 print(mapeTrain, quote=FALSE)
 print("", quote=FALSE)
 print("", quote=FALSE)
 if(validateTest == TRUE){
   print("Testing Set Root Mean Square of Prediction Data (RMSE):", quote=FALSE)
   print("", quote=FALSE)
   print(rmseTest, quote=FALSE)
   print("", quote=FALSE)
   print("Testing Set Mean Absolute Error of Predicted Data (MAE):", quote=FALSE)
   print("", quote=FALSE)
   print(maeTest, quote=FALSE)
   print("", quote=FALSE)
   print("Testing Set Mean Percentage Absolute Error of Predicted Data (MAPE):", quote=FALSE)
   print("", quote=FALSE)
   print(mapeTest, quote=FALSE)
   print("", quote=FALSE)
   print("", quote=FALSE)
   print("Testing Set R-Squared:", quote=FALSE)
   print("", quote=FALSE)
   print(r2Test, quote=FALSE)
   print("", quote=FALSE)
   print("Training Set Model R-Squared:", quote=FALSE)
   print("", quote=FALSE)
   print(r2Train, quote=FALSE)
   print("", quote=FALSE)
   print("Training Set R-Squared and Testing Set R-Squared Difference:", quote=FALSE)
   print("", quote=FALSE)
   print(r2Diff, quote=FALSE)
   print("", quote=FALSE)
 }
 sink()


 close(outFile)
 closeAllConnections()

 return(valParams)

}
```

get_outliers.R File:

```
#########################################################################################################################
#############
#
#  get_outliers.R
#
#  This file finds the rows of the data set passed in that contain the outlierCond passed in the columns specified by the
```

```
#  checkVars var.  The rows that match these conditions are then removed from the data set.  The rows removed are printed out.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#########################################################################################################################################
#############

get_outliers <- function(inData, outlierValue, outlierCond, checkVars, outPath, dataDesc){

 # Create output file
 upOutlierCond = toupper(outlierCond)
 outlierValStr = toString(outlierValue)
 outlierValPStr = sub("\\.", "p", outlierValStr)

 dir.create(file.path(outPath, dataDesc), showWarnings = FALSE)
 basePath = paste(outPath, dataDesc, "\\", sep="")
 outFileName = paste(basePath, dataDesc, "_", upOutlierCond, "_", outlierValPStr, "_Outliers_R_Output.txt", sep="")
 outFile = file(outFileName, open="wt")


 # Create Box Cox formula string
 numVars = length(checkVars)
 outData = inData
 allMatchNdx = c()
 for (i in 1:numVars){

  var = checkVars[i]

  if(upOutlierCond == "GTE"){
    matchVals = inData[, var] >= outlierValue
  }else if(upOutlierCond == "GT"){
    matchVals = inData[, var] > outlierValue
  }else if(upOutlierCond == "LTE"){
    matchVals = inData[, var] <= outlierValue
  }else if(upOutlierCond == "LT"){
    matchVals = inData[, var] < outlierValue
  }else if(upOutlierCond == "EQ"){
    matchVals = inData[, var] == outlierValue
  }

  matchNdx = which(matchVals %in% TRUE)

  if(length(inData[matchNdx,var]) > 0){
    allMatchNdx = append(allMatchNdx, matchNdx)
    allMatchNdx = unique(allMatchNdx)
  }
 }

 numOutPresent = length(allMatchNdx)
 numRowsOut = nrow(inData) - length(allMatchNdx)
 perRmv = (numOutPresent/numRowsOut)*100

 if(upOutlierCond == "GTE"){
  condStr = "greater than or equal to"
 }else if(upOutlierCond == "GT"){
  condStr = "greater than"
 }else if(upOutlierCond == "LTE"){
  condStr = "less than or equal to"
 }else if(upOutlierCond == "LT"){
  condStr = "less than"
 }else if(upOutlierCond == "EQ"){
  condStr = "equal to"
 }

 # Print outliers found
```

```
 sink(file=outFile, append=TRUE)
 print("", quote=FALSE)
 print(paste("Number of outliers found ", condStr, " value = ", outlierValStr, ": ", numOutPresent, sep=""), quote=FALSE)
 print("", quote=FALSE)
 print(paste("Number of remaining rows in the dataset: ", numRowsOut, sep=""), quote=FALSE)
 print("", quote=FALSE)
 print(paste("Percentage of dataset removed: ", format(round(perRmv, 2), nsmall = 2), " %", sep=""), quote=FALSE)
 print("", quote=FALSE)
 print("", quote=FALSE)
 print("Rows removed: ", quote=FALSE)
 print("", quote=FALSE)
 print(inData[allMatchNdx,], quote=FALSE)
 print("", quote=FALSE)
 sink()

 outData = outData[-allMatchNdx,]

 close(outFile)
 closeAllConnections()

 return(outData)

}
```

remove_inf_pts.R File:

```
#############################################################################################################################
#############
#
#  remove_inf_pts.R
#
#  This file prints out the rows of the data set given by the infPtsNdx passed in, then removes the rows from the data set.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#############################################################################################################################
#############

remove_inf_pts <- function(inData, infPtsNdx, outPath, ptDesc){

 # Create output file
 outFileName = paste(basePath, "_Remove_", ptDesc, "_R_Output.txt", sep="")
 outFile = file(outFileName, open="wt")

 numOutPresent = length(infPtsNdx)
 numRowsOut = nrow(inData) - length(infPtsNdx)
 perRmv = (numOutPresent/numRowsOut)*100

 # Print outliers found
 sink(file=outFile, append=TRUE)
 print("", quote=FALSE)
 print(paste("Number of influential points found: ", numOutPresent, sep=""), quote=FALSE)
 print("", quote=FALSE)
 print(paste("Number of remaining rows in the dataset: ", numRowsOut, sep=""), quote=FALSE)
 print("", quote=FALSE)
 print(paste("Percentage of dataset removed: ", format(round(perRmv, 2), nsmall = 2), " %", sep=""), quote=FALSE)
 print("", quote=FALSE)
 print("", quote=FALSE)
 print("Rows removed: ", quote=FALSE)
 print("", quote=FALSE)
 print(inData[infPtsNdx,], quote=FALSE)
 print("", quote=FALSE)
 sink()
```

```
 close(outFile)
 closeAllConnections()

 outData = inData[-infPtsNdx,]

 return(outData)

}
```

predict_data.R File:

```
#########################################################################################################################################
#############
#
#  predict_data.R
#
#  This file performs the prediction of dependent data based on the predicted data and model passed in.  It generates the
#  predicted value, std error of prediction, prediction interval, and confidence interval, then applies the inverse Box Cox to
#  all of them except the std error.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#########################################################################################################################################
#############

predict_values <- function(inModel, predictData, sigLevel, yLambda, basePath, modelDesc){

 # Create output file
 outFileName = paste(basePath, modelDesc, "_PredictVals_R_Output.txt", sep="")
 outFile = file(outFileName, open="wt")

 # Perform Prediction
 predSE = predict(inModel, predictData, se.fit=T, level=sigLevel)
 predPredInt = predict(inModel, predictData, interval="prediction", level=sigLevel)
 predConfInt = predict(inModel, predictData, interval="confidence", level=sigLevel)

 se.CI <- predSE$se.fit
 se.PI <- sqrt(predSE$se.fit^2 + predSE$residual.scale^2)
 alpha <- qt((1-sigLevel)/2, df = predSE$df)

 calcConfIntVals = matrix(NA, nrow=length(predSE$fit), ncol=2)
 calcPredIntVals = matrix(NA, nrow=length(predSE$fit), ncol=2)
 for(i in 1:length(predSE$fit)){
  tempCI = predSE$fit[i] + c(alpha, -alpha) * se.CI
  calcConfIntVals[i,1] = tempCI[1]
  calcConfIntVals[i,2] = tempCI[2]

  tempCI = predSE$fit[i] + c(alpha, -alpha) * se.PI
  calcPredIntVals[i,1] = tempCI[1]
  calcPredIntVals[i,2] = tempCI[2]

 }

 # Transformed prediction values and intervals
 transPred = ((predSE$fit*yLambda)+1)^(1/yLambda)
 transCI = ((calcConfIntVals*yLambda)+1)^(1/yLambda)
 transPI = ((calcPredIntVals*yLambda)+1)^(1/yLambda)

 # Print prediction values to output file
 sink(file=outFile, append=TRUE)
 print("", quote=FALSE)
 print("Output of Prediction with Standard Error Values:", quote=FALSE)
 print("", quote=FALSE)
 print(predSE, quote=FALSE)
```

```
 print("", quote=FALSE)
 print("Output of Prediction with Prediction Interval:", quote=FALSE)
 print("", quote=FALSE)
 print(predPredInt, quote=FALSE)
 print("", quote=FALSE)
 print("Output of Prediction with Confidence Interval:", quote=FALSE)
 print("", quote=FALSE)
 print(predConfInt, quote=FALSE)
 print("", quote=FALSE)
 print("Output of Calculated Prediction Interval:", quote=FALSE)
 print("", quote=FALSE)
 print(calcPredIntVals, quote=FALSE)
 print("", quote=FALSE)
 print("Output of Calculated Confidence Interval:", quote=FALSE)
 print("", quote=FALSE)
 print(calcConfIntVals, quote=FALSE)
 print("", quote=FALSE)
 print("Output of Inverted Y Box Cox Transform:", quote=FALSE)
 print("", quote=FALSE)
 print(transPred, quote=FALSE)
 print("", quote=FALSE)
 print("Output of Inverted CI Box Cox Transform:", quote=FALSE)
 print("", quote=FALSE)
 print(transCI, quote=FALSE)
 print("", quote=FALSE)
 print("Output of Inverted PI Box Cox Transform:", quote=FALSE)
 print("", quote=FALSE)
 print(transPI, quote=FALSE)
 print("", quote=FALSE)
 sink()

 close(outFile)
 closeAllConnections()

 return(TRUE)

}
```

Project Orig Vars Rev.R File (largest VIF over 10 first, then larges p value over 0.05 elimination code):

```
#############################################################################################################################
#############
#
#  Project Orig Vars Rev.R
#
#  This file contains the high level code used in creating intermediate models.  The first model created uses all the indep vars.
#  The second model removes influential points.  The third model removes extreme outliers.  Next models are made starting at the
#  backward and stepwise model selection recommendations with the regular Rent variable.  After that, the Box Cox of Rent is
#  applied to the data after the extreme outliers were removed.  After that, models starting with the backward and stepwise
#  model selection recommendations with the Box Cox of the Rent variable.  During the creation of these models, first the coeffs
#  with the coeffs with the max VIF over 10 are eliminated, then the max p value above 0.05.  After the backward and stepwise
#  models of both Rent and Box Cox Rent are generated, the data set is divided into training and testing sets and a new model is
#  generated for each combo (Rent backward, Rent stepwise, Box Cox Rent backward, Box Cox Rent stepwise) using the training set.
#  Validatio is then done using the testing set.  Finally, new data points are imported and used to generate predictions for each
#  model combination.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#############################################################################################################################
#############

library(psych)
library(QuantPsyc)
library(car)
```

```
library(leaps)
library(DAAG)

# Assign output file directory based on relative path of this R script
mainPath = getSrcDirectory(function(x) {x})
mainPath = gsub("/", "\\\\", mainPath)
mainPath = paste(mainPath, '\\', sep="")
dir.create(file.path(mainPath, "R Anlys Fnl O Rev2"), showWarnings = FALSE)
anlysPath = paste(mainPath, "R Anlys Fnl O Rev2\\", sep="")

# Get subfunctions required
source(paste(mainPath, "Common R Functions\\analyze_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\explore_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\select_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\validate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\crossvalidate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\get_outliers.R", sep=""))
source(paste(mainPath, "Common R Functions\\model_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\remove_inf_pts.R", sep=""))
source(paste(mainPath, "Common R Functions\\predict_values.R", sep=""))

# Import Dataset
rentData = read.table(paste(mainPath, "\\Chicago_rent\\ChicagoProject.txt", sep=""),
                header=T)

###################################### Variable Initialization
#########################################################################

varInfo.xVarNames = c("mdfamy0", "asianp", "whitenhp", "aanhp", "hisp", "renter", "educ8",
                "educ160pro", "Noheat", "totcrime", "murder", "robbery", "personal", "property", "forecl")
varInfo.xVarPlotNames = c("Median Family Income", "Prop of Asian American",
                  "Prop of Non-Hispanic White", "Prop of Non-Hispanic African American",
                  "Prop Hispanic American", "Prop of Renter Occupied", "Prop of With 8 Yrs School",
                  "Prop With Bachelor or Grad School", "Prop With No Heat", "Prop Total Crime",
                  "Prop Murder", "Prop Robbery", "Prop Personal Crime", "Prop Property Crime",
                  "Prop Foreclosures")
varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

percTestPts = 25
percCrossValPts = 20
highRentLimit =  720
lowRentLimit = 450
bcHighRentLimit = 98
bcLowRentLimit = 73

outlierLevel = 5

###################################### Dataset Conversion
#########################################################################

dir.create(file.path(anlysPath, "Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "Init\\", sep="")

# Change the data frame columns from factor type to character type
i <- sapply(rentData, is.factor)
rentData[i] <- lapply(rentData[i], as.character)

# Remove any missing points from the dataset
outData = get_outliers(rentData, "-", "EQ", varInfo.xVarNames, basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}
```

```
# Change all data frame columns from character to numeric (except for the column of neighborhood names)
totVars = append(varInfo.yVarName, varInfo.xVarNames)
numVars = length(totVars)
for (i in 1:numVars){
  var = totVars[i]
  if (is.character(var)){
    rentData[var] = as.numeric(rentData[,var])
  }
}

################################################  Outlier Removal
#####################################################################

# Remove any 0 Rent entries since 0 rent doesn't make logical sense and can sway the data
outData = get_outliers(rentData, 0, "LTE", c("Rent"), basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}


######################################### Neighborhood Dummy Variable Creation
#########################################################

allVarInfo.xVarNames = varInfo.xVarNames
allVarInfo.xVarPlotNames = varInfo.xVarPlotNames

uniqAreas = unique(rentData$canam77)
numAreas = length(uniqAreas)
flagNames = c()
flagDescs = c()
for(i in 1:(numAreas-1)){
  val = uniqAreas[i]
  lowVal = tolower(val)
  matchNdx = which(rentData$canam77 %in% val)

  if(length(matchNdx) > 0){
    temp = gsub("-", "_", lowVal)
    temp2 = gsub("'", "", temp)
    newVal = paste(temp2, "Flag", sep="")

    tempFlag = (rentData$canam77 == val)*1
    rentData[newVal] = tempFlag

    flagNames = append(flagNames, newVal)
    flagDescs = append(flagDescs, val)
  }
}

numFlagVars = length(flagNames)
for(i in 1:numFlagVars){
  allVarInfo.xVarNames = append(allVarInfo.xVarNames, flagNames[i])
  allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, flagDescs[i])
}

########################################### Assign x variables for functions
#############################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

########################################### Create Box Cox of Rent
#####################################################################

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
```

```
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
    xformStr = var
  }else{
    xformStr = paste(xformStr, "+", var, sep="")
  }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda = transInfo$result[1]

if(yLambda > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda) - 1)/yLambda
}else{
  rentData$bcRent = log(rentData$Rent)
}


########################################## Write dataset to a file for other group members
##########################################

write.table(rentData, paste(mainPath, "ChicagoProject_WithDummyNoBadPts.txt", sep=""), sep="\t", row.names=FALSE)

########################################## All Var Modelling
#############################################################################

dir.create(file.path(anlysPath, "M1 All Vars"), showWarnings = FALSE)
basePath = paste(anlysPath, "M1 All Vars\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
baseData = data.frame()

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")


########################################## Remove Influential Pts
#############################################################################

dir.create(file.path(anlysPath, "M2 All Vars No Inf Pts"), showWarnings = FALSE)
basePath = paste(anlysPath, "M2 All Vars No Inf Pts\\", sep="")

infPtsNdx = modelOutParams[[1]]
if(length(infPtsNdx) > 0){

  outData = remove_inf_pts(rentData, infPtsNdx, basePath, "InfPts")
  rentData = outData
}

########################################## All Var Modelling w/o Inf Pts
#############################################################
```

```
# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

########################################## Remove Extreme Outlier Pts
#####################################################################

dir.create(file.path(anlysPath, "M3 All Vars No Out"), showWarnings = FALSE)
basePath = paste(anlysPath, "M3 All Vars No Out\\", sep="")

infPtsNdx = modelOutParams[[2]]
if(length(infPtsNdx) > 0){

 outData = remove_inf_pts(rentData, infPtsNdx, basePath, "ExtremeOut")
 rentData = outData
}


########################################## All Var Modelling w/o Outliers
#####################################################################

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

selNames = modelOutParams[[9]]
backTempNames = selNames[[1]]
stepTempNames = selNames[[2]]

########################################## Backward Sel Modelling
#####################################################################

tempLength = length(backTempNames)
backVarInfo.xVarNames = backTempNames[2:tempLength]
numBackVars = length(backVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numBackVars){
 var = backVarInfo.xVarNames[i]
 matchNdx = which(backVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

backVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M4 Back Init"), showWarnings = FALSE)
```

```
basePath = paste(anlysPath, "M4 Back Init\\", sep="")

varInfo.xVarNames = backVarInfo.xVarNames
varInfo.xVarPlotNames = backVarInfo.xVarPlotNames

maxPVal = 1
maxVIF = 20
modelNdx = 5
backCoeffsRmvd = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

 if (maxVIF > 10){
  tempVals = vifVals[,1]
  maxVifNdx = which(tempVals %in% maxVIF)
  vifNames = rownames(vifVals)
  maxVifName = vifNames[maxVifNdx]
  backCoeffsRmvd = append(backCoeffsRmvd, maxVifName)

  matchNdx = which(varInfo.xVarNames %in% maxVifName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1

 }else if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  backCoeffsRmvd = append(backCoeffsRmvd, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }
}

finalBackVarInfo.xVarNames = varInfo.xVarNames
finalBackVarInfo.xVarPlotNames = varInfo.xVarPlotNames

######################################### Stepwise Sel Modelling
#########################################################################

tempLength = length(stepTempNames)
stepVarInfo.xVarNames = stepTempNames[2:tempLength]
numStepVars = length(stepVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numStepVars){
```

```
 var = stepVarInfo.xVarNames[i]
 matchNdx = which(stepVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

stepVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M4 Step Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 Step Init\\", sep="")

varInfo.xVarNames = stepVarInfo.xVarNames
varInfo.xVarPlotNames = stepVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 5
maxVIF = 20
stepCoeffsRmvd = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

 if (maxVIF > 10){
  tempVals = vifVals[,1]
  maxVifNdx = which(tempVals %in% maxVIF)
  vifNames = rownames(vifVals)
  maxVifName = vifNames[maxVifNdx]
  stepCoeffsRmvd = append(stepCoeffsRmvd, maxVifName)

  matchNdx = which(varInfo.xVarNames %in% maxVifName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1

 }else if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  stepCoeffsRmvd = append(stepCoeffsRmvd, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
```

```
   basePath = paste(anlysPath, folderName, "\\", sep="")

   modelNdx = modelNdx + 1
  }
}

finalStepVarInfo.xVarNames = varInfo.xVarNames
finalStepVarInfo.xVarPlotNames = varInfo.xVarPlotNames

########################################## Box Cox All Var Modelling
#####################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

dir.create(file.path(anlysPath, "M4 All Vars BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 All Vars BC\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

selNames = modelOutParams[[9]]
backTempNames = selNames[[1]]
stepTempNames = selNames[[2]]

########################################## Backward Sel Modelling
#####################################################################

tempLength = length(backTempNames)
backVarInfo.xVarNames = backTempNames[2:tempLength]
numBackVars = length(backVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numBackVars){
  var = backVarInfo.xVarNames[i]
  matchNdx = which(backVarInfo.xVarNames %in% var)
  tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

backVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M5 Back BC Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M5 Back BC Init\\", sep="")

varInfo.xVarNames = backVarInfo.xVarNames
varInfo.xVarPlotNames = backVarInfo.xVarPlotNames

maxPVal = 1
```

```
modelNdx = 6
maxVIF = 20
backCoeffsRmvdBc = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

 if (maxVIF > 10){
  tempVals = vifVals[,1]
  maxVifNdx = which(tempVals %in% maxVIF)
  vifNames = rownames(vifVals)
  maxVifName = vifNames[maxVifNdx]
  backCoeffsRmvdBc = append(backCoeffsRmvdBc, maxVifName)

  matchNdx = which(varInfo.xVarNames %in% maxVifName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1

 }else if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  backCoeffsRmvdBc = append(backCoeffsRmvdBc, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }
}

finalBackBcVarInfo.xVarNames = varInfo.xVarNames
finalBackBcVarInfo.xVarPlotNames = varInfo.xVarPlotNames

######################################### Stepwise Sel Modelling
#########################################################################

tempLength = length(stepTempNames)
stepVarInfo.xVarNames = stepTempNames[2:tempLength]
numStepVars = length(stepVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numStepVars){
 var = stepVarInfo.xVarNames[i]
 matchNdx = which(stepVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}
```

```
stepVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M5 Step BC Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M5 Step BC Init\\", sep="")

varInfo.xVarNames = stepVarInfo.xVarNames
varInfo.xVarPlotNames = stepVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 6
maxVIF = 20
stepCoeffsRmvdBc = c()
while((maxPVal > 0.05) | (maxVIF > 10)){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 vifVals = data.frame(modelOutParams[[8]])
 maxVIF = max(vifVals)

 if (maxVIF > 10){
  tempVals = vifVals[,1]
  maxVifNdx = which(tempVals %in% maxVIF)
  vifNames = rownames(vifVals)
  maxVifName = vifNames[maxVifNdx]
  stepCoeffsRmvdBc = append(stepCoeffsRmvdBc, maxVifName)

  matchNdx = which(varInfo.xVarNames %in% maxVifName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxVifName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1

 }else if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  stepCoeffsRmvdBc = append(stepCoeffsRmvdBc, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }
}
```

```
finalStepBcVarInfo.xVarNames = varInfo.xVarNames
finalStepBcVarInfo.xVarPlotNames = varInfo.xVarPlotNames

######################################## Create Training and Testing Datasets
#########################################################

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

# Create sample indices
selectNdx = sample(1:nrow(rentData), (1-(percTestPts/100))*nrow(rentData))

# Select the training dataset
trainData = rentData[selectNdx, ]

# Select the testing dataset
testData = rentData[-selectNdx, ]

testSetDiff = setdiff(testData$canam77, trainData$canam77)
numDiff = length(testSetDiff)
numOrigRows = length(rentData$canam77)
allMatchNdx = c()
if(numDiff >0){
 for(i in 1:numDiff){
  val = testSetDiff[i]

  matchNdx = which(testData$canam77 %in% val)

  numMatches = length(matchNdx)
  if(matchNdx > 0){
   allMatchNdx = append(allMatchNdx, matchNdx)
  }
 }

 numAllMatches = length(allMatchNdx)
 percMatches = (numAllMatches / numOrigRows) * 100

 if(percMatches < 5){
  tempData = testData[allMatchNdx,]
  trainData = rbind(trainData, setNames(tempData, names(trainData)))
  testData = testData[-allMatchNdx,]
 }
}


testSetDiff = setdiff(testData$canam77, trainData$canam77)
trainSetDiff = setdiff(trainData$canam77, testData$canam77)
trainDataPerc = (length(trainData$canam77)/numOrigRows)*100
testDataPerc = (length(testData$canam77)/numOrigRows)*100


outFileName = paste(anlysPath, "DataSet_Split_Information.txt", sep="")
outFile = file(outFileName, open="wt")

# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Neighborhoods in Testing data that are not in Training data:", quote=FALSE)
print("", quote=FALSE)
print(testSetDiff, quote=FALSE)
print("", quote=FALSE)
print("Neighborhoods in Training data that are not in Testing data:", quote=FALSE)
print("", quote=FALSE)
print(trainSetDiff, quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
```

```
print("Percentage of Original data used for Training data:", quote=FALSE)
print("", quote=FALSE)
print(round(trainDataPerc, digits=2), quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
print("Percentage of Original data used for Training data:", quote=FALSE)
print("", quote=FALSE)
print(round(testDataPerc, digits=2), quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
sink()

close(outFile)
closeAllConnections()


######################################### Backward Train/Test Modelling
################################################################

dir.create(file.path(anlysPath, "M Last Back"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back\\", sep="")

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

varInfo.xVarNames = finalBackVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

######################################### Stepwise Train/Test Modelling
################################################################

dir.create(file.path(anlysPath, "M Last Step"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step\\", sep="")

varInfo.xVarNames = finalStepVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

######################################### Backward BC Train/Test Modelling
################################################################

dir.create(file.path(anlysPath, "M Last Back BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back BC\\", sep="")

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"
```

```
varInfo.xVarNames = finalBackBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackBcVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Stepwise BC Train/Test Modelling
#########################################################

dir.create(file.path(anlysPath, "M Last Step BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step BC\\", sep="")

varInfo.xVarNames = finalStepBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepBcVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modelling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Create Prediction Points
#########################################################################

predictData = read.table(paste(mainPath, "Rent_PredictionData.txt", sep=""), header=T)

# Change the data frame columns from factor type to character type
i <- sapply(predictData, is.factor)
predictData[i] <- lapply(predictData[i], as.character)

# Change all data frame columns from character to numeric (except for the column of neighborhood names)
totVars = length(colnames(predictData))
numVars = length(totVars)
for (i in 1:numVars){
 var = totVars[i]
 if (is.character(var)){
  predictData[var] = as.numeric(predictData[,var])
 }
}

uniqAreas = unique(rentData$canam77)
numAreas = length(uniqAreas)
for(i in 1:(numAreas-1)){
 val = uniqAreas[i]
 lowVal = tolower(val)

 temp = gsub("-", "_", lowVal)
 temp2 = gsub("'", "", temp)
 newVal = paste(temp2, "Flag", sep="")

 tempFlag = (predictData$canam77 == val)*1
 predictData[newVal] = tempFlag
}
```

```
predictData["Rent"] = NULL

########################################### Predict Points With Backward Model
#########################################################

dir.create(file.path(anlysPath, "M Last Back"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back\\", sep="")

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

varInfo.xVarNames = finalBackVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
  xformStr = var
 }else{
  xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

backModel = lm(modelForm, data=rentData)

passFlag = predict_values(backModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################### Predict Points With Stepwise Model
#########################################################

dir.create(file.path(anlysPath, "M Last Step"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step\\", sep="")

varInfo.xVarNames = finalStepVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
  xformStr = var
 }else{
  xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

stepModel = lm(modelForm, data=rentData)

passFlag = predict_values(stepModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################### Backward BC Train/Test Modelling
#############################################################

dir.create(file.path(anlysPath, "M Last Back BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back BC\\", sep="")

varInfo.yVarName = "bcRent"
```

```
varInfo.yVarPlotName = "BoxCox of Rent"

varInfo.xVarNames = finalBackBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackBcVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
  xformStr = var
 }else{
  xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

backBcModel = lm(modelForm, data=rentData)

passFlag = predict_values(backBcModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################## Stepwise BC Train/Test Modelling
#############################################################

dir.create(file.path(anlysPath, "M Last Step BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step BC\\", sep="")

varInfo.xVarNames = finalStepBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepBcVarInfo.xVarPlotNames

numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
  xformStr = var
 }else{
  xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

stepBcModel = lm(modelForm, data=rentData)

passFlag = predict_values(stepBcModel, predictData, 0.95, yLambda, basePath, "AllData")

########################################## Create Output Model Sel Info File
#############################################################

# Create output file
outFileName = paste(anlysPath, "Model_Sel_Info_R_Output.txt", sep="")
outFile = file(outFileName, open="wt")

# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Variables removed during backward selection of Rent (in order or removal):", quote=FALSE)
print("", quote=FALSE)
print(backCoeffsRmvd, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during stepwise selection of Rent (in order or removal):", quote=FALSE)
print(stepCoeffsRmvd, quote=FALSE)
```

```
print("", quote=FALSE)
print("Variables removed during backward selection of Box Cox Rent (in order or removal):", quote=FALSE)
print("", quote=FALSE)
print(backCoeffsRmvdBc, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during stepwise selection of Box Cox Rent (in order or removal):", quote=FALSE)
print(stepCoeffsRmvdBc, quote=FALSE)
print("", quote=FALSE)
sink()

close(outFile)
closeAllConnections()
```

Project Initial.R File (used for initial exploratory and model analysis):

```
#############################################################################################################################
#############
#
#  Project Initial.R
#
#  This file performs exploratory analysis on the entire data set with the regular Rent dependent variable, Rent divided by
#  1000, and Box Cox of Rent.  Models are then created with all three of these dependent variables, with and without influential
#  points and extreme outliers.  Three additional sets of variables are then created (one with dummy vars for non-white and
#  non-african america, one with dummy vars for high rent and low rent, and one with the rent affordability index).  The Box Cox
#  transforms are then generated for all combinations of these new variables and the lambda values generated are printed out.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#############################################################################################################################
#############

library(psych)
library(QuantPsyc)
library(car)
library(leaps)
library(DAAG)

# Assign output file directory based on relative path of this R script
mainPath = getSrcDirectory(function(x) {x})
mainPath = gsub("/", "\\\\", mainPath)
mainPath = paste(mainPath, '\\', sep="")
dir.create(file.path(mainPath, "R Anlys Init"), showWarnings = FALSE)
anlysPath = paste(mainPath, "R Anlys Init\\", sep="")

# Get subfunctions required
source(paste(mainPath, "Common R Functions\\analyze_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\explore_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\select_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\validate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\crossvalidate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\get_outliers.R", sep=""))
source(paste(mainPath, "Common R Functions\\model_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\remove_inf_pts.R", sep=""))

# Import Dataset
rentData = read.table(paste(mainPath, "\\Chicago_rent\\ChicagoProject.txt", sep=""),
                      header=T)

#################################### Variable Initialization
#####################################################################

varInfo.xVarNames = c("mdfamy0", "asianp", "whitenhp", "aanhp", "hisp", "renter", "educ8",
```

```
                "educ160pro", "Noheat", "totcrime", "murder", "robbery", "personal", "property", "forecl")
varInfo.xVarPlotNames = c("Median Family Income", "Prop of Asian American",
                  "Prop of Non-Hispanic White", "Prop of Non-Hispanic African American",
                  "Prop Hispanic American", "Prop of Renter Occupied", "Prop of With 8 Yrs School",
                  "Prop With Bachelor or Grad School", "Prop With No Heat", "Prop Total Crime",
                  "Prop Murder", "Prop Robbery", "Prop Personal Crime", "Prop Property Crime",
                  "Prop Foreclosures")
varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

percTestPts = 25
percCrossValPts = 20
highRentLimit =  720
lowRentLimit = 450
bcHighRentLimit = 98
bcLowRentLimit = 73

outlierLevel = 5

bcLambdas = c()
bcDescs = c()

######################################## Dataset Conversion
#############################################################################

dir.create(file.path(anlysPath, "Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "Init\\", sep="")

# Change the data frame columns from factor type to character type
i <- sapply(rentData, is.factor)
rentData[i] <- lapply(rentData[i], as.character)

# Remove any missing points from the dataset
outData = get_outliers(rentData, "-", "EQ", varInfo.xVarNames, basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}


# Change all data frame columns from character to numeric (except for the column of neighborhood names)
totVars = append(varInfo.yVarName, varInfo.xVarNames)
numVars = length(totVars)
for (i in 1:numVars){
  var = totVars[i]
  if (is.character(var)){
    rentData[var] = as.numeric(rentData[,var])
  }
}

######################################### Neighborhood Dummy Variable Creation
#########################################################

allVarInfo.xVarNames = varInfo.xVarNames
allVarInfo.xVarPlotNames = varInfo.xVarPlotNames

uniqAreas = unique(rentData$canam77)
numAreas = length(uniqAreas)
flagNames = c()
flagDescs = c()
for(i in 1:(numAreas-1)){
  val = uniqAreas[i]
  lowVal = tolower(val)
  matchNdx = which(rentData$canam77 %in% val)

  if(length(matchNdx) > 0){
```

```
   temp = gsub("-", "_", lowVal)
   temp2 = gsub("'", "", temp)
   newVal = paste(temp2, "Flag", sep="")

   tempFlag = (rentData$canam77 == val)*1
   rentData[newVal] = tempFlag

   flagNames = append(flagNames, newVal)
   flagDescs = append(flagDescs, val)
  }
}

numFlagVars = length(flagNames)
for(i in 1:numFlagVars){
  allVarInfo.xVarNames = append(allVarInfo.xVarNames, flagNames[i])
  allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, flagDescs[i])
}

########################################## Perform Exploratory Analysis on Whole Dataset
################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Perform Exploratory Data Analysis on entire dataset
explorePath = explore_data(rentData, varInfo.yVarName, varInfo.yVarPlotName, varInfo, TRUE, TRUE, basePath, "Explore_Init")

################################################  Outlier Removal
#####################################################################

# Remove any 0 Rent entries since 0 rent doesn't make logical sense and can sway the data
outData = get_outliers(rentData, 0, "LTE", c("Rent"), basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}


########################################## Assign x variables for functions
#############################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

########################################## Create Rent By Thousands
#####################################################################

rentData$thousRent = rentData$Rent / 1000

########################################## Create Box Cox of Rent
#####################################################################

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
    xformStr = var
  }else{
    xformStr = paste(xformStr, "+", var, sep="")
  }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)
```

```
transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda1 = transInfo$result[1]

if(yLambda1 > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda1) - 1)/yLambda1
}else{
  rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda1)
bcDescs = append(bcDescs, "Orig X Vars")

######################################### Write dataset to a file for other group members
##########################################

write.table(rentData, paste(mainPath, "ChicagoProject_WithDummyNoBadPts.txt", sep=""), sep="\t", row.names=FALSE)

######################################### Perform Exploratory Analysis on Whole Dataset
##############################################

dir.create(file.path(anlysPath, "M1 All Vars"), showWarnings = FALSE)
basePath = paste(anlysPath, "M1 All Vars\\", sep="")

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Perform Exploratory Data Analysis on entire dataset
explorePath = explore_data(rentData, varInfo.yVarName, varInfo.yVarPlotName, varInfo, TRUE, TRUE, basePath, "Explore_Init")

######################################### Generate Normal Plots of Rent
#################################################################

# Plot Y Normal Plot
fileName = paste(explorePath, "Explore_Init", "_RentNormPlotWithLimits.jpeg", sep="")
plotTitle = paste("Rent Normal Plot")
jpeg(file=fileName)
qqnorm(rentData$Rent, main=plotTitle)
qqline(rentData$Rent)
abline(a=lowRentLimit, b=0, col="blue")
abline(a=highRentLimit, b=0, col="blue")
dev.off()

# Create scatterplots of renter versus y value
fileName = paste(explorePath, "Explore_Init", "_RentVsRentersWithLimits.jpeg", sep="")
plotTitle = paste("Rent Versus Renter")
plotData = data.frame(rentData$renter, rentData$Rent)
jpeg(file=fileName)
plot(plotData, main=plotTitle,
    xlab="renter", ylab="Rent")
abline(a=lowRentLimit, b=0, col="blue")
abline(a=highRentLimit, b=0, col="blue")
dev.off()



######################################### Perform Exploratory Analysis on Whole Dataset with Rent By THousands
######################

dir.create(file.path(anlysPath, "M2 All Vars Thous"), showWarnings = FALSE)
basePath = paste(anlysPath, "M2 All Vars Thous\\", sep="")

varInfo.yVarName = "thousRent"
varInfo.yVarPlotName = "Median Gross Rent By Thousands"
```

```
# Perform Exploratory Data Analysis on entire dataset
explorePath = explore_data(rentData, varInfo.yVarName, varInfo.yVarPlotName, varInfo, TRUE, TRUE, basePath, "Explore_Init")

######################################## Generate Normal Plots of Rent
#####################################################################

# Plot Y Normal Plot
fileName = paste(explorePath, "Explore_Init", "_RentByThousNormPlotWithLimits.jpeg", sep="")
plotTitle = paste("Rent By Thousands Normal Plot")
jpeg(file=fileName)
qqnorm(rentData$thousRent, main=plotTitle)
qqline(rentData$thousRent)
abline(a=(lowRentLimit/1000), b=0, col="blue")
abline(a=(highRentLimit/1000), b=0, col="blue")
dev.off()


# Create scatterplots of renter versus y value
fileName = paste(explorePath, "Explore_Init", "_RentVsThousRentersWithLimits.jpeg", sep="")
plotTitle = paste("Rent in Thousands Versus Renter")
plotData = data.frame(rentData$renter, rentData$thousRent)
jpeg(file=fileName)
plot(plotData, main=plotTitle,
   xlab="renter", ylab="Rent In Thousands")
abline(a=(lowRentLimit/1000), b=0, col="blue")
abline(a=(highRentLimit/1000), b=0, col="blue")
dev.off()


######################################## Perform Exploratory Analysis on Whole Dataset With BC Rent
###################################

dir.create(file.path(anlysPath, "M3 All Vars BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M3 All Vars BC\\", sep="")

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Median Gross Rent"

# Perform Exploratory Data Analysis on entire dataset
explorePath = explore_data(rentData, varInfo.yVarName, varInfo.yVarPlotName, varInfo, TRUE, TRUE, basePath, "Explore_Init")

######################################## Generate  Normal Plots of bcRent
#############################################################

# Plot Y Normal Plot
fileName = paste(explorePath, "Explore_Init", "_bcRentNormPlotWithLimits.jpeg", sep="")
plotTitle = paste("Box Cox Rent Normal Plot")
jpeg(file=fileName)
qqnorm(rentData$bcRent, main=plotTitle)
qqline(rentData$bcRent)
abline(a=bcLowRentLimit, b=0, col="blue")
abline(a=bcHighRentLimit, b=0, col="blue")
dev.off()


# Create scatterplots of renter versus y value
fileName = paste(explorePath, "Explore_Init", "_RentVsBCRentersWithLimits.jpeg", sep="")
plotTitle = paste("Box Cox Rent Versus Renter")
plotData = data.frame(rentData$renter, rentData$bcRent)
jpeg(file=fileName)
plot(plotData, main=plotTitle,
   xlab="renter", ylab="Box Cox Rent")
abline(a=bcLowRentLimit, b=0, col="blue")
abline(a=bcHighRentLimit, b=0, col="blue")
dev.off()
```

```
########################################## All Var Modeling
###############################################################################

dir.create(file.path(anlysPath, "M1 All Vars"), showWarnings = FALSE)
basePath = paste(anlysPath, "M1 All Vars\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
baseData = data.frame()

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, 20, outlierLevel, varInfo, basePath, "AllData")
origInfPtsNdx = modelOutParams[[1]]

########################################## All Var Modeling Thous
###############################################################################

dir.create(file.path(anlysPath, "M2 All Vars Thous"), showWarnings = FALSE)
basePath = paste(anlysPath, "M2 All Vars Thous\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
baseData = data.frame()

varInfo.yVarName = "thousRent"
varInfo.yVarPlotName = "Median Gross Rent By Thousands"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, 20, outlierLevel, varInfo, basePath, "AllData")


########################################## All Var Modeling BC
###############################################################################

dir.create(file.path(anlysPath, "M3 All Vars BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M3 All Vars BC\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
baseData = data.frame()

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, 20, outlierLevel, varInfo, basePath, "AllData")
bcInfPtsNdx = modelOutParams[[1]]
```

```
########################################## Remove Influential Pts
#########################################################################

dir.create(file.path(anlysPath, "M4 All Vars No Inf Pts"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 All Vars No Inf Pts\\", sep="")

if(length(origInfPtsNdx) > 0){

  outData = remove_inf_pts(rentData, origInfPtsNdx, basePath, "InfPts")
  newRentData = outData
}

########################################## All Var Modeling w/o Inf Pts
#################################################################

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

arInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

modelOutParams = model_data(newRentData, baseData, modelFlags, 0, 20, outlierLevel, varInfo, basePath, "AllData")
origOutPtsNdx = modelOutParams[[2]]

########################################## Remove Influential Pts
#########################################################################

dir.create(file.path(anlysPath, "M5 All Vars No Inf BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M5 All Vars No Inf BC\\", sep="")

if(length(bcInfPtsNdx) > 0){

  outData = remove_inf_pts(rentData, bcInfPtsNdx, basePath, "InfPts")
  newBcRentData = outData
}

########################################## All Var Modeling w/o Inf Pts
#################################################################

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"

modelOutParams = model_data(newBcRentData, baseData, modelFlags, 0, 20, outlierLevel, varInfo, basePath, "AllData")
origBcOutPtsNdx = modelOutParams[[2]]

########################################## Remove Extreme Outlier Pts
#########################################################################

dir.create(file.path(anlysPath, "M6 All Vars No Out"), showWarnings = FALSE)
basePath = paste(anlysPath, "M6 All Vars No Out\\", sep="")

if(length(origOutPtsNdx) > 0){
```

```
 outData = remove_inf_pts(newRentData, origOutPtsNdx, basePath, "ExtremeOut")
 newRentData = outData
}


########################################## All Var Modeling w/o Outliers
#############################################################

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

arInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

modelOutParams = model_data(newRentData, baseData, modelFlags, 0, 20, outlierLevel, varInfo, basePath, "AllData")

########################################## Remove Extreme Outlier Pts
#############################################################

dir.create(file.path(anlysPath, "M7 All Vars No Out BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M7 All Vars No Out BC\\", sep="")

if(length(origBcOutPtsNdx) > 0){

 outData = remove_inf_pts(newBcRentData, origBcOutPtsNdx, basePath, "ExtremeOut")
 newBcRentData = outData
}


########################################## High and Low Rent Dummy Variable Creation
#################################################

tempLowFlag = (rentData$Rent < lowRentLimit)*1
rentData$lowRentFlag = tempLowFlag

tempHighFlag = (rentData$Rent < highRentLimit)*1
rentData$highRentFlag = tempHighFlag

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("lowRentFlag", "highRentFlag"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Low Rent Flag", "High Rent Flag"))


########################################## Create Box Cox of Rent
#############################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
  xformStr = var
 }else{
  xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
```

```
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda2 = transInfo$result[1]

if(yLambda2 > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda2) - 1)/yLambda2
}else{
  rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda2)
bcDescs = append(bcDescs, "High/Low Rent")

######################################## Rent Affordability Indices Creation
#########################################################

incomeIndex = rentData$mdfamy0 / mean(rentData$mdfamy0)
rentIndex = rentData$Rent / mean(rentData$Rent)

rentData$rentAffNdx = incomeIndex / rentIndex

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("rentAffNdx"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Rent Affordability Index"))

matchNdx = which(allVarInfo.xVarNames %in% "lowRentFlag")
allVarInfo.xVarNames = allVarInfo.xVarNames[!allVarInfo.xVarNames %in% "lowRentFlag"]
allVarInfo.xVarPlotNames = allVarInfo.xVarPlotNames[!allVarInfo.xVarPlotNames %in% allVarInfo.xVarPlotNames[matchNdx]]

matchNdx = which(allVarInfo.xVarNames %in% "highRentFlag")
allVarInfo.xVarNames = allVarInfo.xVarNames[!allVarInfo.xVarNames %in% "highRentFlag"]
allVarInfo.xVarPlotNames = allVarInfo.xVarPlotNames[!allVarInfo.xVarPlotNames %in% allVarInfo.xVarPlotNames[matchNdx]]

########################################## Create Box Cox of Rent
#########################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
    xformStr = var
  }else{
    xformStr = paste(xformStr, "+", var, sep="")
  }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda3 = transInfo$result[1]

if(yLambda3 > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda3) - 1)/yLambda3
}else{
  rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda3)
```

```
bcDescs = append(bcDescs, "Rent Aff Index")

########################################## All Var Plus  High/Low Rent Dummy Vars Modeling
########################################

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("lowRentFlag", "highRentFlag"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Low Rent Flag", "High Rent Flag"))

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

########################################## Create Box Cox of Rent
#########################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
   xformStr = var
 }else{
   xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda4 = transInfo$result[1]

if(yLambda4 > 0.1){
 rentData$bcRent = ((rentData$Rent^yLambda4) - 1)/yLambda4
}else{
 rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda4)
bcDescs = append(bcDescs, "H/L Rent & Aff Index")

########################################## General Race Variable Creation
#################################################################

rentData$nonWhitenhp = 1 - rentData$whitenhp

rentData$nonAanhp = 1 - rentData$aanhp

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("nonWhitenhp", "nonAanhp"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Non-White", "Non-African American"))

matchNdx = which(allVarInfo.xVarNames %in% "lowRentFlag")
allVarInfo.xVarNames = allVarInfo.xVarNames[!allVarInfo.xVarNames %in% "lowRentFlag"]
allVarInfo.xVarPlotNames = allVarInfo.xVarPlotNames[!allVarInfo.xVarPlotNames %in% allVarInfo.xVarPlotNames[matchNdx]]

matchNdx = which(allVarInfo.xVarNames %in% "highRentFlag")
allVarInfo.xVarNames = allVarInfo.xVarNames[!allVarInfo.xVarNames %in% "highRentFlag"]
allVarInfo.xVarPlotNames = allVarInfo.xVarPlotNames[!allVarInfo.xVarPlotNames %in% allVarInfo.xVarPlotNames[matchNdx]]

matchNdx = which(allVarInfo.xVarNames %in% "rentAffNdx")
allVarInfo.xVarNames = allVarInfo.xVarNames[!allVarInfo.xVarNames %in% "rentAffNdx"]
allVarInfo.xVarPlotNames = allVarInfo.xVarPlotNames[!allVarInfo.xVarPlotNames %in% allVarInfo.xVarPlotNames[matchNdx]]
```

```
######################################## Write dataset to a file for other group members
############################################

write.table(rentData, paste(mainPath, "ChicagoProject_WithDummyAndNewNoBadPts.txt", sep=""), sep="\t", row.names=FALSE)

######################################### Create Box Cox of Rent
#########################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
   xformStr = var
 }else{
   xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda5 = transInfo$result[1]

if(yLambda5 > 0.1){
 rentData$bcRent = ((rentData$Rent^yLambda5) - 1)/yLambda5
}else{
 rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda5)
bcDescs = append(bcDescs, "Race")

######################################### All Var Plus  High/Low Rent Dummy Vars Modeling
########################################

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("lowRentFlag", "highRentFlag"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Low Rent Flag", "High Rent Flag"))

######################################### Create Box Cox of Rent
#########################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
   xformStr = var
 }else{
   xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)
```

```
transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda6 = transInfo$result[1]

if(yLambda6 > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda6) - 1)/yLambda6
}else{
  rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda6)
bcDescs = append(bcDescs, "High/Low Race")

######################################### All Var Plus  High/Low Rent Dummy Vars Modeling
#########################################

matchNdx = which(allVarInfo.xVarNames %in% "lowRentFlag")
allVarInfo.xVarNames = allVarInfo.xVarNames[!allVarInfo.xVarNames %in% "lowRentFlag"]
allVarInfo.xVarPlotNames = allVarInfo.xVarPlotNames[!allVarInfo.xVarPlotNames %in% allVarInfo.xVarPlotNames[matchNdx]]

matchNdx = which(allVarInfo.xVarNames %in% "highRentFlag")
allVarInfo.xVarNames = allVarInfo.xVarNames[!allVarInfo.xVarNames %in% "highRentFlag"]
allVarInfo.xVarPlotNames = allVarInfo.xVarPlotNames[!allVarInfo.xVarPlotNames %in% allVarInfo.xVarPlotNames[matchNdx]]

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("rentAffNdx"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Rent Affordability Index"))

######################################### Create Box Cox of Rent
#########################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
  var = varInfo.xVarNames[j]

  if(j == 1){
    xformStr = var
  }else{
    xformStr = paste(xformStr, "+", var, sep="")
  }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda7 = transInfo$result[1]

if(yLambda7 > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda7) - 1)/yLambda7
}else{
  rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda7)
bcDescs = append(bcDescs, "Rent Aff Race")

######################################### All Var Plus  High/Low Rent Dummy Vars Modeling
#########################################

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("lowRentFlag", "highRentFlag"))
```

```
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Low Rent Flag", "High Rent Flag"))

########################################## Create Box Cox of Rent
#########################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
   xformStr = var
 }else{
   xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
transInfo = summary(transFit)
yLambda8 = transInfo$result[1]

if(yLambda8 > 0.1){
 rentData$bcRent = ((rentData$Rent^yLambda8) - 1)/yLambda8
}else{
 rentData$bcRent = log(rentData$Rent)
}

bcLambdas = append(bcLambdas, yLambda8)
bcDescs = append(bcDescs, "All New")

########################################## Create Output Box Cox Info File
#############################################################

# Create output file
outFileName = paste(anlysPath, "BoxCox_Lambda_Info_R_Output.txt", sep="")
outFile = file(outFileName, open="wt")

# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Variables included in Box Cox Rent formula:", quote=FALSE)
print("", quote=FALSE)
print(bcDescs, quote=FALSE)
print("", quote=FALSE)
print("Box Cox Lambda Variables:", quote=FALSE)
print(bcLambdas, quote=FALSE)
print("", quote=FALSE)
sink()

close(outFile)
closeAllConnections()
```

Project All New.R File (used for modeling with new variable sets added to try to normalize residuals):

```
#########################################################################################################################################
#############
#
#  Project All New.R
```

```
#
#  This file performs models with all combinations of independent variables.  This includes with the three additional sets of
#  variables are then created (one with dummy vars for non-white and non-african america, one with dummy vars for high rent and
#  low rent, and one with the rent affordability index).  Models are generated using the regular Rent dependent variable and the
#  Box Cox of Rent, with the backward and stepwise model selection options.  All combinations of the three additional sets of
#  variables are done by commenting out the ones not desired (to do only the race, the index and high/low rent ones are commented
#  out for example).  All combinations include, no additional sets, high/low by itself, index by itself, race by itself, high/low
#  and index, high/low and race, index and race, and all three.
#
#  Created By    Date
#  Kari Palmier  3/14/2017
#
#########################################################################################################################################
#############

library(psych)
library(QuantPsyc)
library(car)
library(leaps)
library(DAAG)

# Assign output file directory based on relative path of this R script
mainPath = getSrcDirectory(function(x) {x})
mainPath = gsub("/", "\\\\", mainPath)
mainPath = paste(mainPath, '\\', sep="")
dir.create(file.path(mainPath, "R Anlys All"), showWarnings = FALSE)
anlysPath = paste(mainPath, "R Anlys All\\", sep="")

# Get subfunctions required
source(paste(mainPath, "Common R Functions\\analyze_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\explore_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\select_model.R", sep=""))
source(paste(mainPath, "Common R Functions\\validate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\crossvalidate_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\get_outliers.R", sep=""))
source(paste(mainPath, "Common R Functions\\model_data.R", sep=""))
source(paste(mainPath, "Common R Functions\\remove_inf_pts.R", sep=""))

# Import Dataset
rentData = read.table(paste(mainPath, "\\Chicago_rent\\ChicagoProject.txt", sep=""),
                      header=T)

#################################### Variable Initialization
#####################################################################

varInfo.xVarNames = c("mdfamy0", "asianp", "whitenhp", "aanhp", "hisp", "renter", "educ8",
                      "educ160pro", "Noheat", "totcrime", "murder", "robbery", "personal", "property", "forecl")
varInfo.xVarPlotNames = c("Median Family Income", "Prop of Asian American",
                          "Prop of Non-Hispanic White", "Prop of Non-Hispanic African American",
                          "Prop Hispanic American", "Prop of Renter Occupied", "Prop of With 8 Yrs School",
                          "Prop With Bachelor or Grad School", "Prop With No Heat", "Prop Total Crime",
                          "Prop Murder", "Prop Robbery", "Prop Personal Crime", "Prop Property Crime",
                          "Prop Foreclosures")
varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

percTestPts = 25
percCrossValPts = 20
highRentLimit =  720
lowRentLimit = 450
bcHighRentLimit = 98
bcLowRentLimit = 73

outlierLevel = 5
```

```
#################################### Dataset Conversion
#############################################################################

dir.create(file.path(anlysPath, "Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "Init\\", sep="")

# Change the data frame columns from factor type to character type
i <- sapply(rentData, is.factor)
rentData[i] <- lapply(rentData[i], as.character)

# Remove any missing points from the dataset
outData = get_outliers(rentData, "-", "EQ", varInfo.xVarNames, basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}


# Change all data frame columns from character to numeric (except for the column of neighborhood names)
totVars = append(varInfo.yVarName, varInfo.xVarNames)
numVars = length(totVars)
for (i in 1:numVars){
  var = totVars[i]
  if (is.character(var)){
    rentData[var] = as.numeric(rentData[,var])
  }
}

################################################  Outlier Removal
#####################################################################

# Remove any 0 Rent entries since 0 rent doesn't make logical sense and can sway the data
outData = get_outliers(rentData, 0, "LTE", c("Rent"), basePath, "Explore_Init")

if(length(outData[,"Rent"]) != length(rentData[,"Rent"])){
  rentData = outData
}


######################################### Neighborhood Dummy Variable Creation
#########################################################

allVarInfo.xVarNames = varInfo.xVarNames
allVarInfo.xVarPlotNames = varInfo.xVarPlotNames

uniqAreas = unique(rentData$canam77)
numAreas = length(uniqAreas)
flagNames = c()
flagDescs = c()
for(i in 1:(numAreas-1)){
  val = uniqAreas[i]
  lowVal = tolower(val)
  matchNdx = which(rentData$canam77 %in% val)

  if(length(matchNdx) > 0){
    temp = gsub("-", "_", lowVal)
    temp2 = gsub("'", "", temp)
    newVal = paste(temp2, "Flag", sep="")

    tempFlag = (rentData$canam77 == val)*1
    rentData[newVal] = tempFlag

    flagNames = append(flagNames, newVal)
    flagDescs = append(flagDescs, val)
  }
}
```

```
numFlagVars = length(flagNames)
for(i in 1:numFlagVars){
 allVarInfo.xVarNames = append(allVarInfo.xVarNames, flagNames[i])
 allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, flagDescs[i])
}

######################################## High and Low Rent Dummy Variable Creation
################################################

tempLowFlag = (rentData$Rent < lowRentLimit)*1
rentData$lowRentFlag = tempLowFlag

tempHighFlag = (rentData$Rent < highRentLimit)*1
rentData$highRentFlag = tempHighFlag

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("lowRentFlag", "highRentFlag"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Low Rent Flag", "High Rent Flag"))


######################################## Rent Affordability Indices Creation
#########################################################

incomeIndex = rentData$mdfamy0 / mean(rentData$mdfamy0)
rentIndex = rentData$Rent / mean(rentData$Rent)

rentData$rentAffNdx = incomeIndex / rentIndex

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("rentAffNdx"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Rent Affordability Index"))

######################################## General Race Variable Creation
#################################################################

rentData$nonWhitenhp = 1 - rentData$whitenhp

rentData$nonAanhp = 1 - rentData$aanhp

allVarInfo.xVarNames = append(allVarInfo.xVarNames, c("nonWhitenhp", "nonAanhp"))
allVarInfo.xVarPlotNames = append(allVarInfo.xVarPlotNames, c("Non-White", "Non-African American"))

######################################## Assign x variables for functions
#########################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

######################################## Create Box Cox of Rent
#############################################################################

# Compute Box Cox on Rent
numXVars = length(varInfo.xVarNames)
for(j in 1:numXVars){
 var = varInfo.xVarNames[j]

 if(j == 1){
   xformStr = var
 }else{
   xformStr = paste(xformStr, "+", var, sep="")
 }
}

formStr = paste(varInfo.yVarName, "~", xformStr, sep="")
modelForm = as.formula(formStr)

transFit = powerTransform(modelForm, data=rentData, family="bcPower")
```

```
transInfo = summary(transFit)
yLambda = transInfo$result[1]

if(yLambda > 0.1){
  rentData$bcRent = ((rentData$Rent^yLambda) - 1)/yLambda
}else{
  rentData$bcRent = log(rentData$Rent)
}


########################################## Write dataset to a file for other group members
##########################################

write.table(rentData, paste(mainPath, "ChicagoProject_WithDummyNoBadPts.txt", sep=""), sep="\t", row.names=FALSE)

########################################## All Var Modeling
###########################################################################

dir.create(file.path(anlysPath, "M1 All Vars"), showWarnings = FALSE)
basePath = paste(anlysPath, "M1 All Vars\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
baseData = data.frame()

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")


########################################## Remove Influential Pts
###########################################################################

dir.create(file.path(anlysPath, "M2 All Vars No Inf Pts"), showWarnings = FALSE)
basePath = paste(anlysPath, "M2 All Vars No Inf Pts\\", sep="")

infPtsNdx = modelOutParams[[1]]
if(length(infPtsNdx) > 0){

  outData = remove_inf_pts(rentData, infPtsNdx, basePath, "InfPts")
  rentData = outData
}

########################################## All Var Modeling w/o Inf Pts
###############################################################

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

########################################## Remove Extreme Outlier Pts
###############################################################

dir.create(file.path(anlysPath, "M3 All Vars No Out"), showWarnings = FALSE)
```

```
basePath = paste(anlysPath, "M3 All Vars No Out\\", sep="")

infPtsNdx = modelOutParams[[2]]
if(length(infPtsNdx) > 0){

 outData = remove_inf_pts(rentData, infPtsNdx, basePath, "ExtremeOut")
 rentData = outData
}


########################################## All Var Modeling w/o Outliers
#############################################################

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

selNames = modelOutParams[[8]]
backTempNames = selNames[[1]]
stepTempNames = selNames[[2]]

########################################## Backward Sel Modeling
#########################################################################

tempLength = length(backTempNames)
backVarInfo.xVarNames = backTempNames[2:tempLength]
numBackVars = length(backVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numBackVars){
 var = backVarInfo.xVarNames[i]
 matchNdx = which(backVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

backVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M4 Back Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 Back Init\\", sep="")

varInfo.xVarNames = backVarInfo.xVarNames
varInfo.xVarPlotNames = backVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 5
backCoeffsRmvd = c()
while(maxPVal > 0.05){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
```

```
 maxPVal = max(coeffPVals)

 if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  backCoeffsRmvd = append(backCoeffsRmvd, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }
}

finalBackVarInfo.xVarNames = varInfo.xVarNames
finalBackVarInfo.xVarPlotNames = varInfo.xVarPlotNames

######################################### Stepwise Sel Modeling
#########################################################################

tempLength = length(stepTempNames)
stepVarInfo.xVarNames = stepTempNames[2:tempLength]
numStepVars = length(stepVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numStepVars){
 var = stepVarInfo.xVarNames[i]
 matchNdx = which(stepVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

stepVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M4 Step Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 Step Init\\", sep="")

varInfo.xVarNames = stepVarInfo.xVarNames
varInfo.xVarPlotNames = stepVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 5
stepCoeffsRmvd = c()
while(maxPVal > 0.05){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
```

```
   coeffNames = rownames(modelCoeffs)
   maxCoeffName = coeffNames[maxNdx]
   stepCoeffsRmvd = append(stepCoeffsRmvd, maxCoeffName)

   matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
   varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
   varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

   folderName = paste('M', modelNdx, " Step", sep="")
   dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
   basePath = paste(anlysPath, folderName, "\\", sep="")

   modelNdx = modelNdx + 1
  }
}

finalStepVarInfo.xVarNames = varInfo.xVarNames
finalStepVarInfo.xVarPlotNames = varInfo.xVarPlotNames

########################################## Box Cox All Var Modeling
#####################################################################

varInfo.xVarNames = allVarInfo.xVarNames
varInfo.xVarPlotNames = allVarInfo.xVarPlotNames

dir.create(file.path(anlysPath, "M4 All Vars BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M4 All Vars BC\\", sep="")

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = TRUE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

varInfo.yVarName = "bcRent"
varInfo.yVarPlotName = "BoxCox of Rent"

modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

selNames = modelOutParams[[8]]
backTempNames = selNames[[1]]
stepTempNames = selNames[[2]]

########################################## Backward Sel Modeling
#####################################################################

tempLength = length(backTempNames)
backVarInfo.xVarNames = backTempNames[2:tempLength]
numBackVars = length(backVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numBackVars){
 var = backVarInfo.xVarNames[i]
 matchNdx = which(backVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

backVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
```

```
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M5 Back BC Init"), showWarnings = FALSE)
basePath = paste(anlysPath, "M5 Back BC Init\\", sep="")

varInfo.xVarNames = backVarInfo.xVarNames
varInfo.xVarPlotNames = backVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 6
backCoeffsRmvdBc = c()
while(maxPVal > 0.05){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  backCoeffsRmvdBc = append(backCoeffsRmvdBc, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Back BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }
}

finalBackBcVarInfo.xVarNames = varInfo.xVarNames
finalBackBcVarInfo.xVarPlotNames = varInfo.xVarPlotNames

########################################## Stepwise Sel Modeling
#########################################################################

tempLength = length(stepTempNames)
stepVarInfo.xVarNames = stepTempNames[2:tempLength]
numStepVars = length(stepVarInfo.xVarNames)
tempPlotVars = c()
for(i in 1:numStepVars){
 var = stepVarInfo.xVarNames[i]
 matchNdx = which(stepVarInfo.xVarNames %in% var)
 tempPlotVars = append(tempPlotVars, varInfo.xVarPlotNames[matchNdx])

}

stepVarInfo.xVarPlotNames = tempPlotVars

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = FALSE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

dir.create(file.path(anlysPath, "M5 Step BC Init"), showWarnings = FALSE)
```

```
basePath = paste(anlysPath, "M5 Step BC Init\\", sep="")

varInfo.xVarNames = stepVarInfo.xVarNames
varInfo.xVarPlotNames = stepVarInfo.xVarPlotNames

maxPVal = 1
modelNdx = 6
stepCoeffsRmvdBc = c()
while(maxPVal > 0.05){

 modelOutParams = model_data(rentData, baseData, modelFlags, 0, percCrossValPts, outlierLevel, varInfo, basePath, "AllData")

 modelCoeffs = modelOutParams[[7]]
 coeffPVals = modelCoeffs[,4]
 maxPVal = max(coeffPVals)

 if(maxPVal > 0.05){
  maxNdx = which(coeffPVals %in% maxPVal)
  coeffNames = rownames(modelCoeffs)
  maxCoeffName = coeffNames[maxNdx]
  stepCoeffsRmvdBc = append(stepCoeffsRmvdBc, maxCoeffName)

  matchNdx = which(varInfo.xVarNames %in% maxCoeffName)
  varInfo.xVarNames = varInfo.xVarNames[!varInfo.xVarNames %in% maxCoeffName]
  varInfo.xVarPlotNames = varInfo.xVarPlotNames[!varInfo.xVarPlotNames %in% varInfo.xVarPlotNames[matchNdx]]

  folderName = paste('M', modelNdx, " Step BC", sep="")
  dir.create(file.path(anlysPath, folderName), showWarnings = FALSE)
  basePath = paste(anlysPath, folderName, "\\", sep="")

  modelNdx = modelNdx + 1
 }
}

finalStepBcVarInfo.xVarNames = varInfo.xVarNames
finalStepBcVarInfo.xVarPlotNames = varInfo.xVarPlotNames

######################################## Create Training and Testing Datasets
#########################################################

varInfo.yVarName = "Rent"
varInfo.yVarPlotName = "Median Gross Rent"

# Create sample indices
selectNdx = sample(1:nrow(rentData), (1-(percTestPts/100))*nrow(rentData))

# Select the training dataset
trainData = rentData[selectNdx, ]

# Select the testing dataset
testData = rentData[-selectNdx, ]

testSetDiff = setdiff(testData$canam77, trainData$canam77)
numDiff = length(testSetDiff)
numOrigRows = length(rentData$canam77)
allMatchNdx = c()
if(numDiff >0){
 for(i in 1:numDiff){
  val = testSetDiff[i]

  matchNdx = which(testData$canam77 %in% val)

  numMatches = length(matchNdx)
  if(matchNdx > 0){
   allMatchNdx = append(allMatchNdx, matchNdx)
  }
```

```
  }

  numAllMatches = length(allMatchNdx)
  percMatches = (numAllMatches / numOrigRows) * 100

  if(percMatches < 5){
    tempData = testData[allMatchNdx,]
    trainData = rbind(trainData, setNames(tempData, names(trainData)))
    testData = testData[-allMatchNdx,]
  }
}


testSetDiff = setdiff(testData$canam77, trainData$canam77)
trainSetDiff = setdiff(trainData$canam77, testData$canam77)
trainDataPerc = (length(trainData$canam77)/numOrigRows)*100
testDataPerc = (length(testData$canam77)/numOrigRows)*100


outFileName = paste(anlysPath, "DataSet_Split_Information.txt", sep="")
outFile = file(outFileName, open="wt")

# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Neighborhoods in Testing data that are not in Training data:", quote=FALSE)
print("", quote=FALSE)
print(testSetDiff, quote=FALSE)
print("", quote=FALSE)
print("Neighborhoods in Training data that are not in Testing data:", quote=FALSE)
print("", quote=FALSE)
print(trainSetDiff, quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
print("Percentage of Original data used for Training data:", quote=FALSE)
print("", quote=FALSE)
print(round(trainDataPerc, digits=2), quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
print("Percentage of Original data used for Training data:", quote=FALSE)
print("", quote=FALSE)
print(round(testDataPerc, digits=2), quote=FALSE)
print("", quote=FALSE)
print("", quote=FALSE)
sink()

close(outFile)
closeAllConnections()


########################################## Backward Train/Test Modeling
#################################################################

dir.create(file.path(anlysPath, "M Last Back"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back\\", sep="")

varInfo.xVarNames = finalBackVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE
```

```
modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Stepwise Train/Test Modeling
#################################################################

dir.create(file.path(anlysPath, "M Last Step"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step\\", sep="")

varInfo.xVarNames = finalStepVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Backward BC Train/Test Modeling
#################################################################

dir.create(file.path(anlysPath, "M Last Back BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Back BC\\", sep="")

varInfo.xVarNames = finalBackBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalBackBcVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Stepwise BC Train/Test Modeling
#################################################################

dir.create(file.path(anlysPath, "M Last Step BC"), showWarnings = FALSE)
basePath = paste(anlysPath, "M Last Step BC\\", sep="")

varInfo.xVarNames = finalStepBcVarInfo.xVarNames
varInfo.xVarPlotNames = finalStepBcVarInfo.xVarPlotNames

# Perform all Y transforms with all x variables to see if any transforms improve modeling
modelFlags.performExp = TRUE
modelFlags.validateTest = TRUE
modelFlags.performModel = TRUE
modelFlags.makeExpPlots = FALSE
modelFlags.makeExpXXPlots = FALSE
modelFlags.quadXModel = FALSE

modelOutParams = model_data(trainData, testData, modelFlags, percTestPts, percCrossValPts, outlierLevel, varInfo, basePath, "TrainData")

########################################## Create Output Model Sel Info File
#################################################################

# Create output file
outFileName = paste(anlysPath, "Model_Sel_Info_R_Output.txt", sep="")
outFile = file(outFileName, open="wt")
```

```
# Print cp model selection values to output file
sink(file=outFile, append=TRUE)
print("", quote=FALSE)
print("Variables removed during backward selection of Rent (in order or removal):", quote=FALSE)
print("", quote=FALSE)
print(backCoeffsRmvd, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during stepwise selection of Rent (in order or removal):", quote=FALSE)
print(stepCoeffsRmvd, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during backward selection of Box Cox Rent (in order or removal):", quote=FALSE)
print("", quote=FALSE)
print(backCoeffsRmvdBc, quote=FALSE)
print("", quote=FALSE)
print("Variables removed during stepwise selection of Box Cox Rent (in order or removal):", quote=FALSE)
print(stepCoeffsRmvdBc, quote=FALSE)
print("", quote=FALSE)
sink()

close(outFile)
closeAllConnections()
```