**2.1 – Multiple Linear Regression Model**In multiple regression we study the conditional distribution of a response variable () given a set of potential predictor variables . As discussed previously, these predictor variables can have any data type – continuous/discrete, ordinal, or nominal. In this section we focus on the concept of terms. Terms in a multiple regression model are functions of the predictors . We will denote the terms in a multiple regression model using to avoid confusion with the predictors ().The form of the multiple regression model for predicting a response is given by

and typically we assume .

To find the parameter estimates we use Ordinary Least Squares (OLS), i.e. find the tominimize the residual sum of squares (RSS) which is given by,

In matrix notation,

where,

in our regression model and the are the observed values of the *jth* term. The OLS estimates of the parameters are found using matrices as:

Notice the predicted values are a linear combination of the observed response values , namely . This is common to several more advanced modeling strategies that are commonly employed in predictive analytics.

**2.2 - Types of Terms in MLR (and other models)**

In general, in MLR we need to decide which ***terms***, which are functions of the ***predictors*** , to include in our model to adequately predict the response. Below we consider the most commonly used terms () in a multiple linear regression models. These are the basic “*building blocks”* of a MLR model and some of the more advanced methods used in predictive modeling.

**Intercept term**

🡨 This term gives the column of 1’s in the model matrix . We do not   
 need to include an intercept term, but we almost always do!

**Predictor terms**

🡨 Terms can be the ***predictors*** themselves as long as the predictor is   
 meaningfully numeric! (i.e. count or a measurement). This  
 leads to the MLR model you see in most textbooks, i.e.

.

**Polynomial terms**

🡨 These terms are integer powers of numeric predictors, e.g.

**Transformation terms**

🡨 Here is the Tukey Power Transformation Family.

**Dummy terms**

Examples of conditions:

**“Hockey Stick” function**

or

These types of terms are used in fitting Multivariate Adaptive Regression Spline (MARS) models.

**Factor terms (nominal or ordinal variables with more than 2 levels)**

Suppose the predictor is a nominal or ordinal variable with levels (. Then we chose one of the levels as the ***reference group***and create dummy terms for the remaining levels.

E.g. Suppose

We could choose *4 = Oil* as the reference group and create dummy variables for the other two fuel types, i.e.

Why wouldn’t we create a dummy variable for each level of the levels of the variable? For this example that would mean also creating,

The problem with doing this is that the sum of the columns corresponding to these terms would be a column of ones. When one column in the matrix is a linear combination of other columns in the matrix then the inverse of the matrix doesn’t exist, i.e. the matrix is ***singular***.

**Interaction terms**

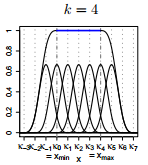
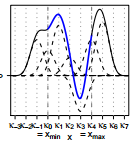
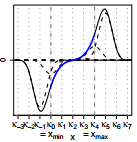
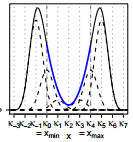
🡨 Here the term is a product of two terms (, where these two terms could be of any other term types.

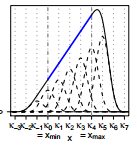
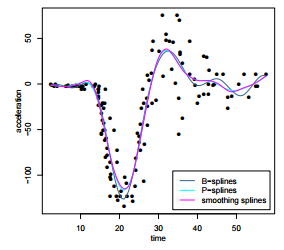
**Linear Combination Terms**

Some advanced modeling methods use linear combinations of predictors as terms. These linear combinations are chosen to reduce dimensionality in the predictor space in cases where the number of predictors () is large. The linear combination is specified by coefficients () which specifies the weight given to the predictor in the term, i.e.

These linear combination terms are used principal component regression (PCR), partial least squares regression (PLSR), and neural networks which are all methods that can be used prediction analytic problems. We will not be discussing any of these methods in this workshop. However I will note that PCR/PLSR can be used in prediction problems where the number of predictors exceeds the number of observations ). This is common in genetic studies where the expression levels of thousands of genes are used to predict a numeric characteristic of patients.

**Spline Basis Terms** - To introduce these types of terms consider the diagrams below.

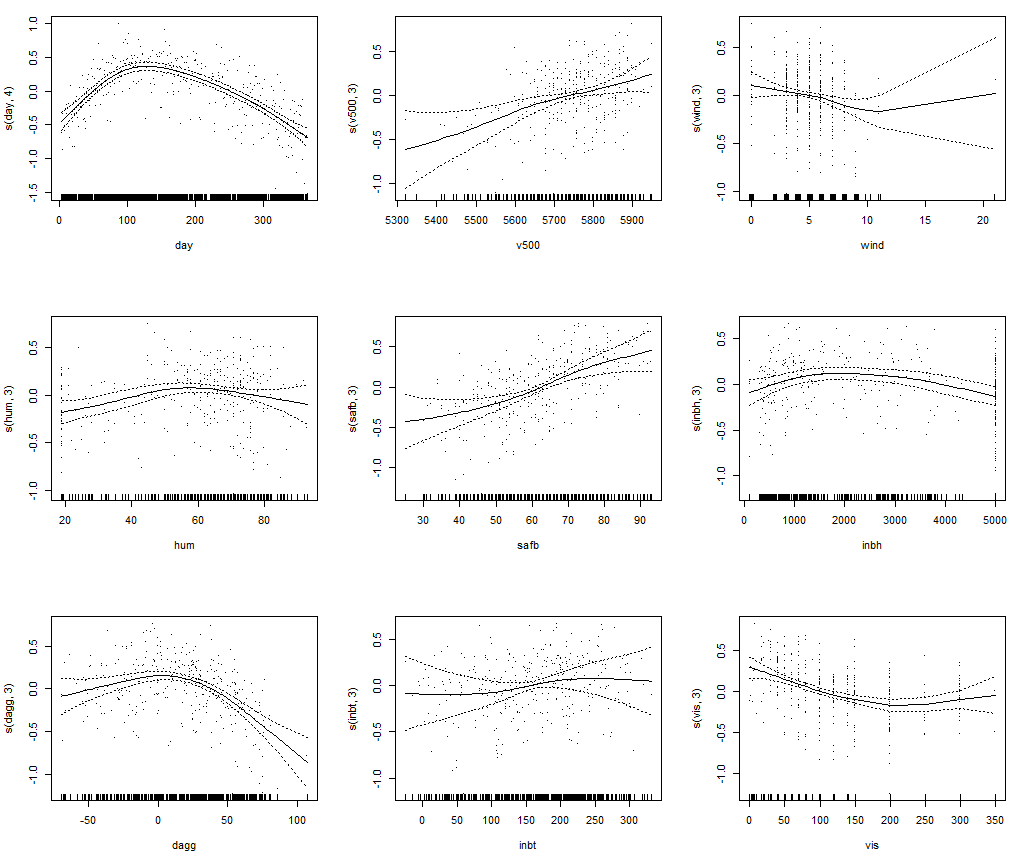
    
Spline basis functions are actually a collection of terms based upon a single predictor For example a spline basis could consist of a collection of terms of the form:

Piece-wise quartic polynomial spline basis functions ().

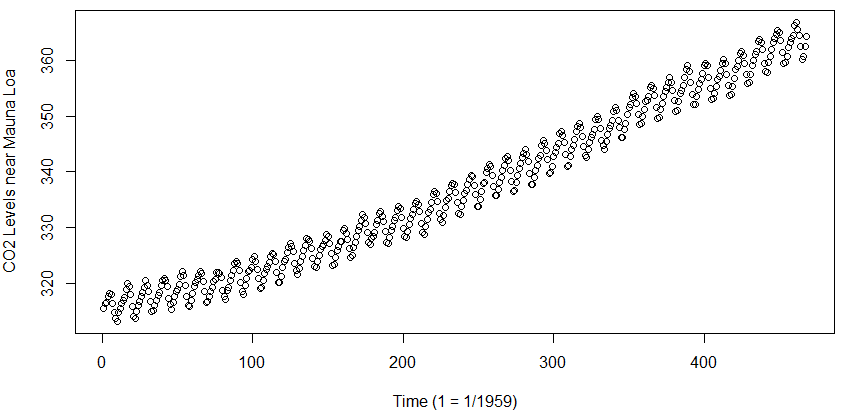
where the are constants withing the range of called *knots* (see diagrams above). Multivariate Adaptive Regression Splines (MARS) models use terms of this form with , along with potential interactions based upon these terms for two different predictors .

**Nonparametric Terms**Terms in a regression model need not have a specific parametric form but rather are estimated during the modeling process. The plots below show the estimated nonparametric terms for a model developed to predict the ozone concentration level in the atmosphere is the Los Angeles Basin. The fitted model is given by

thus here the terms have the form where the are estimated by **smoothing a scatterplot** as shown below. This is an example of a *generalized additive model* (GAM) where the errors are assumed to be normally distributed. Notice the absence of in this model.

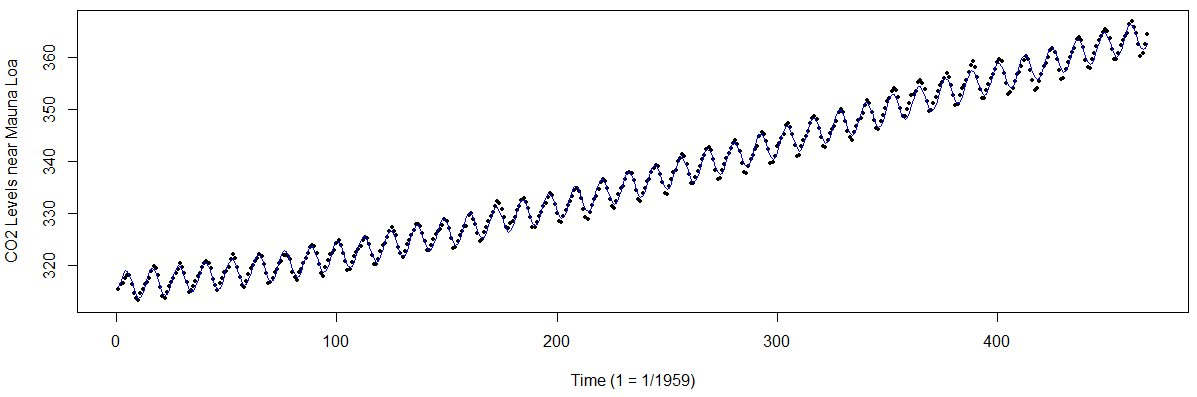


**Trigonometric Terms**

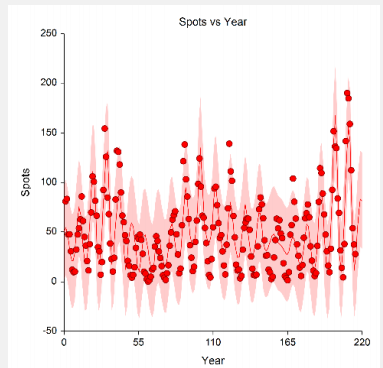
These terms can be used in prediction problems where there is periodic trends in the data. As a simple example consider these data which are levels measured monthly near the Mauna Loa volcano in Hawaii from 1/1959 – 12/1997. This is an example of a *time series*.  


There is clearly a long term increasing trend as well as a strong seasonal pattern that repeats every 12 months, i.e. annually. Adding trigonometric terms to a model of these data should be able to capture the seasonality. If then adding terms,

should allow us to model the strong periodic trend in these data.



Add additional terms and terms with different periodicities can allow for modeling time series with less well-behaved period structure. Adding different trigonometric function terms is referred to as ***harmonic regression***.



Here trigonometric terms with periodicities 9.4, 9.9, 10.6, 11.2, 57, and 91 were used to model the number of sunspots observed annually over period of 215 years.

The model fit to the sunspot data is shown below.

The periodicities used can be based on trial and error, knowledge of the physical phenomenon being studied (sunspots in this case), or using tools like ***spectral analysis*** to identify important periodicities. Will not delve into harmonic analysis in this course, this is only presented to illustrate that by using appropriate terms we can develop models capable of fitting complex relationships between the response and the predictors.

In general, MLR involves deciding which terms to include in our model, and possibly what scale should be used for response (e.g. using rather as the response).  
Given the wide variety of terms that can be added to MLR model given a set of potential predictors this is NO SMALL task, as for the prediction task at hand there are infinitely many possible models that could be constructed! Fortunately there are numerous algorithms in statistical/supervised learning that generally outperform even carefully crafted MLR models that do not require as much thought about the functional form of the predictors. However, we generally still need to give some thought about how to use the information contained in the predictors, so careful term construction can still be important even with “automatic” modeling methods. The construction of terms from a given set of predictors ( is often times referred to as ***feature engineering*** in the machine learning circles.

**Multiple Linear Regression Example 1: Diamond Prices and the 4 C’s**

**A diamond’s quality (and hence price) is determined by the 4C’s.**

**Color** – colorless to light yellow (see below)

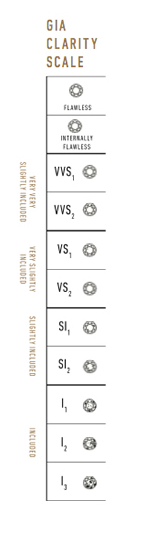
**Clarity** – see GIA scale on the right

**Cut** – grade of cut   
 (good, very good, excellent, ideal)

**Carat** – size/weight of the diamond   
 (1 carat – 200mg or 1/5 of a gram)



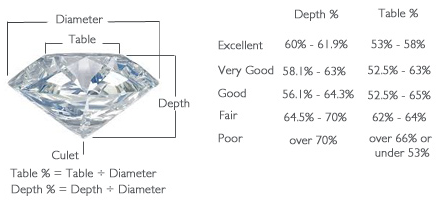


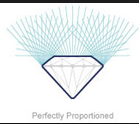
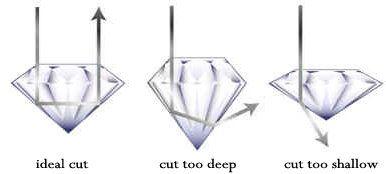


Also the Total Depth (%) and Table (%) are important characteristics

that deal with the shape of the diamond. These measurements are   
shown in the figure below, along with guidelines for quality of the

diamond based on these measurements.





Our goal is to develop an accurate model for predicting or determining the price of a diamond based upon these diamond characteristics. For this purpose will first develop a multiple linear regression model for these data, after doing some preliminary graphical summaries of these data.

**Datafile:** **Diamonds.csv (**http://course1.winona.edu/bdeppa/dsciwork.html)  
These data contain the 4C’s, depth (%), table (%), and price for a sample of n = 2,690 diamonds. There are additional columns containing the natural logarithm of the price (Log Price), the difference between the Table(%) and Depth (%) values as well as their ratio (Table/Depth), i.e. two new terms created by using functions of these predictors. Our goal is to develop a model for the diamond price using the available predictors and terms.

**Variable Descriptions:**

* Price price of diamond ($)
* Carat Carat Size – carat weight of the diamond (e.g. 0.5 = ½ carat)
* Color D,E,F,G,H,I,J,K (ordinal, lowest = K, highest = D) \*see diagram above
* Clarity (ordinal, lowest = *SI2*, highest = *IF*)
* Depth depth (%)
* Table table (%)
* Cut good, very good, excellent, ideal
* TDdiff (Table-Depth
* TDratio Table/Depth,, i.e. the ratio of table to depth
* Test *– used later when we discuss cross-validation for MLR models*

First we will read these data into R-Studio using the read.table command.

> Diamonds = read.table(file.choose(),header=T,sep=",")

> names(Diamonds)

[1] "Price" "Carats" "Color" "Clarity" "Depth" "Table" "Cut"

[8] "TDdiff" "TDratio" "Test"

> head(Diamonds,6)

Price Carats Color Clarity Depth Table Cut TDdiff TDratio Test

1 1000 0.30 E VVS1 60.0 59 Excellent -1.0 0.9833333 0

2 1000 0.44 E VS2 61.9 58 Excellent -3.9 0.9369951 1

3 1000 0.31 E VVS1 61.3 58 Excellent -3.3 0.9461664 0

4 1000 0.66 K SI1 62.8 57 Excellent -5.8 0.9076433 2

5 1000 0.47 H VS2 59.1 64 Very Good 4.9 1.0829103 2

6 1000 0.40 G VS1 62.0 59 Excellent -3.0 0.9516129 2

> str(Diamonds)  
'data.frame': 2690 obs. of 10 variables:

$ Price : int 1000 1000 1000 1000 1000 1000 1000 1000 1001 1001 ...

$ Carats : num 0.3 0.44 0.31 0.66 0.47 0.4 0.36 0.52 0.53 0.43 ...

$ Color : Factor w/ 8 levels "D","E","F","G",..: 2 2 2 8 5 4 1 5 1 3 ...

$ Clarity: Factor w/ 7 levels "IF","SI1","SI2",..: 6 5 6 2 5 4 5 3 3 5 ...

$ Depth : num 60 61.9 61.3 62.8 59.1 62 61.3 61.7 59.4 61.5 ...

$ Table : int 59 58 58 57 64 59 57 61 59 60 ...

$ Cut : Factor w/ 4 levels "Excellent","Good",..: 1 1 1 1 4 1 1 4 4 1 ...

$ TDdiff : num -1 -3.9 -3.3 -5.8 4.9 -3 -4.3 -0.7 -0.4 -1.5 ...

$ TDratio: num 0.983 0.937 0.946 0.908 1.083 ...

$ Test : int 0 1 0 2 2 2 0 0 0 2 ...

> summary(Diamonds)

Price Carats Color Clarity Depth

Min. : 1000 Min. :0.3000 E :504 IF :144 Min. :56.40

1st Qu.: 1801 1st Qu.:0.6000 F :431 SI1 :624 1st Qu.:61.00

Median : 3604 Median :0.9000 G :396 SI2 :530 Median :61.90

Mean : 3971 Mean :0.8701 H :394 VS1 :392 Mean :61.71

3rd Qu.: 5544 3rd Qu.:1.0600 I :316 VS2 :460 3rd Qu.:62.50

Max. :10000 Max. :2.0200 D :277 VVS1:269 Max. :64.30

(Other):372 VVS2:271

Table Cut TDdiff TDratio

Min. :53.00 Excellent:1276 Min. :-10.800 Min. :0.8307

1st Qu.:56.00 Good : 165 1st Qu.: -5.800 1st Qu.:0.9076

Median :58.00 Ideal : 185 Median : -4.200 Median :0.9325

Mean :57.86 Very Good:1064 Mean : -3.851 Mean :0.9382

3rd Qu.:59.00 3rd Qu.: -2.200 3rd Qu.:0.9636

Max. :65.00 Max. : 7.600 Max. :1.1348

It is always a good idea to summarize a data frame when you read it in to make sure the data types are what you think they are. You should see a table with levels and frequencies for categorical/nominal variables and summary statistics (mean, median, etc.) for numeric variables. For numeric variables such as price and carats you could use the num.summary(x) function you saw earlier in the workshop.

Test

Min. :0.0000

1st Qu.:0.0000

Median :0.0000

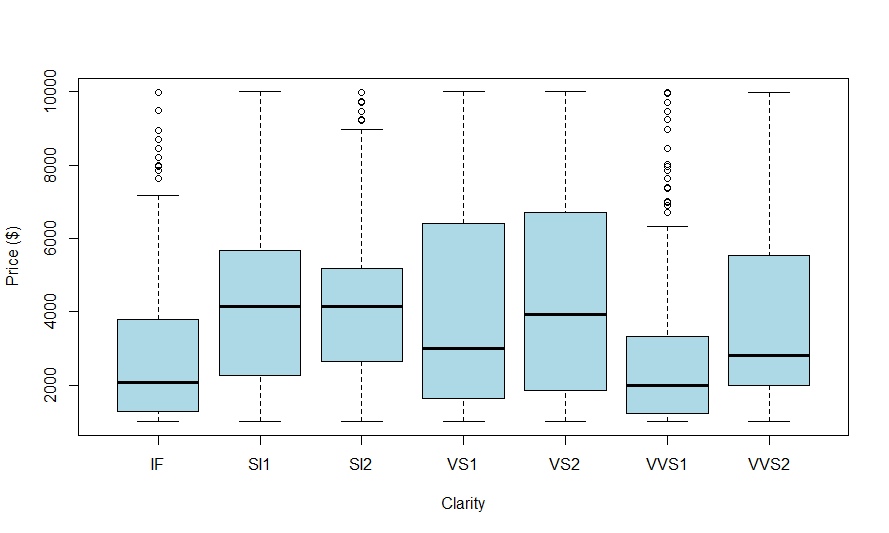
Mean :0.6004

3rd Qu.:1.0000

Max. :2.0000

We begin by considering the regression of Price ($) on Clarity . Below is a descriptive analysis of the conditional distribution of Price ($) given Clarity using boxplots**.**

> boxplot(Price~Clarity,col="lightblue",xlab="Clarity",  
ylab="Price ($)",data=Diamonds)



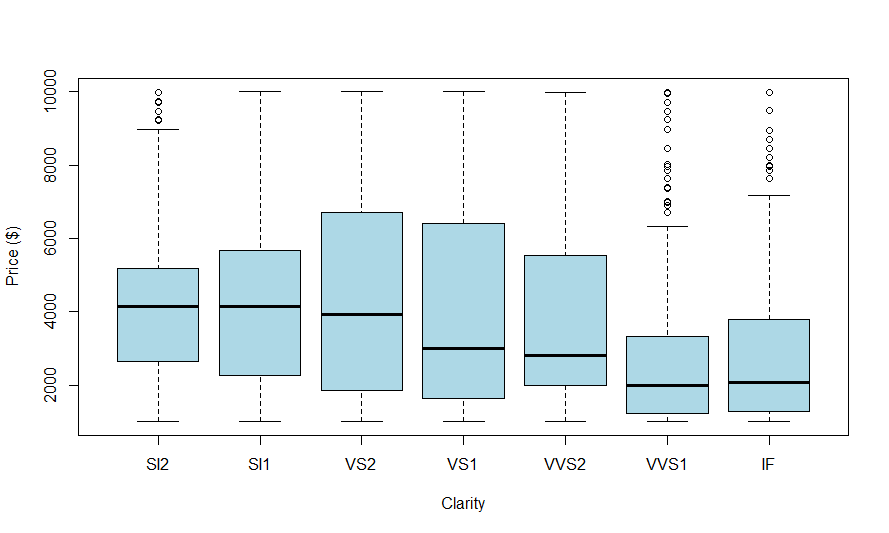
Notice: The ordering of the clarities is lost as R has put them in alphanumeric order!

The command below will put the levels of Clarity in the correct order and saves them in new variable called clarity.

> clarity = ordered(Diamonds$Clarity,levels=c("SI2","SI1","VS2","VS1",

"VVS2","VVS1","IF"))

> boxplot(Price~clarity,data=Diamonds2,col="lightblue",xlab="Clarity",  
ylab="Price ($)")

  
Does something seem counterintuitive here?

Summary statistics confirm what we see evidence of in the boxplot, there is a decrease in the typical diamond price as clarity increases!

> by(Diamonds$Price,clarity,summary)

Diamonds$Clarity: SI2

Min. 1st Qu. Median Mean 3rd Qu. Max.

1000 2660 4136 4245 5189 9996

------------------------------------------------------------

Diamonds$Clarity: SI1

Min. 1st Qu. Median Mean 3rd Qu. Max.

1000 2274 4135 4235 5662 9998

------------------------------------------------------------

Diamonds$Clarity: VS2

Min. 1st Qu. Median Mean 3rd Qu. Max.

1000 1844 3942 4340 6706 9999

------------------------------------------------------------

Diamonds$Clarity: VS1

Min. 1st Qu. Median Mean 3rd Qu. Max.

1000 1628 3004 4075 6380 10000

------------------------------------------------------------

Diamonds$Clarity: VVS2

Min. 1st Qu. Median Mean 3rd Qu. Max.

1001 2002 2800 3839 5538 9988

------------------------------------------------------------

Diamonds$Clarity: VVS1

Min. 1st Qu. Median Mean 3rd Qu. Max.

1000 1215 2002 2742 3337 9985

------------------------------------------------------------

Diamonds$Clarity: IF

Min. 1st Qu. Median Mean 3rd Qu. Max.

1002 1295 2079 2909 3776 9978

Modeling Diamond Price using only Clarity Levels  
As a first MLR model we will fit the following using only dummy terms for Clarity:

where

> lm1 = lm(Price~Clarity,data=Diamonds)

> summary(lm1)

Call:

lm(formula = Price ~ Clarity, data = Diamonds)

Residuals:

Min 1Q Median 3Q Max

-3339.6 -1829.1 -397.8 1435.9 7243.2

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 2909.2 197.2 14.751 < 2e-16 \*\*\*

ClaritySI1 1326.1 218.8 6.061 1.54e-09 \*\*\*

ClaritySI2 1335.9 222.4 6.007 2.15e-09 \*\*\*

ClarityVS1 1166.1 230.6 5.056 4.56e-07 \*\*\*

ClarityVS2 1430.4 226.0 6.330 2.87e-10 \*\*\*

ClarityVVS1 -167.4 244.4 -0.685 0.493336

ClarityVVS2 929.3 244.1 3.808 0.000143 \*\*\*

---

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

Residual standard error: 2367 on 2683 degrees of freedom

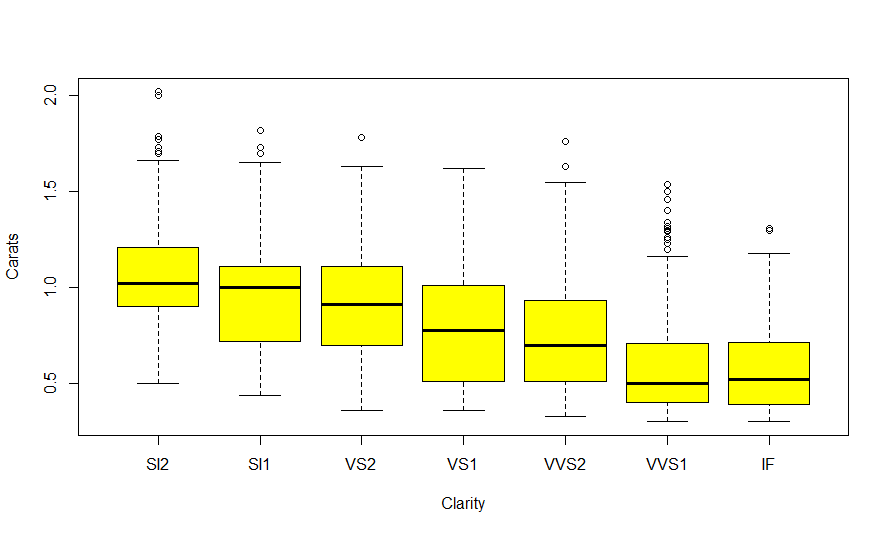
Multiple R-squared: 0.04595, Adjusted R-squared: 0.04382

F-statistic: 21.54 on 6 and 2683 DF, p-value: < 2.2e-16

Again we see that for the most part the predicted price of a diamond increases with decreasing clarity! Why do you think this happening?

Below is an examination of the conditional distribution of given .

> plot(Carats~clarity,data=Diamonds,xlab="Clarity",  
ylab="Carat Size",col="yellow")



How does this help explain the counterintuitive relationship between Price and Clarity?

Clearly knowing the clarity alone is not enough, as we can clearly see that the highest clarity diamonds in these data are also generally much smaller than the lower grade diamonds.

Questions:

1. Fit the MLR model using both the dummy terms for Clarity and Carats as terms in the model and summarize it. To do this in R,

> lm2 = lm(Price~Carats+Clarity,data=Diamonds)  
 > summary(lm2)

What would you say about the role of diamond clarity now?

1. Interpret the estimated coefficients for Carats and Clarity.

> lm2 = lm(Price~Carats+Clarity,data=Diamonds)  
> summary(lm2)

Call:

lm(formula = Price ~ Carats + Clarity, data = Diamonds)

Residuals:

Min 1Q Median 3Q Max

-3594.1 -509.5 -48.7 373.5 4762.5

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1469.01 96.81 -15.174 < 2e-16 \*\*\*

Carats 7458.50 71.12 104.868 < 2e-16 \*\*\*

ClaritySI1 -1536.33 100.67 -15.261 < 2e-16 \*\*\*

ClaritySI2 -2094.38 103.79 -20.180 < 2e-16 \*\*\*

ClarityVS1 -542.34 103.42 -5.244 1.7e-07 \*\*\*

ClarityVS2 -937.88 102.60 -9.141 < 2e-16 \*\*\*

ClarityVVS1 -170.00 108.22 -1.571 0.1163

ClarityVVS2 -235.66 108.65 -2.169 0.0302 \*

---

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

Residual standard error: 1048 on 2682 degrees of freedom

Multiple R-squared: 0.8129, Adjusted R-squared: 0.8125

F-statistic: 1665 on 7 and 2682 DF, p-value: < 2.2e-16

The carat size adjusted coefficients for diamond clarity all show that when comparing diamonds of the same carat weight the lower grade diamonds have prices smaller than that of internally flawless diamonds, e.g. a diamond will cost about $2094.38 less than a diamond of the same carat weight. However, it is also possible that the effect of carat weight differs for each clarity type, e.g. a 0.25 carat increase in carat weight for an diamond might be much more than the same carat weight increase in a diamond. If this is the case then we need to include an ***interaction term*** in our model for price between carat weight and clarity.

> lm3 = lm(Price~Carats\*Clarity,data=Diamonds)

> summary(lm3)

Residuals:

Min 1Q Median 3Q Max

The Carats\*Clarity term will add terms for Carats (1 term) and Clarity (6 dummy terms) as well as their product (6 additional interaction terms).

-3555.4 -503.3 -42.1 374.6 4704.8

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1608.20 221.57 -7.258 5.12e-13 \*\*\*

Carats 7695.61 347.12 22.170 < 2e-16 \*\*\*

ClaritySI1 -1309.86 268.71 -4.875 1.15e-06 \*\*\*

ClaritySI2 -1263.88 282.37 -4.476 7.92e-06 \*\*\*

ClarityVS1 -659.17 266.71 -2.471 0.013519 \*

ClarityVS2 -913.41 272.32 -3.354 0.000807 \*\*\*

ClarityVVS1 -159.64 271.88 -0.587 0.557138

ClarityVVS2 -386.98 281.54 -1.375 0.169395

Carats:ClaritySI1 -327.02 378.37 -0.864 0.387502

Carats:ClaritySI2 -897.44 382.84 -2.344 0.019143 \*

Carats:ClarityVS1 76.61 386.54 0.198 0.842905

Carats:ClarityVS2 -110.29 385.00 -0.286 0.774550

Carats:ClarityVVS1 -17.79 425.08 -0.042 0.966621

Carats:ClarityVVS2 153.78 409.67 0.375 0.707416

---

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

Residual standard error: 1044 on 2676 degrees of freedom

Multiple R-squared: 0.8147, Adjusted R-squared: 0.8138

F-statistic: 905.1 on 13 and 2676 DF, p-value: < 2.2e-16

> anova(lm3)

Analysis of Variance Table

Response: Price

Df Sum Sq Mean Sq F value Pr(>F)

Carats 1 1.1697e+10 1.1697e+10 10725.0117 < 2.2e-16 \*\*\*

Clarity 6 1.1081e+09 1.8468e+08 169.3412 < 2.2e-16 \*\*\*

Carats:Clarity 6 2.7870e+07 4.6449e+06 4.2591 0.0002815 \*\*\*

Residuals 2676 2.9184e+09 1.0906e+06

---

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

plot(Price~Carats,data=Diamonds)

abline(-1608.2,7695.61,lwd=3,col=1)

abline(-1608.2 - 1309.86,7695.61-327.02,lwd=3,col=2)

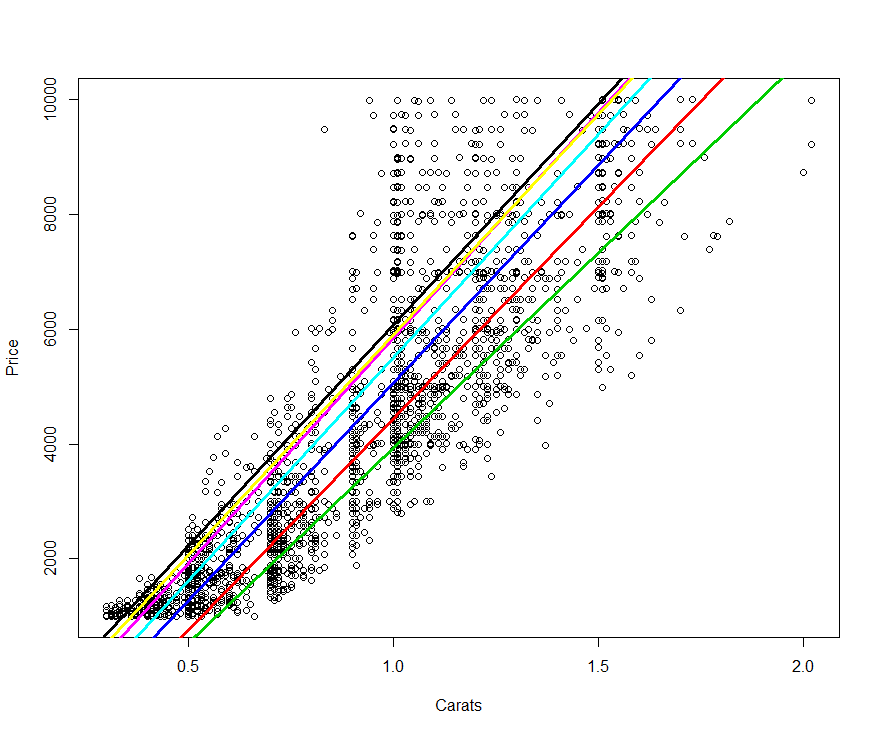
abline(-1608.2 - 1263.88,7695.61-897.44,lwd=3,col=3)

abline(-1608.2 - 913.41,7695.61-110.29,lwd=3,col=4)

abline(-1608.2 - 659.17,7695.61+76.61,lwd=3,col=5)

abline(-1608.2 - 386.98,7695.61+153.78,lwd=3,col=6)

abline(-1608.2 - 159.64,7695.61-17.79,lwd=3,col=7)



Comments:

Well this model seems to give a much better representation of the relationship between clarity, carat size, and price. However is this a good model? Does it fit the training data well? Will it predict the price of diamonds well? We will explore the answers to these questions next.

**Using Residual Plots to Assess Model Adequacy**

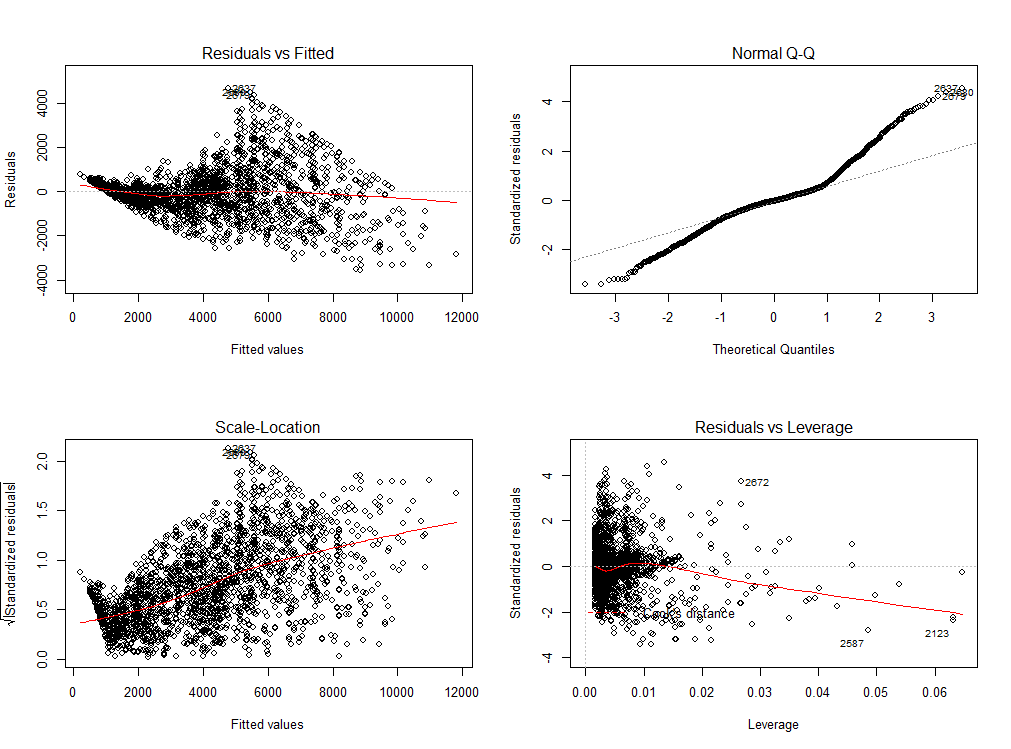
Assessing the adequacy of a fitted model generally involves plotting the residuals,

vs. the fitted values (or any other function of the model terms), plotting the absolute value of the residuals or squared root of the absolute residuals vs. the fitted values , and examining the distribution of residuals.

**Ideally these plots will look similar to those shown below.**

This sequence of plots is easily obtained from a fitted MLR model in R by simply typing plot(*model name*).

> plot(lm3)



These plots display several model deficiencies with our current MLR model.

Question: What are they?

**Response Transformations**Again the general form of a multiple regression model is given by

and typically we assume . However, if examination of plots based upon the residuals provides visual evidence that the assumed mean function is not correct and/or the then it may be the case that a transformation of the response, satisfies the assumed model in terms of the mean and variance. Normality of the response (and to some degree for the numeric predictors) is a desirable property in regression, thus we may consider whether the distribution of is approximately normal in our choice of response transformation.

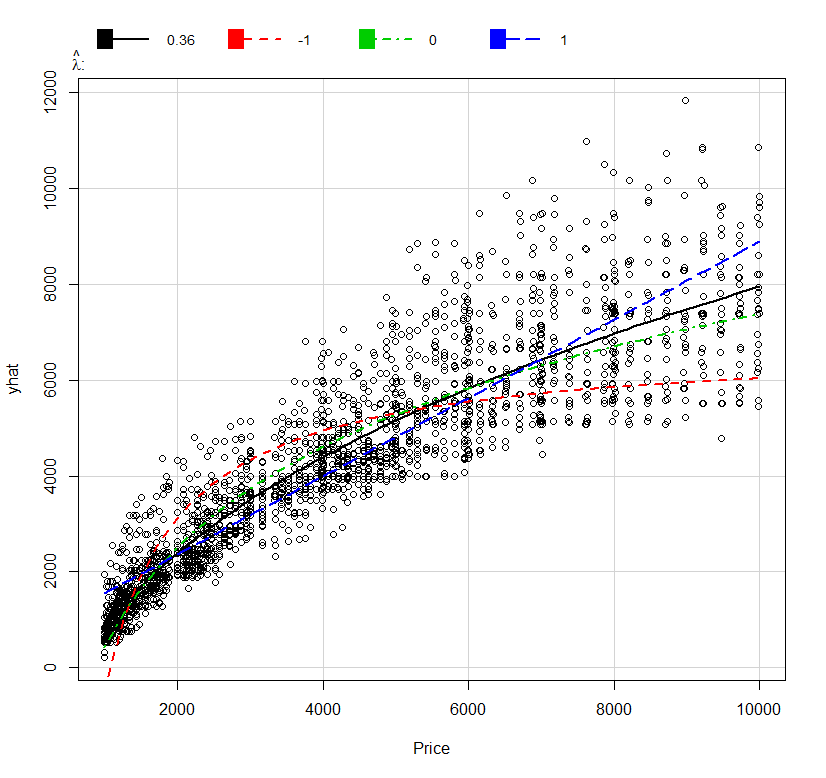
We can summarize theses reasons for transforming the response as follows.

**Transforming to Linearity -** In cases where we have clear visual evidence of curvature in the residual plots we can generally assume that  **,** i.e. are model does not reflect the relationship between the response and the predictors. However it may be possible that the assumed model holds when the response is transformed the assumed model holds, i.e. . If this works, then we have saved ourselves from having to change the terms in our model to account for the curvature.

**Transforming to Stabilize the Variance –** In cases where we have clear visual evidence from a plot of residuals vs. the fitted values and/or a non-constant variance plot () that it is possible a response transformation will give , i.e. is constant.

**Transforming to Normality –** In cases where the errors do not appear to be normally distributed, as evidenced by a normal quantile plot or histogram of the residuals , a transformation of the response, , can possibly improve normality of the errors and the response. Such a transformation will also imply that the conditional distribution of normal, in particular the conditional distribution . This transformation is primarily done for inference purposes in the case of MLR, however in predictive modeling a normally distributed response can be easier to predict accurately in general.

**Transformations for Improving Linearity (Inverse Response Plot)**Adding a smooth to a plot of the fitted values () vs. actual response values () can give a visual impression of the ***power transformation*** that might improve linearity. Below an inverse response plot for the MLR model for price based upon carat size, clarity, and their interaction.

> library(car)  
> lm3 = lm(Price~Carats\*Clarity,data=Diamonds)  
> inverseResponsePlot(lm3)  


Comments:

**Common Variance Stabilizing Transformations**

|  |  |
| --- | --- |
|  | **Comments** |
|  | Appropriate when for example when has a  Poisson distribution. The latter form is called the Freeman-Tukey deviate, and it gives better results if the are small or some . |
|  | Though most commonly used to achieve linearity, this is also a variance stabilizing transformation when . It can be appropriate if the errors are a percentage of the response, like , rather than an absolute deviation like units. |
|  | The inverse transformation stabilizes variance when  . It can be appropriate when responses are close to zero, but occasionally when large values occur. |
|  | This is usually called the *arcsine square-root* transformation. It stabilizes the variance when is a proportion between zero and one, but it can be used more generally if has a limited range by first transforming to the range (0,1) and then applying the transformation. To scale a variable use |

Residual Plot and NCV Plot for Current Diamond Price Model



**Transformations to Approximate Normality**

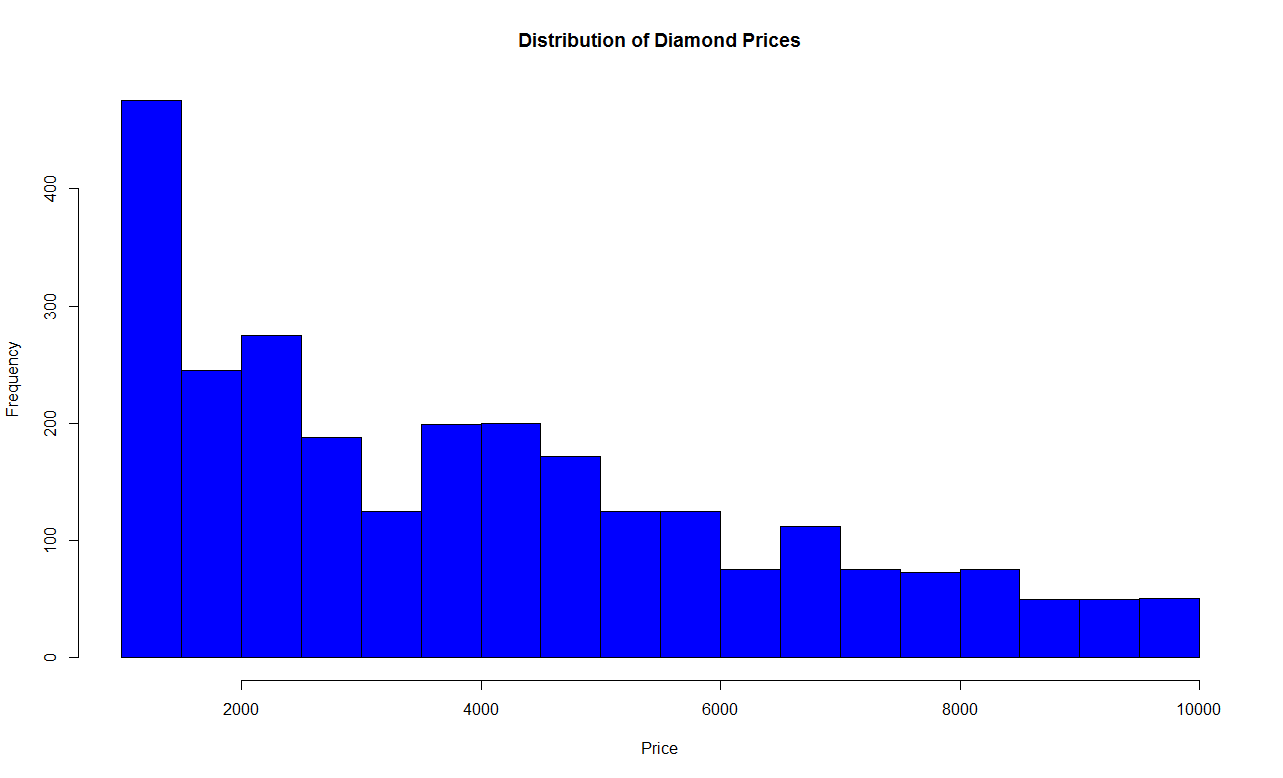
**Tukey’s Ladder of Powers** is useful when transforming a numeric variable using the power transformation family to have an approximately normal distribution in the transformed scale.One of the most common departures from normality that we see when working with continuous numeric data is skewness – either left or more commonly right.

***Ladder of Powers*** (Tukey, 1977) – choosing power transformations to remove skewness

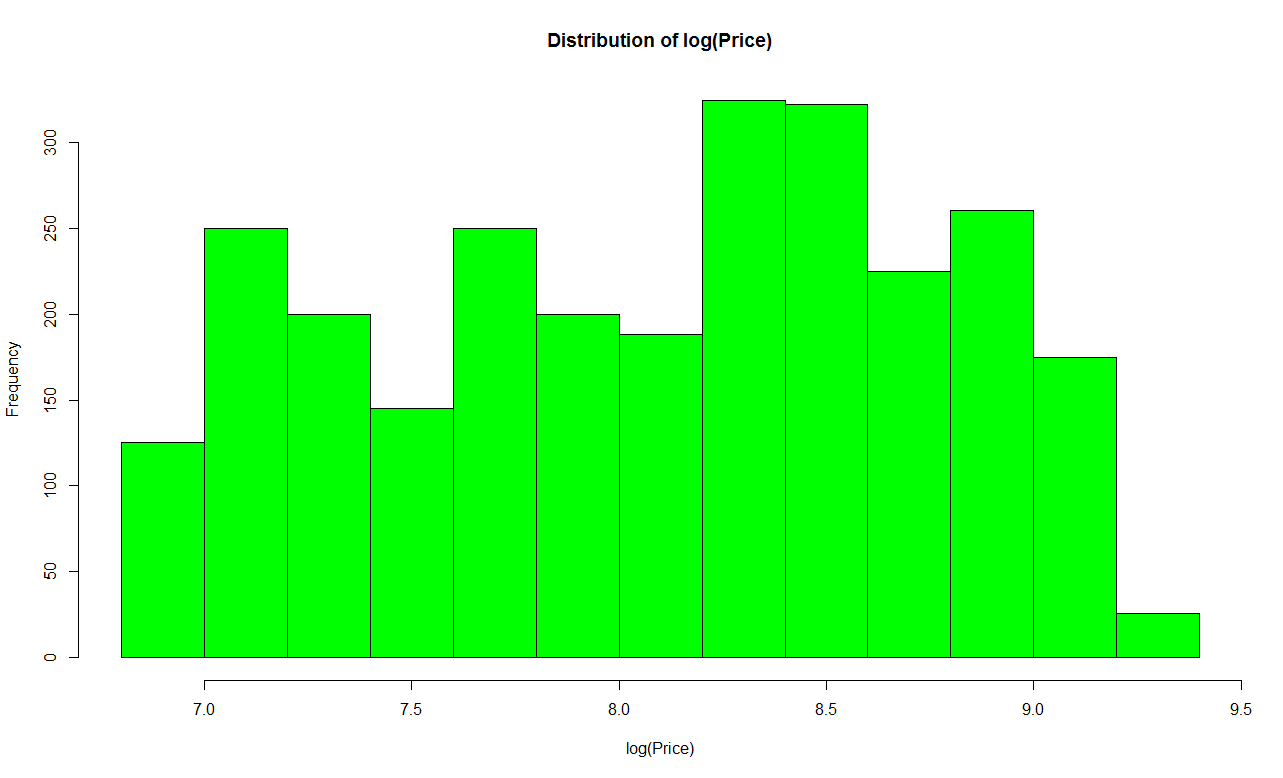
There is a numerical method for finding the “optimal” transformation to achieve approximate normality called the *Box-Cox Method*. It can be used find the transformation to achieve normality in the univariate and the multivariate case. In situations where the we have evidence of non-normality in the residuals from a fitted model , which will usually be accompanied by curvature and/or non-constant variation also, we generally start by using the Box-Cox method to find an optimal response transformation.

> hist(Diamonds$Price,xlab="Price",col="blue",

main="Distribution of Diamond Prices")



> hist(log(Diamonds$Price),xlab="log(Price)",

main="Distribution of log(Price)",col="green")  


Scaled-Power Transformation Family   
(Box-Cox Transformation)

Tukey Power Transformation Family

**Box-Cox Transformation Family & Method**

The Box-Cox method chooses the optimal by maximizing ,

(or minimizing )

Here is the residual sum of squares from fitting the model with as the response. The maximization is usually done by evaluating over a sequence of values from - 2 to 2.

We will now consider the use of the Box-Cox transformation procedure for finding an “optimal” transformation for diamond prices in these data.

> library(car)  
> results = powerTransform(Diamonds$Price)  
> summary(results)

bcPower Transformation to Normality

Est.Power Std.Err. Wald Lower Bound Wald Upper Bound

Diamonds$Price 0.1483 0.0322 0.0852 0.2114

Likelihood ratio tests about transformation parameters

LRT df pval

LR test, lambda = (0) 21.1972 1 4.143687e-06

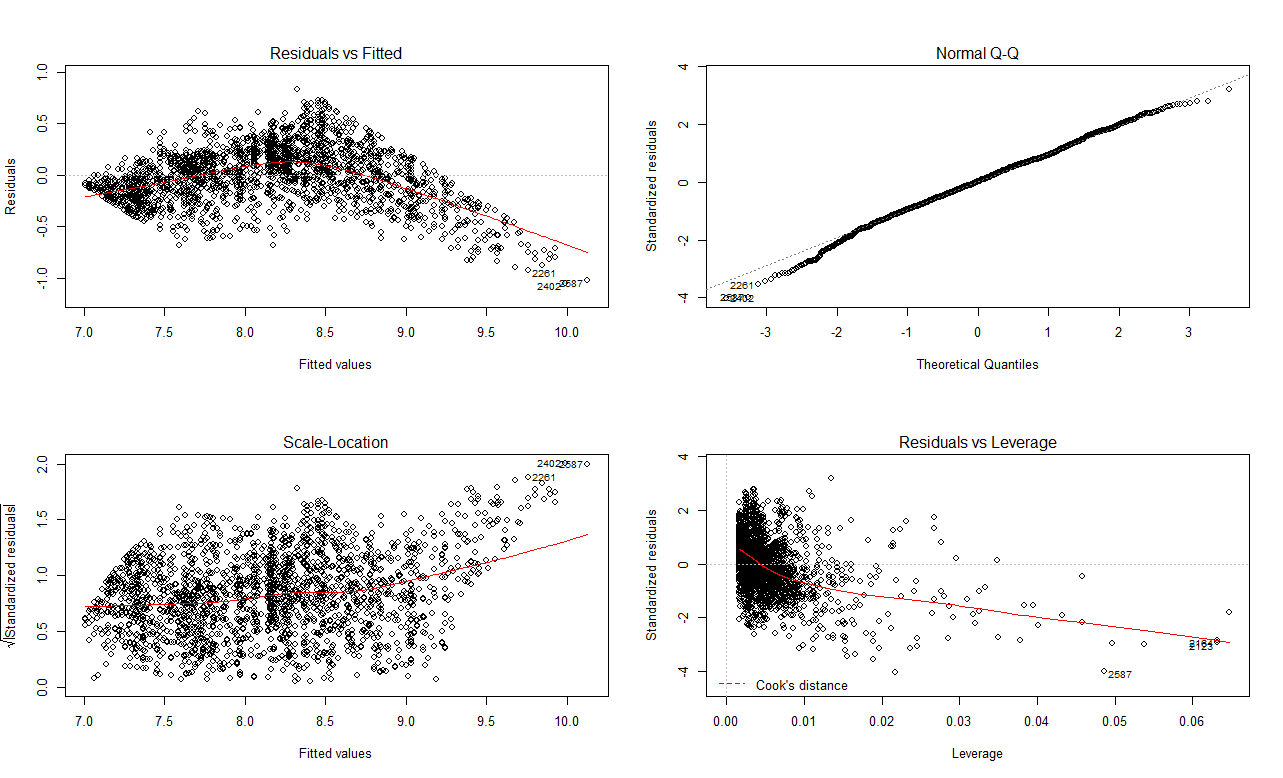
LR test, lambda = (1) 679.5061 1 0.000000e+00

The Box-Cox procedure recommends using which is not a standard transformation, i.e. it is not a rung on Tukey’s Ladder of Powers. Thus as the lower bound is near , we will opt for using a logarithmic transformation of price.

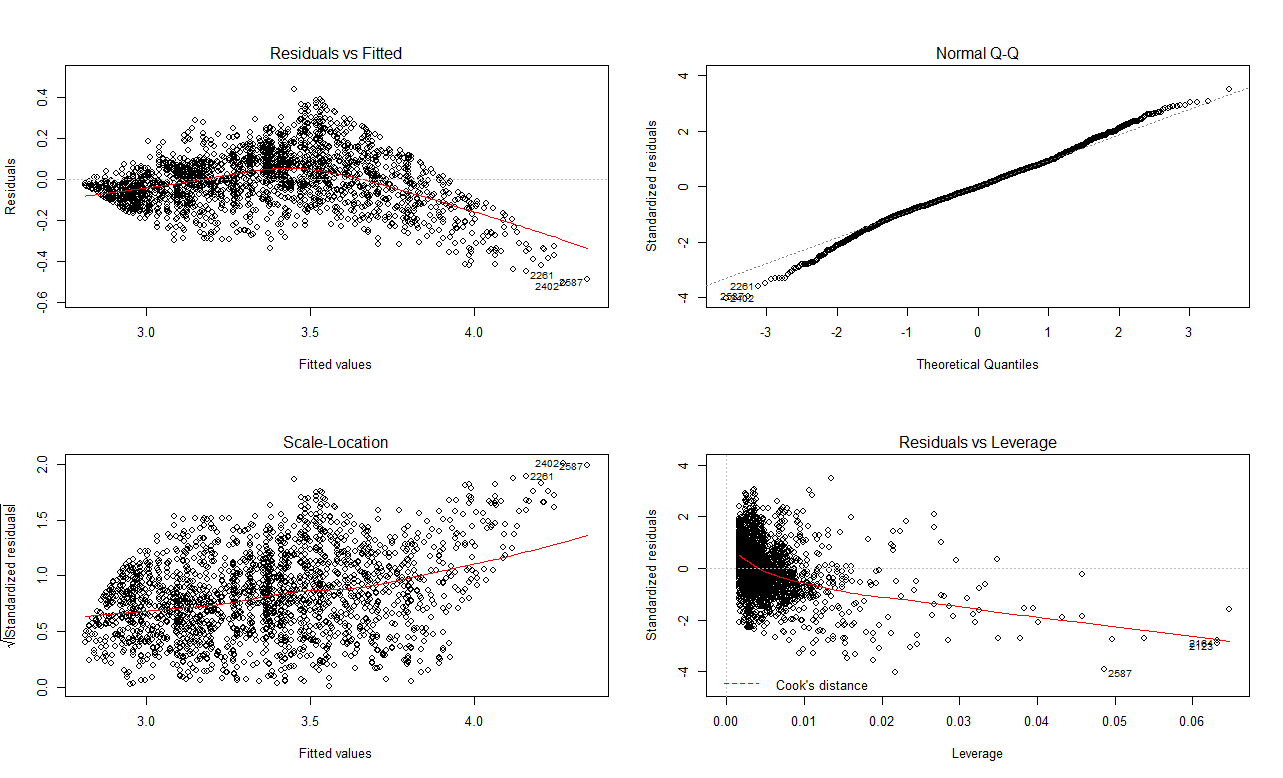
> temp = Diamonds

> temp$Price = log(Diamonds$Price)

> lm4 = lm(Price~Carats\*Clarity,data=temp)

> plot(lm4)

We can compare these plots to those using the recommended transformation from Box-Cox (.



There is virtually no difference so we will use as the response.

What should we consider doing now? Clearly, we have unaddressed curvature in the plot of the residuals vs. the fitted values. Let’s consider the relationship between log(Price) and carat size of the diamonds. The commands below will create a scatterplot of log(Price) vs. Carats, add a scatterplot smoother for each clarity type to the plot, and create a legend. For creating the legend we make use of the properly ordered version of diamond clarity.

Diamonds$Price = log(Diamonds$Price)

plot(Diamonds$Carat,Diamonds$Price,xlab="Carat Size",ylab="Price ($)")

lines(lowess(Diamonds$Carat[Diamonds$Clarity=="SI2"],  
Diamonds$Price[Diamonds$Clarity=="SI2"]),col=1,lty=1,lwd=3)

lines(lowess(Diamonds$Carat[Diamonds$Clarity=="SI1"],  
Diamonds$Price[Diamonds$Clarity=="SI1"]),col=2,lty=1,lwd=3)

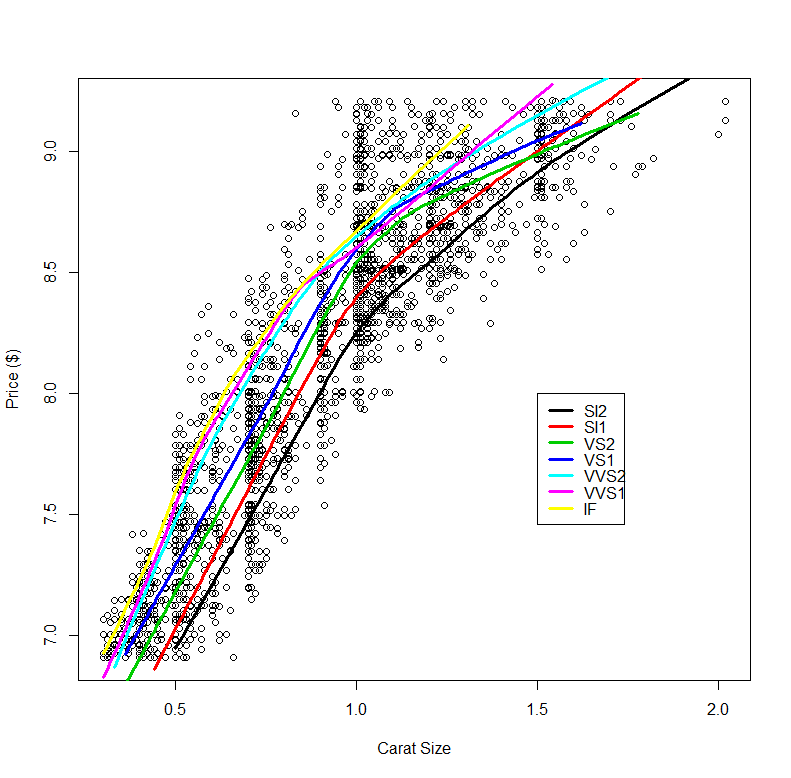
lines(lowess(Diamonds$Carat[Diamonds$Clarity=="VS2"],  
Diamonds$Price[Diamonds$Clarity=="VS2"]),col=3,lty=1,lwd=3)

lines(lowess(Diamonds$Carat[Diamonds$Clarity=="VS1"],  
Diamonds$Price[Diamonds$Clarity=="VS1"]),col=4,lty=1,lwd=3)

lines(lowess(Diamonds$Carat[Diamonds$Clarity=="VVS2"],  
Diamonds$Price[Diamonds$Clarity=="VVS2"]),col=5,lty=1,lwd=3)

lines(lowess(Diamonds$Carat[Diamonds$Clarity=="VVS1"],  
Diamonds$Price[Diamonds$Clarity=="VVS1"]),col=6,lty=1,lwd=3)

lines(lowess(Diamonds$Carat[Diamonds$Clarity=="IF"],  
Diamonds$Price[Diamonds$Clarity=="IF"]),col=7,lty=1,lwd=3)  
  
legend(1.5,8.0,levels(clarity),lty=1,lwd=3,col=1:7)



We can clearly see the curvature present suggesting we might consider adding a polynomial terms based on carat size to our model., perhaps quadratic or cubic.

The following two models are equivalent in every way. They both add linear and squared term based on carat size to the model and take their interactions with Clarity.

> lm5 = lm(Price~poly(Carats,2)\*Clarity,data=Diamonds)

> lm5.2 = lm(Price~(Carats+I(Carats^2))\*Clarity,data=Diamonds)

> summary(lm5)

Call:

lm(formula = Price ~ poly(Carats, 2) \* Clarity, data = Diamonds)

Residuals:

Min 1Q Median 3Q Max

-0.86366 -0.11814 0.01965 0.13087 0.69568

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 8.323891 0.028845 288.569 < 2e-16 \*\*\*

poly(Carats, 2)1 25.828205 2.092351 12.344 < 2e-16 \*\*\*

poly(Carats, 2)2 -13.013167 1.626757 -7.999 1.84e-15 \*\*\*

ClaritySI1 -0.400284 0.030461 -13.141 < 2e-16 \*\*\*

ClaritySI2 -0.506786 0.031434 -16.122 < 2e-16 \*\*\*

ClarityVS1 -0.159285 0.030760 -5.178 2.41e-07 \*\*\*

ClarityVS2 -0.266585 0.030533 -8.731 < 2e-16 \*\*\*

ClarityVVS1 -0.007132 0.034449 -0.207 0.83599

ClarityVVS2 -0.032402 0.031978 -1.013 0.31103

poly(Carats, 2)1:ClaritySI1 12.081872 2.172276 5.562 2.93e-08 \*\*\*

poly(Carats, 2)2:ClaritySI1 2.856121 1.719302 1.661 0.09679 .

poly(Carats, 2)1:ClaritySI2 11.276694 2.221849 5.075 4.13e-07 \*\*\*

poly(Carats, 2)2:ClaritySI2 4.945606 1.710077 2.892 0.00386 \*\*

poly(Carats, 2)1:ClarityVS1 7.969316 2.173173 3.667 0.00025 \*\*\*

poly(Carats, 2)2:ClarityVS1 2.227663 1.749541 1.273 0.20303

poly(Carats, 2)1:ClarityVS2 9.906343 2.165376 4.575 4.98e-06 \*\*\*

poly(Carats, 2)2:ClarityVS2 1.573812 1.745053 0.902 0.36721

poly(Carats, 2)1:ClarityVVS1 4.167730 2.352494 1.772 0.07657 .

poly(Carats, 2)2:ClarityVVS1 1.363659 1.863605 0.732 0.46440

poly(Carats, 2)1:ClarityVVS2 5.041043 2.227709 2.263 0.02372 \*

poly(Carats, 2)2:ClarityVVS2 1.511083 1.818622 0.831 0.40611

---

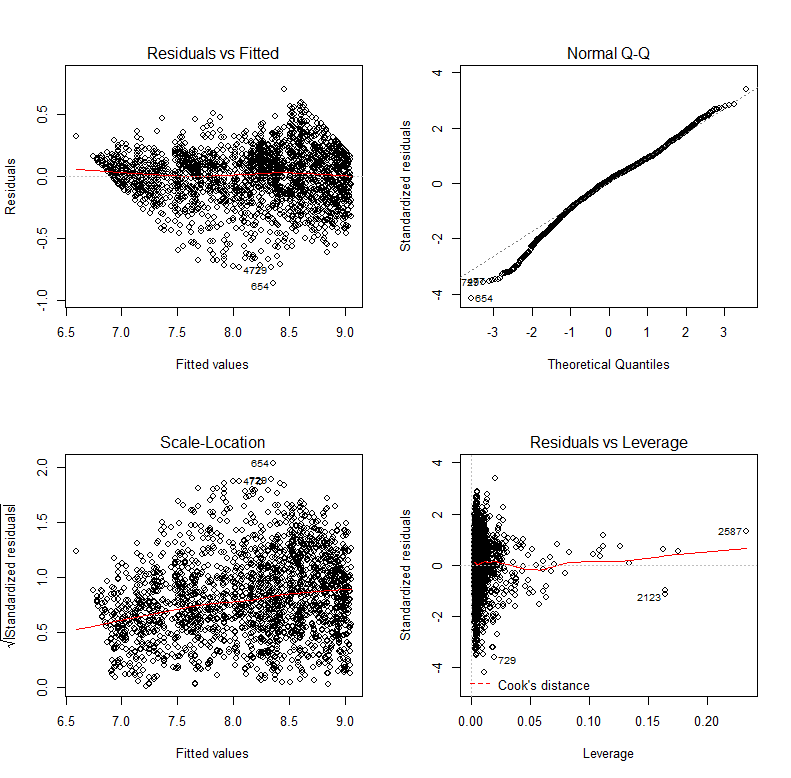
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2082 on 2669 degrees of freedom

Multiple R-squared: 0.9014, Adjusted R-squared: 0.9006

F-statistic: 1219 on 20 and 2669 DF, p-value: < 2.2e-16

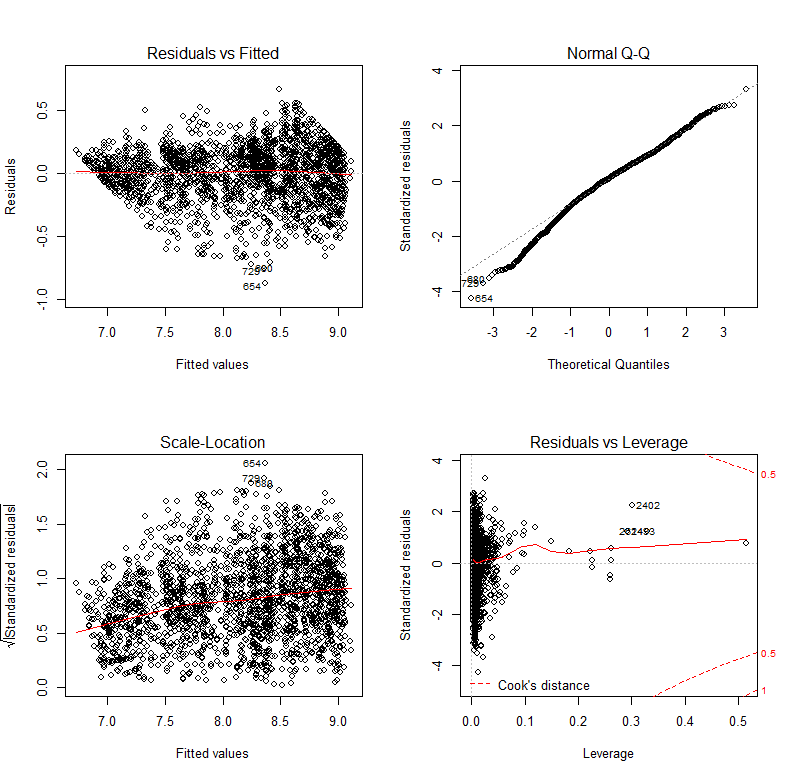
> plot(lm5)

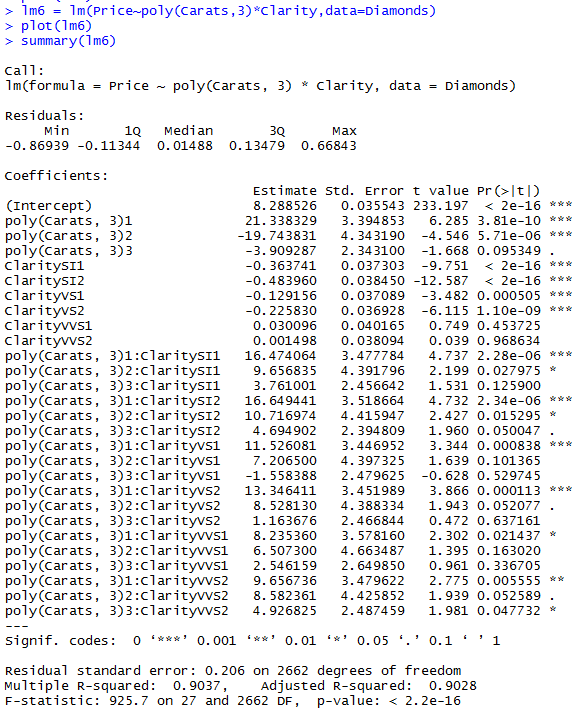


Adding cubic polynomial terms based on carat size gives.

> lm6 = lm(Price~poly(Carats,3)\*Clarity,data=Diamonds)

> plot(lm6)





**Tasks**

1. Form a training and test dataset from the diamonds data with the response log transformed.

Diamonds = read.table(file.choose(),header=T,sep=”,”)Diamonds$Price = log(Diamonds$Price)  
Diamonds.train = Diamonds[Diamonds$Test<2,]  
Diamonds.test = Diamonds[Diamonds$Test==2,]

**2.** Using the training data only, develop the “best” MLR model you can for   
 log(Price) using a cubic polynomial based upon carat size, Clarity, Color, Cut,   
 TDdiff, and TDratio only.

e.g. a model might look like (however as this model doesn’t include clarity is not   
 going to be good).

lm.poo = lm(Price~poly(Carats,3) + Color + Cut,data=Diamonds.train)

Examine residual plots, model summaries, and comment on the quality of your   
 model.

**3.** Predict log(Price) and then the Price for the diamonds in the test data. To do  
 use the code below:

ypredlog = predict(model name,newdata=Diamonds.test)  
 ypred = exp(ypredlog)

Construct a scatterplot the predicted prices vs. actual prices and the line to

the plot.

yactual = exp(Diamonds$Price)

plot(yactual,ypred,xlab=”Actual Price”,ylab=”Predicted Price”)

abline(0,1,lwd=3,col=”blue”)

**4.** Use the function PredAcc below to measure the accuracy of your predicted prices  
 of the diamonds in the test set.

PredAcc = function(y,ypred){

RMSEP = sqrt(mean((y-ypred)^2))

MAE = mean(abs(y-ypred))

MAPE = mean(abs(y-ypred)/y)\*100

cat("RMSEP\n")

cat("===============\n")

cat(RMSEP,"\n\n")

cat("MAE\n")

cat("===============\n")

cat(MAE,"\n\n")

cat("MAPE\n")

cat("===============\n")

cat(MAPE,"\n\n")

return(data.frame(RMSEP=RMSEP,MAE=MAE,MAPE=MAPE))

}

Then run function on your predictions for the test diamonds.

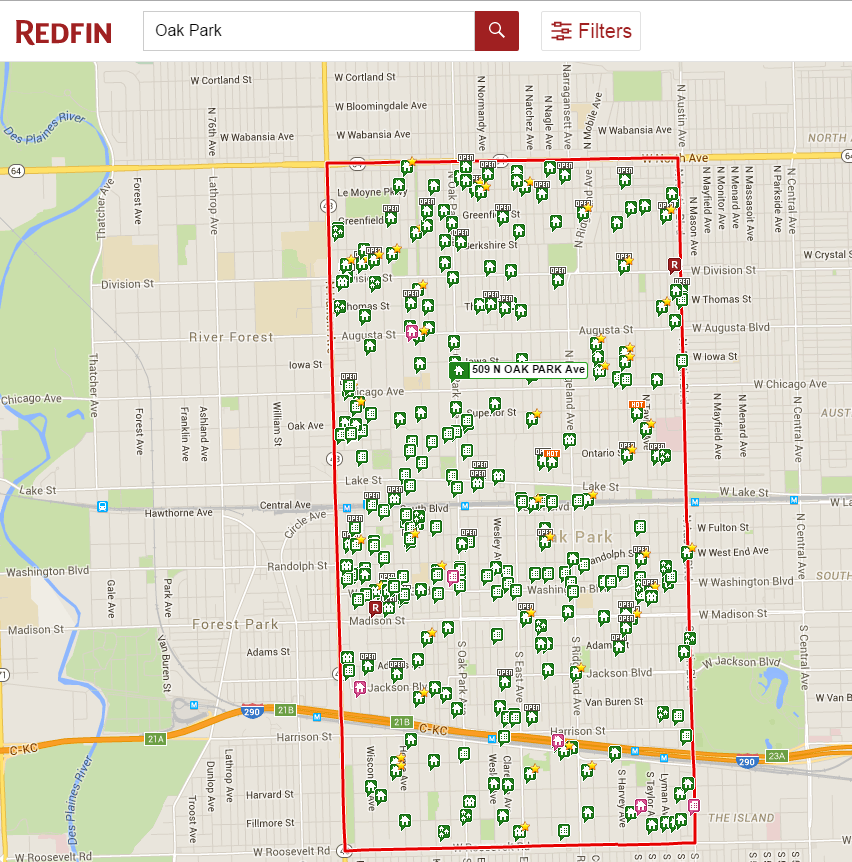
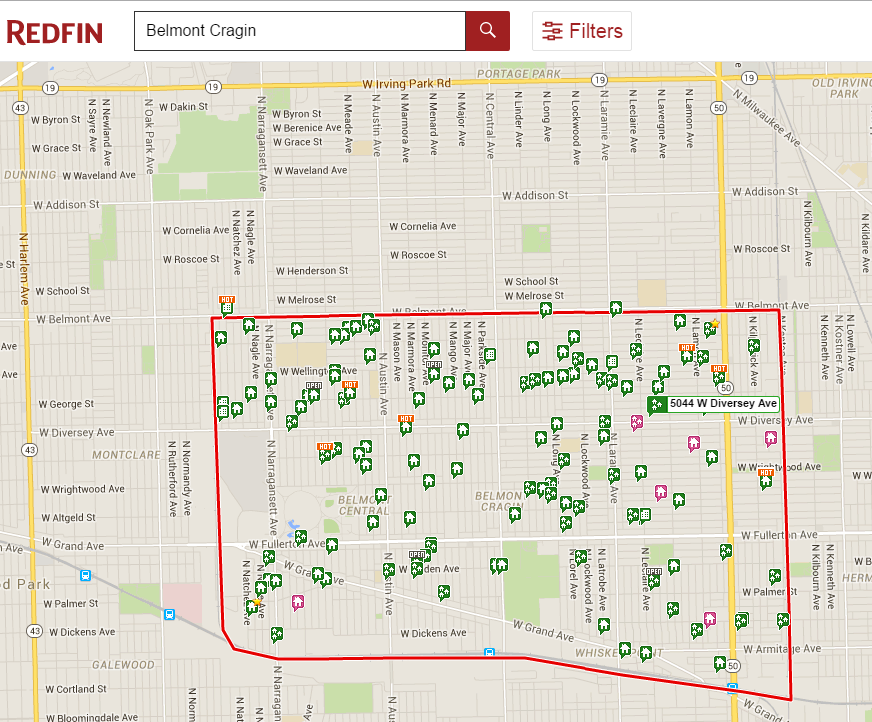
PredAcc(yactual,ypred)

Who built the best MLR model in the workshop?

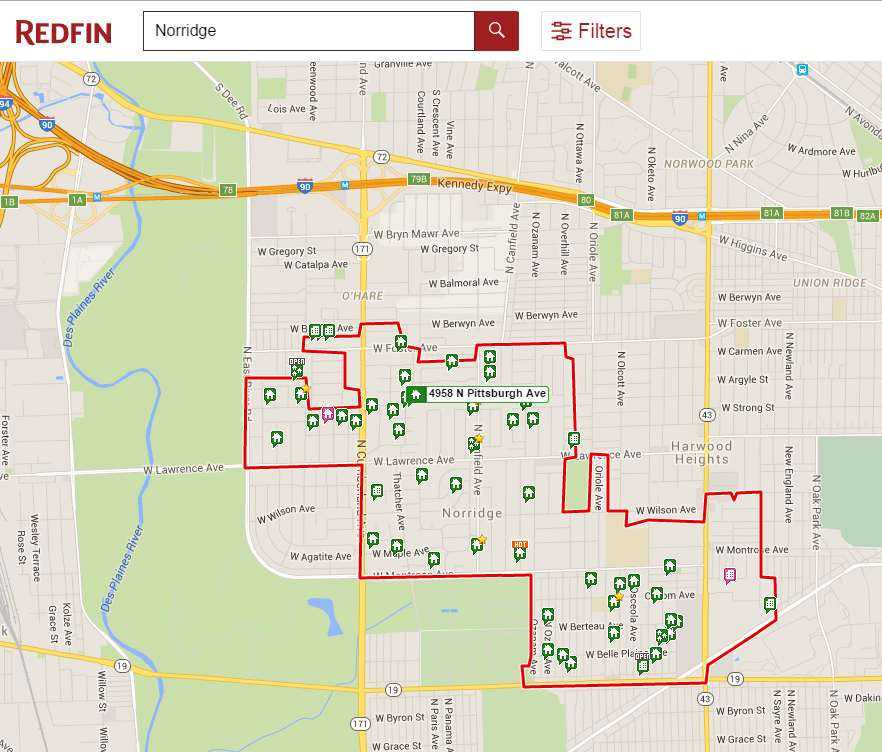
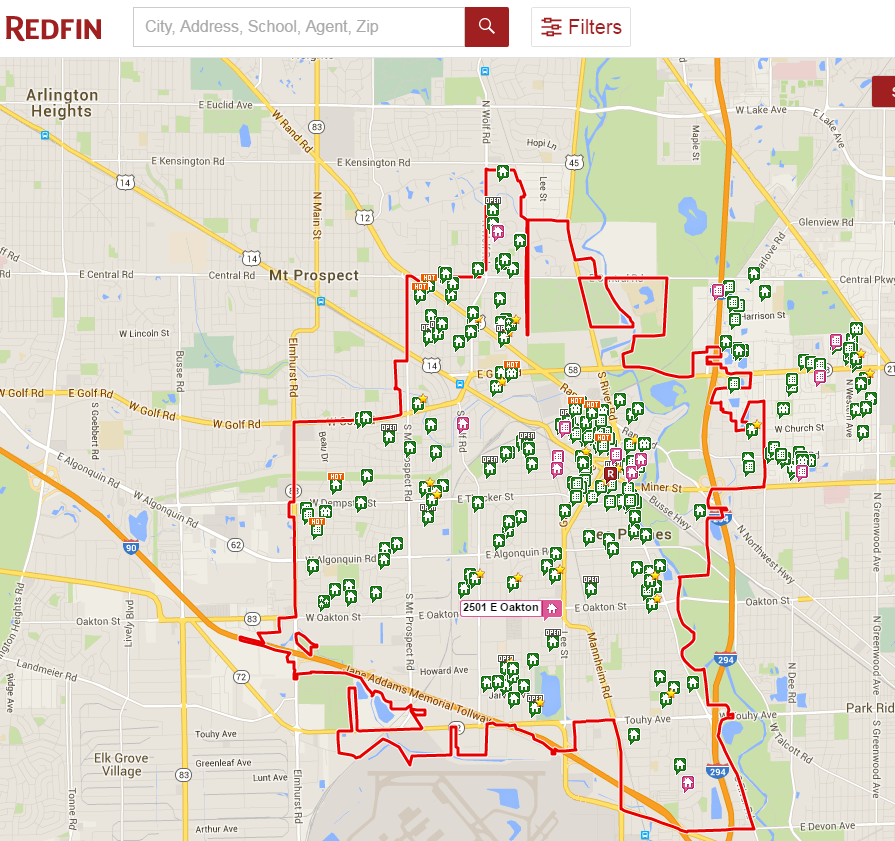
**Multiple Linear Regression Example 2: Chicago Area Home Prices**

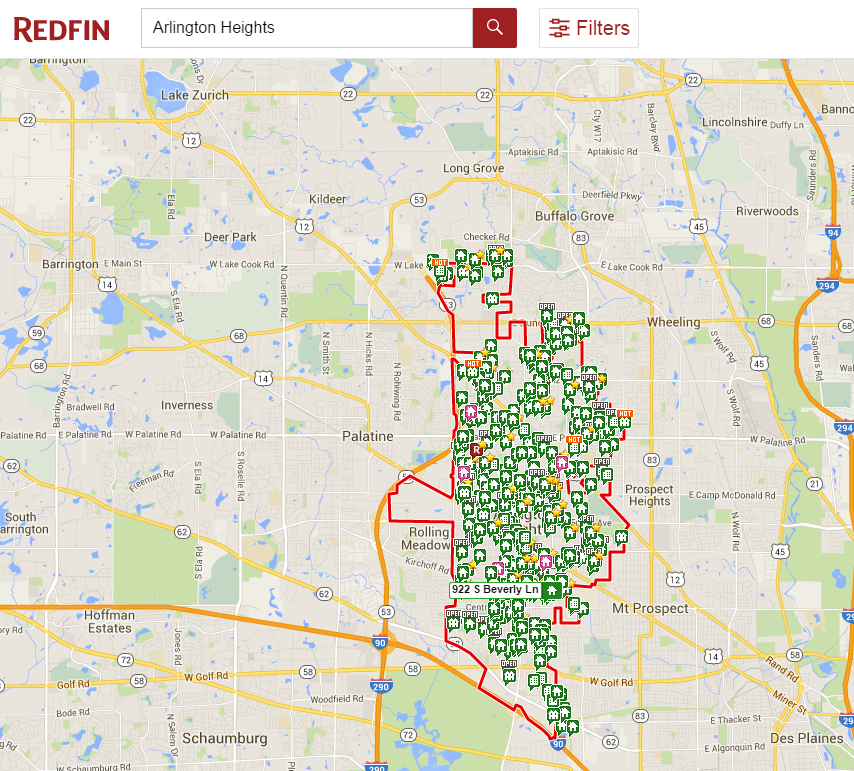
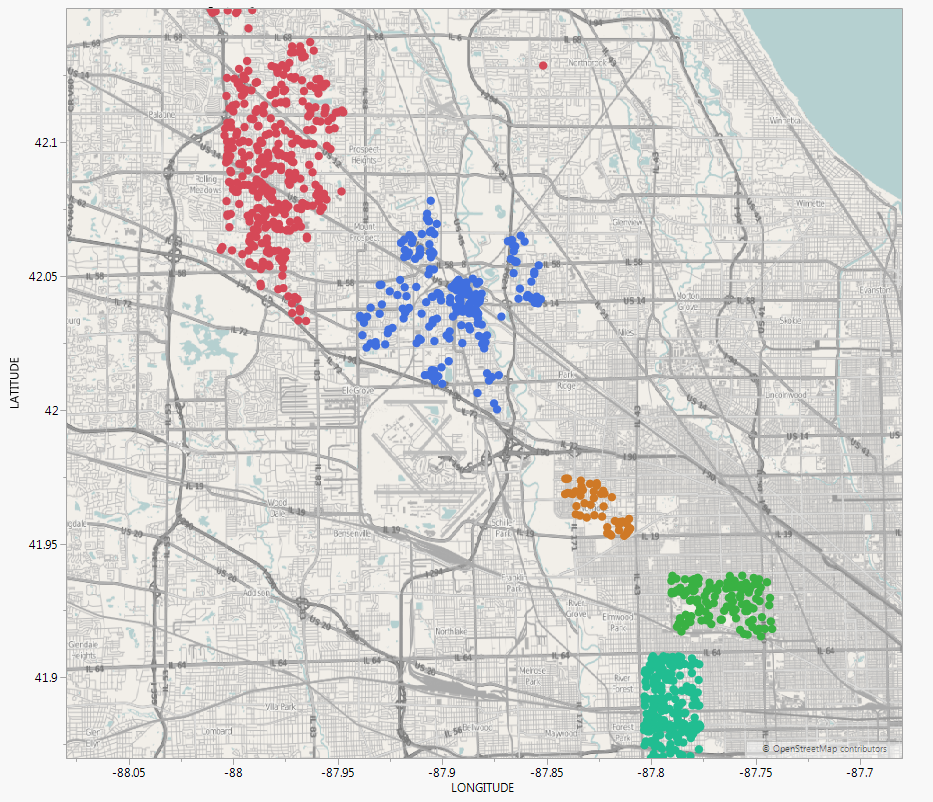
The real estate listing site Redfin ® ([www.redfin.com](http://www.redfin.com)) is a very helpful resource when it comes to buying and selling a home. It is also provides a means for data geeks to download the current list price and characteristics of homes currently on the market across the U.S. Below are some images from Redfin ® of the homes in a database I created for homes currently for sale around Oak Park, IL.

Oak Park Homes Belmont Cragin Homes

Norridge Homes Des Plaines Homes

Arlington Height Homes All Homes Sampled 

The training and test datasets are contain in the .csv files **ChiHomes(train).csv** and **ChiHomes(test).csv** on the workshop website.

> ChiTrain = read.table(file.choose(),header=T,sep=",")

> ChiTest = read.table(file.choose(),header=T,sep=",")

> names(ChiTrain)

[1] "Type" "City" "ZIP" "ListPrice" "BEDS"

[6] "BATHS" "ImputedSQFT" "ParkSpots" "HasGarage" "DOM"

[11] "BeenReduced" "SoldPrev" "LATITUDE" "LONGITUDE"

> head(ChiTrain)

Type City ZIP ListPrice BEDS BATHS

1 Townhouse Arlington Heights 60005 155000 3 1.5

2 Condo/Coop Arlington Heights 60005 159000 2 2.0

3 Single Family Residential Arlington Heights 60005 569900 4 2.5

4 Single Family Residential Arlington Heights 60004 1550000 4 4.5

5 Single Family Residential Arlington Heights 60004 575000 4 3.5

6 Single Family Residential Arlington Heights 60004 284900 3 1.0

ImputedSQFT ParkSpots HasGarage DOM BeenReduced SoldPrev LATITUDE LONGITUDE

1 866 2 None 853 No Yes 42.06740 -87.99018

2 1200 2 None 546 No Yes 42.04242 -87.97369

3 2184 1 Garage 381 Yes Yes 42.07701 -87.97501

4 4200 3 Garage 381 Yes Yes 42.09479 -87.95910

5 3500 2 Garage 343 Yes No 42.09479 -87.97472

6 1150 2 Garage 337 Yes Yes 42.10714 -88.00290

> str(ChiTrain)

'data.frame': 736 obs. of 14 variables:

$ Type : Factor w/ 4 levels "Condo/Coop","Multi-Family",..: 4 1 3 3 3 3 3 3 3 3 ...

$ City : Factor w/ 5 levels "Arlington Heights",..: 1 1 1 1 1 1 1 1 1 1 ...

$ ZIP : int 60005 60005 60005 60004 60004 60004 60004 60004 60005 60005 ...

$ ListPrice : int 155000 159000 569900 1550000 575000 284900 830000 437000 779900   
 509000 ...

$ BEDS : int 3 2 4 4 4 3 5 3 4 5 ...

$ BATHS : num 1.5 2 2.5 4.5 3.5 1 3.5 2.5 3 2.5 ...

$ ImputedSQFT: int 866 1200 2184 4200 3500 1150 3890 2980 2700 3000 ...

$ ParkSpots : int 2 2 1 3 2 2 3 2 2 2 ...

$ HasGarage : Factor w/ 2 levels "Garage","None": 2 2 1 1 1 1 1 1 1 1 ...

$ DOM : int 853 546 381 381 343 337 309 305 289 276 ...

$ BeenReduced: Factor w/ 2 levels "No","Yes": 1 1 2 2 2 2 1 2 1 1 ...

$ SoldPrev : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 1 1 1 ...

$ LATITUDE : num 42.1 42 42.1 42.1 42.1 ...

$ LONGITUDE : num -88 -88 -88 -88 -88 ...

**Variable Descriptions:**

* = type of home (Condo/Coop, Multi-Family, Single Family Residential, or Townhouse.
* = city/suburb (Arlington Hts., Chicago (Belmont Cragin), Des Plaines, Norridge, or Oak Park.
* = ZIP code 
* = current list price ($)
* = number of bedrooms
* = number of bathrooms
* = ***imputed*** square footage (ft2). For many of the homes sampled the square footage was not given, however the number of bedrooms and bathrooms was always given. To fill in the missing square footage of homes, I used MLR to estimate the square footage using the number of bedrooms and bathrooms as predictors. I used the predicted values from this model to fill in the missing square footage for the homes without. For homes that had square footage I obviously used the value given.
* = number of parking spots
* = home has a garage (Garage or None)
* = current number of days the home has been on the market.
* = has the price of home been reduced since first listed (Yes or No)
* = has the home been sold previously (Yes or No)
* = latitude of the home (degrees, accurate to hundred thousandths)
* = longitude of the home (degrees, accurate to hundred thousandths and negative because the U.S. is west of the Prime Meridian.

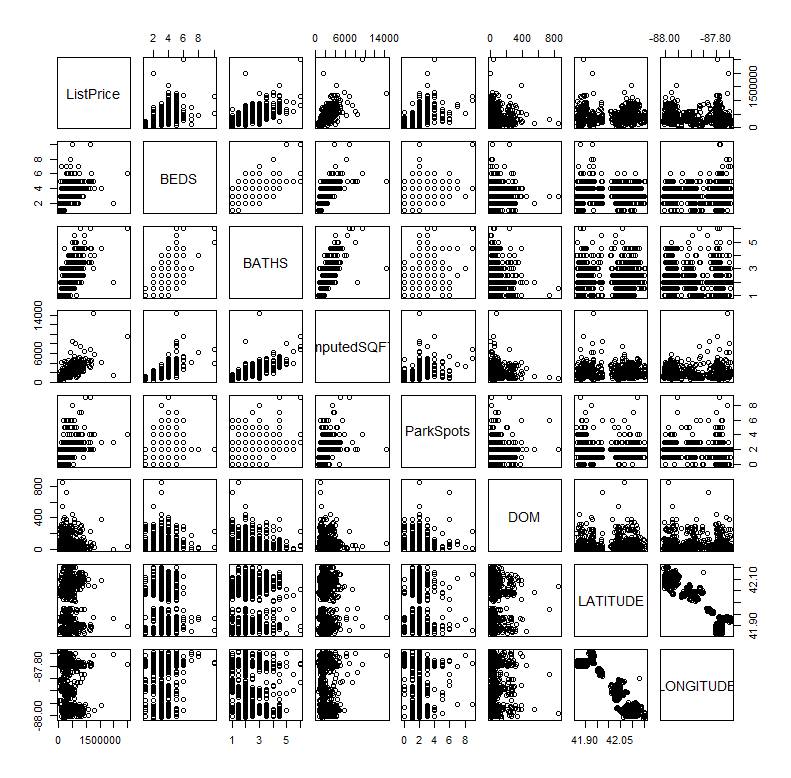
Our goal is develop a predictive model for list price using the available predictors. We will then examine the performance of our model by predicting the list price of the homes in the test data set. We can begin by fitting a base model using default terms for all predictors. This means dummy variable will be created for all nominal variables and the numeric predictors will simply be the predictors themselves. Also before beginning we need to convert ZIP code to a factor (i.e. categorical/nominal variable) as it should not be interpreted as numeric.

> ChiTrain$ZIP = as.factor(ChiTrain$ZIP)

> ChiTest$ZIP = as.factor(ChiTest$ZIP)

Before fitting a preliminary model we will examine a scatterplot matrix of the response and the number variables.

> pairs(ChiTrain[,c(4:8,10,13,14)])



If you want to fit a model for a response (Y) where all of the other variables in the data frame are potential predictors you can use model ***wild card*** notation (y ~ .) to specify the model. We will make use of this in several examples in the remainder of the workshop.

> base.lm = lm(ListPrice~.,data=ChiTrain)

> summary(base.lm)

Call:

lm(formula = ListPrice ~ ., data = ChiTrain)

Residuals:

Min 1Q Median 3Q Max

-493484 -71347 -3737 47646 1765111

Coefficients: (4 not defined because of singularities)

Estimate Std. Error t value Pr(>|t|)

(Intercept) -7.544e+06 3.298e+07 -0.229 0.819144

TypeMulti-Family 3.276e+04 3.653e+04 0.897 0.370082

TypeSingle Family Residential 1.224e+05 1.862e+04 6.574 9.47e-11 \*\*\*

TypeTownhouse -3.212e+04 2.503e+04 -1.283 0.199765

CityChicago -6.524e+04 1.218e+05 -0.536 0.592288

CityDes Plaines -1.008e+05 5.150e+04 -1.957 0.050722 .

CityNorridge -2.501e+04 7.848e+04 -0.319 0.750028

CityOak Park 1.034e+05 1.117e+05 0.926 0.354683

ZIP60005 1.799e+04 2.324e+04 0.774 0.439151

ZIP60016 5.069e+04 3.249e+04 1.560 0.119145

ZIP60018 NA NA NA NA

ZIP60301 -2.790e+04 5.328e+04 -0.524 0.600718

ZIP60302 4.917e+04 2.455e+04 2.003 0.045585 \*

ZIP60304 NA NA NA NA

ZIP60634 -2.323e+04 8.319e+04 -0.279 0.780151

ZIP60639 -1.558e+04 8.130e+04 -0.192 0.848076

ZIP60641 -7.187e+04 8.548e+04 -0.841 0.400722

ZIP60706 NA NA NA NA

ZIP60707 NA NA NA NA

BEDS -2.525e+04 7.685e+03 -3.285 0.001070 \*\*

BATHS 7.856e+04 8.892e+03 8.836 < 2e-16 \*\*\*

ImputedSQFT 9.973e+01 7.170e+00 13.909 < 2e-16 \*\*\*

ParkSpots 2.224e+04 5.489e+03 4.052 5.64e-05 \*\*\*

HasGarageNone -5.403e+04 1.634e+04 -3.306 0.000994 \*\*\*

DOM 1.186e+02 6.010e+01 1.974 0.048809 \*

BeenReducedYes -2.346e+04 1.099e+04 -2.134 0.033198 \*

SoldPrevYes -1.432e+04 1.011e+04 -1.416 0.157262

LATITUDE 3.125e+05 4.132e+05 0.756 0.449758

LONGITUDE 6.381e+04 3.420e+05 0.187 0.852021

---

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

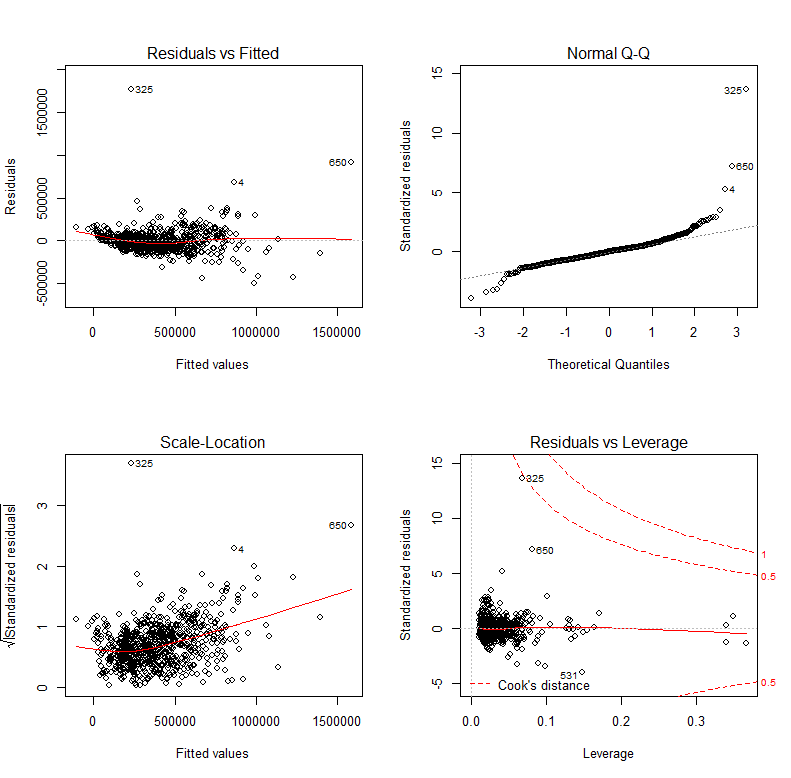
Residual standard error: 134100 on 711 degrees of freedom

Multiple R-squared: 0.7342, Adjusted R-squared: 0.7252

F-statistic: 81.83 on 24 and 711 DF, p-value: < 2.2e-16

> par(mfrow=c(2,2))

> plot(base.lm)



Comments on Model Adequacy:

> par(mfrow=c(1,1))

Also notice that the estimated coefficients for several of the ZIP code dummy variables are missing (NA) – see highlights above. Why do you think this is happening?

It is easy to confirm this by creating a table of ZIP code by City (or vise versa).

> table(ChiTrain$City,ChiTrain$ZIP)

60004 60005 60016 60018 60301 60302 60304 60634 60639 60641

Arlington Heights 178 95 0 0 0 0 0 0 0 0

Chicago 0 0 0 0 0 0 0 21 47 19

Des Plaines 0 0 135 24 0 0 0 0 0 0

Norridge 0 0 0 0 0 0 0 0 0 0

Oak Park 0 0 0 0 8 129 46 0 0 0

All of the ZIP codes are unique to the cities/suburbs in these data. Thus using both in our model is redundant. As city/suburb is more identifiable to users, we will opt to ignore ZIP code. Also some of the ZIP codes have low counts, e.g. 60707 has only 3 homes! City does not have this problem, see below.

60706 60707

Arlington Heights 0 0

Chicago 0 3

Des Plaines 0 0

Norridge 31 0

Oak Park 0 0

> table(ChiTrain$City)

Arlington Heights Chicago Des Plaines Norridge Oak Park

273 90 159 31 183

We will now fit the base model without ZIP code.

> base.lm2 = lm(ListPrice~.-ZIP,data=ChiTrain)

> summary(base.lm2)

Call:

lm(formula = ListPrice ~ . - ZIP, data = ChiTrain)

Residuals:

Min 1Q Median 3Q Max

-487733 -72437 -5323 50077 1783024

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.441e+07 3.148e+07 -0.458 0.64723

TypeMulti-Family 1.975e+04 3.555e+04 0.556 0.57865

TypeSingle Family Residential 1.144e+05 1.821e+04 6.283 5.76e-10 \*\*\*

TypeTownhouse -3.649e+04 2.489e+04 -1.466 0.14302

CityChicago -7.918e+04 8.634e+04 -0.917 0.35943

CityDes Plaines -5.857e+04 3.433e+04 -1.706 0.08840 .

CityNorridge -1.660e+04 6.806e+04 -0.244 0.80742

CityOak Park 1.479e+05 8.674e+04 1.705 0.08855 .

BEDS -2.447e+04 7.704e+03 -3.177 0.00155 \*\*

BATHS 7.892e+04 8.892e+03 8.876 < 2e-16 \*\*\*

ImputedSQFT 9.873e+01 7.132e+00 13.843 < 2e-16 \*\*\*

ParkSpots 2.308e+04 5.468e+03 4.221 2.74e-05 \*\*\*

HasGarageNone -5.370e+04 1.637e+04 -3.281 0.00109 \*\*

DOM 1.227e+02 6.027e+01 2.036 0.04211 \*

BeenReducedYes -2.356e+04 1.095e+04 -2.152 0.03174 \*

SoldPrevYes -1.448e+04 1.012e+04 -1.430 0.15312

LATITUDE 3.481e+05 2.900e+05 1.200 0.23035

LONGITUDE 2.698e+03 3.303e+05 0.008 0.99349

---

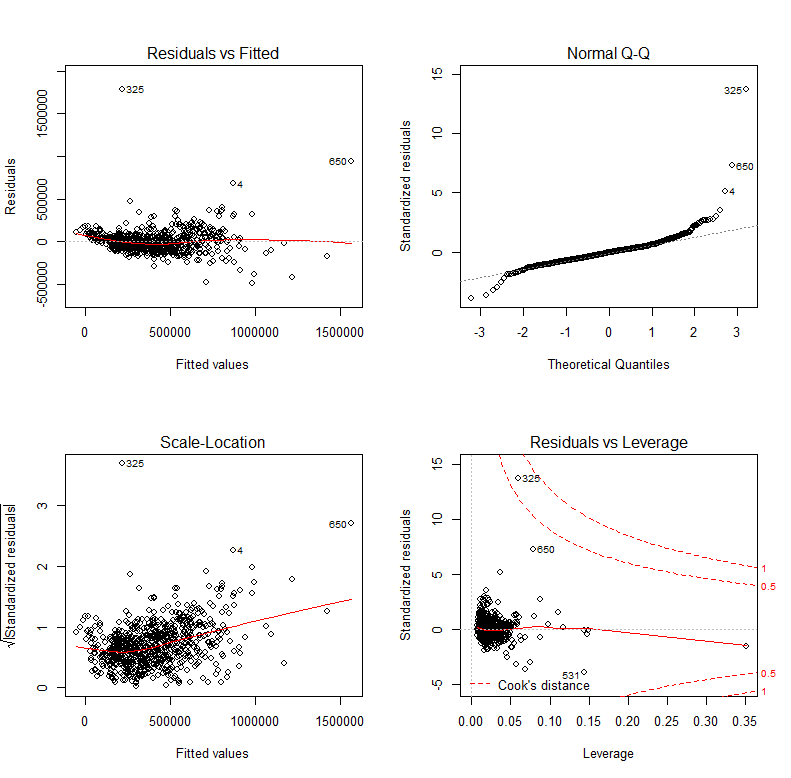
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 134600 on 718 degrees of freedom

Multiple R-squared: 0.7295, Adjusted R-squared: 0.7231

F-statistic: 113.9 on 17 and 718 DF, p-value: < 2.2e-16

Residual Plots for Base Model without ZIP Code.



> ChiTrain[c(325,650),]

Type City ZIP ListPrice BEDS BATHS ImputedSQFT

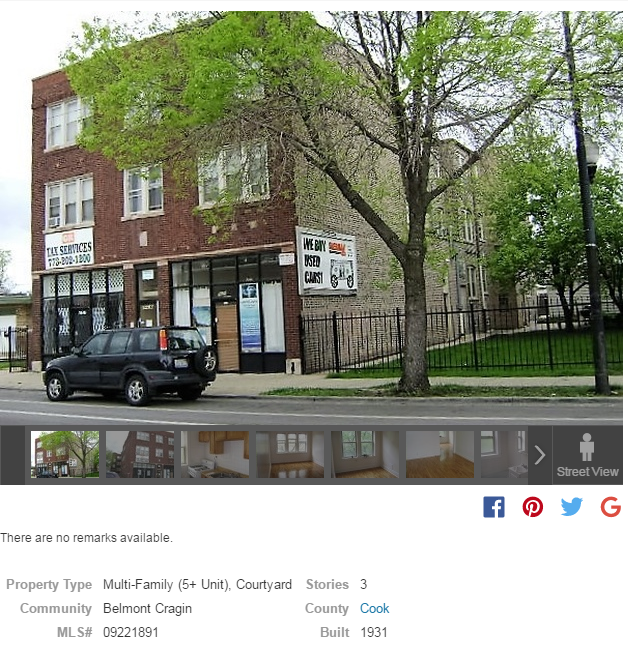
325 Multi-Family Chicago 60639 2000000 2 2 1500

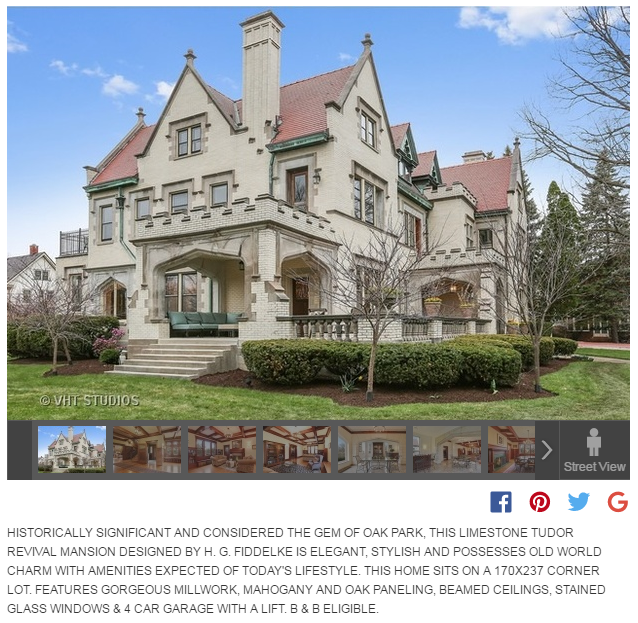
650 Single Family Residential Oak Park 60302 2500000 6 6 9500

ParkSpots HasGarage DOM BeenReduced SoldPrev LATITUDE LONGITUDE

325 3 Garage 3 No No 41.93171 -87.75334

650 4 Garage 32 No No 41.89483 -87.79511





Obs. 650

Obs. 325

Clearly 325 and 650 are exceptional cases, particularly troubling is 325 as the reported predictor values seem incorrect (1500 sq. ft., 2 beds, 2 baths, etc.) and it seems to be commercial space as well as residential. Thus we will *“remove”* this property from our model or from the data frame entirely.

> base.lm3 = lm(ListPrice~.-ZIP,data=ChiTrain,subset=-325)

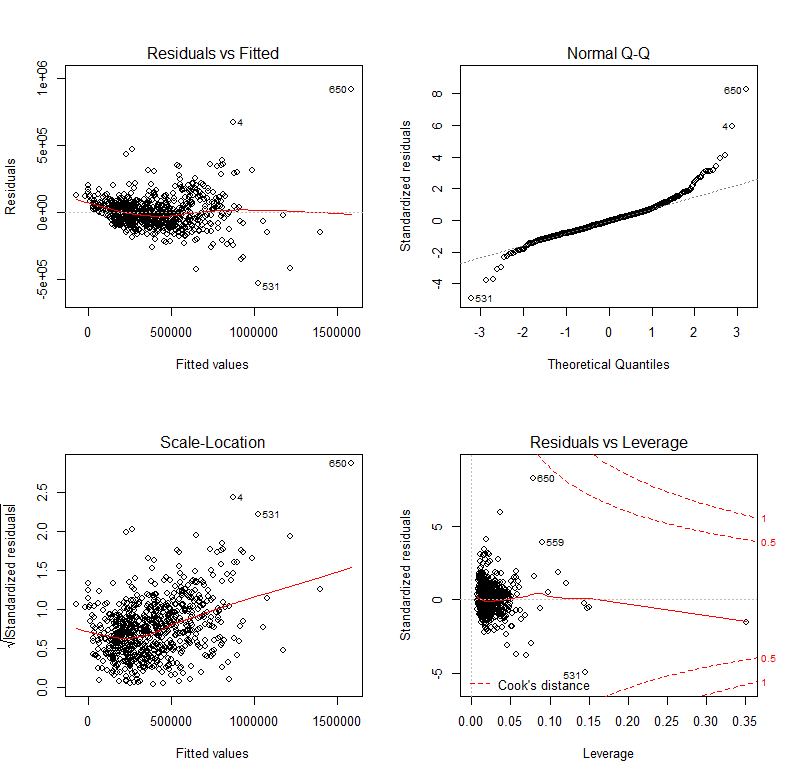
We could create a new data frame without observation (row) 325 and ZIP code while we are at it.

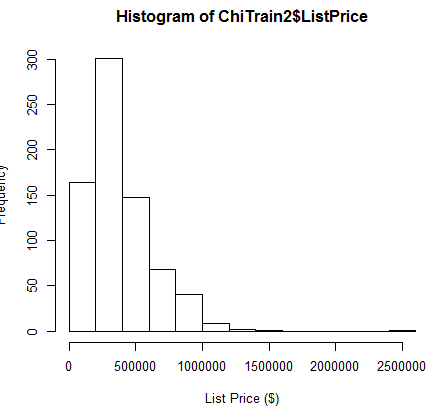
> ChiTrain2 = ChiTrain[-325,-3]

> plot(base.lm3)

> hist(ChiTrain2$ListPrice,xlab=”List Price ($)”)

Residual plots from the model above and a histogram of the list prices without observation 325 are shown below.





> summary(ChiTrain2$ListPrice)

Min. 1st Qu. Median Mean 3rd Qu. Max.

51000 224700 330000 390700 509000 2500000

Box-Cox Transformation for List Price

> results = powerTransform(ChiTrain2$ListPrice)

> summary(results)

bcPower Transformation to Normality

Est.Power Std.Err. Wald Lower Bound Wald Upper Bound

ChiTrain2$ListPrice 0.1543 0.0495 0.0573 0.2513

Likelihood ratio tests about transformation parameters

LRT df pval

LR test, lambda = (0) 9.770945 1 0.001772913

LR test, lambda = (1) 287.481246 1 0.000000000

> log.lm1 = lm(log(ListPrice)~.,data=ChiTrain2)

> summary(log.lm1)

Call:

lm(formula = log(ListPrice) ~ ., data = ChiTrain2)

Residuals:

Min 1Q Median 3Q Max

-1.08471 -0.15435 0.00441 0.17500 1.24421

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -5.030e+01 6.588e+01 -0.764 0.4454

TypeMulti-Family 9.482e-02 7.558e-02 1.255 0.2100

TypeSingle Family Residential 3.995e-01 3.819e-02 10.460 <2e-16 \*\*\*

TypeTownhouse 8.446e-02 5.209e-02 1.622 0.1053

CityChicago -2.789e-01 1.807e-01 -1.544 0.1230

CityDes Plaines -1.331e-01 7.184e-02 -1.853 0.0643 .

CityNorridge 3.355e-03 1.424e-01 0.024 0.9812

CityOak Park 2.806e-01 1.815e-01 1.546 0.1226

BEDS 3.216e-02 1.628e-02 1.975 0.0486 \*

BATHS 1.774e-01 1.863e-02 9.522 <2e-16 \*\*\*

ImputedSQFT 1.549e-04 1.493e-05 10.377 <2e-16 \*\*\*

ParkSpots 2.085e-02 1.145e-02 1.821 0.0691 .

HasGarageNone -3.156e-01 3.426e-02 -9.212 <2e-16 \*\*\*

DOM 2.355e-04 1.262e-04 1.867 0.0623 .

BeenReducedYes -4.635e-02 2.291e-02 -2.023 0.0434 \*

SoldPrevYes -2.745e-02 2.121e-02 -1.294 0.1961

LATITUDE 2.392e-01 6.070e-01 0.394 0.6937

LONGITUDE -5.903e-01 6.914e-01 -0.854 0.3935

---

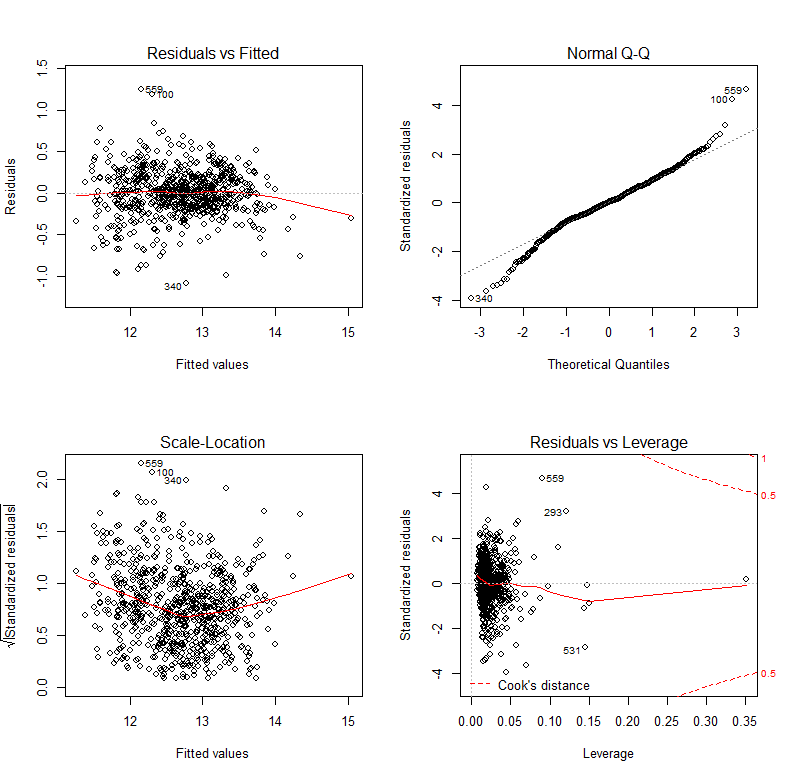
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2817 on 717 degrees of freedom

Multiple R-squared: 0.8104, Adjusted R-squared: 0.8059

F-statistic: 180.3 on 17 and 717 DF, p-value: < 2.2e-16

> plot(log.lm1)



> results = powerTransform(ChiTrain2$ImputedSQFT)

> summary(results)

bcPower Transformation to Normality

Est.Power Std.Err. Wald Lower Bound Wald Upper Bound

ChiTrain2$ImputedSQFT -0.2736 0.0656 -0.4022 -0.145

Likelihood ratio tests about transformation parameters

LRT df pval

LR test, lambda = (0) 17.85198 1 2.387712e-05

LR test, lambda = (1) 450.14029 1 0.000000e+00

> tSQFT = -(1/(ChiTrain2$ImputedSQFT)^.25)

> ChiTrain3 = ChiTrain2

> ChiTrain3$ImputedSQFT = tSQFT

> log.lm2 = lm(log(ListPrice)~.,data=ChiTrain3)

> summary(log.lm2)

Call:

lm(formula = log(ListPrice) ~ ., data = ChiTrain3)

Residuals:

Min 1Q Median 3Q Max

-1.08112 -0.14645 0.00672 0.15804 1.18744

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -6.463e+01 6.225e+01 -1.038 0.29951

TypeMulti-Family 1.001e-01 7.120e-02 1.406 0.16020

TypeSingle Family Residential 3.697e-01 3.610e-02 10.240 < 2e-16 \*\*\*

TypeTownhouse 3.599e-02 4.911e-02 0.733 0.46396

CityChicago -2.014e-01 1.709e-01 -1.178 0.23902

CityDes Plaines -1.001e-01 6.796e-02 -1.472 0.14139

CityNorridge 9.484e-02 1.349e-01 0.703 0.48216

CityOak Park 3.019e-01 1.715e-01 1.760 0.07876 .

BEDS -1.543e-02 1.618e-02 -0.954 0.34044

BATHS 1.243e-01 1.844e-02 6.744 3.18e-11 \*\*\*

ImputedSQFT 1.653e+01 1.150e+00 14.381 < 2e-16 \*\*\*

ParkSpots 2.587e-02 1.079e-02 2.398 0.01674 \*

HasGarageNone -2.299e-01 3.242e-02 -7.092 3.17e-12 \*\*\*

DOM 2.409e-04 1.192e-04 2.021 0.04368 \*

BeenReducedYes -5.969e-02 2.165e-02 -2.758 0.00597 \*\*

SoldPrevYes -2.907e-02 1.999e-02 -1.454 0.14635

LATITUDE 1.135e-02 5.740e-01 0.020 0.98423

LONGITUDE -8.977e-01 6.538e-01 -1.373 0.17021

---

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

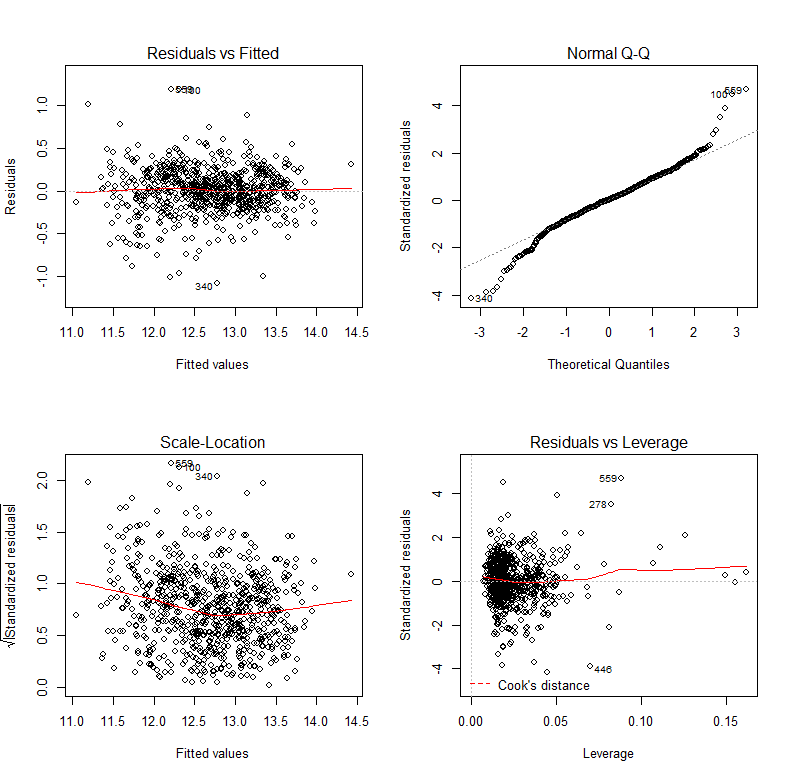
Residual standard error: 0.2662 on 717 degrees of freedom

Multiple R-squared: 0.8307, Adjusted R-squared: 0.8267

F-statistic: 207 on 17 and 717 DF, p-value: < 2.2e-16

There several potential predictors that do not appear to be significant. Perhaps surprisingly the latitude and longitude of the home are not significant.

> plot(log.lm2)



This model seems quite adequate from a residual standpoint, however we want to build the simplest model we can thus we will use a stepwise selection method called backward elimination to reduce the model.

> log.step = step(log.lm2)

Start: AIC=-1927.89

log(ListPrice) ~ Type + City + BEDS + BATHS + ImputedSQFT + ParkSpots +

HasGarage + DOM + BeenReduced + SoldPrev + LATITUDE + LONGITUDE

Df Sum of Sq RSS AIC

- LATITUDE 1 0.0000 50.801 -1929.9

- BEDS 1 0.0645 50.865 -1929.0

- LONGITUDE 1 0.1335 50.934 -1928.0

<none> 50.801 -1927.9

- SoldPrev 1 0.1498 50.951 -1927.7

- DOM 1 0.2893 51.090 -1925.7

- ParkSpots 1 0.4075 51.208 -1924.0

- BeenReduced 1 0.5388 51.340 -1922.1

- BATHS 1 3.2221 54.023 -1884.7

- HasGarage 1 3.5637 54.365 -1880.0

- City 4 7.9979 58.799 -1828.4

- Type 3 12.3953 63.196 -1773.4

- ImputedSQFT 1 14.6537 65.455 -1743.6

Step: AIC=-1929.89

log(ListPrice) ~ Type + City + BEDS + BATHS + ImputedSQFT + ParkSpots +

HasGarage + DOM + BeenReduced + SoldPrev + LONGITUDE

Df Sum of Sq RSS AIC

- BEDS 1 0.0645 50.865 -1931.0

- LONGITUDE 1 0.1336 50.935 -1930.0

<none> 50.801 -1929.9

- SoldPrev 1 0.1503 50.951 -1929.7

- DOM 1 0.2901 51.091 -1927.7

- ParkSpots 1 0.4076 51.209 -1926.0

- BeenReduced 1 0.5416 51.343 -1924.1

- BATHS 1 3.2307 54.032 -1886.6

- HasGarage 1 3.5690 54.370 -1882.0

- Type 3 12.4031 63.204 -1775.3

- ImputedSQFT 1 14.7515 65.552 -1744.5

- City 4 16.7084 67.509 -1728.9

Step: AIC=-1930.96

log(ListPrice) ~ Type + City + BATHS + ImputedSQFT + ParkSpots +

HasGarage + DOM + BeenReduced + SoldPrev + LONGITUDE

Df Sum of Sq RSS AIC

- LONGITUDE 1 0.1287 50.994 -1931.1

<none> 50.865 -1931.0

- SoldPrev 1 0.1450 51.010 -1930.9

- DOM 1 0.2878 51.153 -1928.8

- ParkSpots 1 0.3838 51.249 -1927.4

- BeenReduced 1 0.5484 51.414 -1925.1

- BATHS 1 3.1700 54.035 -1888.5

- HasGarage 1 3.5543 54.420 -1883.3

- Type 3 12.8595 63.725 -1771.3

- City 4 16.7453 67.611 -1729.8

- ImputedSQFT 1 16.8586 67.724 -1722.6

Step: AIC=-1931.1

log(ListPrice) ~ Type + City + BATHS + ImputedSQFT + ParkSpots +

HasGarage + DOM + BeenReduced + SoldPrev

Df Sum of Sq RSS AIC

<none> 50.994 -1931.1

- SoldPrev 1 0.1473 51.141 -1931.0

- DOM 1 0.2809 51.275 -1929.1

- ParkSpots 1 0.3649 51.359 -1927.9

- BeenReduced 1 0.5638 51.558 -1925.0

- BATHS 1 3.3093 54.303 -1886.9

- HasGarage 1 3.6410 54.635 -1882.4

- Type 3 13.3014 64.296 -1766.7

- City 4 16.8426 67.837 -1729.3

- ImputedSQFT 1 16.7395 67.734 -1724.5

The criterion used to select the “optimal” reduced model is Akaike Information Criterion (AIC) which is given by,

and is minimized in order to find an “optimal” submodel. It basically balances the RSS against the number of estimated parameters in our model (*k*). Here LATITUDE, BEDS, and LONGITUDE have been removed in that order.

> summary(log.step)

Call:

lm(formula = log(ListPrice) ~ Type + City + BATHS + ImputedSQFT +

ParkSpots + HasGarage + DOM + BeenReduced + SoldPrev, data = ChiTrain3)

Residuals:

Min 1Q Median 3Q Max

-1.11723 -0.14494 0.00438 0.15462 1.22290

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 14.6975673 0.1996732 73.608 < 2e-16 \*\*\*

TypeMulti-Family 0.0723806 0.0638677 1.133 0.25747

TypeSingle Family Residential 0.3600461 0.0326190 11.038 < 2e-16 \*\*\*

TypeTownhouse 0.0336826 0.0487107 0.691 0.48948

CityChicago -0.3983691 0.0369054 -10.794 < 2e-16 \*\*\*

CityDes Plaines -0.1771730 0.0273690 -6.474 1.77e-10 \*\*\*

CityNorridge -0.0459900 0.0510346 -0.901 0.36781

CityOak Park 0.1297827 0.0268657 4.831 1.66e-06 \*\*\*

BATHS 0.1234094 0.0180541 6.836 1.74e-11 \*\*\*

ImputedSQFT 15.9262962 1.0359456 15.374 < 2e-16 \*\*\*

ParkSpots 0.0243675 0.0107353 2.270 0.02351 \*

HasGarageNone -0.2318336 0.0323342 -7.170 1.86e-12 \*\*\*

DOM 0.0002368 0.0001189 1.992 0.04679 \*

BeenReducedYes -0.0608307 0.0215601 -2.821 0.00491 \*\*

SoldPrevYes -0.0287845 0.0199612 -1.442 0.14973

---

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

Residual standard error: 0.2661 on 720 degrees of freedom

Multiple R-squared: 0.8301, Adjusted R-squared: 0.8268

F-statistic: 251.2 on 14 and 720 DF, p-value: < 2.2e-16

As we have created a new term based on ImputedSQFT in our training data we need to do the same thing for our test data otherwise our predictions using ImputedSQFT in the original scale will be WAY OFF!

> ChiTest2 = ChiTest

> ChiTest2$ImputedSQFT = -(1/(ChiTest$ImputedSQFT)^.25)

> ypredlog = predict(log.step,newdata=ChiTest2)

> ypred = exp(ypredlog)

> PredAcc(ChiTest2$ListPrice,ypred)

RMSEP

===============

123147.2

MAE

===============

77544.15

MAPE

===============

21.24778

> plot(ChiTest2$ListPrice,ypred,xlab="Actual List Price",

ylab="Predicted List Price")

> abline(0,1,lwd=3,col="blue")



We can see that we missed badly one some of the properties in the test data. These could be exceptional cases like those we examined in the training data. We can remove the four most poorly fit cases and recalculate our prediction accuracy measures using the commands below. After running the identify command click on 4 cases to remove then click Finish in the upper-right hand corner of the plot window.

> bad = identify(ChiTest2$ListPrice,ypred)

> bad

[1] 1 106 293 311

> yact = ChiTest2$ListPrice[-bad]

> ypred2 = ypred[-bad]

Examining the details for the 4 poorly fit cases

> ChiTest2[bad,]

Type City ZIP ListPrice BEDS BATHS

1 Single Family Residential Arlington Heights 60005 1599000 4 5

106 Multi-Family Arlington Heights 60004 479900 10 6

293 Single Family Residential Oak Park 60302 1249000 5 3

311 Single Family Residential Oak Park 60302 1995000 6 4

ImputedSQFT ParkSpots HasGarage DOM BeenReduced SoldPrev LATITUDE LONGITUDE

1 -0.1227826 3 Garage 351 Yes Yes 42.06740 -87.97385

106 -0.0911891 12 None 10 No No 42.15185 -88.01309

293 -0.1308518 2 Garage 18 Yes Yes 41.89187 -87.79982

311 -0.1155987 5 Garage 25 No No 41.89917 -87.79296

> PredAcc(yact,ypred2)

RMSEP

===============

95002.97

MAE

===============

70014.44

MAPE

===============

20.5968