Final Project - DATS 6101 - Intro to Data Science

Aaron A. Gauthier

June 27, 2018

My Question: What is the predicted price of a diamond based on the attributes of cut, color, clarity and carat? Is our prediction close?

```
#I used the Diamonds dataset from the following link: https://www.kaggle.com/shivam2503/diamonds. After downloa
ding the dataset, I decided to answer the the following question: What is the predicted price of a diamond base
d on the attributes of cut, color, clarity and carat? What is the accuracy of the prediction? Additionally, I d
ecided to to answer additional questions in order to learn some additional methods / processes and code that ma
y be of use in the future.

#Lots of research and partial projects have been completed on certain aspects of the diamonds dataset. I could
not find one full project that brought the diamonds dataset through the entire data science lifecycle. I decide
d to pick this dataset so that I could learn and leverage others work. I learn the most when I follow someone's
 thought process and learn from them. I took some ideas from multiple posts / projects / homeworks and cited th
eir respective work within this project. I learned a lot about how to think about this data set that I will app
ly to other datasets in the future. I also learned about some really great ways to plot information that I may
not have found otherwise if I hadn't decided to basically take on the mindset of find some sort of project(s) a
nd improve upon it. I believe the following project is much more complete than those that came before and shows
 alternative methods and insights into the data.

#I also incorporated additional plots, graphs and developed a methodology that I believe closely follows the st
art of a Data Science methodology. Following the "Epicycles of Analysis", I based my methodology on the followi
ng core activities and steps.

#Lines 21-36 was taken from the following website: https://gohighbrow.com/the-data-analysis-epicycle/.

#There are five core activities of data analysis:

#1. Stating and refining the question
#2. Exploring the data
#3. Building formal statistical models
#4. Interpreting the results
#5. Communicating the results

#These five activities can occur at different time scales; for example, you might go through all five in the co
urse of a day but also deal with each (for a large project) over the course of many months. Although there are
many different types of activities that you might engage in while doing data analysis, every aspect of the enti
re process can be approached through an iterative process that we call the epicycle of data analysis. More spec
ifically, for each of the five core activities, it is critical that you engage in the following steps:

#1. Setting expectations,
#2. Collecting information (data), comparing the data to your expectations, and if the expectations don't match
,
#3. Revising your expectations or fixing the data so your data and your expectations match. As you go through e
very stage of an analysis, you will need to go through the epicycle to continuously refine your question, your
exploratory data analysis, your formal models, your interpretation, and your communication.

#The repeated cycling through each of these five core activities that is done to complete a data analysis forms
 the larger circle of data analysis.

#I also learned the following from this project and am starting to develop my own methodology to follow (as may
be a sub-methodology) for me to apply to my data analysis in the future or maybe they are better defined design
 principles / considerations:

#1. I learned to look for mathematical relationships where they exist! In this example,

#I never would have equated Carat Weight to Volume in three dimensional space. Volume is measured in cubic unit
s. This mathematical relationship is very important to know and understand in developing a model. I have learne
d to look at, think about and consider any mathematical relationships where they exist especially when dealing
with physical dimensions in three dimensional space.

#2. I learned to first develop my plan for Exploratory Data Analysis (EDA) and adjust it accordingly.

#3. I learned to truly think about how to visually represent the data in such a way that it communicates the st
ory of the data. It should be immediately obvious what you are communicating through your visual representation
s of the data.

#4. I learned to truly think about and disect what the results means and what could explain the possible deviat
ions to the model. Meaning what additional factors may account for the model not being accurate or possibly how
 to improve upon the model and make it more accurate.
```

```
#In the diamonds prediction phase, I noticed some intersting observations when tryin to forecast a branded diam
ond with the model versus an unbranded diamond in the model. The branded diamond was not within the limits of t
he model. The unbranded diamond was within the limits of the model. I learned to always be thinking about, "Wha
t conclusions can you draw from that information?"

#Lastly, what additional information or analysis might improve my model results or work to control limitations.
 I think more specialized domain information could help as well as a better understanding of how to predict wit
h Lasso and GLMNET models. I was not able to get to this point in the project which was my goal. I think to gai
n deeper insight into the data, some sort of customer survey might be needed to gain insight into the data.

#Libraries for use in this project
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(ggplot2)
library(ResourceSelection)
```

```
## ResourceSelection 0.3-2   2017-02-28
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(pscl)
```

```
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
##
## Attaching package: 'glmnet'
```

```
## The following object is masked from 'package:pROC':
##
##     auc
```

```
library(plotmo)
```

```
## Loading required package: plotrix
```

```
##
## Attaching package: 'plotrix'
```

```
## The following object is masked from 'package:gplots':
##
##     plotCI
```

```
## Loading required package: TeachingDemos
```

```
library(gradDescent)
```

```
##
## Attaching package: 'gradDescent'
```

```
## The following object is masked from 'package:ROCR':
##
##     prediction
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:gradDescent':
##
##     RMSE
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:plotrix':
##
##     rescale
```

```
library(GGally)
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(gplots)
library(hexbin)
library(colorspace)
```

```
##
## Attaching package: 'colorspace'
```

```
## The following object is masked from 'package:pROC':
##
##     coords
```

```
library(modelr)
library(leaps)
library(Hmisc)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:TeachingDemos':
##
##     cnvrt.coords, subplot
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

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

```
library(xkcd)
```

```
## Loading required package: extrafont
```

```
## Registering fonts with R
```

```
library(xkcdcolors)
library(XKCDdata)
library(memisc)
```

```
##
## Attaching package: 'memisc'
```

```
## The following objects are masked from 'package:Hmisc':
##
##     %nin%, html
```

```
## The following object is masked from 'package:scales':
##
##     percent
```

```
## The following object is masked from 'package:foreach':
##
##     foreach
```

```
## The following object is masked from 'package:Matrix':
##
##     as.array
```

```
## The following objects are masked from 'package:dplyr':
##
##     collect, recode, rename, syms
```

```
## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts
```

```
## The following object is masked from 'package:base':
##
##     as.array
```

```
library(lattice)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:memisc':
##
##     recode
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(forcats)
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:memisc':
##
##     rename
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

```
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:reshape':
##
##     rename, round_any
```

```
## The following object is masked from 'package:memisc':
##
##     rename
```

```
## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, mutate, rename, summarise
```

```
## The following object is masked from 'package:reshape':
##
##     rename
```

```
## The following objects are masked from 'package:memisc':
##
##     rename, style
```

```
## The following object is masked from 'package:Hmisc':
##
##     subplot
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following object is masked from 'package:TeachingDemos':
##
##     subplot
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(foreign)
library(RColorBrewer)
library(glmnet)
library(nnet)
library(mlogit)
```

```
## Loading required package: maxLik
```

```
## Loading required package: miscTools
```

```
##
## Please cite the 'maxLik' package as:
## Henningsen, Arne and Toomet, Ott (2011). maxLik: A package for maximum likelihood estimation in R. Computati
onal Statistics 26(3), 443-458. DOI 10.1007/s00180-010-0217-1.
##
## If you have questions, suggestions, or comments regarding the 'maxLik' package, please use a forum or 'track
er' at maxLik's R-Forge site:
## https://r-forge.r-project.org/projects/maxlik/
```

```
library(Matrix)
library(ggcorrplot)
library(tinytex)

#Let's set our working directory
setwd("C:/Users/Gaming_DataScience/OneDrive/Documents/R")

diamonds <- read.csv("diamonds.csv")#read diamonds file
head(diamonds)#display some of the data
```

```
##   X carat       cut color clarity depth table price    x    y    z
## 1 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

```
str(diamonds)#data frame by observations and variables
```

```
## 'data.frame':    53940 obs. of  11 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Factor w/ 5 levels "Fair","Good",..: 3 4 2 4 2 5 5 5 1 5 ...
##  $ color  : Factor w/ 7 levels "D","E","F","G",..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Factor w/ 8 levels "I1","IF","SI1",..: 4 3 5 6 4 8 7 3 6 5 ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

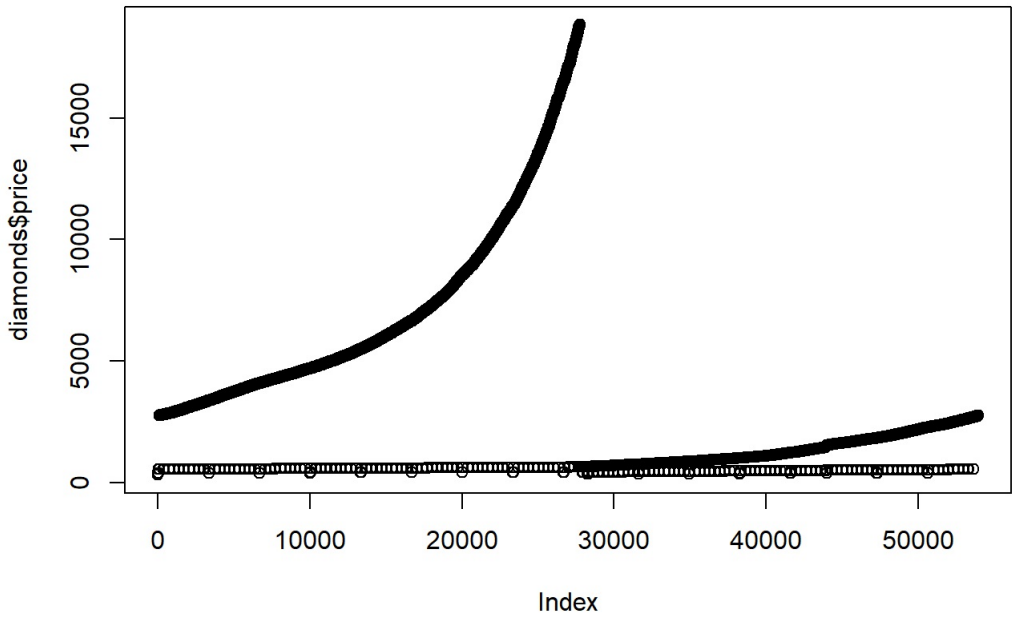
```
sum(is.na(diamonds)) # check the values to see if I have any NAs in my data and it doesn't seem to indicate tha
t I do.
```

```
## [1] 0
```

```
which(is.na(diamonds)) # check to see the locations of any NA values in my data. It says there are none! Yea, s
o our data set appears to be clean.
```

```
## integer(0)
```

```
plot(diamonds$price) #plotting our dependent variable price - Looks like mess - very strange - I think this typ
e of plot has to do with the ordinal factor variables. There appears to be an exponential relationship based on
 price? But how, what contributes to this? I hope further EDA will determine this or make it more apparent!? It
 seems like there is a second strange, but not as pronounced exponential relationship; followed by a flat relat
ionship? Really strange, wish I knew what this was attributed too...
```




```
data(diamonds)
summary(diamonds)
```

```
##      carat               cut        color        clarity     
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065  
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258  
##  Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194  
##  Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171  
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066  
##  Max.   :5.0100                     I: 5422   VVS1    : 3655  
##                                     J: 2808   (Other): 2531  
##      depth           table           price             x         
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000  
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710  
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700  
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731  
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540  
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740  
##                                                                  
##        y                z        
##  Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 4.720   1st Qu.: 2.910  
##  Median : 5.710   Median : 3.530  
##  Mean   : 5.735   Mean   : 3.539  
##  3rd Qu.: 6.540   3rd Qu.: 4.040  
##  Max.   :58.900   Max.   :31.800  
##
```

```
dim(diamonds)#this will give us the number of observations and variables. Taken from www.rpubs.com/amelij/EDA_lesson3
```

```
## [1] 53940    10
```

```
samp_index = sample(1:nrow(diamonds), 30, replace = FALSE) # Creating a random sample to look at the data set.
samp_index
```

```
##  [1] 44608 50727 45674 42316 53505 14581 52375 10591 35959 48364 38951
## [12] 47349 15884 24060 23387 15395 46376 22678 20179 50335 22509  6462
## [23]  7835 37727 23515 28521 10735  1597 37518 23382
```

```
diamonds[samp_index,] # Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
```

```
## # A tibble: 30 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 0.53  Premium   E     SI1      61.9  56    1607  5.22  5.19  3.22
##  2 0.7   Ideal     H     SI1      62.6  55    2294  5.65  5.7   3.55
##  3 0.52  Very Good E     VS2      63.4  57    1694  5.16  5.12  3.26
##  4 0.41  Premium   D     VVS2     61.8  58    1295  4.75  4.79  2.95
##  5 0.71  Fair      G     VS1      65.5  55    2686  5.62  5.53  3.65
##  6 1.13  Premium   H     VS2      60.6  58    5885  6.81  6.72  4.1
##  7 0.54  Premium   D     VVS2     62    59    2505  5.17  5.22  3.22
##  8 0.91  Very Good E     VS2      62.9  55    4823  6.13  6.17  3.87
##  9 0.33  Very Good H     SI1      63    58     475  4.4   4.42  2.78
## 10 0.56  Ideal     D     VS2      60.6  56    1963  5.33  5.37  3.24
## 11 0.31  Good      H     VS2      63.1  58     489  4.27  4.32  2.71
## 12 0.5   Premium   D     VS2      62.1  58    1845  5.08  5.03  3.14
## 13 1     Premium   F     VS2      62.4  60    6352  6.35  6.38  3.97
## 14 0.26  Ideal     E     VVS1     61.7  56     635  4.09  4.11  2.53
## 15 0.3   Good      I     IF       63.3  57     631  4.23  4.3   2.7
## 16 1.2   Very Good I     VS1      62.4  56    6167  6.78  6.74  4.22
## 17 0.580 Ideal     E     VS2      62.1  55    1761  5.35  5.34  3.32
## 18 1.29  Ideal     F     VS1      62.3  54.4 10727  6.93  7     4.34
## 19 1.03  Premium   E     VS1      62.8  58    8629  6.46  6.41  4.04
## 20 0.55  Ideal     F     VVS2     62.4  55    2242  5.24  5.28  3.28
## 21 1.03  Premium   F     VVS1     59.7  60   10546  6.63  6.57  3.94
## 22 0.71  Ideal     E     VS1      61.1  57    4053  5.74  5.78  3.52
## 23 1.01  Very Good F     SI2      63.3  57    4299  6.39  6.37  4.04
## 24 0.34  Premium   G     VVS2     61.7  56     995  4.51  4.47  2.77
## 25 1.56  Ideal     G     SI1      61.2  56   11522  7.49  7.51  4.59
## 26 0.290 Very Good E     VVS2     60.9  57     674  4.29  4.32  2.62
## 27 0.33  Ideal     I     VS2      61.5  57     594  4.49  4.42  2.74
## 28 1     Very Good G     I1       62.5  62    3011  6.42  6.35  3.99
## 29 0.41  Good      G     VVS1     61    61     986  4.77  4.8   2.92
## 30 0.36  Good      E     SI1      63.1  58     631  4.49  4.54  2.85
```

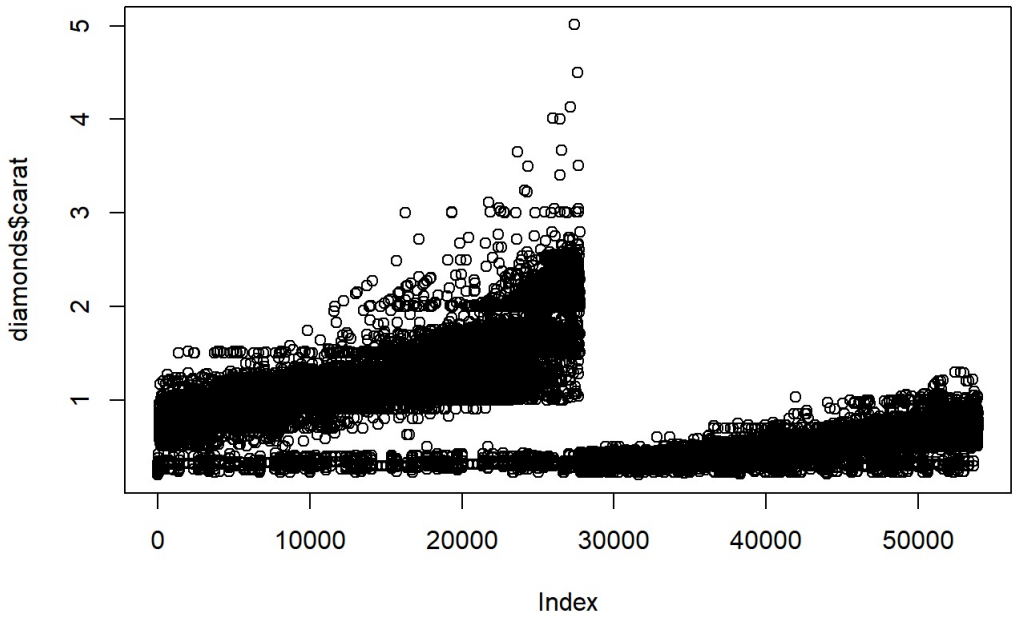
```
#Lets take a closer look at the numberic variables. Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
str(diamonds$carat)
```

```
##  num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
```

```
levels(diamonds$carat)
```

```
## NULL
```

```
plot(diamonds$carat) # Wow, looks like a messy distribution - let's hope I can prove it's correlated to the price. This appears to be correlated with the plot I did above with the diamond price. Again, a very strange graph, but not as pronounced an exponential relationship. You can see how the first graph I did comes out like this, but I still can't explain it.
```



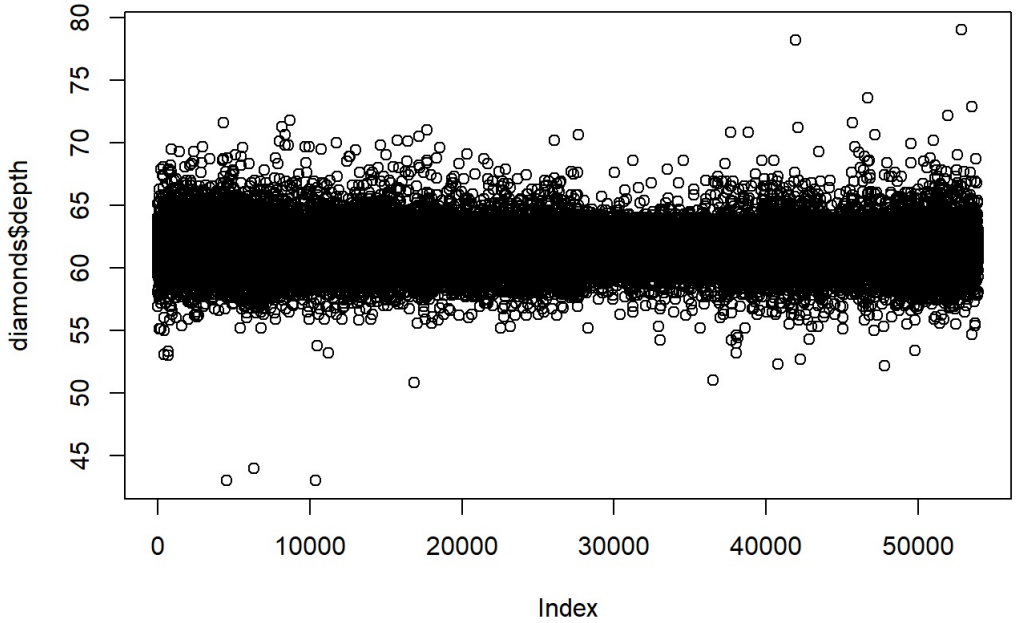

```
#Lets take a closer look at the numeric variables. Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
str(diamonds$depth)
```

```
##  num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
```

```
levels(diamonds$depth)
```

```
## NULL
```

```
plot(diamonds$depth) #Interesting plot, depth is essentially a straight line with little change. Does this mean there is no correlation? Maybe we should exclude this variable?
```



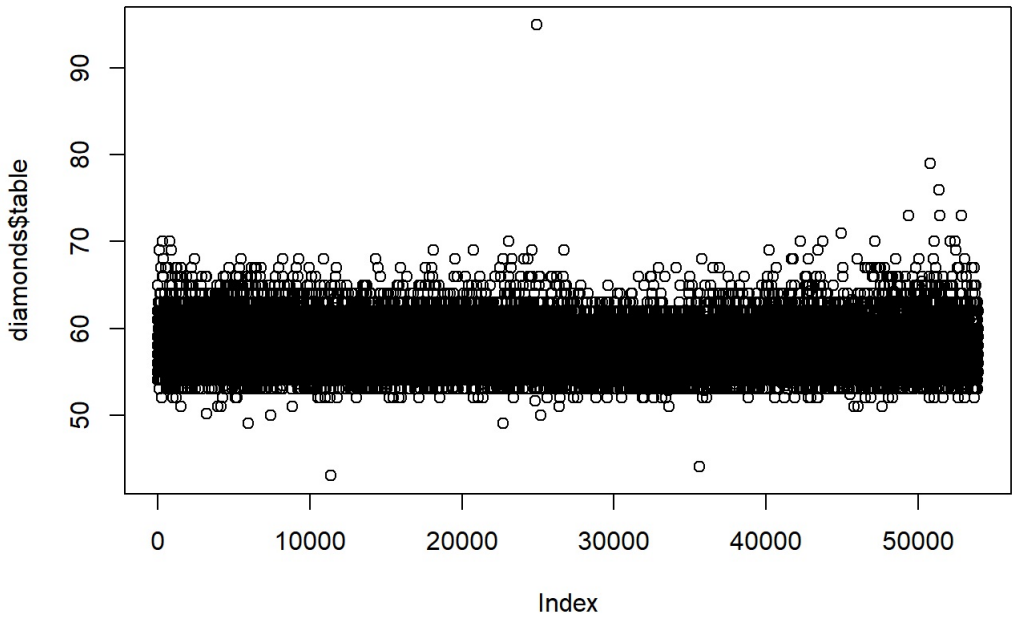

```
#Lets take a closer look at the numeric variables; Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
str(diamonds$table)
```

```
##  num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
```

```
levels(diamonds$table)
```

```
## NULL
```

```
plot(diamonds$table)#Looks like this variable should possibly be removed from any model, it's flat!
```



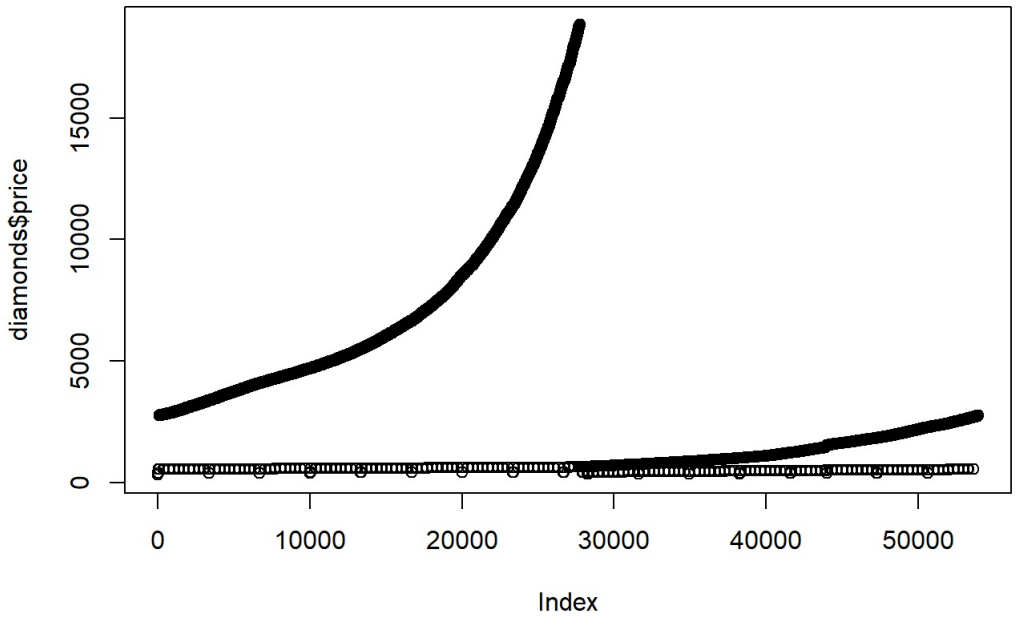

```
#Lets take a closer look at the numeric variables. Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
str(diamonds$price)
```

```
##  int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
```

```
levels(diamonds$price)
```

```
## NULL
```

```
plot(diamonds$price) #Again, it's a strange graph, but shows there is an exponential relationship. It's consistent with the first plot we did for the data set above.
```



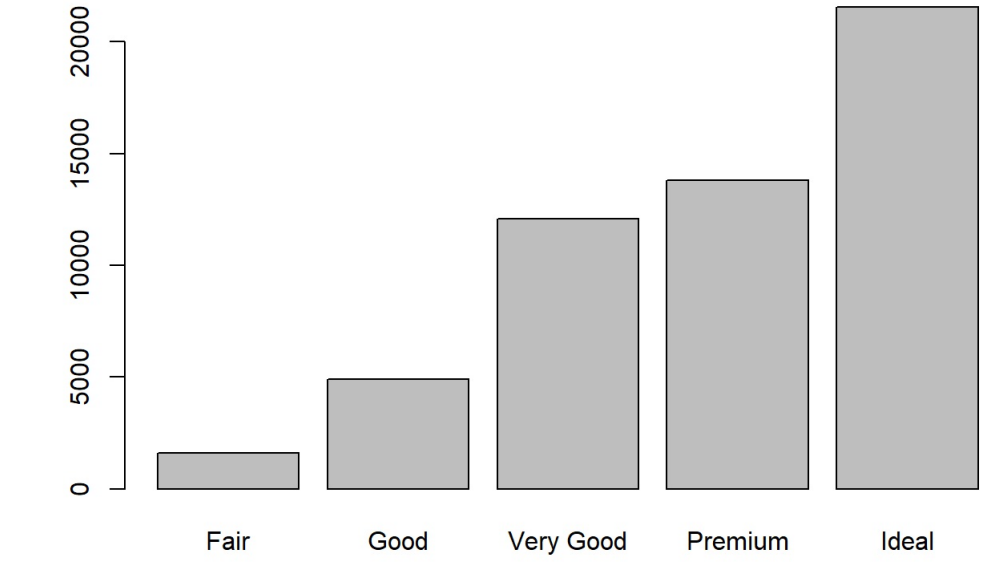

```
#Lets take a closer look at the other ordinal factor variables. Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
str(diamonds$cut)
```

```
##  Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
```

```
levels(diamonds$cut)
```

```
## [1] "Fair"      "Good"      "Very Good" "Premium"   "Ideal"
```

```
plot(diamonds$cut) #There's more ideal cut diamonds than anything else - my guess these are the best sellers! ( i.e. very popular choice amongst consumers)
```



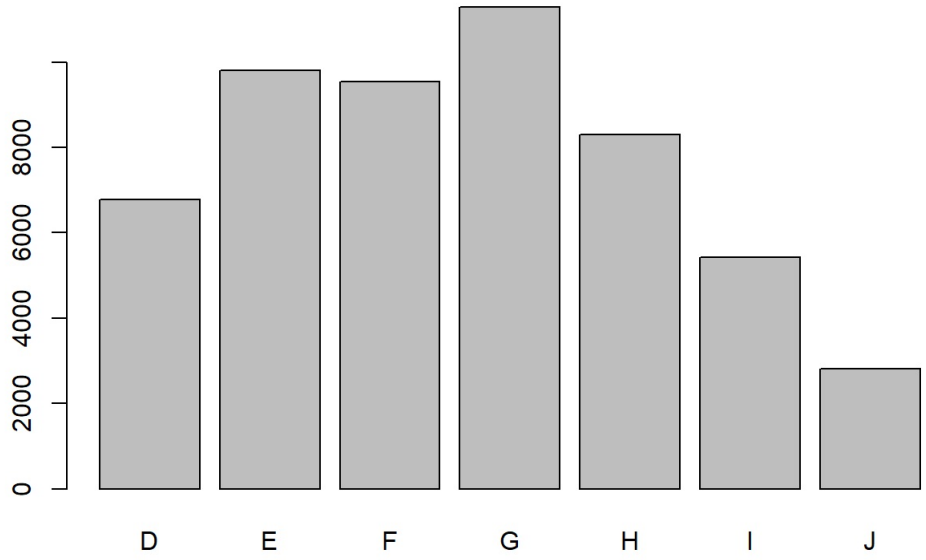

```
#Lets take a closer look at the other ordinal factor variables. Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
str(diamonds$color)
```

```
##  Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
```

```
levels(diamonds$color)
```

```
## [1] "D" "E" "F" "G" "H" "I" "J"
```

```
plot(diamonds$color)#There's less of the lower quality diamonds as we would expect! Overall there are less I & J diamonds (lesser quality) than the higher quality diamonds. There are also less D color diamonds. There seems to be a close to even distribution between diamonds that are colored between E, F, and G. This may mean that customer demand is most for diamonds of color E, F and G.
```



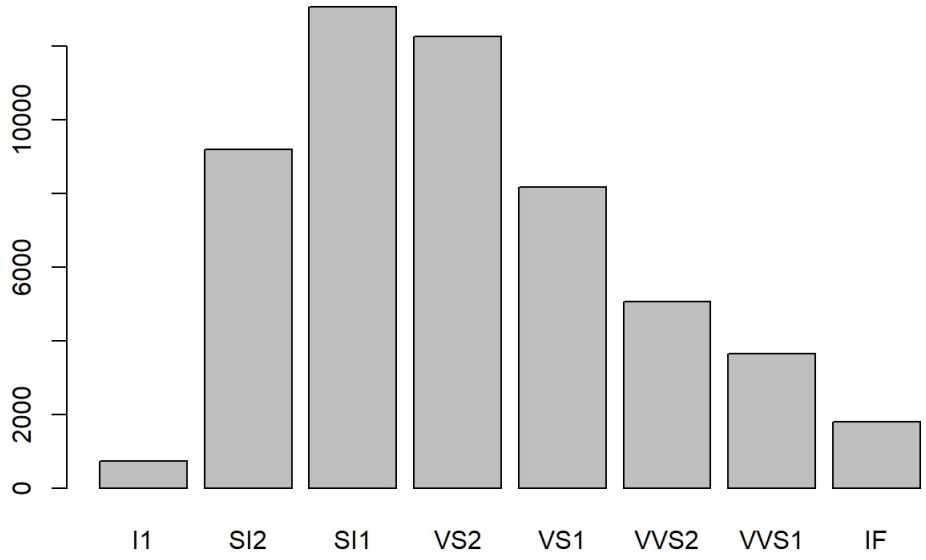

```
#Lets take a closer look at the other ordinal factor variables. Taken from youtube video https://www.youtube.com/watch?v=GgthMorTsn0
str(diamonds$clarity)
```

```
##  Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
```

```
levels(diamonds$clarity)
```

```
## [1] "I1"   "SI2"  "SI1"  "VS2"  "VS1"  "VVS2" "VVS1" "IF"
```

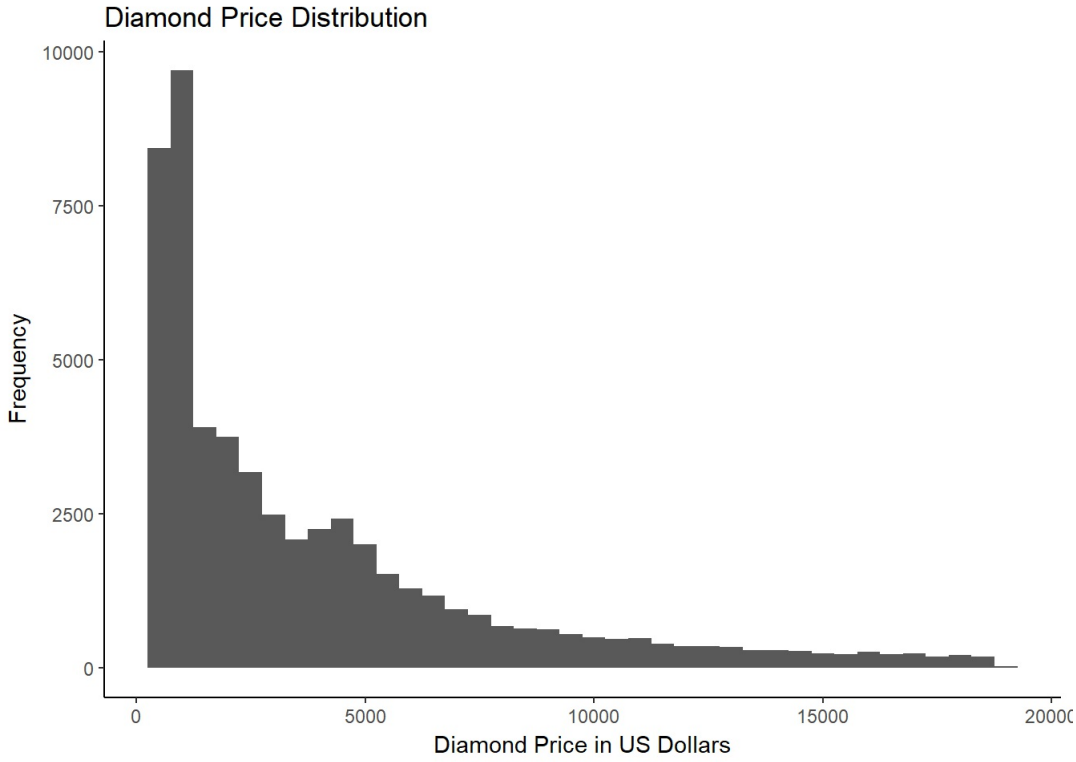
```
plot(diamonds$clarity)#There's more of the lesser quality diamonds as we would expect! As the quality increases, the overall demand appears to decrease. I think this is due to the fact that only people who have more money can afford to buy the higher priced diamonds. That would make sense, since there are less people with more money overall that can afford a more expensive diamond.
```




```
#Lets take a closer look at the variables and their correlations
#Lets look the relationship between carat and price the two continuous variables
data(diamonds)
names(diamonds)#I excluded Depth and Table from the data set due to the fact that the graphs were essentially a
 flat line distribution of the data. I did not see a reason to include them. I also excluded X, Y, and Z which
are the three dimensional measurements that go into mathematically making the formulas for Depth and Table. The
 Depth and Table values are derived from an equation that include the variable measurements x, y and z.
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

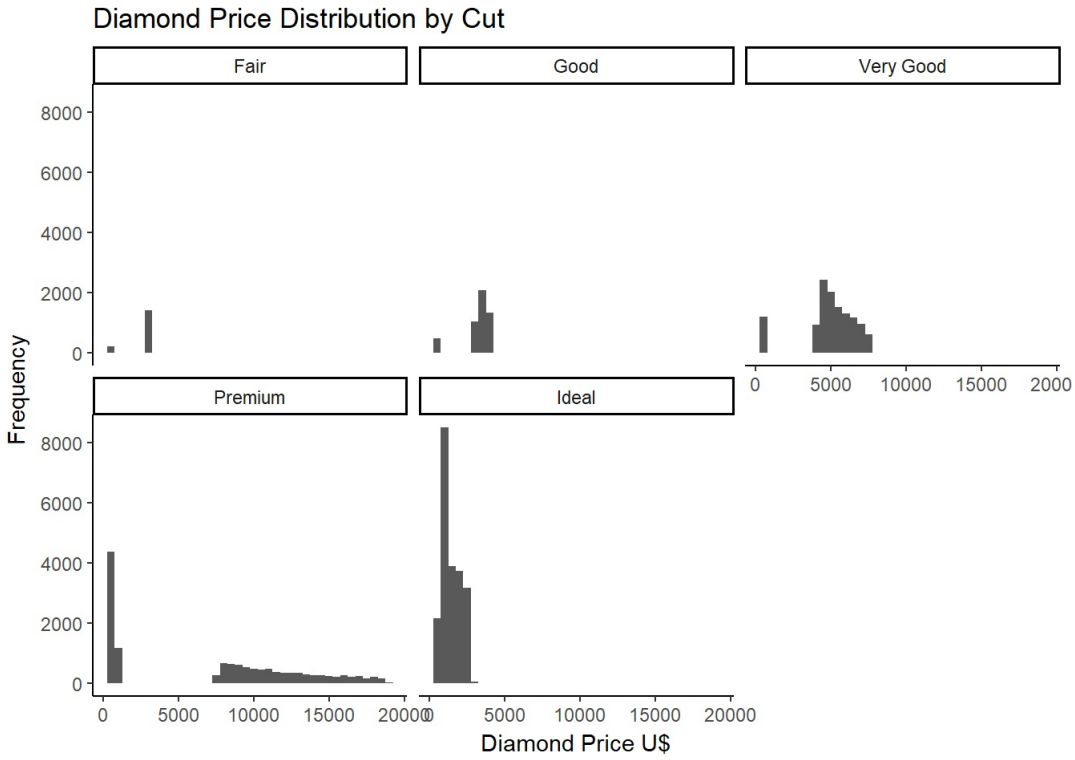
```
ggplot(data=diamonds) + geom_histogram(binwidth = 500, aes(x=diamonds$price)) + ggtitle("Diamond Price Distribu
tion") + xlab ("Diamond Price in US Dollars") + ylab("Frequency") + theme_classic() #Taken from https://rpubs.c
om/ameliji/EDA_Lesson3
```



```
#This is a long tail distribution. It has a very high concentration of obversations below US $5,000 mark. Why i
s this significant? It seems to show that there is a demand for the "higher quality diamonds." The demand shows
 there are less people who are willing to pay more for such higher quality diamonds. Supports my comments on th
e previous graphs.

#Now I am curious about the mean price of the diamonds. What is the mean?

mean(diamonds$price) # Mean is $3,932.80. This seems to be pretty low, but the bulk of the data is before $5,00
0 based on looking at the graphs. So this result would make sense. This value may become more valuable when we
are comparing the models later. I can use this to turn price into a binary result. I will verify this later wit
h a box plot of the mean price for the diamonds data set. Could this possibly be a differentiator between high
quality and low quality diamonds? I think this is a moving target and preference amongst customers, but using t
he mean may make sense as an established cut line.
```

```
## [1] 3932.8
```

```
#Let's take a look at the Diamond Price Distribution by Cut
ggplot(data=diamonds) + geom_histogram(binwidth=500, aes(x=diamonds$price)) + ggtitle("Diamond Price Distributi
on by Cut") + xlab("Diamond Price U$") + ylab("Frequency") + theme_classic() + facet_wrap(~cut) #Taken from htt
ps://rpubs.com/ameliji/EDA_Lesson3
```



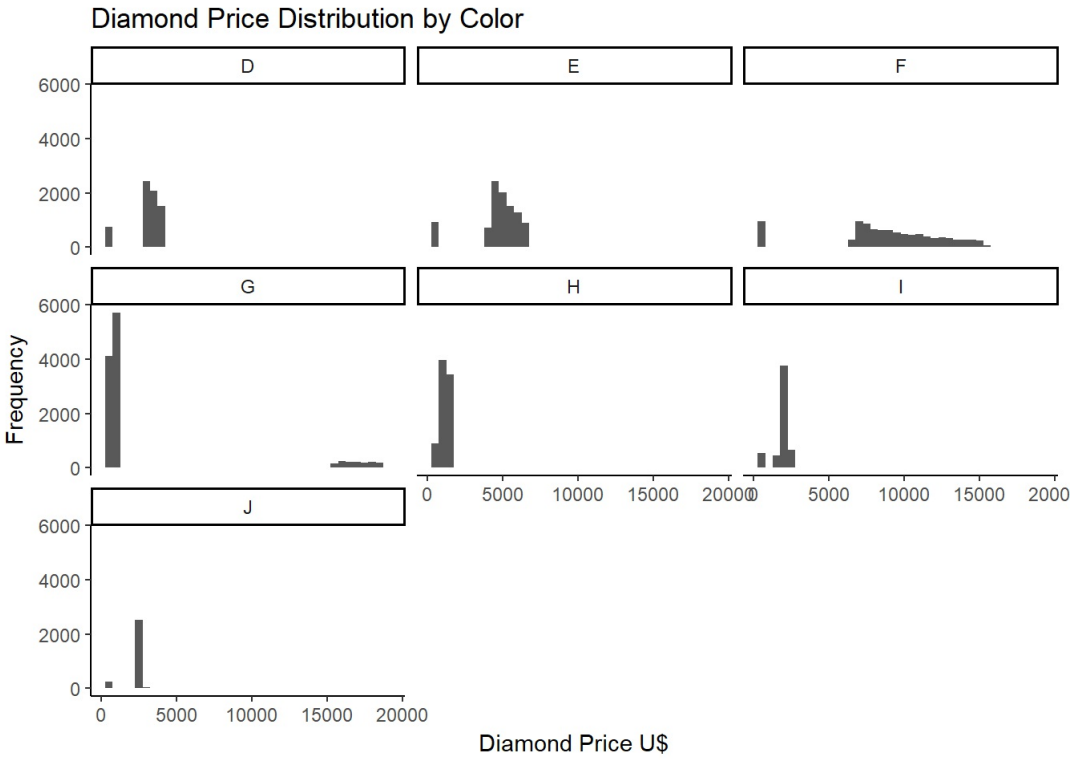

```
#The bulk of the data seems to be distributed in the Premium and Ideal category diamonds. This seems to point t
o as the most popular categories amongst diamonds buyers for this attribute of Cut.

#I am curious about what is the highest priced diamond?
subset(diamonds, price == max(price)) # Taken from https://rpubs.com/ameilij/EDA_lesson3
```

```
## # A tibble: 1 x 10
##   carat cut     color clarity depth table price     x     y     z
##   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  2.29 Premium I     VS2      60.8    60 18823   8.5  8.47  5.16
```

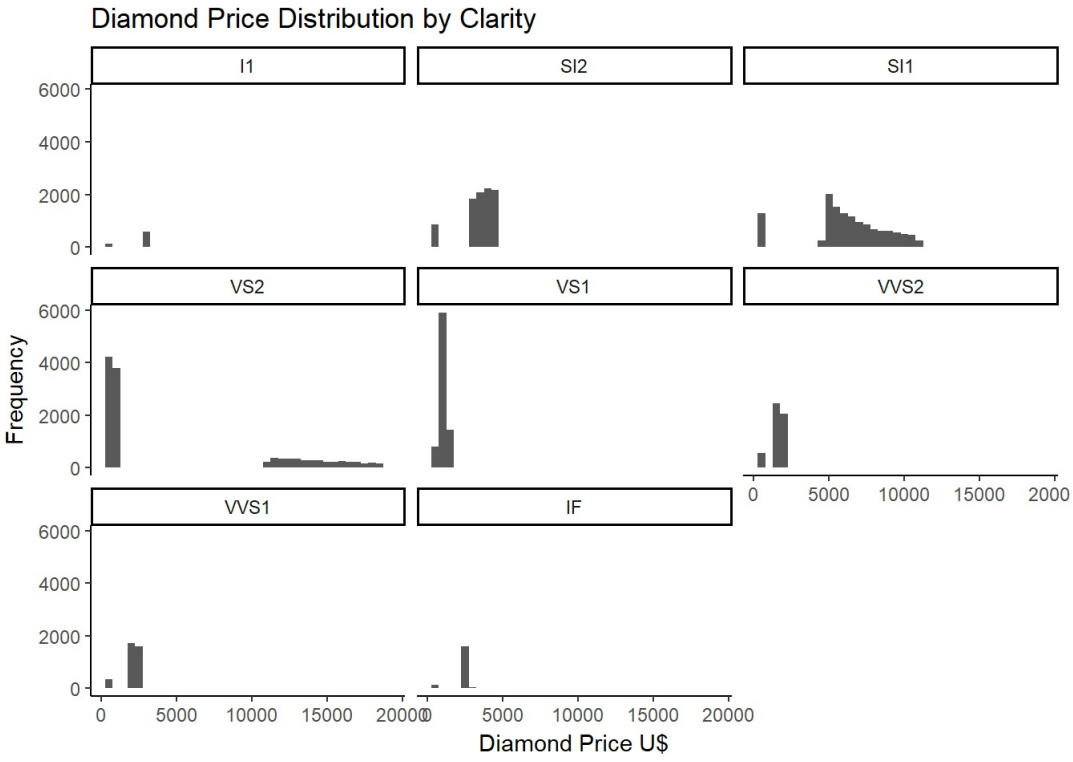
```
#Should return the highest priced diamond - It's a 2.29 Carat, Premium Cut, Color I, clarity VS2 with a depth o
f 60.8, a table of 60 and a price of $18,823.

#I wonder what the least expensive diamond is?
subset(diamonds, price == min(price)) #Taken from https://rpubs.com/ameilij/EDA_lesson3
```

```
## # A tibble: 2 x 10
##   carat cut     color clarity depth table price     x     y     z
##   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal   E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium E     SI1      59.8    61   326  3.89  3.84  2.31
```

```
# It's a tie between two diamonds for $326.

#Let's take a look at the Diamond Price Distribution by Color
ggplot(data=diamonds) + geom_histogram(binwidth=500, aes(x=diamonds$price)) + ggtitle("Diamond Price Distributi
on by Color") + xlab("Diamond Price U$") + ylab("Frequency") + theme_classic() + facet_wrap(~color) #Taken from
 https://rpubs.com/ameliji/EDA_Lesson3
```




```
#This plot is in line with previous  plots. It is separated by "bins". It's interesting to see the bins broken
out by distribution.

#Let's take a look at the Diamond Price Distribution by Clarity
ggplot(data=diamonds) + geom_histogram(binwidth=500, aes(x=diamonds$price)) + ggtitle("Diamond Price Distributi
on by Clarity") + xlab("Diamond Price U$") + ylab("Frequency") + theme_classic() + facet_wrap(~clarity) #Taken
from https://rpubs.com/ameliji/EDA_Lesson3
```

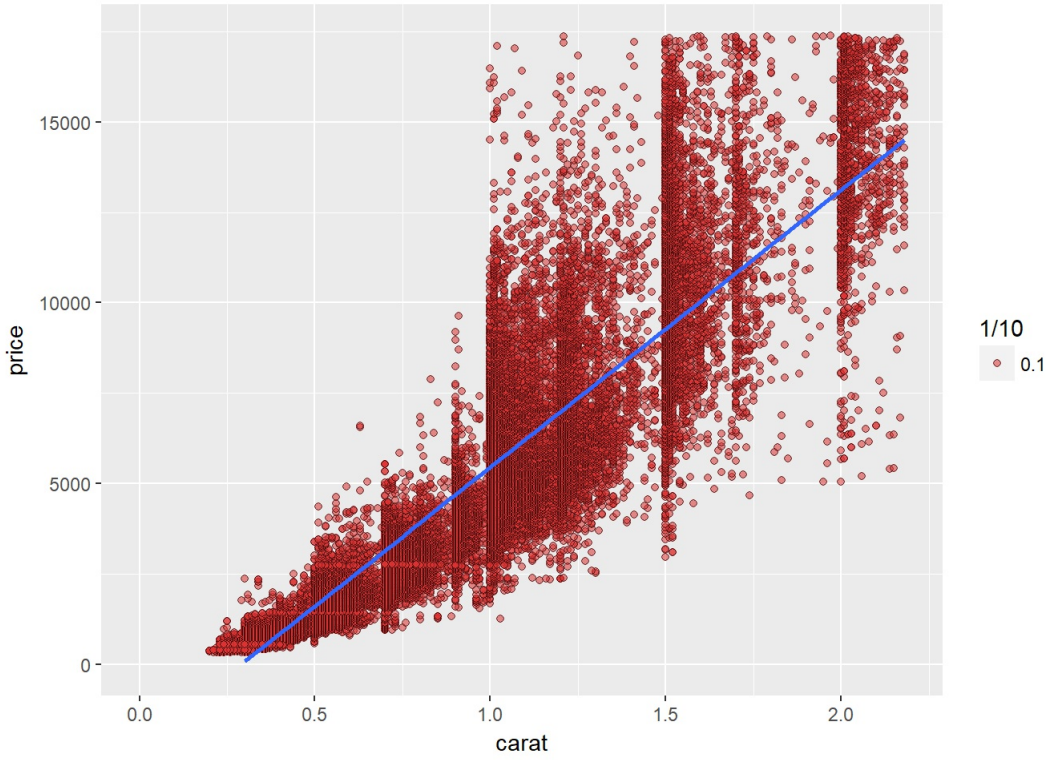


```
ggplot(data=diamonds, aes(x=carat, y=price)) +
  # get rid of top percentile as they could skew the data
  scale_x_continuous(lim=c(0,quantile(diamonds$carat,0.99))) +
  scale_y_continuous(lim=c(0,quantile(diamonds$price,0.99))) +
  geom_point(fill=I('#dd3333'), color= I("black"), aes(alpha=1/10),shape=21) +
  stat_smooth(method='lm')# Taken from https://rpubs.com/taylorwhite/diamondPricing
```

```
## Warning: Removed 926 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 926 rows containing missing values (geom_point).
```

```
## Warning: Removed 4 rows containing missing values (geom_smooth).
```



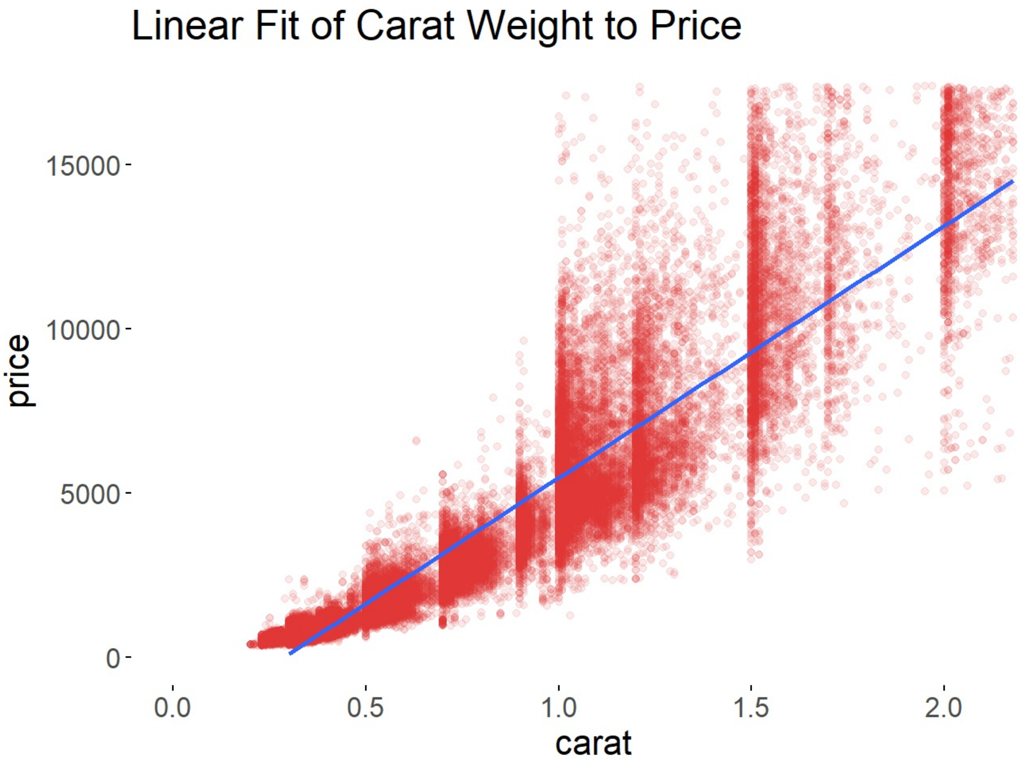

```
#It appears that there is a positive, linear relationship between price and carat weight. We need to further in
vestigate this...

ggplot(data=diamonds, aes(x=carat, y=price)) +
  # get rid of top percentile
  scale_x_continuous(lim=c(0,quantile(diamonds$carat,0.99))) +
  scale_y_continuous(lim=c(0,quantile(diamonds$price,0.99))) +
  geom_point(color=I('#dd3333'),alpha=1/10) +
  stat_smooth(method='lm') +
  ggtitle("Linear Fit of Carat Weight to Price") +
  theme_xkcd()#Taken from https://rpubs.com/taylorwhite/diamondPricing - It's a cool plot. A bit different from
 the one before, but learning lots of different methods on how to plot through this project. I like learning ne
w methods of plotting on this project. Though, I do like the previous plot better than this one. Both plots sho
w the same thing, just in a slightly different way.
```

```
## Warning in theme_xkcd(): Not xkcd fonts installed! See vignette("xkcd-
## intro")
```

```
## Warning: Removed 926 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 926 rows containing missing values (geom_point).
```

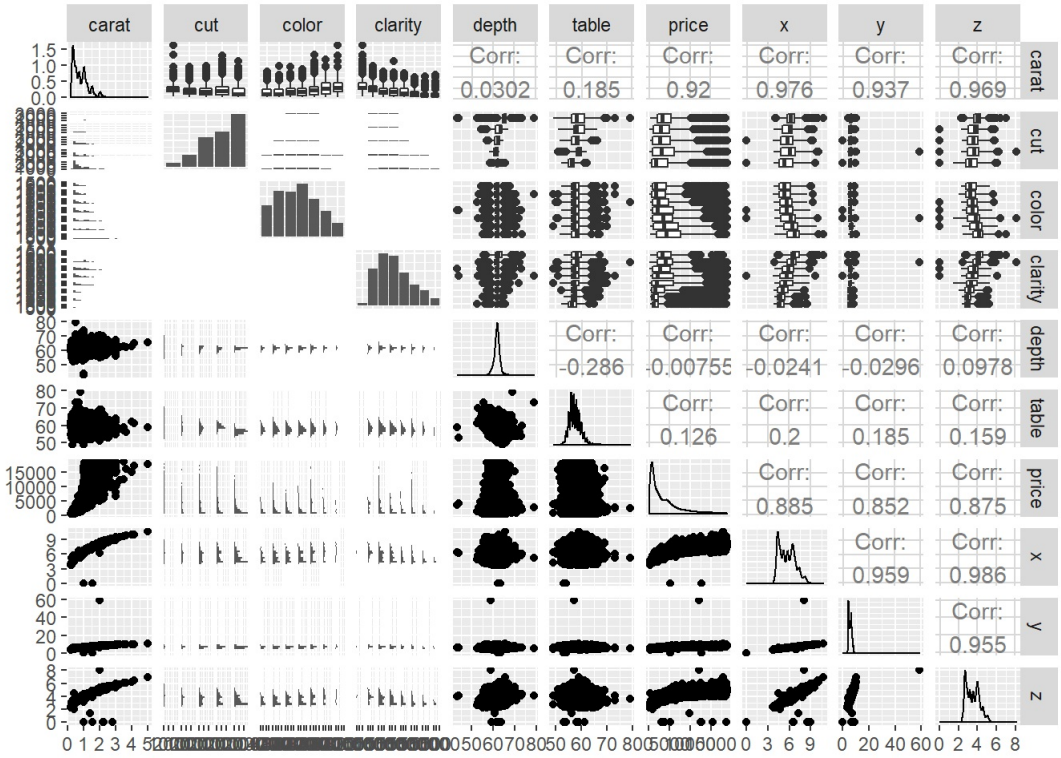
```
## Warning: Removed 4 rows containing missing values (geom_smooth).
```




```
set.seed(42) #Yep, inserting the "cool data science seed thing to do #42" :) Taken from https://rpubs.com/antho
nycerna/diamondspredictions
diamond_samp <- diamonds[sample(1:length(diamonds$price), 25000), ] #Looking at the first 25,000 diamonds
ggpairs(diamond_samp, outlier.shape = I('.')) #Taken from https://rpubs.com/taylorwhite/diamondPricing
```

```
## Warning in warn_if_args_exist(list(...)): Extra arguments: "outlier.shape"
## are being ignored. If these are meant to be aesthetics, submit them
## using the 'mapping' variable within ggpairs with ggplot2::aes or
## ggplot2::aes_string.
```

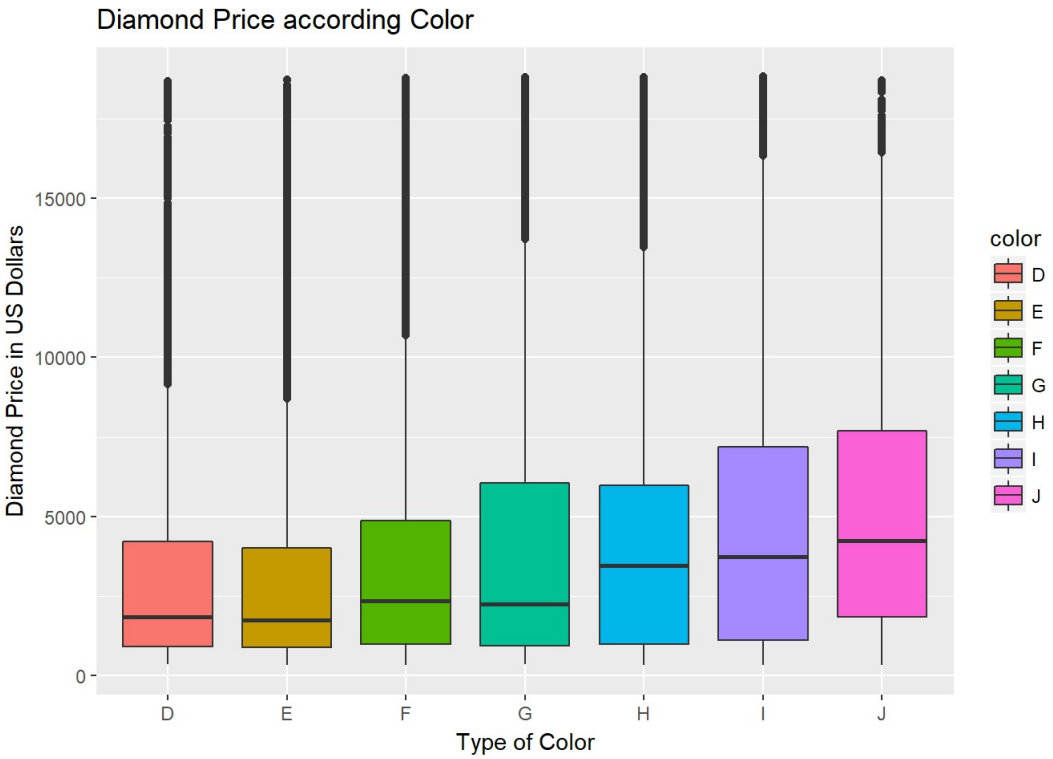
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



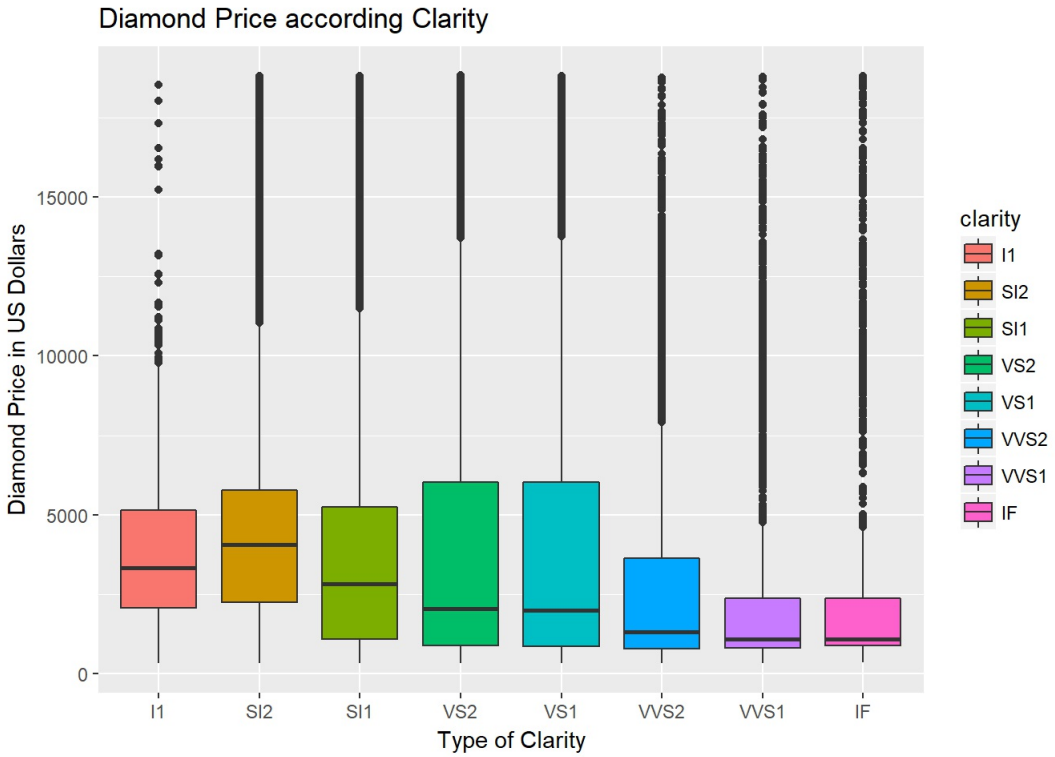

```
#The diagnol plots for color, clarity, depth and possibly table seem to be normally distributed. The x, y, and z variable appear to be highly correlated with each other, which makes perfect sense since these variables are what makes up the depth and table of a diamond or rather x, y and z are the diamond measurements / dimensions. This makes total sense! When you look at price versus the x, y and z variables there is a logarithmic relationship. This becomes hard to analyze, so either I will need to use the logit function / link for a glm or I need to figure out a way to make the log function linear. I need to look at my mathematics books to figure this one out - it's been a while!

#Looking back at previous notes, it appears that carat versus price is an exponential relationship. Part of the reason this could be is if we use our domain knowledge about diamonds, then we know that the higher quality diamonds are very rare to find. Hence a more logarithmic function or rather they become so rare, that there are only a few that are available. This seems to make sense because the variables xyz contribute to the carat weight of a diamond (i.e. xyz in three dimensions is volume of a function). What does this suggest? After look through some mathematics books its highly possible that the cubed root of the carat weight might be of help in our model. We will have to remember to carry this forward!
```

```
#Let's look at some boxplots now

#Diamond Price According to Cut
ggplot(diamonds, aes(factor(cut), price, fill=cut)) + geom_boxplot() + ggtitle("Diamond Price according Cut") + xlab("Type of Cut") + ylab("Diamond Price in US Dollars") #Taken from https://rpubs.com/ameliji/EDA_lesson3 - I modified the code.
```

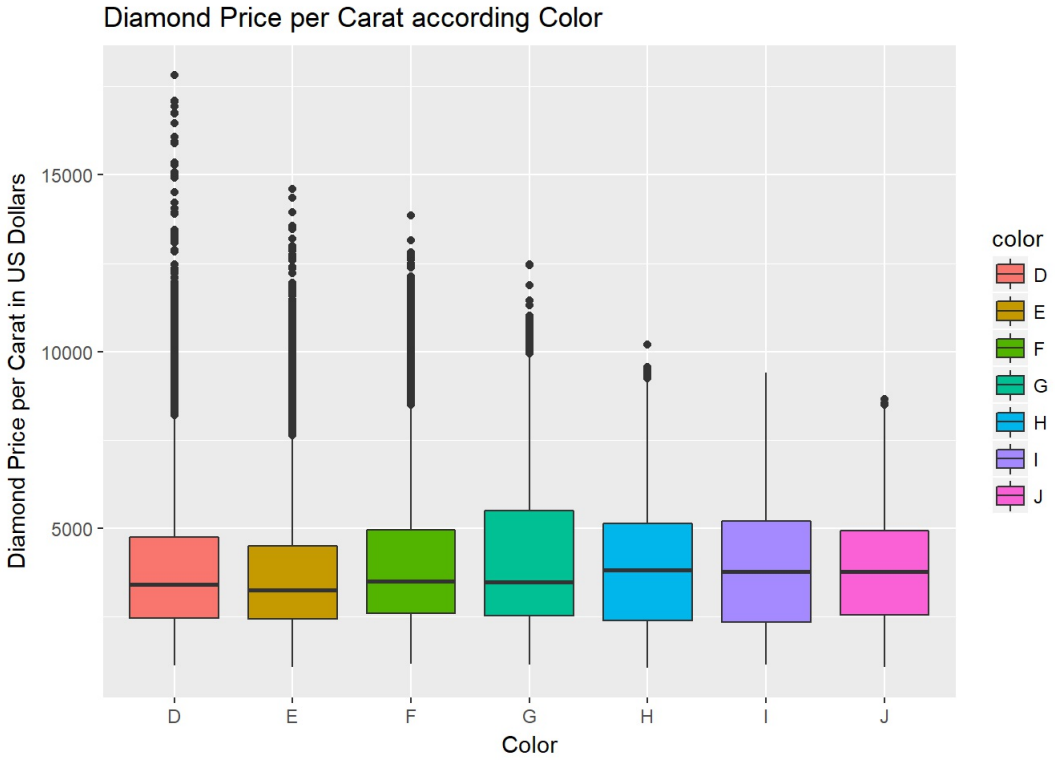

```
#It doesn't appear that Cut is a good way to determine the quality or whether or not a diamond will be expensive. Might want to exclude this variable from the model - think about it. But if we exclude it, it would be the three C's instead of the four C's. If cut does not seem to affect the price very much, then why is it included? Maybe it's a marketing thing or branding effect? It's yet another way to make folks feel like they are picking a higher quality diamond? Maybe it's a "feel better" about your purchase metric? It's interesting why it's included when it does not seem to affect price that much.

#Diamond Price According to Color
ggplot(diamonds, aes(factor(color), price, fill=color)) + geom_boxplot() + ggtitle("Diamond Price according Color") + xlab("Type of Color") + ylab("Diamond Price in US Dollars") #Taken from https://rpubs.com/ameliji/EDA_lesson3 - I modified the code.
```



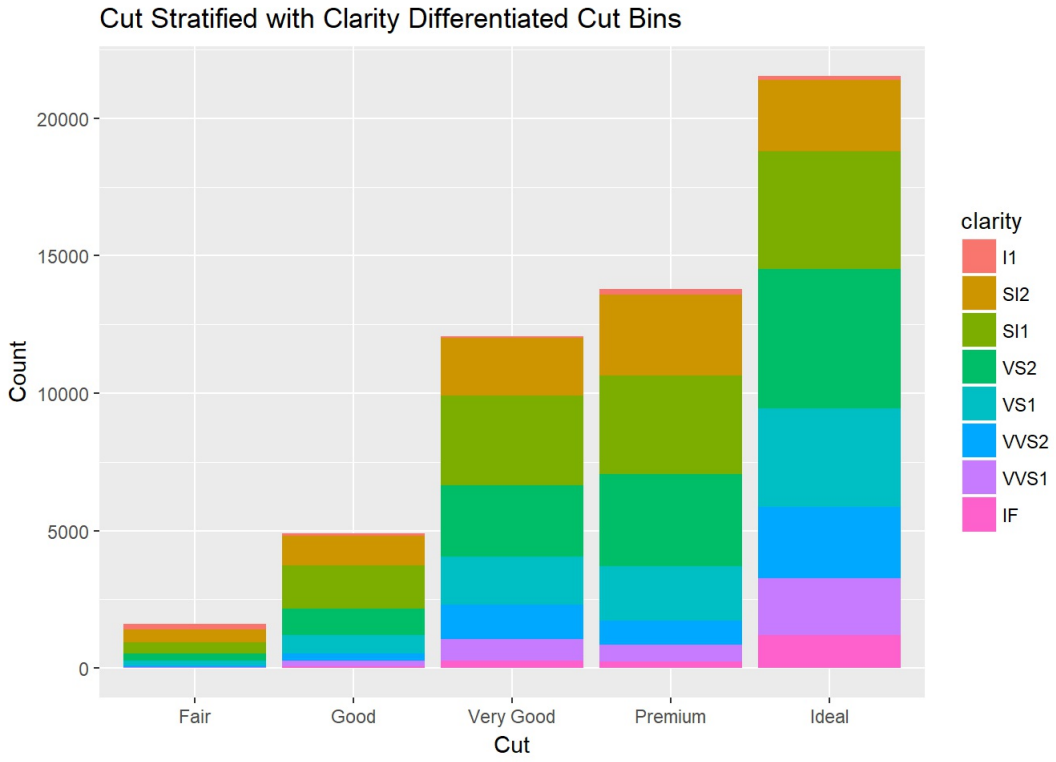

```
#Color looks like it makes a difference in the quality or whether or not a diamond will be expensive - as we would expect. Color is a meaningful variable as compared to cut.

#Diamond Price According to Clarity
ggplot(diamonds, aes(factor(clarity), price, fill=clarity)) + geom_boxplot() + ggtitle("Diamond Price according Clarity") + xlab("Type of Clarity") + ylab("Diamond Price in US Dollars") #Taken from https://rpubs.com/ameliji/EDA_lesson3 - I modified the code.
```




```
#Looks like this matters. Clarity is a meaningful variable as compared to cut.

#Lets take a look at the price per Carat of diamonds across the various Colors using a boxplot.
ggplot(diamonds, aes(factor(color), (price/carat), fill=color)) + geom_boxplot() + ggtitle("Diamond Price per C
arat according Color") + xlab("Color") + ylab("Diamond Price per Carat in US Dollars")#Taken from https://rpubs
.com/ameliji/EDA_lesson3 - I modified the code.
```

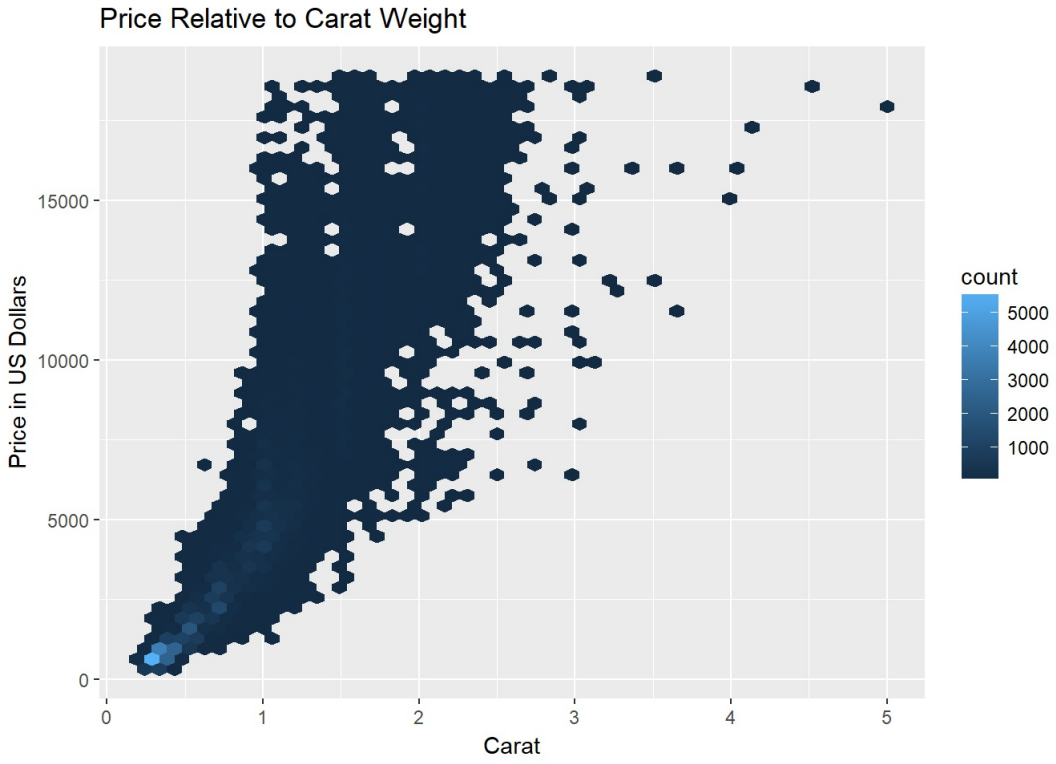



```
#After looking through these plots, it looks like the lesser quality diamonds seem to be more expensive? What e
xplains this? I think the Carat weight comes into play here. Carat weight seems to be the single most determini
ng factor in deciding the price of a diamond.

#Lets try to do some more helpful visualizations for our project to really pack in the power of what our data i
s saying
ggplot(data = diamonds) + geom_bar(mapping = aes(x = cut, fill = clarity)) + ggtitle("Cut Stratified with Clari
ty Differentiated Cut Bins") + xlab("Cut") + ylab("Count") #Wow, I love this and it's a great example of how to
 pack in as much information as possible to make a meaning visualization. I love how I can see the clarity with
in the categorical variable of cut (i.e. the different buckets of cut) #Code was taken from R for Data Science
by Hadley Wickham and Garrett Grolemund, page 27.
```

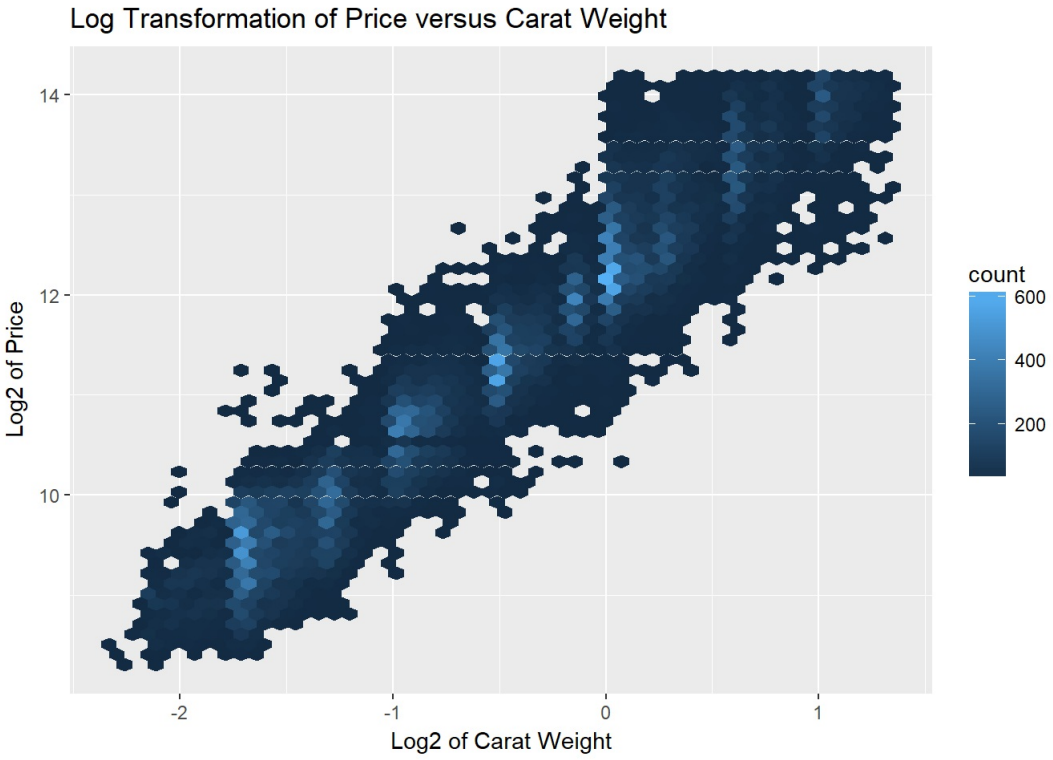



```
#This graph is useful in showing by types of Cut, what the distribution is by clarity. As we would expect, as t
he quality increases in clarity, it gets harder to find or becomes more rare. Though it's interesting that this
 graph could possibly show that maybe jewelers who cut diamonds target an ideal cut for all diamonds, but maybe
 for some reason if it doesn't work out to be an ideal cut, then it becomes a lesser cut? This seems to make se
nse and is a likely conclusion, however I can't prove it. It's anecdotal for now...
```

```
#Let's look at these characteristics with a different plot to see what we can determine for a best model fit? I
s it linear or logarithmic? The below code (lines 248- 301) is taken from pages 378-383 in R for Data Science b
y Hadley Wickham and Garrett Grolemund.

ggplot(diamonds, aes(carat, price)) + geom_hex(bins = 50) + xlab("Carat") + ylab("Price in US Dollars") + ggtit
le("Price Relative to Carat Weight") #Looks like Carat is the most important factor in deciding price, but lets
 do the same with the other attributes / characteristics - this also looks to be exponential in nature as oppos
ed to linear.
```

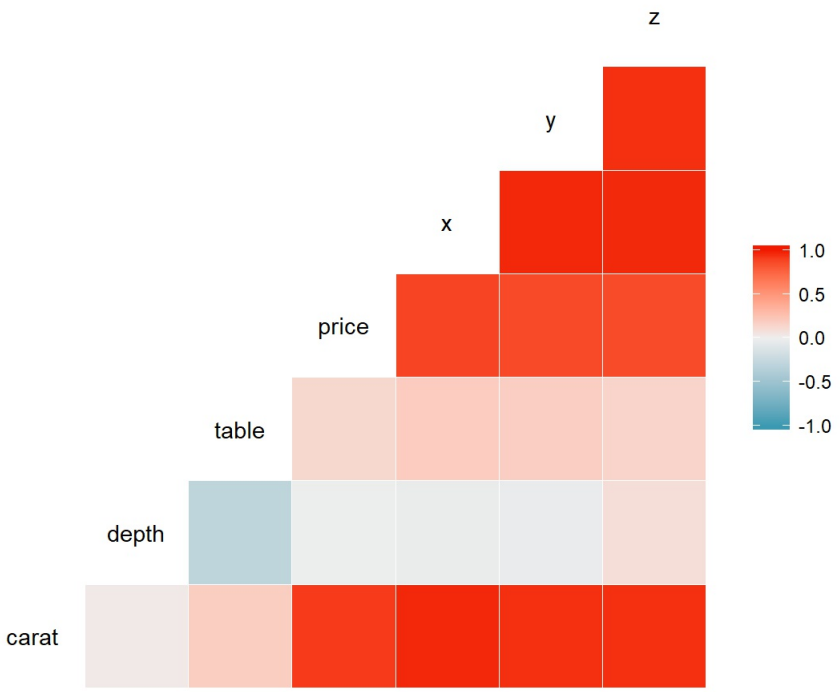


```
#Lets look at diamonds smaller than 2.5 Carats as that seems to be the cut off score for most of the data.

#Using the log function we are transforming the data from logarithmic to that of a linear pattern which makes i
t easier to work with the data.
diamonds2 <- diamonds %>%
  filter(carat <= 2.5) %>%
  mutate(lprice = log2(price), lcarat = log2(carat)) # Got this from the book example from R for Data Science b
y Hadley Wickham and Garrett Grolemund, page 378.

ggplot(diamonds2, aes(lcarat, lprice)) + geom_hex(bins = 50) + ggtitle("Log Transformation of Price versus Cara
t Weight") + xlab("Log2 of Carat Weight") + ylab("Log2 of Price") #Plots the linear relationsip between the var
iables - so taking the Log2 gives us a linear distribution of Carat Weight versus Price.
```




```
# Got this from the book example from R for Data Science by Hadley Wickham and Garrett Grolemund, page 378.

#Let's look at the correlation plot a bit. Price and Carat Weight.
ggcorr(diamonds[,1:10]) #Taken from https://briatte.github.io/ggcorr/. Carat to price seems to be strongly corr
elated as we thought. Let's build a simple model.
```

```
## Warning in ggcorr(diamonds[, 1:10]): data in column(s) 'cut', 'color',
## 'clarity' are not numeric and were ignored
```



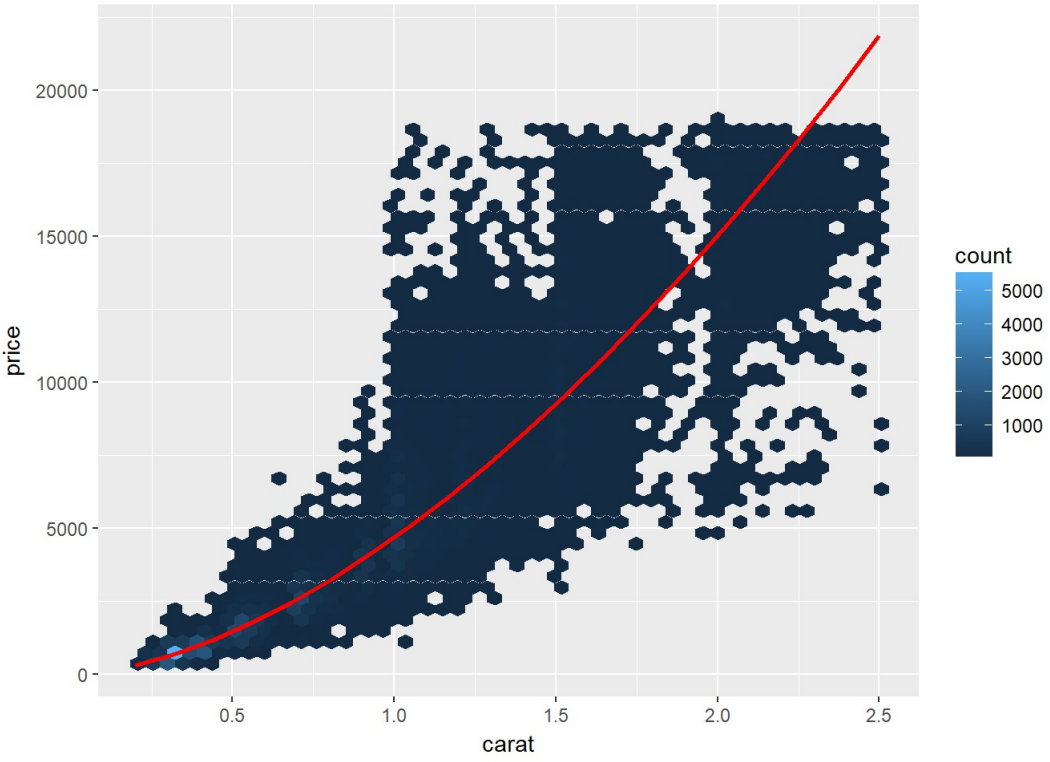

```
model_diamonds <- lm(lprice ~ lcarat, data = diamonds2) #Created linear model - simple - starting off with cara
t first, we will get more complex later. Let's start with price as the predictor value and carat as an independ
ent variable since it seems there is a strong linear relationship between the two variables.
summary(model_diamonds) #Our Adjusted R-squared: 0.9334 which validates that carat weight is a strong determina
nt of price. This relationship will be very important for our predictions later on.
```

```
##
## Call:
## lm(formula = lprice ~ lcarat, data = diamonds2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.96407 -0.24549 -0.00844  0.23930  1.93486
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.193863   0.001969  6194.5   <2e-16 ***
## lcarat       1.681371   0.001936   868.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3767 on 53812 degrees of freedom
## Multiple R-squared:  0.9334, Adjusted R-squared:  0.9334
## F-statistic: 7.542e+05 on 1 and 53812 DF,  p-value: < 2.2e-16
```

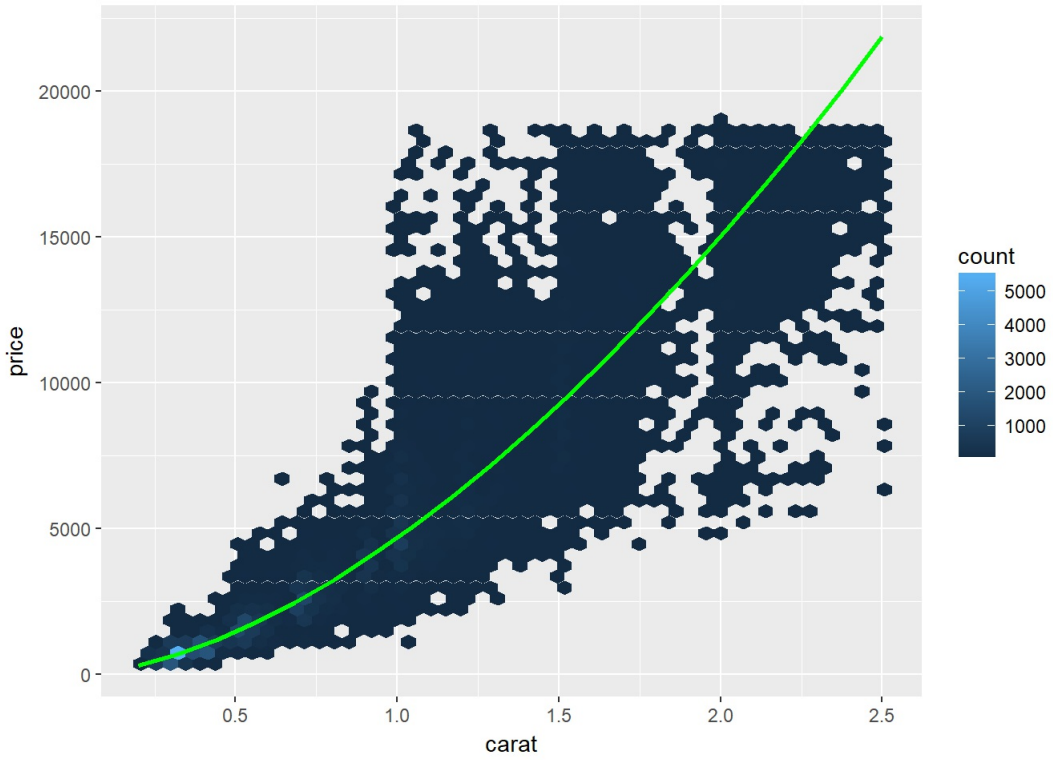
```
model_diamonds1 <- glm(lprice ~ lcarat, data = diamonds2) #Created generalized linear model- starting off with
carat first, we will get more complex later
summary(model_diamonds1) #This output shows a very high AIC: 47,654. Is this model off? Let's plot the line to
see how it fits.
```

```
##
## Call:
## glm(formula = lprice ~ lcarat, data = diamonds2)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -1.96407  -0.24549  -0.00844   0.23930   1.93486
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.193863   0.001969  6194.5   <2e-16 ***
## lcarat       1.681371   0.001936   868.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1419315)
##
##     Null deviance: 114683.4  on 53813  degrees of freedom
## Residual deviance:   7637.6  on 53812  degrees of freedom
## AIC: 47654
##
## Number of Fisher Scoring iterations: 2
```

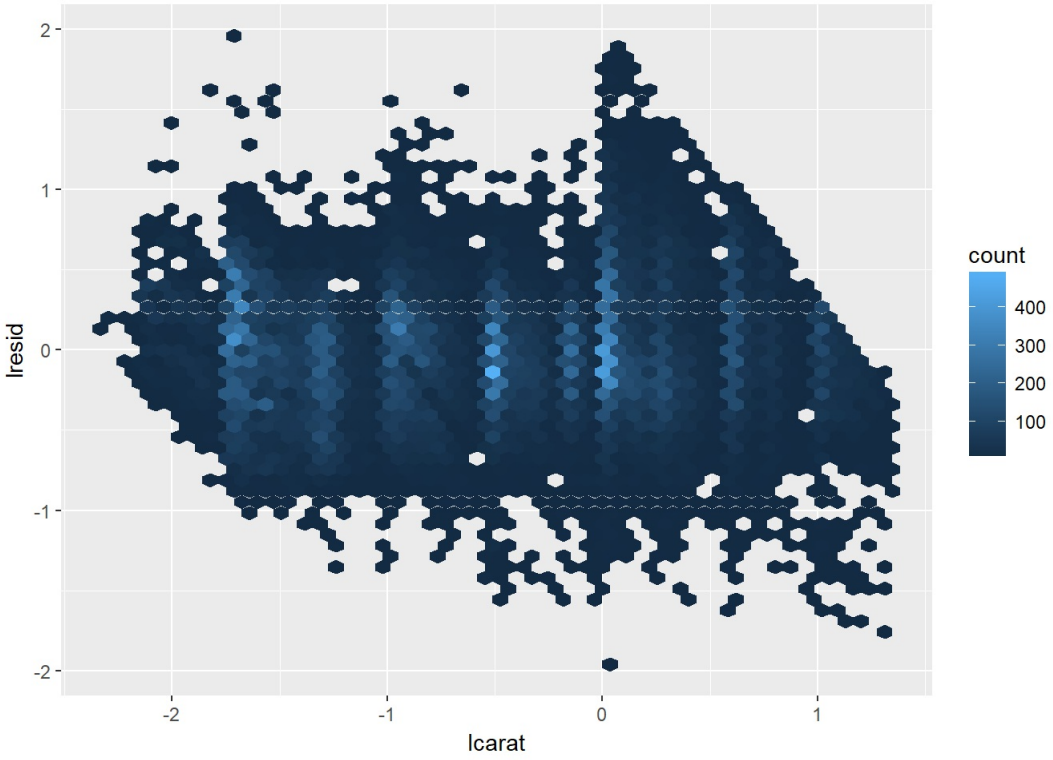
```
#Let's look at what the models tell us about the data

#Linear Model First
grid <- diamonds2 %>%
  data_grid(carat = seq_range(carat, 20)) %>%
  mutate(lcarat = log2(carat)) %>%
  add_predictions(model_diamonds, "lprice") %>%
  mutate(price = 2 ^ lprice)
 #Got this from the book example (Lines 274-305) from R for Data Science by Hadley Wickham and Garrett Grolemun
d, page 379.


ggplot(diamonds2, aes(carat, price)) + geom_hex(bins = 50) + geom_line(data = grid, color = "red", size = 1) #L
ooks like a pretty good fit for the simple linear model
```



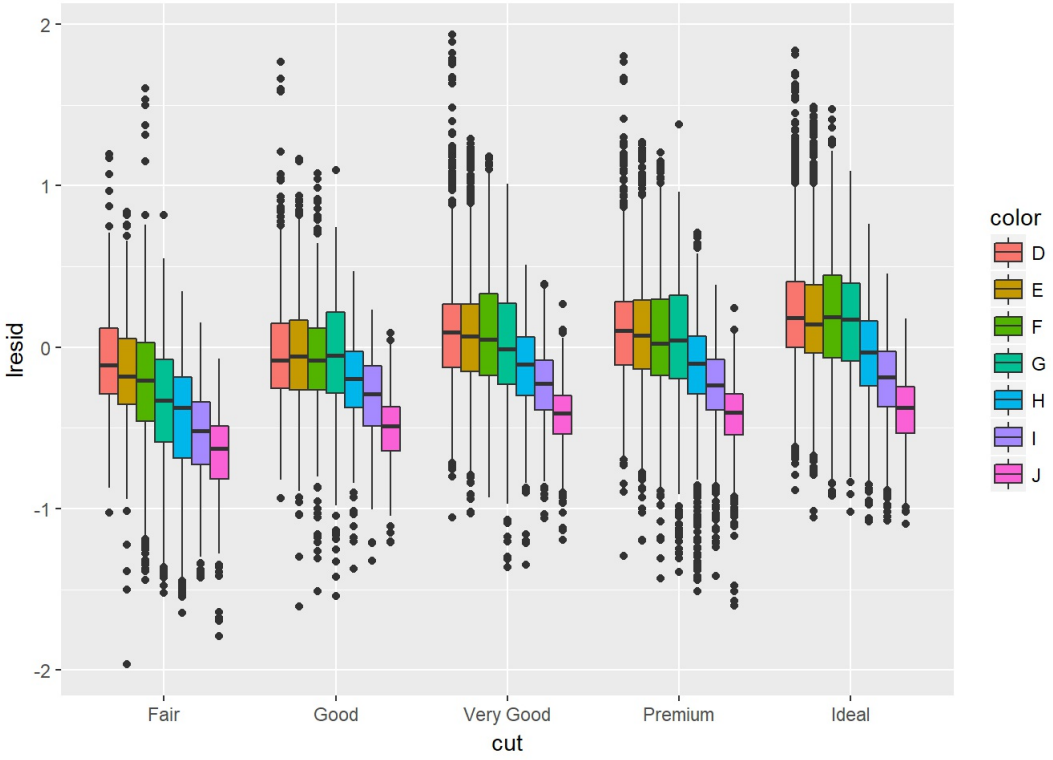
```
#Generalized Linear Model Next
grid1 <- diamonds2 %>%
  data_grid(carat = seq_range(carat, 20)) %>%
  mutate(lcarat = log2(carat)) %>%
  add_predictions(model_diamonds1, "lprice") %>%
  mutate(price = 2 ^ lprice)

ggplot(diamonds2, aes(carat, price)) + geom_hex(bins = 50) + geom_line(data = grid, color = "green", size = 1)
# Plots look the same for the glm
```



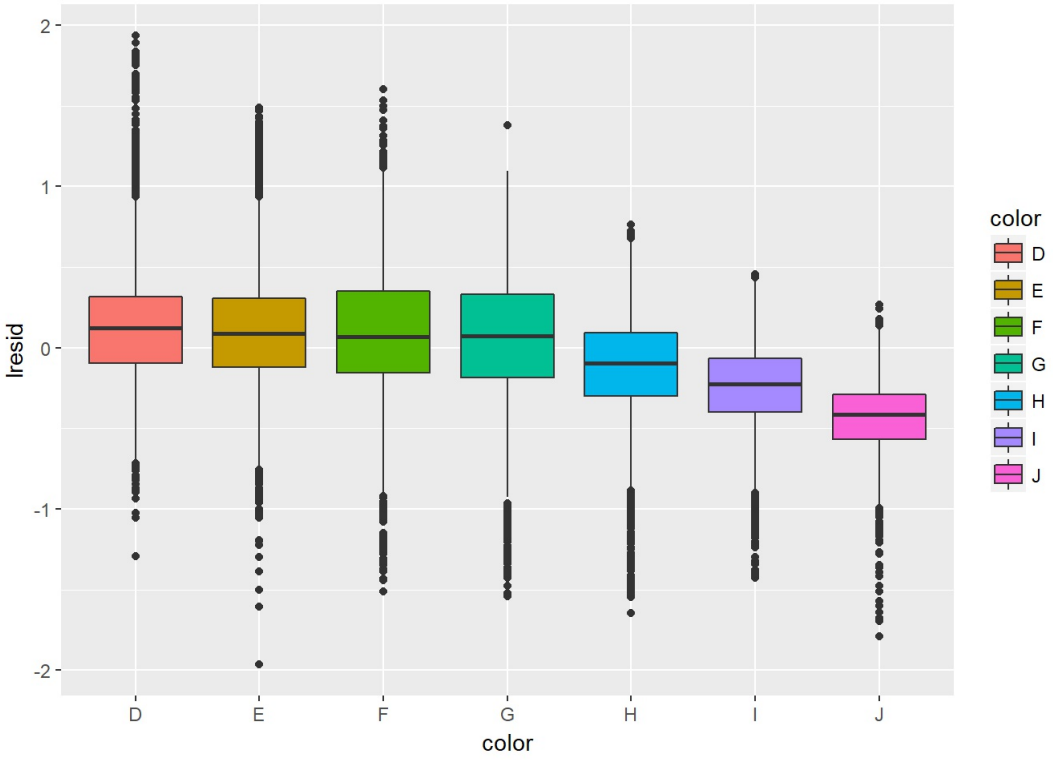

```
#Lets look at the residuals which will help us verify our linear relationship

#Starting with linear model first
diamonds2 <- diamonds2 %>%
  add_residuals(model_diamonds, "lresid")

ggplot(diamonds2, aes(lcarat, lresid)) + geom_hex(bins = 50) #This plot seems to indicate that larger diamonds seem to be cheaper than less expensive diamonds overall. This is an interesting find! A residual closer to 1 means that the diamond price is almost double that of a diamond that has a -1 residual. So I think this means that you get more value for your money if you buy a larger diamond. I hope this to be the case! Gents, if I interpreted this correctly, you may just want to buy the larger diamond so you get more "value" from your purchase. Your money goes further if you buy a larger diamond. Very interesting find! I was not expecting this best value proposition.
```



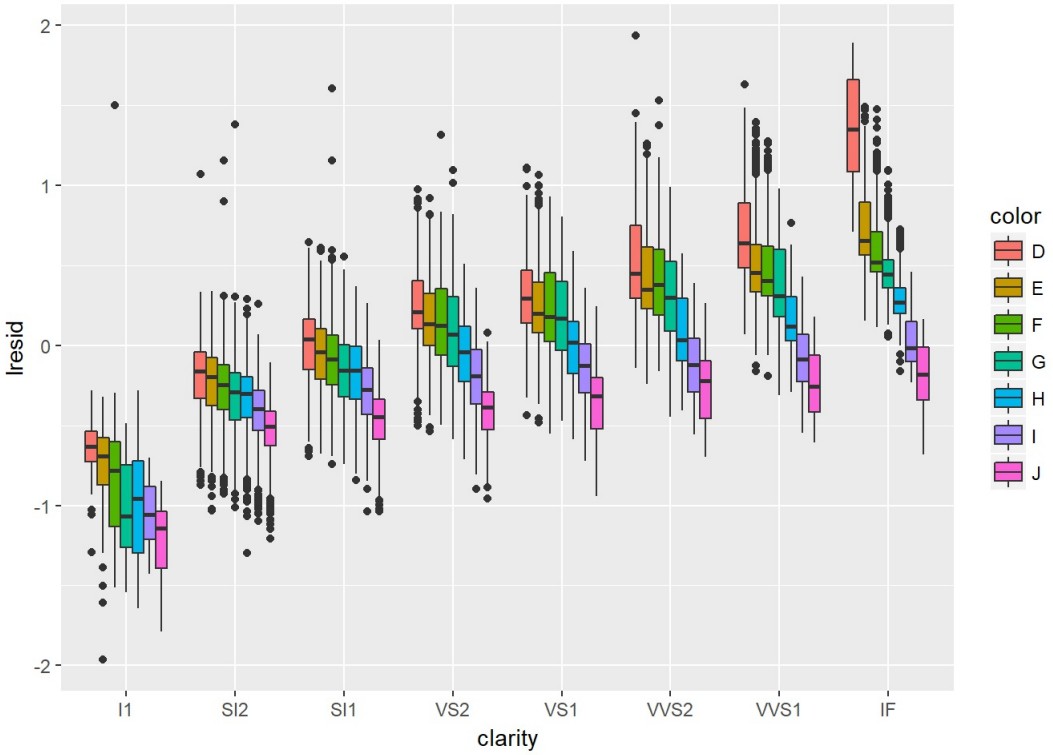

```
#Let's redo our plots using residuals instead of price
ggplot(diamonds2, aes(cut, lresid, fill = color)) + geom_boxplot() #Cut does not seem to affect price that much
 as it increases slightly, but not by much. Does not look like it affects the price much!
```




```
ggplot(diamonds2, aes(color, lresid, fill = color)) + geom_boxplot() #Color seems to affect price. As the quali
ty decreases, so does the price!
```

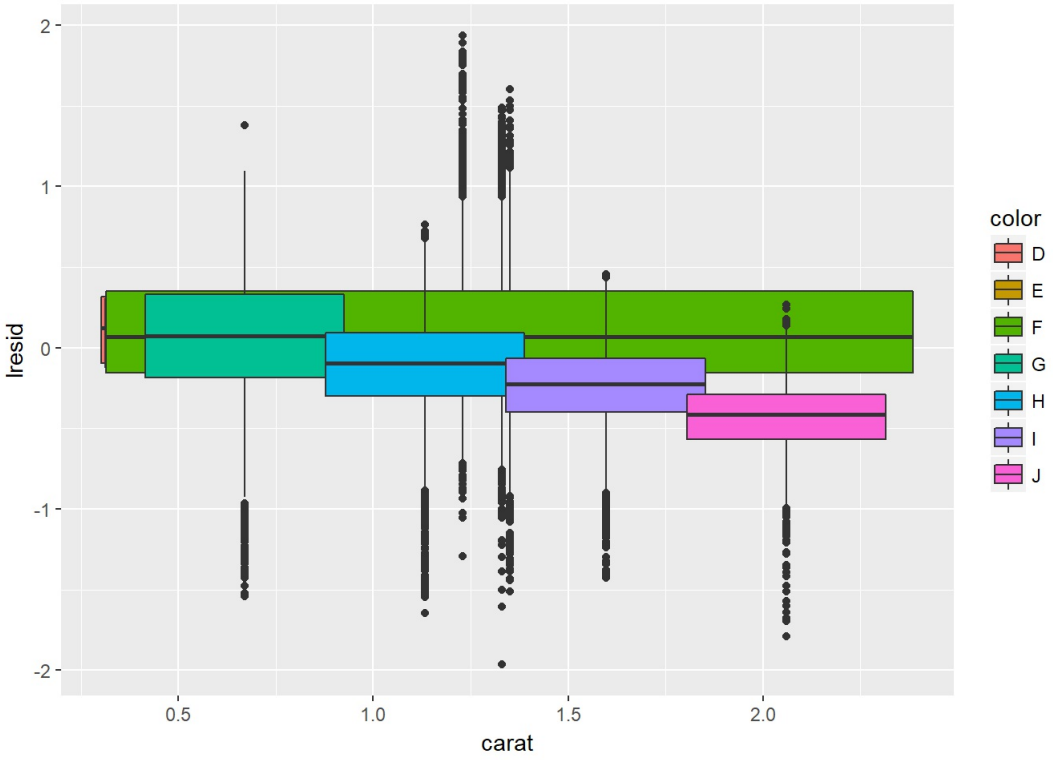



```
ggplot(diamonds2, aes(clarity, lresid, fill = color)) + geom_boxplot() #There looks like a strong residual corr
elation to price for the attribute clarity! The higher the quality of the diamond in terms of clarity, the high
er the price!
```




```
ggplot(diamonds2, aes(carat, lresid, fill = color)) + geom_boxplot() # Aughh, this is an ugly chart. Better to
use the hex bins chart from earlier. This looks horrible! I wonder what happened here? #Meaningless!
```

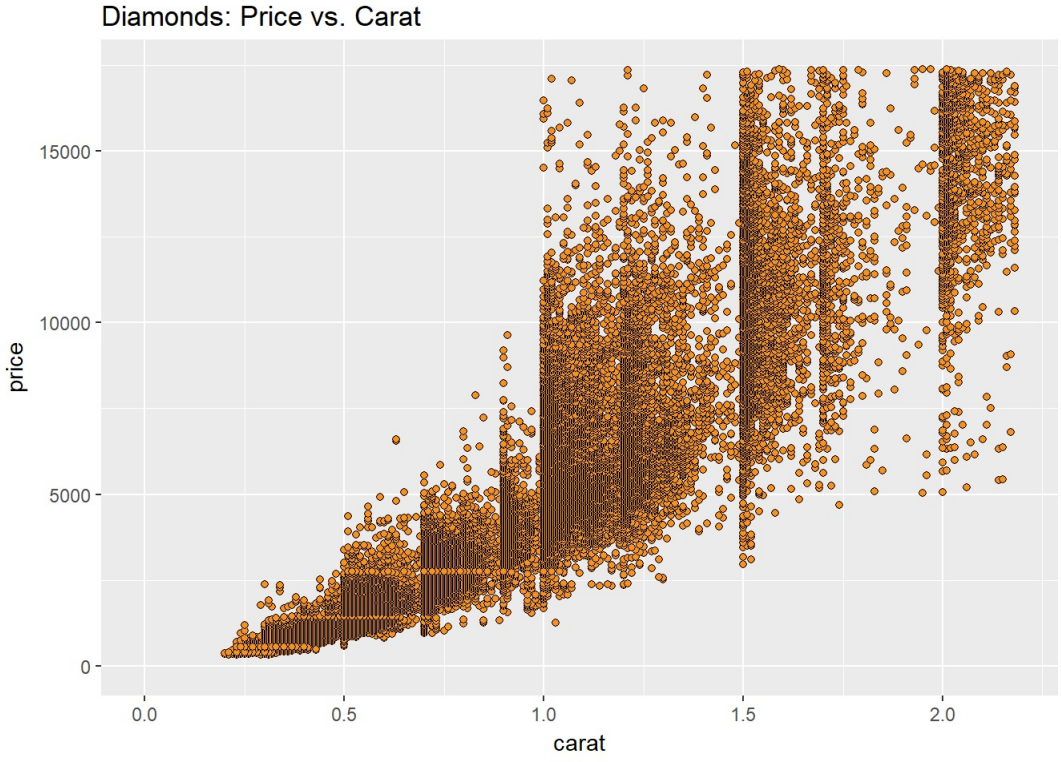
```
## Warning: position_dodge requires non-overlapping x intervals
```




```
#Perfect! This shows the relationship we expect. As the quality of the diamond increases then so does the relat
ive price.Taken from page 381 in R for Data Science by Hadley Wickham and Garrett Grolemund.

#A residual of -1 indicates that lprice was 1 unit lower than a prediction based solely on its weight. This mea
ns that the residual values of -1 are half the predicted price, and residuals with value 1 are twice the predic
ted price.

#Lets now move to a more complex model
```

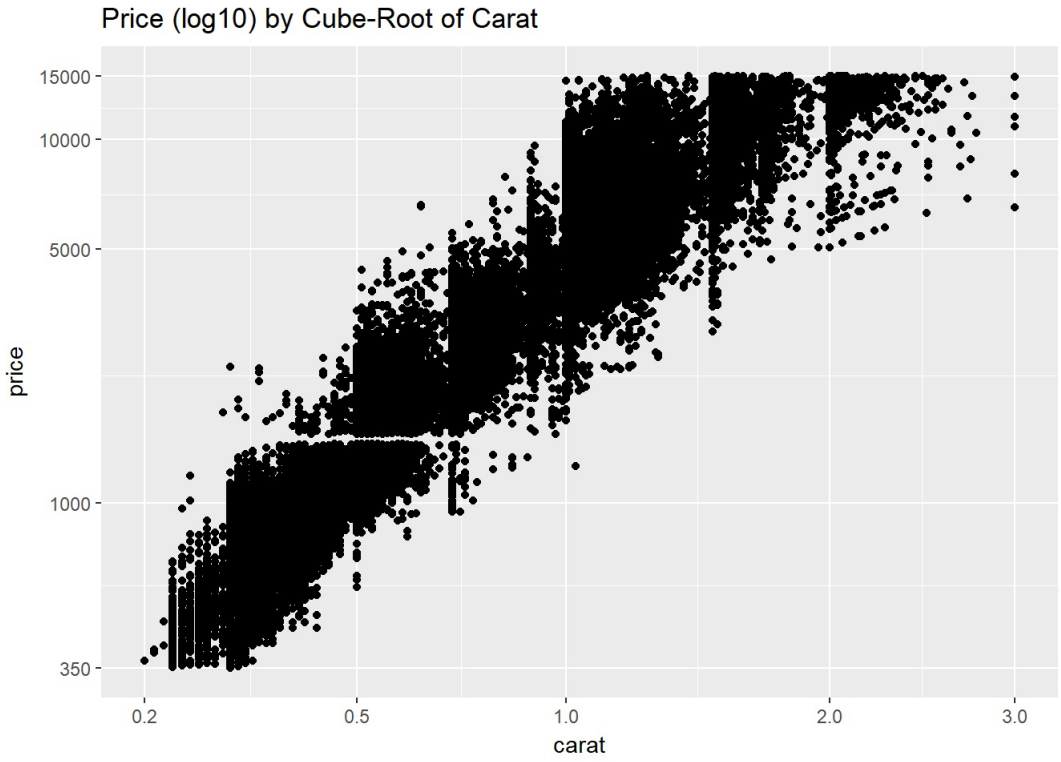
```
#More complex model to evaluate all of the characteristics or attributes of a diamond that affects its price

ggplot(aes(x=carat, y=price), data=diamonds) +
  geom_point(fill=I("#F79420"), color=I("black"), shape=21) +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99)) ) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99)) ) +
  ggtitle("Diamonds: Price vs. Carat") #https://rpubs.com/anthonycerna/diamondspredictions
```

```
## Warning: Removed 926 rows containing missing values (geom_point).
```



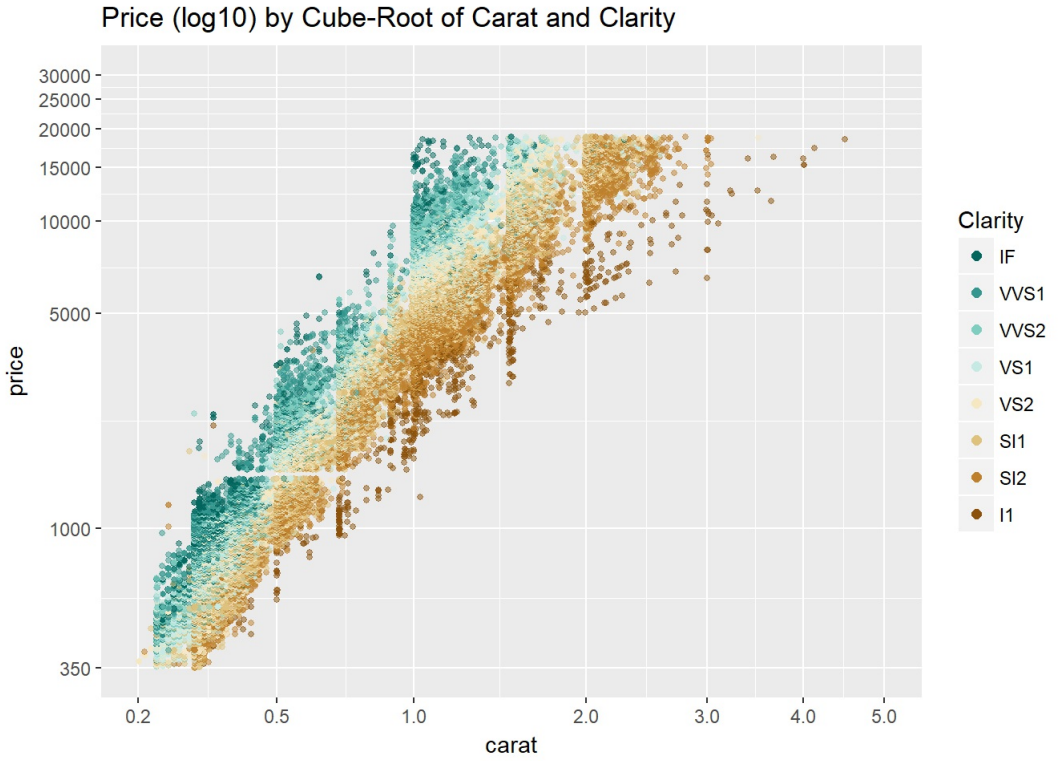
```
#Wow, this is a really cool plot! If you look closely at the plot it shows the supply / demand side of the diam
ond business. That's why at certain Carat Weights the plots look sort of "segmented" for lack of a better term.
 Customers who are looking for less expensive diamonds are more sentive to price versus someone looking for a m
ore expensive diamond. This makes perfect sense, folks with more money aren't worried about price as much - I a
m sure they still worry, but not like someone who can't afford a more expensive diamond. The market for larger
diamonds is not as competitive, so it makes sense that we see a larger variance and price increase.

#Let's remember that earlier it was stated that the volume of a diamond carat = xyz or measured in units cubed
in three dimensional space. So hence this is why we take the cubed root of our carat function. Here's how we do
 it.....
cuberoot_trans = function() trans_new('cuberoot',
                                      transform = function(x) x^(1/3),
                                      inverse = function(x) x^3) #taken from https://rpubs.com/anthonycerna/dia
mondspredictions

#Now that we executed the function - Lets plot it! taken from https://rpubs.com/anthonycerna/diamondspredictio
ns
ggplot(aes(carat, price), data = diamonds) +
  geom_point() +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat')
```

```
## Warning: Removed 1683 rows containing missing values (geom_point).
```



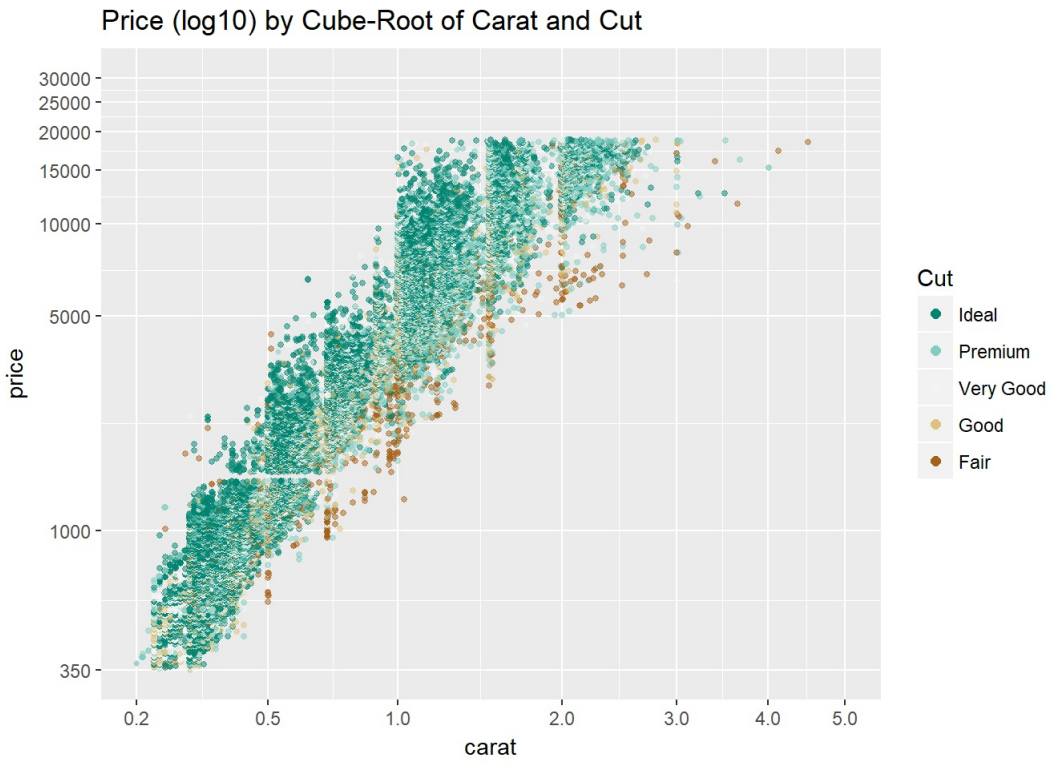

```
#This is almost linear - I think about as close as we'll get!

#Let's look at the above plot but with some added color and of course let's plot it with the clarity "bins" for
 the diamonds, just to see what it gives us...
ggplot(aes(x = carat, y = price, color=clarity), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type='div',
                     guide=guide_legend(title='Clarity',
                                        reverse=T,
                                        override.aes=list(alpha=1, size=2))) +
  scale_x_continuous(trans = cuberoot_trans(),
                     limits = c(0.2, 5),
                     breaks = c(0.2, 0.5, 1, 2, 3, 4, 5)) +
  scale_y_continuous(trans = log10_trans(),
                     limits = c(350, 30000),
                     breaks = c(350, 1000, 5000, 10000, 15000, 20000, 25000, 30000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Clarity') #This was taken from https://rpubs.com/anthonycern
a/diamondspredictions
```

```
## Warning: Removed 23 rows containing missing values (geom_point).
```



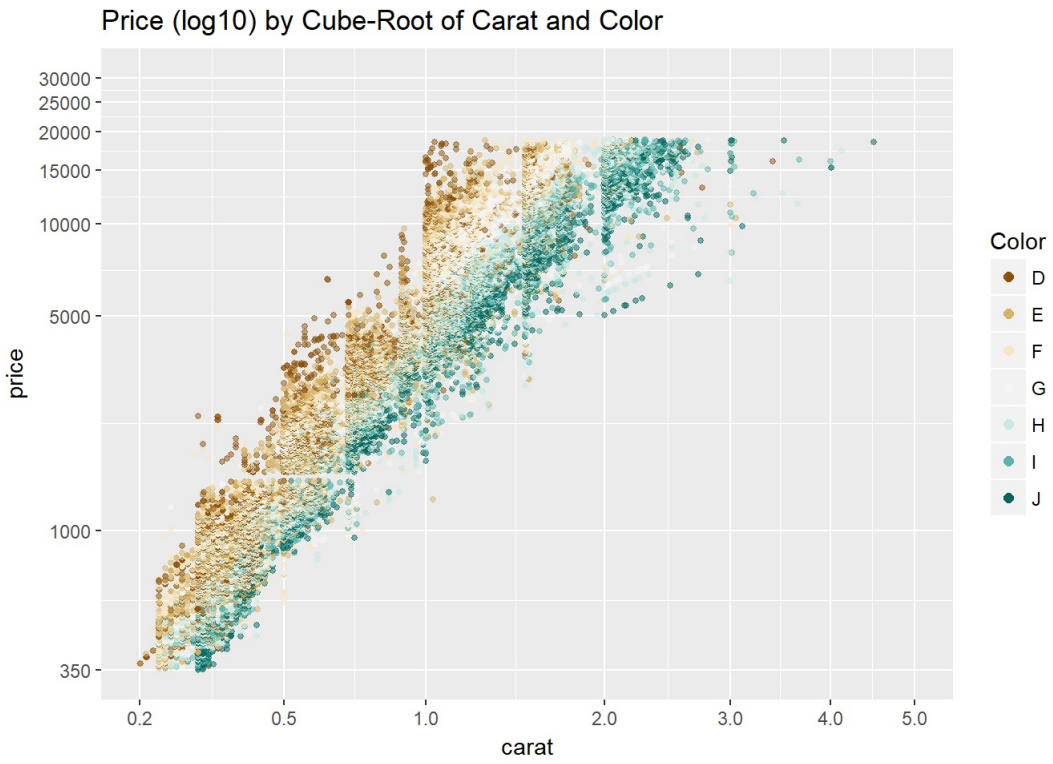

```
#Wow, this tells us a lot about the diamond business as it relates to price and the distribution of the various
 diamonds. The larger the diamond more it seems from our data the quality available seems to decrease as eviden
ced by the lack of IF at higher levels. It's very clear the IF for Internally Flawless are the most expensive d
iamonds (as expected) and the I and SI2 diamonds are the least expensive. This is in line with what we expect b
ased on our domain knowledge of diamonds.

#I really like this plot. I am glad I picked this data set and researched the various plots and found good exam
ples. This is something I will throw in my "kit bag" for later use! I never would have learned this if I had no
t researched these different plots. This will come in useful for other factor variable data sets. As an aside,
I need to learn how to manipulate the colors found in this graph.

#Lets do the same plot, but for cut to see if we see a variance in the price.

ggplot(aes(x = carat, y = price, color = cut), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Cut',
                                          reverse = T,
                                          override.aes = list(alpha = 1,
                                                              size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 5),
                     breaks = c(0.2, 0.5, 1, 2, 3, 4, 5)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 30000),
                     breaks = c(350, 1000, 5000, 10000, 15000, 20000, 25000, 30000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Cut') #This was taken from https://rpubs.com/anthonycerna/di
amondspredictions
```

```
## Warning: Removed 24 rows containing missing values (geom_point).
```



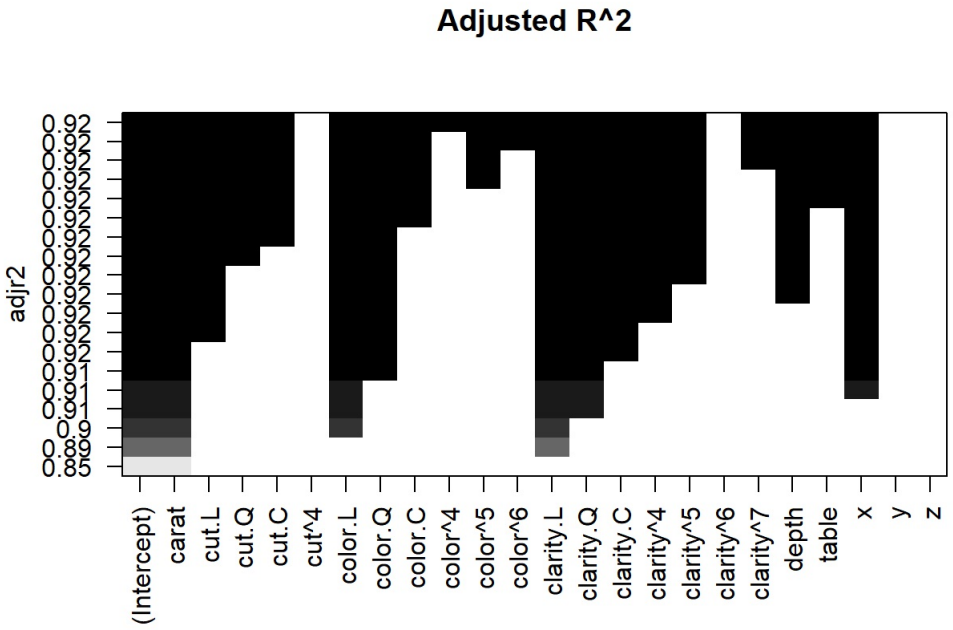

```
#Looking at this data, it seems that cut does not account for a variance in the price. The data shows the bulk of the cut are either ideal or premium. It appears that Carat weight is still the most significant contributing  factor to diamond price. I can see why a plot like this can help you make sense of the data. I learned a lot f rom researching and using these codes. I learned a whole new way of looking at data and how to interpret it. I definitely encourage others to play with various plots to see you can go deeper into the data.

#Lastly, lets look at the price against the Carat and Color and see what we get.....
ggplot(aes(x = carat, y = price, color = color), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = "Color",
                                          reverse = F,
                                          override.aes = list(alpha = 1,
                                                              size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 5),
                     breaks = c(0.2, 0.5, 1, 2, 3, 4, 5)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 30000),
                     breaks = c(350, 1000, 5000, 10000, 15000, 20000, 25000, 30000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Color') #Taken from https://rpubs.com/anthonycerna/diamondsp redictions
```

```
## Warning: Removed 25 rows containing missing values (geom_point).
```




```
#We see the same thing here that we did for clarity. Diamond color does contribute to price. This is something we would expect for those of us who have bought diamonds before. This is consistent with our domain knowledge o f diamonds.
```

```
#Variable selection - Let's validate we were right with our variables....from the above analysis.

#Lets start with the best line fit even though it's "computationally expensive".
View(diamonds) #Opens up the data set so we can view it.

reg.best <- regsubsets(price~., data = diamonds, nvmax = 19)#Price is the dependent variable
#The plot will show the Adjust R^2 when using the variables across the bottom
plot(reg.best, scale = "adjr2", main = "Adjusted R^2")
```

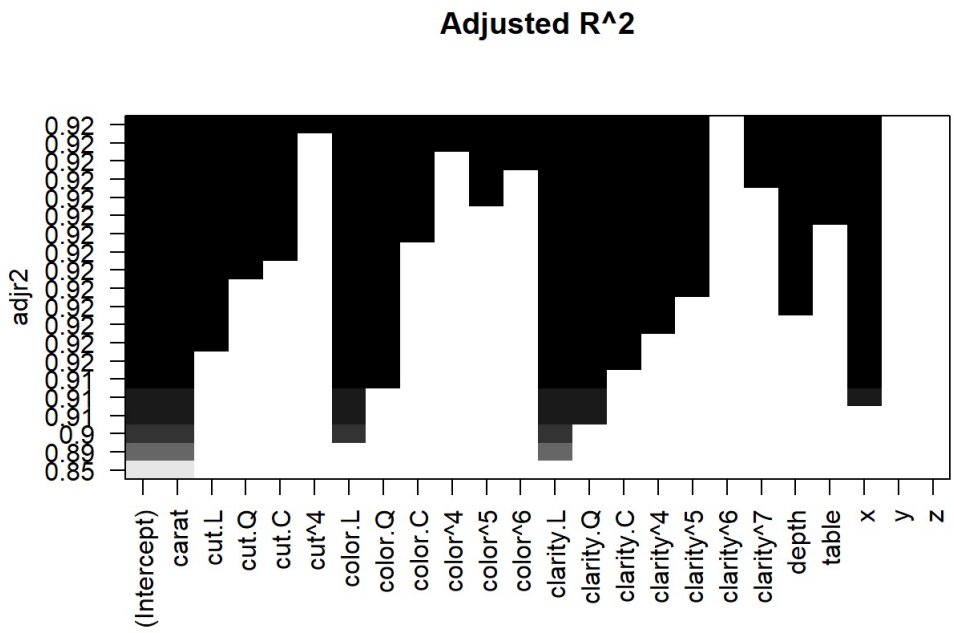


```
summary(reg.best) # basically what I get out of this is the variable categories of carat, cut, color and clarit
y are important. I excluded x as it doesn't make sense to include this variable. Maybe this value and Depth exh
ibit some multicolinearity? Or this could be possible due to some sort of preference amongst customers? Or mayb
e it's just a fluke variable interaction? Maybe jewelers unknowingly end up cutting certain diamonds to exhibit
 more "sparkle" or "fire"? Maybe this is related to the cut, such as Ideal? Would have to do somoe more explori
ng to see if this is simply due to chance or if there's a more significant reason involved. I can only speculat
e for right now...
```

```
## Subset selection object
## Call: regsubsets.formula(price ~ ., data = diamonds, nvmax = 19)
## 23 Variables  (and intercept)
##           Forced in Forced out
## carat         FALSE      FALSE
## cut.L         FALSE      FALSE
## cut.Q         FALSE      FALSE
## cut.C         FALSE      FALSE
## cut^4         FALSE      FALSE
## color.L       FALSE      FALSE
## color.Q       FALSE      FALSE
## color.C       FALSE      FALSE
## color^4       FALSE      FALSE
## color^5       FALSE      FALSE
## color^6       FALSE      FALSE
## clarity.L     FALSE      FALSE
## clarity.Q     FALSE      FALSE
## clarity.C     FALSE      FALSE
## clarity^4     FALSE      FALSE
## clarity^5     FALSE      FALSE
## clarity^6     FALSE      FALSE
## clarity^7     FALSE      FALSE
## depth         FALSE      FALSE
## table         FALSE      FALSE
## x             FALSE      FALSE
## y             FALSE      FALSE
## z             FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##           carat cut.L cut.Q cut.C cut^4 color.L color.Q color.C color^4
## 1  ( 1 )  "*"   " "   " "   " "   " "   " "     " "     " "     " "
## 2  ( 1 )  "*"   " "   " "   " "   " "   " "     " "     " "     " "
## 3  ( 1 )  "*"   " "   " "   " "   " "   "*"     " "     " "     " "
## 4  ( 1 )  "*"   " "   " "   " "   " "   "*"     " "     " "     " "
## 5  ( 1 )  "*"   " "   " "   " "   " "   "*"     " "     " "     " "
## 6  ( 1 )  "*"   " "   " "   " "   " "   "*"     "*"     " "     " "
## 7  ( 1 )  "*"   " "   " "   " "   " "   "*"     "*"     " "     " "
## 8  ( 1 )  "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 9  ( 1 )  "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 10  ( 1 ) "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 11  ( 1 ) "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 12  ( 1 ) "*"   "*"   "*"   " "   " "   "*"     "*"     " "     " "
## 13  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     " "     " "
## 14  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 15  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 16  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 17  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 18  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 19  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     "*"
##           color^5 color^6 clarity.L clarity.Q clarity.C clarity^4
## 1  ( 1 )  " "     " "     " "       " "       " "       " "
## 2  ( 1 )  " "     " "     "*"       " "       " "       " "
## 3  ( 1 )  " "     " "     "*"       " "       " "       " "
## 4  ( 1 )  " "     " "     "*"       "*"       " "       " "
## 5  ( 1 )  " "     " "     "*"       "*"       " "       " "
## 6  ( 1 )  " "     " "     "*"       "*"       " "       " "
## 7  ( 1 )  " "     " "     "*"       "*"       "*"       " "
## 8  ( 1 )  " "     " "     "*"       "*"       "*"       " "
## 9  ( 1 )  " "     " "     "*"       "*"       "*"       "*"
## 10  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 11  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 12  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 13  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 14  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 15  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 16  ( 1 ) "*"     " "     "*"       "*"       "*"       "*"
```

```
## 17  ( 1 ) "*"      " "      "*"        "*"        "*"        "*"
## 18  ( 1 ) "*"      "*"      "*"        "*"        "*"        "*"
## 19  ( 1 ) "*"      "*"      "*"        "*"        "*"        "*"
##           clarity^5 clarity^6 clarity^7 depth table x   y   z
## 1  ( 1 )  " "        " "        " "        " "   " "   " " " " " "
## 2  ( 1 )  " "        " "        " "        " "   " "   " " " " " "
## 3  ( 1 )  " "        " "        " "        " "   " "   " " " " " "
## 4  ( 1 )  " "        " "        " "        " "   " "   " " " " " "
## 5  ( 1 )  " "        " "        " "        " "   " "   "*" " " " "
## 6  ( 1 )  " "        " "        " "        " "   " "   "*" " " " "
## 7  ( 1 )  " "        " "        " "        " "   " "   "*" " " " "
## 8  ( 1 )  " "        " "        " "        " "   " "   "*" " " " "
## 9  ( 1 )  " "        " "        " "        " "   " "   "*" " " " "
## 10  ( 1 ) " "        " "        " "        "*"   " "   "*" " " " "
## 11  ( 1 ) "*"        " "        " "        "*"   " "   "*" " " " "
## 12  ( 1 ) "*"        " "        " "        "*"   " "   "*" " " " "
## 13  ( 1 ) "*"        " "        " "        "*"   " "   "*" " " " "
## 14  ( 1 ) "*"        " "        " "        "*"   " "   "*" " " " "
## 15  ( 1 ) "*"        " "        " "        "*"   "*"   "*" " " " "
## 16  ( 1 ) "*"        " "        " "        "*"   "*"   "*" " " " "
## 17  ( 1 ) "*"        " "        "*"        "*"   "*"   "*" " " " "
## 18  ( 1 ) "*"        " "        "*"        "*"   "*"   "*" " " " "
## 19  ( 1 ) "*"        " "        "*"        "*"   "*"   "*" " " " "
```

```
#Lets try backward selection to see if we get something different
reg.back <- regsubsets(price~., data = diamonds, method = "backward", nvmax = 20)
#The plot will show the Adjust R^2 when using the variables across the bottom
plot(reg.back, scale = "adjr2", main = "Adjusted R^2")
```




```
summary(reg.back)
```

```
## Subset selection object
## Call: regsubsets.formula(price ~ ., data = diamonds, method = "backward",
##     nvmax = 20)
## 23 Variables  (and intercept)
##           Forced in Forced out
## carat         FALSE      FALSE
## cut.L         FALSE      FALSE
## cut.Q         FALSE      FALSE
## cut.C         FALSE      FALSE
## cut^4         FALSE      FALSE
## color.L       FALSE      FALSE
## color.Q       FALSE      FALSE
## color.C       FALSE      FALSE
## color^4       FALSE      FALSE
## color^5       FALSE      FALSE
```

```
## color^6       FALSE      FALSE
## clarity.L     FALSE      FALSE
## clarity.Q     FALSE      FALSE
## clarity.C     FALSE      FALSE
## clarity^4     FALSE      FALSE
## clarity^5     FALSE      FALSE
## clarity^6     FALSE      FALSE
## clarity^7     FALSE      FALSE
## depth         FALSE      FALSE
## table         FALSE      FALSE
## x             FALSE      FALSE
## y             FALSE      FALSE
## z             FALSE      FALSE
## 1 subsets of each size up to 20
## Selection Algorithm: backward
##           carat cut.L cut.Q cut.C cut^4 color.L color.Q color.C color^4
## 1  ( 1 )  "*"   " "   " "   " "   " "   " "     " "     " "     " "
## 2  ( 1 )  "*"   " "   " "   " "   " "   " "     " "     " "     " "
## 3  ( 1 )  "*"   " "   " "   " "   " "   "*"     " "     " "     " "
## 4  ( 1 )  "*"   " "   " "   " "   " "   "*"     " "     " "     " "
## 5  ( 1 )  "*"   " "   " "   " "   " "   "*"     " "     " "     " "
## 6  ( 1 )  "*"   " "   " "   " "   " "   "*"     "*"     " "     " "
## 7  ( 1 )  "*"   " "   " "   " "   " "   "*"     "*"     " "     " "
## 8  ( 1 )  "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 9  ( 1 )  "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 10  ( 1 ) "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 11  ( 1 ) "*"   "*"   " "   " "   " "   "*"     "*"     " "     " "
## 12  ( 1 ) "*"   "*"   "*"   " "   " "   "*"     "*"     " "     " "
## 13  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     " "     " "
## 14  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 15  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 16  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 17  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 18  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     " "
## 19  ( 1 ) "*"   "*"   "*"   "*"   " "   "*"     "*"     "*"     "*"
## 20  ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"     "*"     "*"     "*"
##           color^5 color^6 clarity.L clarity.Q clarity.C clarity^4
## 1  ( 1 )  " "     " "     " "       " "       " "       " "
## 2  ( 1 )  " "     " "     "*"       " "       " "       " "
## 3  ( 1 )  " "     " "     "*"       " "       " "       " "
## 4  ( 1 )  " "     " "     "*"       "*"       " "       " "
## 5  ( 1 )  " "     " "     "*"       "*"       " "       " "
## 6  ( 1 )  " "     " "     "*"       "*"       " "       " "
## 7  ( 1 )  " "     " "     "*"       "*"       "*"       " "
## 8  ( 1 )  " "     " "     "*"       "*"       "*"       " "
## 9  ( 1 )  " "     " "     "*"       "*"       "*"       "*"
## 10  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 11  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 12  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 13  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 14  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 15  ( 1 ) " "     " "     "*"       "*"       "*"       "*"
## 16  ( 1 ) "*"     " "     "*"       "*"       "*"       "*"
## 17  ( 1 ) "*"     " "     "*"       "*"       "*"       "*"
## 18  ( 1 ) "*"     "*"     "*"       "*"       "*"       "*"
## 19  ( 1 ) "*"     "*"     "*"       "*"       "*"       "*"
## 20  ( 1 ) "*"     "*"     "*"       "*"       "*"       "*"
##           clarity^5 clarity^6 clarity^7 depth table x   y   z
## 1  ( 1 )  " "       " "       " "       " "   " "   " " " " " "
## 2  ( 1 )  " "       " "       " "       " "   " "   " " " " " "
## 3  ( 1 )  " "       " "       " "       " "   " "   " " " " " "
## 4  ( 1 )  " "       " "       " "       " "   " "   " " " " " "
## 5  ( 1 )  " "       " "       " "       " "   " "   "*" " " " "
## 6  ( 1 )  " "       " "       " "       " "   " "   "*" " " " "
## 7  ( 1 )  " "       " "       " "       " "   " "   "*" " " " "
## 8  ( 1 )  " "       " "       " "       " "   " "   "*" " " " "
## 9  ( 1 )  " "       " "       " "       " "   " "   "*" " " " "
## 10  ( 1 ) " "       " "       " "       "*"   " "   "*" " " " "
## 11  ( 1 ) "*"       " "       " "       "*"   " "   "*" " " " "
## 12  ( 1 ) "*"       " "       " "       "*"   " "   "*" " " " "
## 13  ( 1 ) "*"       " "       " "       "*"   " "   "*" " " " "
## 14  ( 1 ) "*"       " "       " "       "*"   " "   "*" " " " "
## 15  ( 1 ) "*"       " "       " "       "*"   "*"   "*" " " " "
## 16  ( 1 ) "*"       " "       " "       "*"   "*"   "*" " " " "
## 17  ( 1 ) "*"       " "       "*"       "*"   "*"   "*" " " " "
## 18  ( 1 ) "*"       " "       "*"       "*"   "*"   "*" " " " "
```

```
## 19  ( 1 ) "*"       " "       "*"       "*"   "*"   "*" " " " " "
## 20  ( 1 ) "*"       " "       "*"       "*"   "*"   "*" " " " " "
```

```
#I think this confirms the best line fit recommendations of variables, looks to me like carat, cut, color and c
larity are most important in determining the price of a diamond. This is consistent with the above analysis, th
ough the above analysis was more beneficial as we know that cut seems to be the least response or rather add th
e least to the price of a diamond. This seems to validate our best fit method. Again going with the industry st
andard four C's and excluding x and depth. Interesting that these showed up again!

#Finally, let's move on to building a more complex model and then eventually predicting some results
```

```
#Model Creation, on our way to building the best model to predict the price of diamonds based on its contributi
ng attributes.
#We need to remember that since carat weight is determined by volume in three dimensional space xyz= volume and
 hence carat weight, then we can use the following equation: lm(log(price) ~ carat^(1/3)). This seems very reas
onable since we can see that as the carat weight increases the rarer and rarer a diamond becomes until we don't
 see it any more, or its so rare it is not depicted in the sample size. Makes perfect sense! The larger the dia
mond, the exponentially harder it is to find an internally flawless (IF) diamond.

#I will predict the diamond prices from the model and compare it to that of the same diamond with same specific
ations from the BlueNile.com website and see what we get. Should be an interesting experiment.

#Let's create the variables first

diamondsdata <- data.frame(diamonds, header = TRUE)

#Create variables
carat <- diamonds$carat
cut <- diamonds$cut
color <- diamonds$color
clarity <- diamonds$clarity
price <- diamonds$price

#this will tells us if we have missing values
sum(is.na(carat)) # Does not appear that we have any missing values for carat
```

```
## [1] 0
```

```
#Let's first use the Linear Model based on the example...
lm1 <- lm(I(log(price)) ~ I(carat^(1/3)), data = diamonds)
lm2 <- update(lm1, ~ . + carat)
lm3 <- update(lm2, ~ . + cut)
lm4 <- update(lm3, ~ . + color)
lm5 <- update(lm4, ~ . + clarity)
mtable(lm1, lm2, lm3, lm4, lm5)
```

```
##
## Calls:
## lm1: lm(formula = I(log(price)) ~ I(carat^(1/3)), data = diamonds)
## lm2: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat, data = diamonds)
## lm3: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut, data = diamonds)
## lm4: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut + color,
##     data = diamonds)
## lm5: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut + color +
##     clarity, data = diamonds)
##
## ================================================================================================
##                         lm1            lm2            lm3            lm4            lm5
## ------------------------------------------------------------------------------------------------
##   (Intercept)          2.821***       1.039***       0.874***       0.932***       0.415***
##                       (0.006)        (0.019)        (0.019)        (0.017)        (0.010)
##   I(carat^(1/3))       5.558***       8.568***       8.703***       8.438***       9.144***
##                       (0.007)        (0.032)        (0.031)        (0.028)        (0.016)
##   carat                              -1.137***      -1.163***      -0.992***      -1.093***
##                                      (0.012)        (0.011)        (0.010)        (0.006)
##   cut: .L                                            0.224***       0.224***       0.120***
##                                                     (0.004)        (0.004)        (0.002)
##   cut: .Q                                           -0.062***      -0.062***      -0.031***
##                                                     (0.004)        (0.003)        (0.002)
##   cut: .C                                            0.051***       0.052***       0.014***
##                                                     (0.003)        (0.003)        (0.002)
##   cut: ^4                                            0.018***       0.018***      -0.002
##                                                     (0.003)        (0.002)        (0.001)
##   color: .L                                                        -0.373***      -0.441***
##                                                                    (0.003)        (0.002)
##   color: .Q                                                        -0.129***      -0.093***
##                                                                    (0.003)        (0.002)
##   color: .C                                                         0.001         -0.013***
##                                                                    (0.003)        (0.002)
##   color: ^4                                                         0.029***       0.012***
##                                                                    (0.003)        (0.002)
##   color: ^5                                                        -0.016***      -0.003*
##                                                                    (0.003)        (0.001)
##   color: ^6                                                        -0.023***       0.001
##                                                                    (0.002)        (0.001)
##   clarity: .L                                                                      0.907***
##                                                                                   (0.003)
##   clarity: .Q                                                                     -0.240***
##                                                                                   (0.003)
##   clarity: .C                                                                      0.131***
##                                                                                   (0.003)
##   clarity: ^4                                                                     -0.063***
##                                                                                   (0.002)
##   clarity: ^5                                                                      0.026***
##                                                                                   (0.002)
##   clarity: ^6                                                                     -0.002
##                                                                                   (0.002)
##   clarity: ^7                                                                      0.032***
##                                                                                   (0.001)
## ------------------------------------------------------------------------------------------------
##   R-squared            0.924          0.935          0.939          0.951          0.984
##   adj. R-squared       0.924          0.935          0.939          0.951          0.984
##   sigma                0.280          0.259          0.250          0.224          0.129
##   F               652012.063     387489.366     138654.523     87959.467     173791.084
##   p                    0.000          0.000          0.000          0.000          0.000
##   Log-likelihood   -7962.499      -3631.319      -1837.416      4235.240      34091.272
##   Deviance          4242.831       3613.360       3380.837      2699.212        892.214
##   AIC              15930.999       7270.637       3690.832     -8442.481     -68140.544
##   BIC              15957.685       7306.220       3761.997     -8317.942     -67953.736
##   N                53940          53940          53940          53940          53940
## ================================================================================================
```

#This was taken from https://rpubs.com/anthonycerna/diamondspredictions This is a really interesting way to bre ak down a model - but I think I may modify it. I will use random selection to see what it does. Wow, the R-squa red increases as variables are introduced into the model. This is an excellent way to see how each variable con tributes and builds upon each other to get to a really solid model. I like this approach!

```
summary(lm5)
```

```
##
## Call:
## lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut + color +
##     clarity, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81377 -0.08307 -0.00080  0.07976  1.93542
##
## Coefficients:
##                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)     0.414792   0.009855   42.090  < 2e-16 ***
## I(carat^(1/3))  9.144314   0.016156  565.988  < 2e-16 ***
## carat          -1.092551   0.005965 -183.164  < 2e-16 ***
## cut.L           0.119825   0.002264   52.926  < 2e-16 ***
## cut.Q          -0.031025   0.001992  -15.577  < 2e-16 ***
## cut.C           0.013578   0.001730    7.849 4.28e-15 ***
## cut^4          -0.001884   0.001385   -1.360   0.1739
## color.L        -0.440905   0.001973 -223.494  < 2e-16 ***
## color.Q        -0.092790   0.001796  -51.658  < 2e-16 ***
## color.C        -0.013299   0.001676   -7.936 2.13e-15 ***
## color^4         0.012047   0.001540    7.824 5.20e-15 ***
## color^5        -0.003204   0.001454   -2.203   0.0276 *
## color^6         0.001330   0.001322    1.006   0.3142
## clarity.L       0.907144   0.003438  263.861  < 2e-16 ***
## clarity.Q      -0.239602   0.003214  -74.552  < 2e-16 ***
## clarity.C       0.130897   0.002749   47.624  < 2e-16 ***
## clarity^4      -0.062759   0.002195  -28.593  < 2e-16 ***
## clarity^5       0.025752   0.001792   14.371  < 2e-16 ***
## clarity^6      -0.002090   0.001561   -1.339   0.1806
## clarity^7       0.031982   0.001378   23.213  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1286 on 53920 degrees of freedom
## Multiple R-squared:  0.9839, Adjusted R-squared:  0.9839
## F-statistic: 1.738e+05 on 19 and 53920 DF,  p-value: < 2.2e-16
```

```
#First let's split our data in train and test, using say 80% train, 20% test and target/predictor variables
n_obs = dim(diamonds)[1]
n_obs
```

```
## [1] 53940
```

```
prop_split = 0.80
train_index = sample(1:n_obs, round(n_obs * prop_split))
predictors <- diamonds[c(1:4)] #specifies the predictor variables within the diamonds dataset variables 1-4 whi
ch correspond to carat, cut, color and clarity
head(predictors) #Validates the variable selection by showing only those variables selected - thank god it work
ed!
```

```
## # A tibble: 6 x 4
##   carat cut       color clarity
##   <dbl> <ord>     <ord> <ord>
## 1 0.23  Ideal     E     SI2
## 2 0.21  Premium   E     SI1
## 3 0.23  Good      E     VS1
## 4 0.290 Premium   I     VS2
## 5 0.31  Good      J     SI2
## 6 0.24  Very Good J     VVS2
```

```
target <- diamonds$price #Selecting price as the output / target of the model
head(target)
```

```
## [1] 326 326 327 334 335 336
```

```
#Lasso also Regression Requires a Matrix vice a Dataframe...
predictors <- model.matrix(price~., predictors)
str(predictors)
```

```
##  num [1:53940, 1:19] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:53940] "1" "2" "3" "4" ...
##   ..$ : chr [1:19] "(Intercept)" "carat" "cut.L" "cut.Q" ...
##  - attr(*, "assign")= int [1:19] 0 1 2 2 2 2 3 3 3 3 ...
##  - attr(*, "contrasts")=List of 3
##   ..$ cut    : chr "contr.poly"
##   ..$ color  : chr "contr.poly"
##   ..$ clarity: chr "contr.poly"
```

```
head(predictors)
```

```
##   (Intercept) carat      cut.L      cut.Q         cut.C      cut^4
## 1           1  0.23  0.6324555  0.5345225  3.162278e-01  0.1195229
## 2           1  0.21  0.3162278 -0.2672612 -6.324555e-01 -0.4780914
## 3           1  0.23 -0.3162278 -0.2672612  6.324555e-01 -0.4780914
## 4           1  0.29  0.3162278 -0.2672612 -6.324555e-01 -0.4780914
## 5           1  0.31 -0.3162278 -0.2672612  6.324555e-01 -0.4780914
## 6           1  0.24  0.0000000 -0.5345225 -4.095972e-16  0.7171372
##      color.L      color.Q    color.C    color^4    color^5     color^6
## 1 -0.3779645 9.690821e-17  0.4082483 -0.5640761  0.4364358 -0.19738551
## 2 -0.3779645 9.690821e-17  0.4082483 -0.5640761  0.4364358 -0.19738551
## 3 -0.3779645 9.690821e-17  0.4082483 -0.5640761  0.4364358 -0.19738551
## 4  0.3779645 0.000000e+00 -0.4082483 -0.5640761 -0.4364358 -0.19738551
## 5  0.5669467 5.455447e-01  0.4082483  0.2417469  0.1091089  0.03289758
## 6  0.5669467 5.455447e-01  0.4082483  0.2417469  0.1091089  0.03289758
##     clarity.L   clarity.Q  clarity.C  clarity^4  clarity^5  clarity^6
## 1 -0.38575837  0.07715167  0.3077287 -0.5237849  0.4921546 -0.3077287
## 2 -0.23145502 -0.23145502  0.4308202 -0.1208734 -0.3637664  0.5539117
## 3  0.07715167 -0.38575837 -0.1846372  0.3626203  0.3209704 -0.3077287
## 4 -0.07715167 -0.38575837  0.1846372  0.3626203 -0.3209704 -0.3077287
## 5 -0.38575837  0.07715167  0.3077287 -0.5237849  0.4921546 -0.3077287
## 6  0.23145502 -0.23145502 -0.4308202 -0.1208734  0.3637664  0.5539117
##    clarity^7
## 1  0.1194880
## 2 -0.3584641
## 3 -0.5974401
## 4  0.5974401
## 5  0.1194880
## 6  0.3584641
```

```
pred_tr = predictors[train_index,]
pred_te = predictors[-train_index,]
target_tr = target[train_index]
target_te = target[-train_index]
set.seed(42) #Doing the data science cool thing per class and setting seed to #42.

#
lasso.diamonds <- glmnet(pred_te,target_te, alpha = 0, nlambda = 100, lambda.min.ratio = .0001)

#here we are going to train our lambda then embed it into our lasso model
cv.diamonds.lasso <- cv.glmnet(pred_tr, target_tr, family="gaussian", alpha=1, nlambda=100, lambda.min.ratio=.0
001)

coef(cv.diamonds.lasso, s=cv.diamonds.lasso$lambda.1se)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -3506.234724
## (Intercept)     .
## carat        8794.984202
## cut.L         531.290024
## cut.Q        -162.066654
## cut.C          51.258002
## cut^4           8.439808
## color.L     -1778.293367
## color.Q      -533.629880
## color.C       -87.670590
## color^4         3.809124
## color^5       -30.768306
## color^6       -15.131767
## clarity.L    3842.728496
## clarity.Q   -1498.143586
## clarity.C     632.666025
## clarity^4    -201.032731
## clarity^5      69.942425
## clarity^6       .
## clarity^7      52.966396
```

```
#if we embed it back into another lasso we get the same result
cv.diamonds.lasso.1 <- glmnet(pred_tr, target_tr, alpha=1,lambda = cv.diamonds.lasso$lambda.1se)
#same coefs

coef(cv.diamonds.lasso.1)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -3506.882561
## (Intercept)     .
## carat        8795.112152
## cut.L         532.492389
## cut.Q        -163.048844
## cut.C          51.767100
## cut^4           8.247662
## color.L     -1778.105294
## color.Q      -533.469326
## color.C       -87.526295
## color^4         3.886202
## color^5       -30.735583
## color^6       -15.093807
## clarity.L    3844.328906
## clarity.Q   -1499.505638
## clarity.C     633.644896
## clarity^4    -201.643137
## clarity^5      70.348944
## clarity^6       .
## clarity^7      53.034988
```

```
plot(cv.diamonds.lasso, xvar = 'lambda') #Shows how the lamba is targeted to approach 1. We are decreasing the error within the model.
```

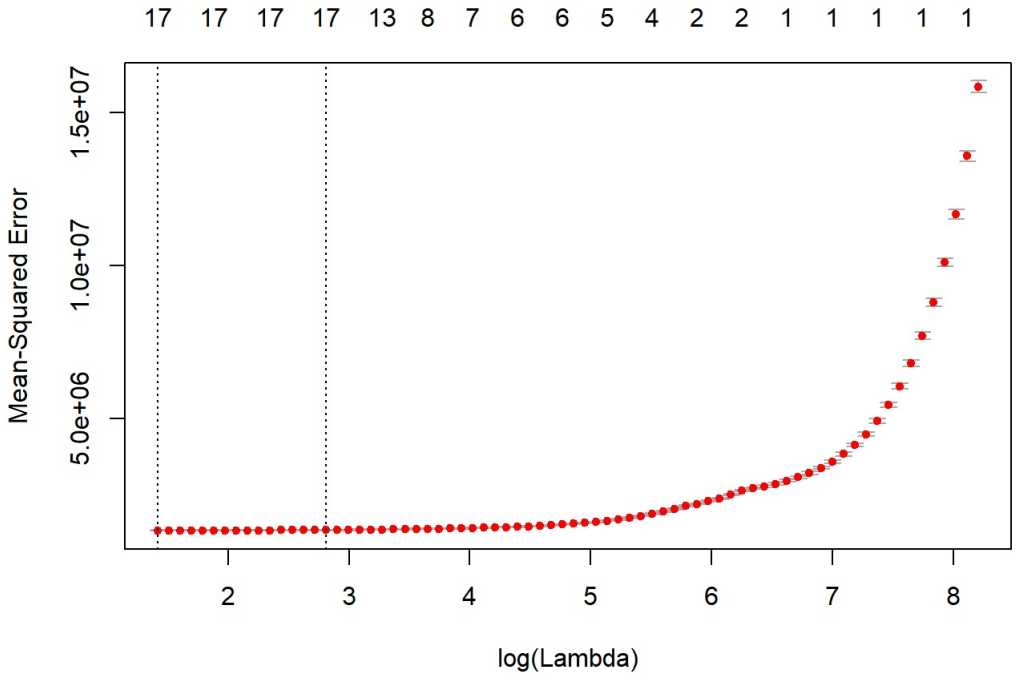
```
## Warning in plot.window(...): "xvar" is not a graphical parameter
```

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

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

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

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

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



Ok now take the diamonds dataset

and run lasso, but also adjust the hyper-parameters including the k-fold number and lambda ratio (alpha) and compare results.

Now let's add a Elastic Net example, a hyper-parameter that locates the optimal middle point between lasso and ridge will also need to be trained

Elastic net does a better job with highly correlated variables

```
cv.diamonds.elnet <- cv.glmnet(pred_tr, target_tr, family="gaussian", alpha=.05, nlambda=100, lambda.min.ratio=
.0001)

coef(cv.diamonds.elnet, s=cv.diamonds.elnet$lambda.1se)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) -3347.60595
## (Intercept)     .
## carat        8554.97783
## cut.L         603.73696
## cut.Q        -249.67973
## cut.C          98.00590
## cut^4          10.29049
## color.L     -1676.68426
## color.Q      -547.68670
## color.C      -125.51836
## color^4        18.96693
## color^5       -69.33364
## color^6       -42.43914
## clarity.L    3668.02895
## clarity.Q   -1526.04659
## clarity.C     660.26819
## clarity^4    -237.76398
## clarity^5     126.08311
## clarity^6       .
## clarity^7      89.49316
```

```
y_hat_lasso <- predict(cv.diamonds.lasso, pred_te)
RMSE_Lasso <- sqrt(mean((target_te-y_hat_lasso)^2))
RMSE_Lasso
```

```
## [1] 1175.564
```

```
#Still looks like Lasso performs a bit better, maybe because our variables aren't all that correlated
y_hat_elnet <- predict(cv.diamonds.elnet, pred_te)
RMSE_elnet<- sqrt(mean((target_te-y_hat_elnet)^2))
RMSE_elnet
```

```
## [1] 1180.141
```

```
#I can't go beyond this point to complete the prediction. I have denoted any code with ### i.e. these are the a
ctual code lines that I would have used.

# Display regression coefficients
###coef(lasso.diamonds)
# Make predictions on the test data
###x.test <- data.frame(pred_te ~., predictors)[,1]#I don't know what I did wrong and I can't move forward to m
ake predictions. I went wrong somewhere.
###probabilities <- model %>% predict(newx = x.test)
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)
#fine the optimal value of lambda
###set.seed(123)
###cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
###plot(cv.lasso)

# Final model with lambda.min
###lasso.model <- glmnet(x, y, alpha = 1, family = "binomial",
 ###                      lambda = cv.lasso$lambda.min)
# Make prediction on test data
###x.test <- model.matrix(price ~., test.data)[,-1]
###probabilities <- lasso.model %>% predict(newx = x.test)
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)

# Final model with lambda.1se
###lasso.model <- glmnet(x, y, alpha = 1, family = "binomial",
###                      lambda = cv.lasso$lambda.1se)
# Make prediction on test data
###x.test <- model.matrix(price ~., test.data)[,-1]
###probabilities <- lasso.model %>% predict(newx = x.test)
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy rate
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)

# Fit the model
###full.model <- glm(price ~., data = train.data, family = binomial)
# Make predictions
###probabilities <- full.model %>% predict(test.data, type = "response")
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)
```

```
#Let's do some predictions with our models to see how we did....

#Our Example Diamond from BlueNile:

#1-- Round, Carat: 1.10, Cut: Astor by Blue Nile Ideal, Color: F, Clarity: VVS2, Price: $11,212     Taken from:
 https://www.bluenile.com/build-your-own-ring/diamond-details/LD09800963?refTab=DIAMONDS&track=viewDiamondDetai
ls&action=newTab

#2-- Round, Carat: 0.90, Cut: Very Good, Color: F, Clarity: VS2, Price: $4,188      Taken from: https://www.blu
enile.com/build-your-own-ring/diamond-details/LD10490866?refTab=DIAMONDS&track=viewDiamondDetails&action=newTab


#Linear Model First......

#1--Blue Nile Astor Diamond
BlueNileDiamondLM = data.frame(carat = 1.10, cut = "Ideal",
                         color = "F", clarity="VVS2")
# data.Frame creates a data frame, we created a dataframe with one value, BlueNileDiamondLM
modelEstimate = predict(lm5, newdata = BlueNileDiamondLM,
                        interval="prediction", level = .95)
exp(modelEstimate) # this will give us the actual price because our model outputs log 10.
```

```
##        fit     lwr      upr
## 1 8635.849 6711.02 11112.75
```

```
#The output is based on a 95% chance that the diamond will fall within the price range. It's off by a little bi
t, but in all fairness, I did pick a specific branding diamond, so maybe the additional $100 is due to branding
. Let's try a second example without the Blue Nile Astor branding...

#2--Blue Nile Diamond

BlueNileDiamondLM1 = data.frame(carat = 0.90, cut = "Very Good",
                         color = "F", clarity="VS2")
# data.Frame creates a data frame, we created a dataframe with one value, BlueNileDiamondLM1
modelEstimate1 = predict(lm5, newdata = BlueNileDiamondLM1,
                        interval="prediction", level = .95)
exp(modelEstimate1) # this will give us the actual price because our model outputs log 10.
```

```
##        fit      lwr      upr
## 1 4548.009 3534.359 5852.372
```

```
#The output is based on a 95% chance that the diamond will fall within the price range. It's within the range b
etween the fit and the lwr prediction. So it's without a doubt within the bounds of the model. It's a good pred
iction and fit, so the Astor Diamonds may be a bit more pricey due to the branding, but does not seem like that
 much over priced. Would have to perform some analysis of this to be sure, but it certaintly looks to be the ca
se. I can't wait to see what the other models do....
```

```
#I can't get this to work. I have errors in my code and I don't know how to correct it. I have denoted all code
 lines that I would have used as ### i.e. actual code lines.

#Lasso Logistic Regression Model prediction...

###table(diamonds$price)
###quantile(as.numeric(diamonds$price))
###boxplot(as.numeric(diamonds$price)) #This box plot shows the price distribution based on quantile. 50% (lowe
r bound) of the data falls from $2,401 and below to $326. 50% (upper bound) falls above $2,401 up to $18,823. T
hose on the lower bound will be to a 0 and those on the upper bound will be set to a 1.
###data("diamonds") # Lines 550-580 taken from http://www.sthda.com/english/articles/36-classification-methods-
essentials/149-penalized-logistic-regression-essentials-in-r-ridge-lasso-and-elastic-net/.
###diamonds <- na.omit(diamonds)
###sample_n(diamonds, 10) #Inspecting the data
###set.seed(42)
###training.samples <- diamonds$price %>%
###  createDataPartition(p = 0.8, list = FALSE)
###train.data <- diamonds[training.samples, ]
###test.data <- diamonds[-training.samples, ]
#Creates moel matrix of predictors and converts categorical predictors to the appropriate dummy variables requi
red for glmnet function use
###x <- model.matrix(price~carat + cut + color +clarity, train.data)[,-1]#Dummy code categorical predictor vari
ables
###y <- ifelse(train.data$price == "pos", 0, 1)#Convert the outcome to a numberical variable

#Create the penalized logistic regression model
###model = glmnet(x, y, family = "binomial", alpha = 1, lambda = NULL) # I get an error here that I can't fix
#I can't move beyond this point due to the above model error and I don't know what I did incorrectly so I canno
t fix it.

#Sample code that would hopefully get me to the final model prediction, though I don't know because I can't run
 it to completion.
 #Find the best lambda using cross-validation
###set.seed(123)
###cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
# Fit the final model on the training data
###model <- glmnet(x, y, alpha = 1, family = "binomial",
###                lambda = cv.lasso$lambda.min)
# Display regression coefficients
###coef(model)
# Make predictions on the test data
###x.test <- model.matrix(price ~., test.data)[,-1]
###probabilities <- model %>% predict(newx = x.test)
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)
```

```
#fine the optimal value of lambda
###set.seed(123)
###cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
###plot(cv.lasso)

# Final model with lambda.min
###lasso.model <- glmnet(x, y, alpha = 1, family = "binomial",
###                      lambda = cv.lasso$lambda.min)
# Make prediction on test data
###x.test <- model.matrix(price ~., test.data)[,-1]
###probabilities <- lasso.model %>% predict(newx = x.test)
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)

# Final model with lambda.1se
###lasso.model <- glmnet(x, y, alpha = 1, family = "binomial",
###                      lambda = cv.lasso$lambda.1se)
# Make prediction on test data
###x.test <- model.matrix(price ~., test.data)[,-1]
###probabilities <- lasso.model %>% predict(newx = x.test)
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy rate
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)

# Fit the model
###full.model <- glm(price ~., data = train.data, family = binomial)
# Make predictions
###probabilities <- full.model %>% predict(test.data, type = "response")
###predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
###observed.classes <- test.data$price
###mean(predicted.classes == observed.classes)
```

```
#Despite not being able to complete this project with the prediction models, I still learned a lot. I wish I wa
s able to get this to completion, but unfortunately I cannot figure this out in r. Hopefully someone can help i
n finalizing this.
```