McKnight Midterm PHP2560

Olivia McKnight

10/19/2020

###Introduction

Diamonds are one of the most sought after gemstones in the world, and they have a variety of different attributes by which they are described. These include carat, or the weight of the diamond, cut, or the quality of the cut of the diamond, color (color of the diamond), clarity (clarity of the diamond), depth percentage, which is a computation that takes into account the length, width, and depth of the diamond, and table (which is the width of the top of the diamond relative to the widest point). These all affect how a diamond is appraised and how much it will cost in USD. For my midterm, I will be analyzing an open source dataset, Diamonds Data, that consists of 53940 observations of the 11 variables listed below. Through statistical programming, I hope to reveal which variable is most closely correlated with price. I will begin this process by conducting an exploratory data analysis. Then, I will conduct a regression analysis to model the relationship between significant variables and price. Then, based on the model I develop, I will determine if it is possible to write a function that will return a diamond's price based on its other characteristics, using sample diamonds from the dataset. Then, I will generate randomly simulated diamonds and evaulate if my function is still effective. #ABOUT THE DATASET

Dataset: Diamonds Data 53940 observations of 11 variables

Variables:

diamondnum number assigned to each diamond in the datset (1 – 53940)

price price in US dollars ($326–$18,823)

carat weight of the diamond (0.2–5.01)

cut quality of the cut (Fair, Good, Very Good, Premium, Ideal)

color diamond colour, from J (worst) to D (best)

clarity a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

depth total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43–79)

table width of top of diamond relative to widest point (43–95)

price price in US dollars ($326–$18,823)

x length in mm (0–10.74)

y width in mm (0–58.9)

z depth in mm (0–31.8)

Qualitative Features (Categorical) : *Cut, Color, Clarity*.

Quantitative Features (Numerical) : *Carat, Depth , Table , Price , X , Y, Z*.

#Guiding questions *Which of the variables above is most closely correlated to price?*

Is there a way to model how each of the other variables relates to price?

Based on the above findings, is there a way to write a function that will return a price based on diamond characteristics?

Can prices be generated for randomly generated diamonds?

###Exploratory Data Analysis

```
To begin the exploratory data analysis,  all necessary packages must be installed and pl
aced into the library. Then, one must get an idea of the structure of the original datas
et by viewing the structure of "diamonds_data". The "diamonds_data" is not in a format t
hat is easy to understand, so it must be cleaned and modified. The categorical variables
cut, color, and clarity are all characters. It will be difficult to conduct analyses wit
h them in this format so using the mutate() functions they are turned into factors. The
diamondnum variable is also dropped as it is simply an naming variable for the different
diamonds and will not be helpful in the analysis. Next, several columns should be rename
d as their current names are confusing. X is renamed to length, Y is renamed to width, Z
is renamed to depth, and depth is renamed to depth_perc. The new dataset, called diamond
s, is much easier to interpret.
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✔ ggplot2 3.3.2     ✔ purrr   0.3.4
## ✔ tibble  2.1.3     ✔ dplyr   0.8.3
## ✔ tidyr   1.0.0     ✔ stringr 1.4.0
## ✔ readr   1.3.1     ✔ forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
##
## Attaching package: 'docstring'
```

```
## The following object is masked from 'package:utils':
##
##     ?
```

```
## Warning: package 'scales' was built under R version 3.6.2
```

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

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

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

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

```
## Warning: package 'survival' was built under R version 3.6.2
```

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

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

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

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

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

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

```
## Warning: package 'reshape2' was built under R version 3.6.2
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
## corrplot 0.84 loaded
```

```
## Warning: package 'plotly' was built under R version 3.6.2
```

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

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

```
## Warning: package 'lmtest' was built under R version 3.6.2
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.6.2
```

```
##
## Attaching package: 'zoo'
```

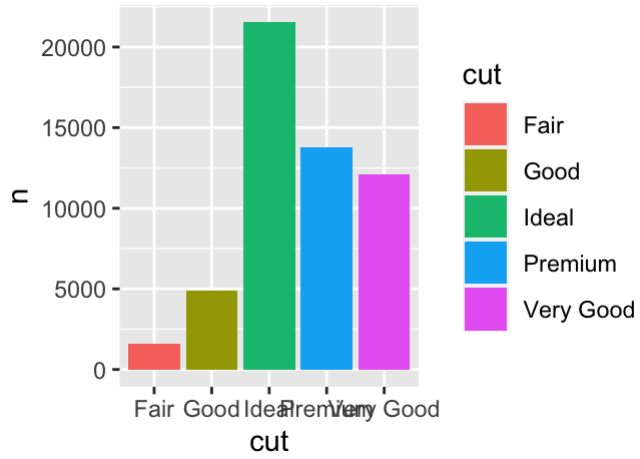
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

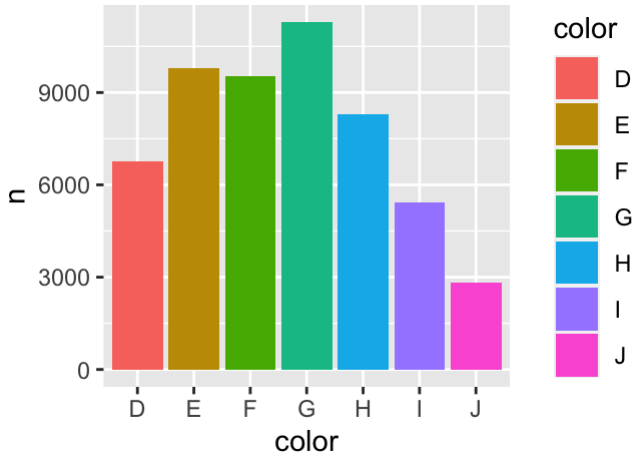
```
#drop unnecessary columns (diamondnum) and rename for clarity
diamonds <- diamonds_data %>%
    arrange(cut, color, clarity) %>%
    mutate(cut = as.factor(cut),
           color = as.factor(color),
           clarity = as.factor(clarity)) %>%
    select(-diamondnum) %>%
    rename(depth_perc = "depth", length = "x", width = "y", depth = "z")
head(diamonds)
```

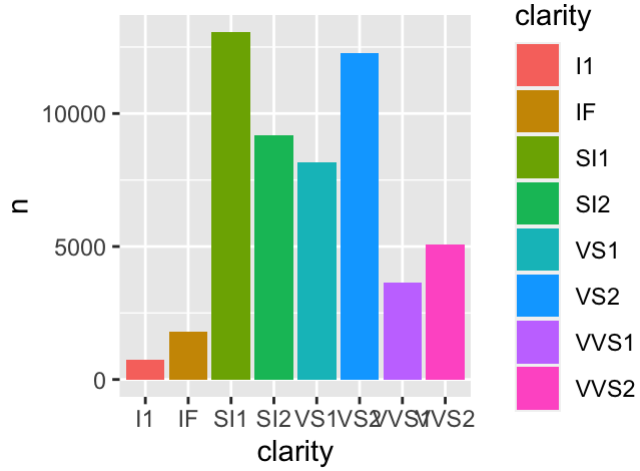
```
##   carat  cut color clarity depth_perc table price length width depth
## 1  1.50 Fair     D      I1       64.7    62  5460   7.19  7.04  4.60
## 2  1.70 Fair     D      I1       64.7    56  5617   7.46  7.37  4.80
## 3  3.40 Fair     D      I1       66.8    52 15964   9.42  9.34  6.27
## 4  0.91 Fair     D      I1       66.2    57  2491   6.00  5.94  3.95
## 5  0.30 Fair     D      IF       60.5    57  1208   4.47  4.35  2.67
## 6  0.37 Fair     D      IF       61.2    57  1440   4.68  4.73  2.88
```

This dataset is comprised of three categorical variables: cut, color, and clarity. In order to get acquainted with the data, one must determine how many categories are in each variable, and how many diamonds are in each of those categories. Those counts were discerned using tidyverse functions, and made easy to understand using ggplot2 functions to make barcharts. Then, indicator variables were created for these variables, and those were plotted as histograms along with the continuous variables.

```
#before exploring visually, get counts of our categorical variables
##cut
cut_count <- diamonds %>%
  group_by(cut) %>%
  count()
cut_plot <- ggplot(cut_count, aes(x = cut, y = n, fill=cut)) + geom_col()

##color
color_count <- diamonds %>%
  group_by(color) %>%
  count()
color_plot <- ggplot(color_count, aes(x = color, y = n, fill=color)) + geom_col()

##clarity
clarity_count <- diamonds %>%
  group_by(clarity) %>%
  count()
clarity_plot <- ggplot(clarity_count, aes(x = clarity, y = n, fill=clarity)) + geom_col
()

#show plots together
grid.arrange(cut_plot, color_plot, clarity_plot, ncol=2)
```








```
#create indicator variables
#cat varaibles <- cut, color, clarity
diamonds$cut <- factor(diamonds$cut, levels=c("Fair", "Good", "Very Good", "Premium", "I
deal"))
diamonds$color <- factor(diamonds$color, levels=c("J", "I", "H", "G", "F", "E", "D"))
diamonds$clarity <- factor(diamonds$clarity, levels=c("I1", "SI2", "SI1", "VS2", "VS1",
"VVS2", "VVS1", "IF"))
str(diamonds)
```

```
## 'data.frame':    53940 obs. of  10 variables:
##  $ carat     : num  1.5 1.7 3.4 0.91 0.3 0.37 0.47 0.9 0.9 0.9 ...
##  $ cut       : Factor w/ 5 levels "Fair","Good",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ color     : Factor w/ 7 levels "J","I","H","G",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ clarity   : Factor w/ 8 levels "I1","SI2","SI1",..: 1 1 1 1 8 8 8 3 3 3 ...
##  $ depth_perc: num  64.7 64.7 66.8 66.2 60.5 61.2 60.6 66.4 64.8 64.5 ...
##  $ table     : num  62 56 52 57 57 57 60 59 59 61 ...
##  $ price     : int  5460 5617 15964 2491 1208 1440 2211 3382 3689 3689 ...
##  $ length    : num  7.19 7.46 9.42 6 4.47 4.68 5.09 5.97 6.1 6.05 ...
##  $ width     : num  7.04 7.37 9.34 5.94 4.35 4.73 4.98 5.92 6.03 6.01 ...
##  $ depth     : num  4.6 4.8 6.27 3.95 2.67 2.88 3.05 3.95 3.93 3.89 ...
```

```
diamonds$cut2 <- as.numeric(diamonds$cut)
diamonds$color2 <- as.numeric(diamonds$color)
diamonds$clarity2 <- as.numeric(diamonds$clarity)
```

A comprehensive but relatively simple exploratory data analysis was completed using the funModeling and Hmisc packages. Cumulative percentage values were generated for the cut, color, and clarity using this method. The cut with the highest cumulative percentage was Ideal, the color with the highest cumulative percentage was G, and the clarity with the highest cumulative percentage was SI1. The continuous variables (carat, depth_perc, table, price, length, width and depth) along with dummy variables created for cut, color, and clarity, were all plotted as histograms. From those plots, one can see that the majority of diamonds are under one carat. The majority of diamonds have a depth percentage around 62%. The most common table for the diamodns in this dataset is 58. Nearly all of the diamonds have a price that is lower than 5,000 USD. The lengths of diamonds is more varied than the other variables, with a range of about 3.3mm to 9mm. All of the diamonds have a width of under 10mm, and nearly all of the diamonds have a depth of under 5mm. The histograms for cut, color, and clarity align with the visualization in the bar charts in this analysis.

```
## funModeling v.1.9.3 :)
## Examples and tutorials at livebook.datascienceheroes.com
##  / Now in Spanish: librovivodecienciadedatos.ai
```

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

```
## 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:plotly':
##
##     arrange, mutate, rename, summarise
```

```
## 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
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
  summary(diamonds)
```

```
##      carat               cut        color        clarity
##  Min.   :0.2000   Fair      : 1610   J: 2808   SI1    :13065
##  1st Qu.:0.4000   Good      : 4906   I: 5422   VS2    :12258
##  Median :0.7000   Very Good:12082   H: 8304   SI2    : 9194
##  Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171
##  3rd Qu.:1.0400   Ideal     :21551   F: 9542   VVS2   : 5066
##  Max.   :5.0100                     E: 9797   VVS1   : 3655
##                                     D: 6775   (Other): 2531
##    depth_perc        table           price           length
##  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
##
##      width            depth             cut2           color2
##  Min.   : 0.000   Min.   : 0.000   Min.   :1.000   Min.   :1.000
##  1st Qu.: 4.720   1st Qu.: 2.910   1st Qu.:3.000   1st Qu.:3.000
##  Median : 5.710   Median : 3.530   Median :4.000   Median :4.000
##  Mean   : 5.735   Mean   : 3.539   Mean   :3.904   Mean   :4.406
##  3rd Qu.: 6.540   3rd Qu.: 4.040   3rd Qu.:5.000   3rd Qu.:6.000
##  Max.   :58.900   Max.   :31.800   Max.   :5.000   Max.   :7.000
##
##     clarity2
##  Min.   :1.000
##  1st Qu.:3.000
##  Median :4.000
##  Mean   :4.051
##  3rd Qu.:5.000
##  Max.   :8.000
##
```

```
df_status(diamonds)
```

```
##      variable q_zeros p_zeros q_na p_na q_inf p_inf    type unique
## 1       carat       0    0.00    0    0     0     0 numeric    273
## 2         cut       0    0.00    0    0     0     0  factor      5
## 3       color       0    0.00    0    0     0     0  factor      7
## 4     clarity       0    0.00    0    0     0     0  factor      8
## 5  depth_perc       0    0.00    0    0     0     0 numeric    184
## 6       table       0    0.00    0    0     0     0 numeric    127
## 7       price       0    0.00    0    0     0     0 integer  11602
## 8      length       8    0.01    0    0     0     0 numeric    554
## 9       width       7    0.01    0    0     0     0 numeric    552
## 10      depth      20    0.04    0    0     0     0 numeric    375
## 11       cut2       0    0.00    0    0     0     0 numeric      5
## 12     color2       0    0.00    0    0     0     0 numeric      7
## 13   clarity2       0    0.00    0    0     0     0 numeric      8
```

```
freq(diamonds)
```

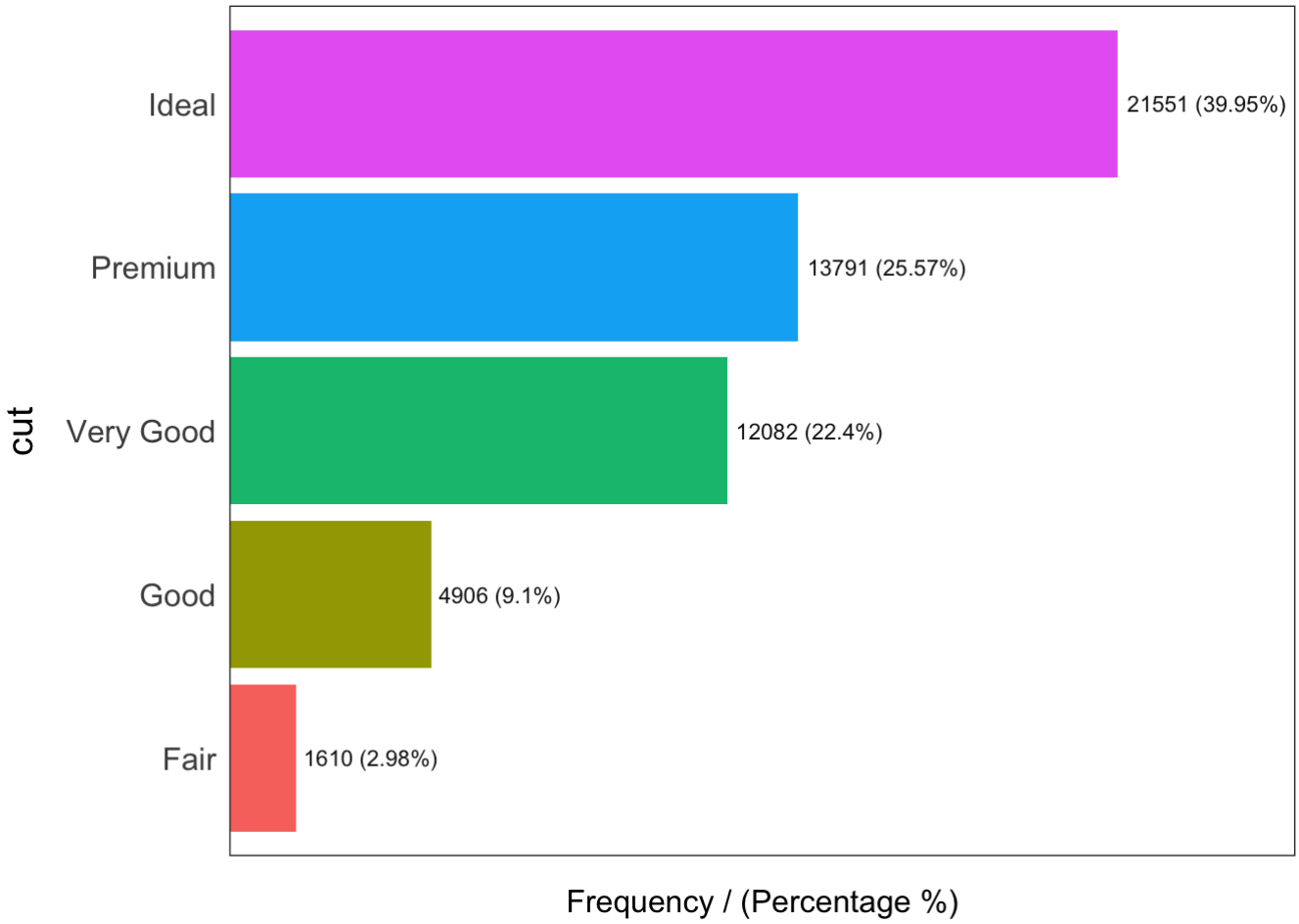
```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```

```
## Warning: Use of `tbl_plot$frequency` is discouraged. Use `frequency`
## instead.
```

```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.

## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```

```
## Warning: Use of `tbl_plot$frequency` is discouraged. Use `frequency`
## instead.
```

```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```




```
##         cut frequency percentage cumulative_perc
## 1     Ideal     21551      39.95           39.95
## 2   Premium     13791      25.57           65.52
## 3 Very Good     12082      22.40           87.92
## 4      Good      4906       9.10           97.02
## 5      Fair      1610       2.98          100.00
```

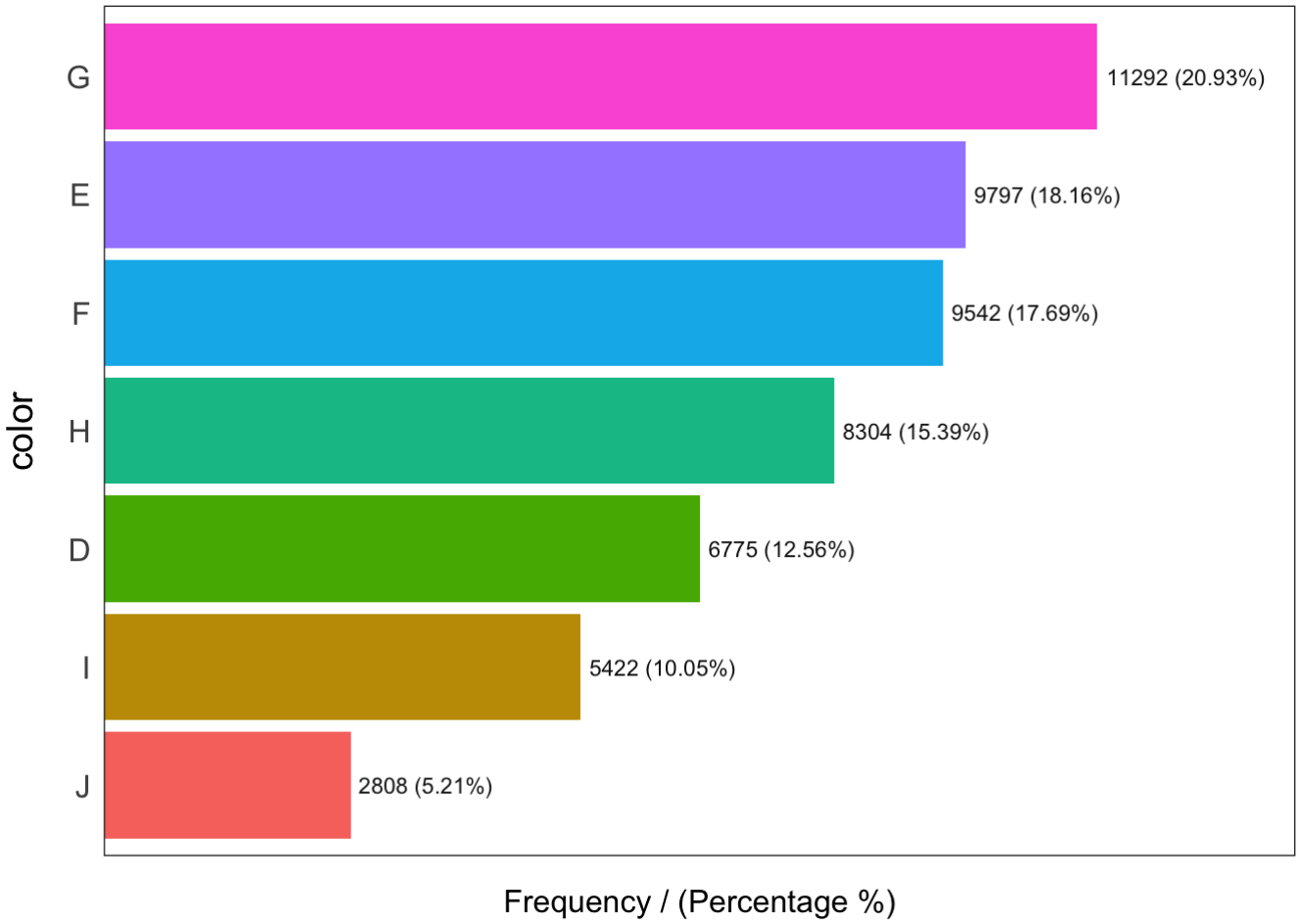
```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```

```
## Warning: Use of `tbl_plot$frequency` is discouraged. Use `frequency`
## instead.
```

```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.

## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```

```
## Warning: Use of `tbl_plot$frequency` is discouraged. Use `frequency`
## instead.
```

```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```




```
##   color frequency percentage cumulative_perc
## 1     G     11292      20.93           20.93
## 2     E      9797      18.16           39.09
## 3     F      9542      17.69           56.78
## 4     H      8304      15.39           72.17
## 5     D      6775      12.56           84.73
## 6     I      5422      10.05           94.78
## 7     J      2808       5.21          100.00
```

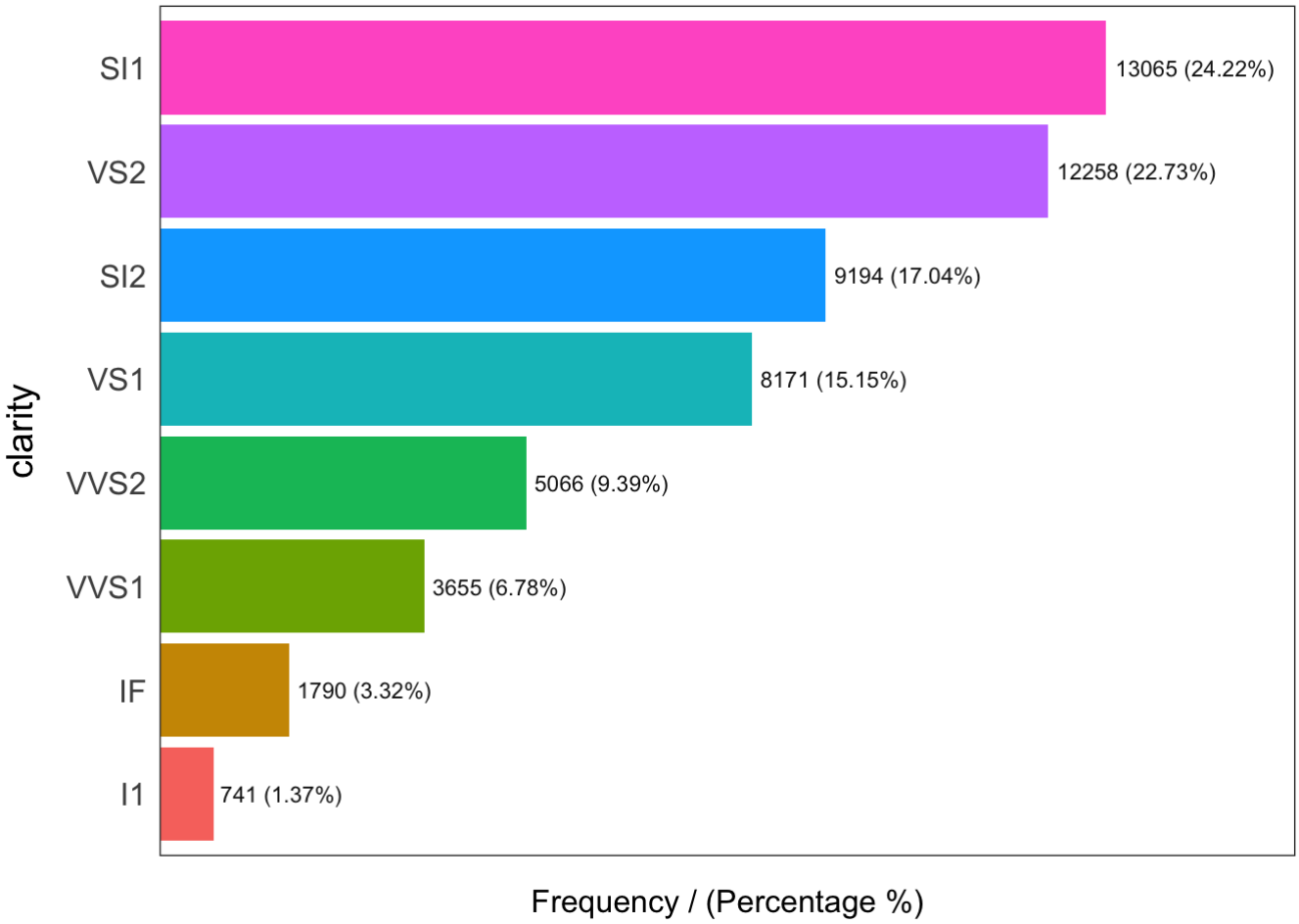
```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```

```
## Warning: Use of `tbl_plot$frequency` is discouraged. Use `frequency`
## instead.
```

```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.

## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```

```
## Warning: Use of `tbl_plot$frequency` is discouraged. Use `frequency`
## instead.
```

```
## Warning: Use of `tbl_plot$category` is discouraged. Use `category` instead.
```



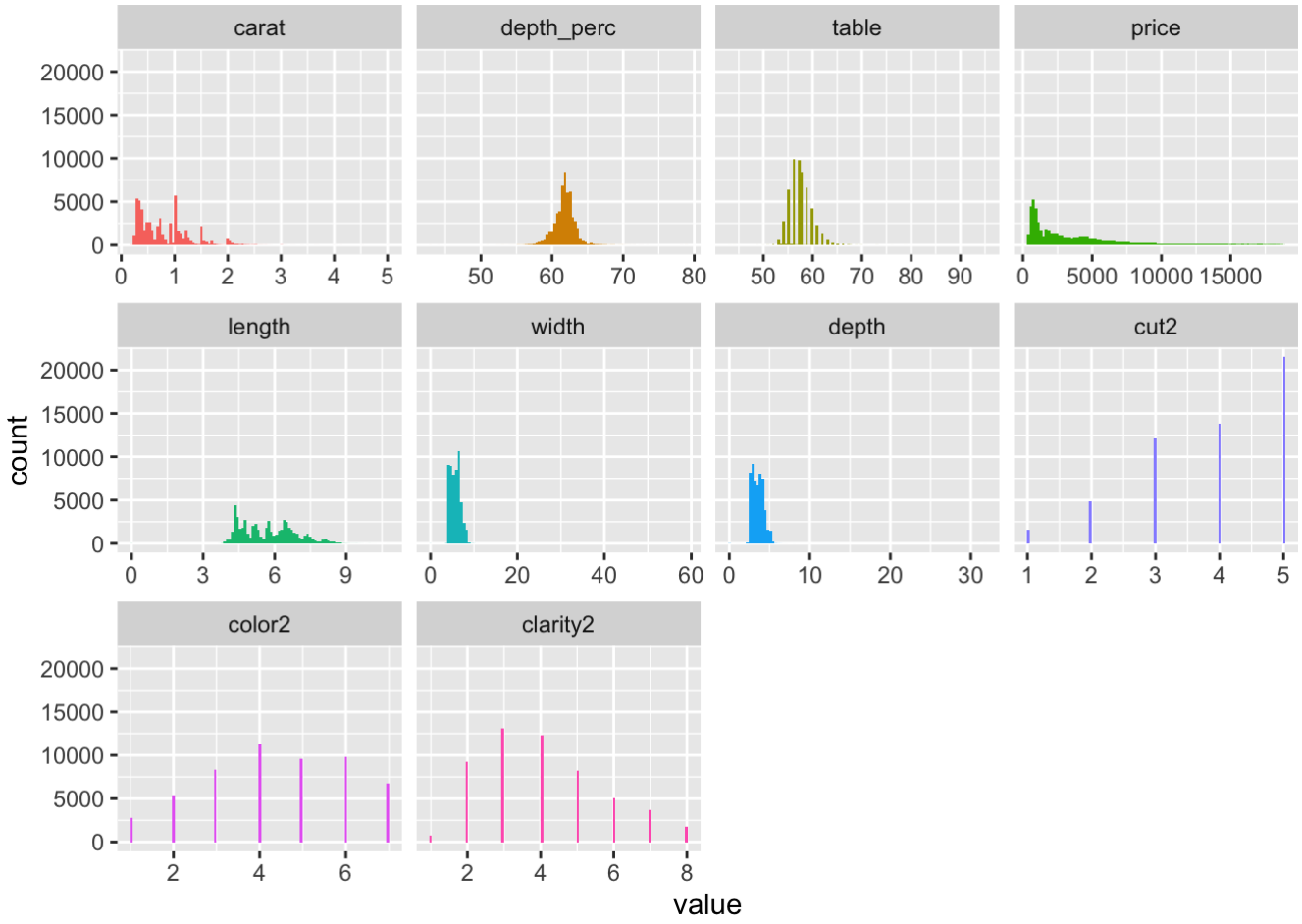
```
##   clarity frequency percentage cumulative_perc
## 1     SI1     13065      24.22           24.22
## 2     VS2     12258      22.73           46.95
## 3     SI2      9194      17.04           63.99
## 4     VS1      8171      15.15           79.14
## 5    VVS2      5066       9.39           88.53
## 6    VVS1      3655       6.78           95.31
## 7      IF      1790       3.32           98.63
## 8      I1       741       1.37          100.00
```

```
## [1] "Variables processed: cut, color, clarity"
```

```
  profiling_num(diamonds)
```

```
##      variable         mean      std_dev variation_coef   p_01   p_05
## 1       carat    0.7979397    0.4740112     0.59404391   0.24   0.30
## 2  depth_perc   61.7494049    1.4326213     0.02320057  57.90  59.30
## 3       table   57.4571839    2.2344906     0.03888966  53.00  54.00
## 4       price 3932.7997219 3989.4397381     1.01440196 429.00 544.00
## 5      length    5.7311572    1.1217607     0.19573023   4.02   4.29
## 6       width    5.7345260    1.1421347     0.19916811   4.04   4.30
## 7       depth    3.5387338    0.7056988     0.19942129   2.48   2.65
## 8        cut2    3.9040971    1.1165999     0.28600720   1.00   2.00
## 9      color2    4.4058027    1.7011048     0.38610552   1.00   1.00
## 10   clarity2    4.0510197    1.6471361     0.40659790   1.00   2.00
##      p_25    p_50    p_75     p_95     p_99    skewness  kurtosis     iqr
## 1    0.40    0.70    1.04     1.70     2.18  1.11661487  4.256408    0.64
## 2   61.00   61.80   62.50    63.80    65.60 -0.08229174  8.738771    1.50
## 3   56.00   57.00   59.00    61.00    64.00  0.79687369  5.801486    3.00
## 4  950.00 2401.00 5324.25 13107.10 17378.22  1.61835028  5.177383 4374.25
## 5    4.71    5.70    6.54     7.66     8.36  0.37866581  2.381785    1.83
## 6    4.72    5.71    6.54     7.65     8.34  2.43409903 94.205991    1.82
## 7    2.91    3.53    4.04     4.73     5.15  1.52238022 50.082143    1.13
## 8    3.00    4.00    5.00     5.00     5.00 -0.71716051  2.601952    2.00
## 9    3.00    4.00    6.00     7.00     7.00 -0.18936064  2.133207    3.00
## 10   3.00    4.00    5.00     7.00     8.00  0.55142221  2.605159    2.00
##           range_98     range_80
## 1     [0.24, 2.18] [0.31, 1.51]
## 2     [57.9, 65.6]   [60, 63.3]
## 3         [53, 64]     [55, 60]
## 4  [429, 17378.22]  [646, 9821]
## 5     [4.02, 8.36] [4.36, 7.31]
## 6     [4.04, 8.34]  [4.36, 7.3]
## 7     [2.48, 5.15] [2.69, 4.52]
## 8           [1, 5]       [2, 5]
## 9           [1, 7]       [2, 7]
## 10          [1, 8]       [2, 7]
```

```
  plot_num(diamonds, bins = 100)
```




```
describe(diamonds)
```

```
## diamonds
##
##  13  Variables      53940  Observations
## --------------------------------------------------------------------------
## carat
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    53940        0      273    0.999   0.7979   0.5122     0.30     0.31
##      .25      .50      .75      .90      .95
##     0.40     0.70     1.04     1.51     1.70
##
## lowest : 0.20 0.21 0.22 0.23 0.24, highest: 4.00 4.01 4.13 4.50 5.01
## --------------------------------------------------------------------------
## cut
##        n  missing distinct
##    53940        0        5
##
## lowest : Fair      Good      Very Good Premium   Ideal
## highest: Fair      Good      Very Good Premium   Ideal
##
## Value           Fair      Good Very Good   Premium     Ideal
## Frequency       1610      4906     12082     13791     21551
## Proportion     0.030     0.091     0.224     0.256     0.400
## --------------------------------------------------------------------------
## color
##        n  missing distinct
##    53940        0        7
##
## lowest : J I H G F, highest: H G F E D
##
## Value          J     I     H     G     F     E     D
## Frequency   2808  5422  8304 11292  9542  9797  6775
## Proportion 0.052 0.101 0.154 0.209 0.177 0.182 0.126
## --------------------------------------------------------------------------
## clarity
##        n  missing distinct
##    53940        0        8
##
## lowest : I1   SI2  SI1  VS2  VS1 , highest: VS2  VS1  VVS2 VVS1 IF
##
## Value         I1   SI2   SI1   VS2   VS1  VVS2  VVS1    IF
## Frequency    741  9194 13065 12258  8171  5066  3655  1790
## Proportion 0.014 0.170 0.242 0.227 0.151 0.094 0.068 0.033
## --------------------------------------------------------------------------
## depth_perc
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    53940        0      184    0.999    61.75    1.515     59.3     60.0
##      .25      .50      .75      .90      .95
##     61.0     61.8     62.5     63.3     63.8
##
## lowest : 43.0 44.0 50.8 51.0 52.2, highest: 72.2 72.9 73.6 78.2 79.0
## --------------------------------------------------------------------------
## table
##        n  missing distinct     Info     Mean      Gmd      .05      .10
```

```
##    53940        0      127     0.98    57.46    2.448       54       55
##      .25      .50      .75      .90      .95
##       56       57       59       60       61
##
## lowest : 43.0 44.0 49.0 50.0 50.1, highest: 71.0 73.0 76.0 79.0 95.0
## ---------------------------------------------------------------------------
## price
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    53940        0    11602        1     3933     4012      544      646
##      .25      .50      .75      .90      .95
##      950     2401     5324     9821    13107
##
## lowest :   326   327   334   335   336, highest: 18803 18804 18806 18818 18823
## ---------------------------------------------------------------------------
## length
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    53940        0      554        1    5.731    1.276     4.29     4.36
##      .25      .50      .75      .90      .95
##     4.71     5.70     6.54     7.31     7.66
##
## lowest :  0.00  3.73  3.74  3.76  3.77, highest: 10.01 10.02 10.14 10.23 10.74
## ---------------------------------------------------------------------------
## width
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    53940        0      552        1    5.735    1.269     4.30     4.36
##      .25      .50      .75      .90      .95
##     4.72     5.71     6.54     7.30     7.65
##
## lowest :  0.00  3.68  3.71  3.72  3.73, highest: 10.10 10.16 10.54 31.80 58.90
##
## Value        0.0   3.5   4.0   4.5   5.0   5.5   6.0   6.5   7.0   7.5
## Frequency      7     5  1731 12305  7817  5994  6742  9260  4298  3402
## Proportion 0.000 0.000 0.032 0.228 0.145 0.111 0.125 0.172 0.080 0.063
##
## Value        8.0   8.5   9.0   9.5  10.0  10.5  32.0  59.0
## Frequency   1635   652    69    14     6     1     1     1
## Proportion 0.030 0.012 0.001 0.000 0.000 0.000 0.000 0.000
##
## For the frequency table, variable is rounded to the nearest 0.5
## ---------------------------------------------------------------------------
## depth
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##    53940        0      375        1    3.539   0.7901     2.65     2.69
##      .25      .50      .75      .90      .95
##     2.91     3.53     4.04     4.52     4.73
##
## lowest :  0.00  1.07  1.41  1.53  2.06, highest:  6.43  6.72  6.98  8.06 31.80
##
## Value        0.0   1.0   1.5   2.0   2.5   3.0   3.5   4.0   4.5   5.0
## Frequency     20     1     2     3  8807 13809  9474 13682  5525  2352
## Proportion 0.000 0.000 0.000 0.000 0.163 0.256 0.176 0.254 0.102 0.044
##
## Value        5.5   6.0   6.5   7.0   8.0  32.0
## Frequency    237    20     5     1     1     1
```

```
## Proportion 0.004 0.000 0.000 0.000 0.000 0.000
##
## For the frequency table, variable is rounded to the nearest 0.5
## ---------------------------------------------------------------------------
## cut2
##        n  missing distinct     Info     Mean      Gmd
##    53940        0        5    0.907    3.904    1.202
##
## lowest : 1 2 3 4 5, highest: 1 2 3 4 5
##
## Value          1     2     3     4     5
## Frequency   1610  4906 12082 13791 21551
## Proportion 0.030 0.091 0.224 0.256 0.400
## ---------------------------------------------------------------------------
## color2
##        n  missing distinct     Info     Mean      Gmd
##    53940        0        7    0.973    4.406    1.927
##
## lowest : 1 2 3 4 5, highest: 3 4 5 6 7
##
## Value          1     2     3     4     5     6     7
## Frequency   2808  5422  8304 11292  9542  9797  6775
## Proportion 0.052 0.101 0.154 0.209 0.177 0.182 0.126
## ---------------------------------------------------------------------------
## clarity2
##        n  missing distinct     Info     Mean      Gmd
##    53940        0        8    0.964    4.051    1.829
##
## lowest : 1 2 3 4 5, highest: 4 5 6 7 8
##
## Value          1     2     3     4     5     6     7     8
## Frequency    741  9194 13065 12258  8171  5066  3655  1790
## Proportion 0.014 0.170 0.242 0.227 0.151 0.094 0.068 0.033
## ---------------------------------------------------------------------------
```

```
  correlation_table(data=diamonds, target="price")
```

```
##      Variable price
## 1       price  1.00
## 2       carat  0.92
## 3      length  0.88
## 4       width  0.87
## 5       depth  0.86
## 6       table  0.13
## 7  depth_perc -0.01
## 8        cut2 -0.05
## 9    clarity2 -0.15
## 10     color2 -0.17
```

In order to determine which variables are most strongly correlated with price, it is necessary to generate a correlation table. This table shows all of the variables and their correlation coefficient when compared to price. Next, correlation matrices must be generated so that a correlation heatmap can be complied for easy reference. The first correlation matrix is comprised of the variables and their correlation coefficients when compared to one another. The next matrix is a correlation matrix comprised of the p values for each of the variable combinations. As one can see in the output. Those matrices are not very easy to interpret. So, the final correlation matrix, "corr_matrix_fixed", was generated using a function of the p-values and correlation coefficients. While fixed matrix is more uniform than the others, a correlation heatmap is necessary to easily visualize the data.
The correlation heatmap was generated using ggcorrplot. Each cell in the visualization has the correlation coefficient of the variables with which it aligns. The cells in red have strong positive correlation, and the cells in blue have strong negative correlation. Those with light pink and light blue have weak correlation. The interactive aspect of this heatmap were added using ggplotly. By examining this visualization and the correlation table, it can be deduced that carat is most strongly correlated with price, with a correlation coefficient of 0.92. This is closely followed by length, width, and depth, which have correllation coefficients of 0.88, 0.87, 0.86 respectively. Chi square tests and t tests were conducted on all the variables in relationship to price, but they proved inconclusive.

```
#examine all correlations in relation to price
cor_table <- correlation_table(data=diamonds, target="price")
cor_table
```

```
##      Variable price
## 1       price  1.00
## 2       carat  0.92
## 3      length  0.88
## 4       width  0.87
## 5       depth  0.86
## 6       table  0.13
## 7  depth_perc -0.01
## 8        cut2 -0.05
## 9    clarity2 -0.15
## 10     color2 -0.17
```

```
#select all continous variables
cor_data <- diamonds %>%
  select(price, carat, length, width, depth, depth_perc, table, cut2, color2, clarity2)

#create a corelation matrix of correlation coefficients
cor_matrix <- cor(cor_data, method = "pearson", use = "complete.obs")
cor_matrix
```

```
##                  price       carat      length       width       depth
## price       1.00000000  0.92159130  0.88443516  0.86542090  0.86124944
## carat       0.92159130  1.00000000  0.97509423  0.95172220  0.95338738
## length      0.88443516  0.97509423  1.00000000  0.97470148  0.97077180
## width       0.86542090  0.95172220  0.97470148  1.00000000  0.95200572
## depth       0.86124944  0.95338738  0.97077180  0.95200572  1.00000000
## depth_perc -0.01064740  0.02822431 -0.02528925 -0.02934067  0.09492388
## table       0.12713390  0.18161755  0.19534428  0.18376015  0.15092869
## cut2       -0.05349066 -0.13496702 -0.12556524 -0.12146187 -0.14932254
## color2     -0.17251093 -0.29143675 -0.27028669 -0.26358440 -0.26822688
## clarity2   -0.14680007 -0.35284057 -0.37199853 -0.35841962 -0.36695200
##             depth_perc      table        cut2      color2    clarity2
## price      -0.01064740  0.1271339 -0.05349066 -0.17251093 -0.14680007
## carat       0.02822431  0.1816175 -0.13496702 -0.29143675 -0.35284057
## length     -0.02528925  0.1953443 -0.12556524 -0.27028669 -0.37199853
## width      -0.02934067  0.1837601 -0.12146187 -0.26358440 -0.35841962
## depth       0.09492388  0.1509287 -0.14932254 -0.26822688 -0.36695200
## depth_perc  1.00000000 -0.2957785 -0.21805501 -0.04727923 -0.06738444
## table      -0.29577852  1.0000000 -0.43340461 -0.02646520 -0.16032684
## cut2       -0.21805501 -0.4334046  1.00000000  0.02051852  0.18917474
## color2     -0.04727923 -0.0264652  0.02051852  1.00000000 -0.02563128
## clarity2   -0.06738444 -0.1603268  0.18917474 -0.02563128  1.00000000
```

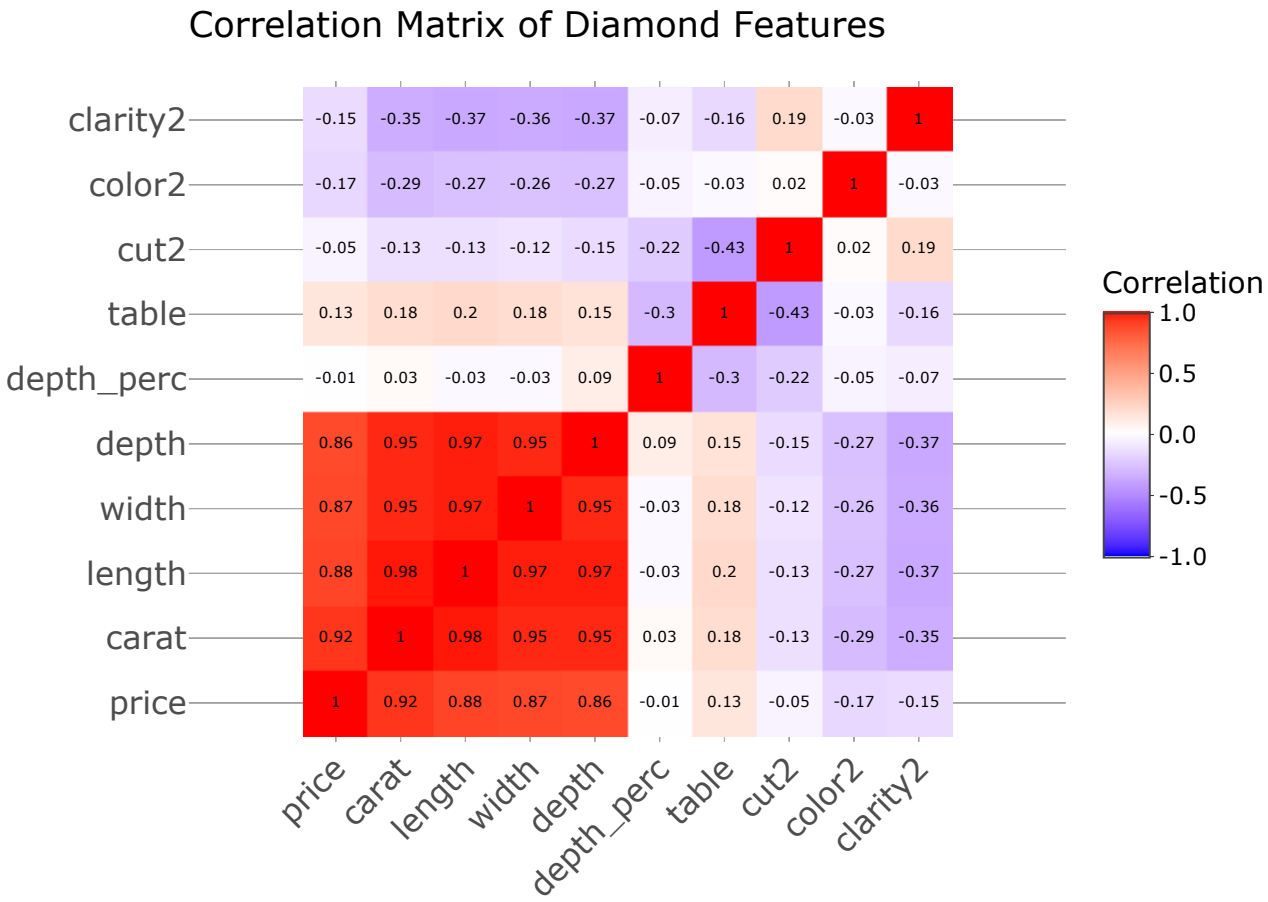
```
#create a correlation matrix of p values using pearson correlation
cor_matrix_2 <- rcorr(as.matrix(cor_data))
cor_matrix_2
```

```
##            price carat length width depth depth_perc table  cut2 color2
## price       1.00  0.92   0.88  0.87  0.86      -0.01  0.13 -0.05  -0.17
## carat       0.92  1.00   0.98  0.95  0.95       0.03  0.18 -0.13  -0.29
## length      0.88  0.98   1.00  0.97  0.97      -0.03  0.20 -0.13  -0.27
## width       0.87  0.95   0.97  1.00  0.95      -0.03  0.18 -0.12  -0.26
## depth       0.86  0.95   0.97  0.95  1.00       0.09  0.15 -0.15  -0.27
## depth_perc -0.01  0.03  -0.03 -0.03  0.09       1.00 -0.30 -0.22  -0.05
## table       0.13  0.18   0.20  0.18  0.15      -0.30  1.00 -0.43  -0.03
## cut2       -0.05 -0.13  -0.13 -0.12 -0.15      -0.22 -0.43  1.00   0.02
## color2     -0.17 -0.29  -0.27 -0.26 -0.27      -0.05 -0.03  0.02   1.00
## clarity2   -0.15 -0.35  -0.37 -0.36 -0.37      -0.07 -0.16  0.19  -0.03
##            clarity2
## price         -0.15
## carat         -0.35
## length        -0.37
## width         -0.36
## depth         -0.37
## depth_perc    -0.07
## table         -0.16
## cut2           0.19
## color2        -0.03
## clarity2       1.00
##
## n= 53940
##
##
## P
##            price  carat  length width  depth  depth_perc table  cut2
## price             0.0000 0.0000 0.0000 0.0000 0.0134     0.0000 0.0000
## carat      0.0000        0.0000 0.0000 0.0000 0.0000     0.0000 0.0000
## length     0.0000 0.0000        0.0000 0.0000 0.0000     0.0000 0.0000
## width      0.0000 0.0000 0.0000        0.0000 0.0000     0.0000 0.0000
## depth      0.0000 0.0000 0.0000 0.0000        0.0000     0.0000 0.0000
## depth_perc 0.0134 0.0000 0.0000 0.0000 0.0000            0.0000 0.0000
## table      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000            0.0000
## cut2       0.0000 0.0000 0.0000 0.0000 0.0000 0.0000     0.0000
## color2     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000     0.0000 0.0000
## clarity2   0.0000 0.0000 0.0000 0.0000 0.0000 0.0000     0.0000 0.0000
##            color2 clarity2
## price      0.0000 0.0000
## carat      0.0000 0.0000
## length     0.0000 0.0000
## width      0.0000 0.0000
## depth      0.0000 0.0000
## depth_perc 0.0000 0.0000
## table      0.0000 0.0000
## cut2       0.0000 0.0000
## color2            0.0000
## clarity2   0.0000
```

```
#since both of these tables are tough to interpret, we want to get it into a way that we can understand what is and is not significant

cor_matrix_fixed <- function(rvals, pvals) {
  #' @param pvals gets the p-values from corr_matrix_2
  #' @param rvals gets the correlation coefficients from corr_matrix_2
  #' creates a data frame of the rows, columns, correlation coefficients, and p-values for the correlation matrix
  pvals <- cor_matrix_2$P #gets p values from corr_matrix_2
  rvals <- cor_matrix_2$r #gets correlation coefs from corr_matrix_2
  ut <- upper.tri(rvals)
  data.frame(
    row = rownames(rvals)[row(rvals)[ut]],
    column = rownames(rvals)[col(rvals)[ut]],
    cor  = (rvals)[ut],
    p = pvals[ut]
    )
}
cor_matrix_fixed(cor_matrix_2$P, cor_matrix_2$r)
```

```
##           row     column         cor            p
## 1       price      carat  0.92159130 0.000000e+00
## 2       price     length  0.88443516 0.000000e+00
## 3       carat     length  0.97509423 0.000000e+00
## 4       price      width  0.86542090 0.000000e+00
## 5       carat      width  0.95172220 0.000000e+00
## 6      length      width  0.97470148 0.000000e+00
## 7       price      depth  0.86124944 0.000000e+00
## 8       carat      depth  0.95338738 0.000000e+00
## 9      length      depth  0.97077180 0.000000e+00
## 10      width      depth  0.95200572 0.000000e+00
## 11      price depth_perc -0.01064740 1.340325e-02
## 12      carat depth_perc  0.02822431 5.517875e-11
## 13     length depth_perc -0.02528925 4.248197e-09
## 14      width depth_perc -0.02934067 9.382273e-12
## 15      depth depth_perc  0.09492388 0.000000e+00
## 16      price      table  0.12713390 0.000000e+00
## 17      carat      table  0.18161755 0.000000e+00
## 18     length      table  0.19534428 0.000000e+00
## 19      width      table  0.18376015 0.000000e+00
## 20      depth      table  0.15092869 0.000000e+00
## 21 depth_perc      table -0.29577852 0.000000e+00
## 22      price       cut2 -0.05349066 0.000000e+00
## 23      carat       cut2 -0.13496702 0.000000e+00
## 24     length       cut2 -0.12556524 0.000000e+00
## 25      width       cut2 -0.12146187 0.000000e+00
## 26      depth       cut2 -0.14932254 0.000000e+00
## 27 depth_perc       cut2 -0.21805501 0.000000e+00
## 28      table       cut2 -0.43340461 0.000000e+00
## 29      price     color2 -0.17251093 0.000000e+00
## 30      carat     color2 -0.29143675 0.000000e+00
## 31     length     color2 -0.27028669 0.000000e+00
## 32      width     color2 -0.26358440 0.000000e+00
## 33      depth     color2 -0.26822688 0.000000e+00
## 34 depth_perc     color2 -0.04727923 0.000000e+00
## 35      table     color2 -0.02646520 7.872680e-10
## 36       cut2     color2  0.02051852 1.880911e-06
## 37      price   clarity2 -0.14680007 0.000000e+00
## 38      carat   clarity2 -0.35284057 0.000000e+00
## 39     length   clarity2 -0.37199853 0.000000e+00
## 40      width   clarity2 -0.35841962 0.000000e+00
## 41      depth   clarity2 -0.36695200 0.000000e+00
## 42 depth_perc   clarity2 -0.06738444 0.000000e+00
## 43      table   clarity2 -0.16032684 0.000000e+00
## 44       cut2   clarity2  0.18917474 0.000000e+00
## 45     color2   clarity2 -0.02563128 2.621447e-09
```

```
library(ggcorrplot)
# Create a correlation matrix of  all of the variables
mat <- ggcorrplot(cor_matrix, method ="square", type = "full", ggtheme = ggplot2::theme_
minimal, title = "Correlation Matrix of Diamond Features", show.legend = TRUE, legend.ti
tle = "Correlation", show.diag = FALSE, colors = c("blue", "white", "red"), outline.colo
r = "white", hc.order = FALSE, hc.method = "complete", lab = TRUE, lab_col = "black", la
b_size = 2)

#make the heatmap interactive
q <- ggplotly(p = ggplot2::last_plot())
q
```



```
#Write function to perform T tests
cols_to_test <- c("cut2", "color2", "clarity2", "carat", "depth_perc", "table", "length"
, "width", "depth")
t_results <- ldply(
  cols_to_test,
  function(colname) {
    #' @param colname is the name of the column in the dataset
    #' @param t_val is the value generated from the t-test compared to price
    #' @param p_val is the p-value associated with the t-test
    t_val = t.test(diamonds[[colname]], diamonds$price)$statistic
    p_val = 2*pt(-abs(t_val), df=length(diamonds)-1)
    return(data.frame(colname=colname, t_value=t_val, p_value=p_val))
    })

#returns a table that has each variable the p value of its t test vs price
print(t_results)
```

```
##      colname   t_value      p_value
## 1       cut2 -228.7252 3.281528e-23
## 2     color2 -228.6960 3.286562e-23
## 3   clarity2 -228.7166 3.283002e-23
## 4      carat -228.9060 3.250560e-23
## 5 depth_perc -225.3577 3.920664e-23
## 6      table -225.6075 3.868887e-23
## 7     length -228.6188 3.299892e-23
## 8      width -228.6186 3.299926e-23
## 9      depth -228.7465 3.277868e-23
```

```
#write function to perform chi square test
cols_to_test <- c("cut", "color", "clarity", "carat", "depth_perc", "table", "length",
"width", "depth")
chi_results <- ldply(
  cols_to_test,
  function(colname) {
    #' @param colname is the name of the column in the dataset
    #' @param chisq_val is the value generated from the chisquare test of a value vs pri
ce
    #' @param p_val is the p-value associated with the t-test
    chisq_val = chisq.test(diamonds[[colname]], diamonds$price, correct = T)$statistic
    p_val = chisq.test(diamonds[[colname]], diamonds$price, correct = T)$p.val
    return(data.frame(colname=colname, chisq_value=chisq_val, p_value=p_val))
    })
```

```
## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect

## Warning in chisq.test(diamonds[[colname]], diamonds$price, correct = T):
## Chi-squared approximation may be incorrect
```

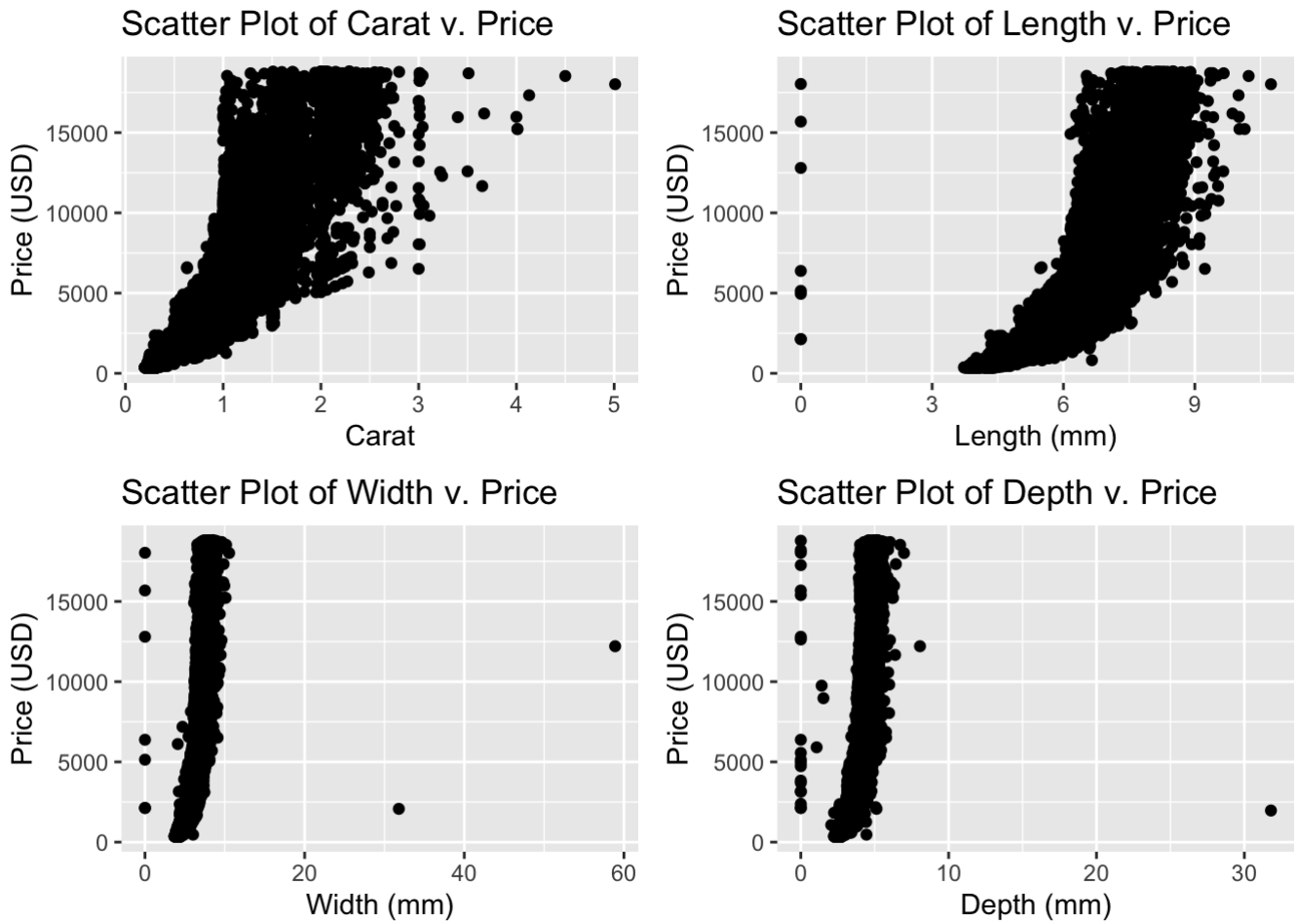
```
#return table of chisquare values and p values for each variable when compared to price
print(chi_results)
```

```
##      colname chisq_value       p_value
## 1        cut    58970.25 1.163753e-316
## 2      color   111326.28  0.000000e+00
## 3    clarity   127923.17  0.000000e+00
## 4      carat  8368939.00  0.000000e+00
## 5 depth_perc  2092629.52  1.000000e+00
## 6      table  1458379.49  9.749052e-01
## 7     length 11143992.81  0.000000e+00
## 8      width 10997185.07  0.000000e+00
## 9      depth  7797366.28  0.000000e+00
```

###Modeling

Now that the four variables that are most strongly correlated with price have been established, the type of regression analysis must be chosen. A multivariate regression is going to be necessary to compare all four variables to price at the same time. In order to get a better idea of the shape of the data, scatter plots were created for each of the variabels vs. price using the ggplot2 function geom_point(). It is clear from the visualization that these relationships are not linear. because of the upward curvature of the data, an exponential regression analysis is the best suited. The output of the model and the plots to analyze fit are shown below.The intercept is 1.803406, the coefficient for carat is -0.645099, the coefficient for length is 0.988625, for width is 0.037068, and for depth is 0.175085. These are all statistically significant values. Therefore, the formula for price prediction is:
price = 1.803406 + (-0.645099)xcarat + 0.988625xlength + 0.037068xwidth + 0.175085xdepth
Based on the straight line in the residuals plot, the shape of the Q-Q plot, and the straight line in the scale-location plot, it is clear that, although it is not a perfect fit. This is an acceptable and valid model to predict price.

```
#Scatter plots for strongly correlated variables to determine what kind of model I should use (carat, length, width, depth)
library(ggplot2)
carat_scatter <- ggplot(diamonds, aes(x=carat, y=price)) + geom_point() + ggtitle("Scatter Plot of Carat v. Price") +
  xlab("Carat") + ylab("Price (USD)")

length_scatter <- ggplot(diamonds, aes(x=length, y=price)) + geom_point() + ggtitle("Scatter Plot of Length v. Price") + xlab("Length (mm)") + ylab("Price (USD)")

width_scatter <- ggplot(diamonds, aes(x=width, y=price)) + geom_point() + ggtitle("Scatter Plot of Width v. Price") +
  xlab("Width (mm)") + ylab("Price (USD)")

depth_scatter <- ggplot(diamonds, aes(x=depth, y=price)) + geom_point() + ggtitle("Scatter Plot of Depth v. Price") +
  xlab("Depth (mm)") + ylab("Price (USD)")

grid.arrange(carat_scatter, length_scatter, width_scatter, depth_scatter, ncol=2)
```




```
## Warning: package 'jtools' was built under R version 3.6.2
```

```
##
## Attaching package: 'jtools'
```

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

```
## Warning: package 'survey' was built under R version 3.6.2
```

```
## Loading required package: grid
```

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

```
##
## Attaching package: 'Matrix'
```

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

```
##
## Attaching package: 'survey'
```

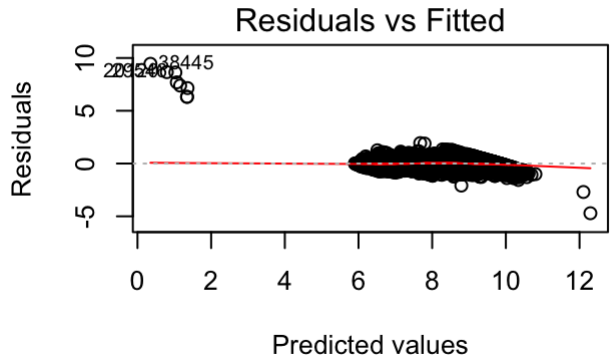
```
## The following object is masked from 'package:Hmisc':
##
##     deff
```

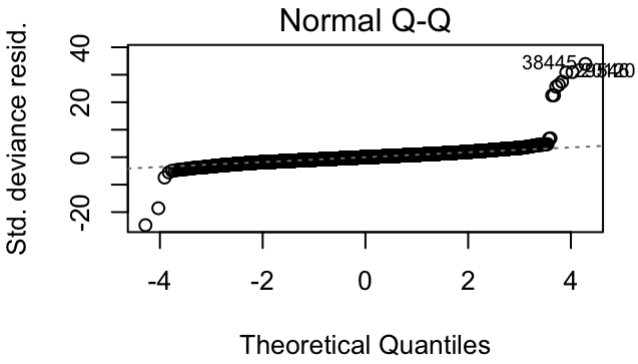
```
## The following object is masked from 'package:graphics':
##
##     dotchart
```

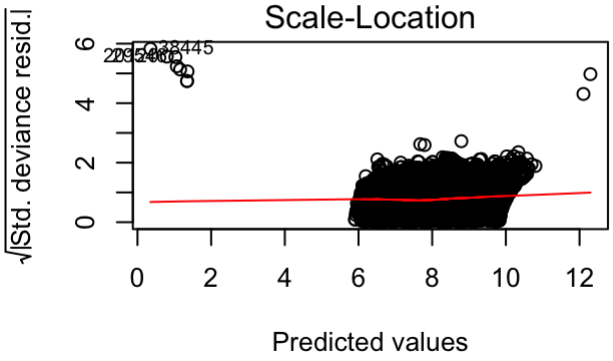
```
#make an exponential model of the variable vs price
diamond_model <- glm(log(price) ~ carat + length + width + depth, data = diamonds)

#print out a sumamary of the model
summary(diamond_model, exp=TRUE)
```

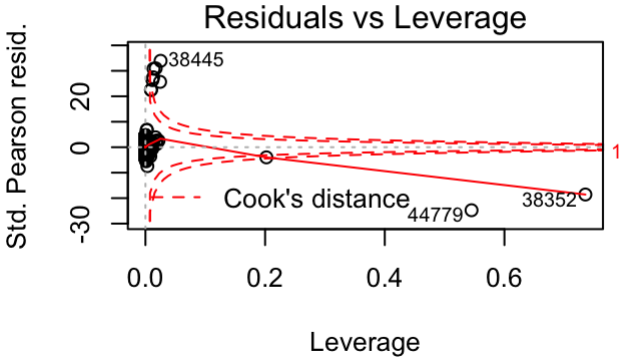
```
##
## Call:
## glm(formula = log(price) ~ carat + length + width + depth, data = diamonds)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7090  -0.1722  -0.0023   0.1661   9.4481
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.803406   0.019318  93.352  < 2e-16 ***
## carat       -0.645099   0.011649 -55.379  < 2e-16 ***
## length       0.988625   0.007491 131.982  < 2e-16 ***
## width        0.037068   0.004786   7.745 9.74e-15 ***
## depth        0.175085   0.007271  24.081  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.07956713)
##
##     Null deviance: 55530.9  on 53939  degrees of freedom
## Residual deviance:  4291.5  on 53935  degrees of freedom
## AIC: 16552
##
## Number of Fisher Scoring iterations: 2
```

```
#plot the model to assess for fit
par(mfrow=c(2,2))
plot1 <- plot(diamond_model)
```









###Predictive Function

In order to test the predictive ability of the model, it should be tested through a function. The function below, price_predict, is a function that takes in a vector of values and, based on their designation, places them in the correct place for evaluation in the model. Then, it returns the predicted price. It is important to note that while it is more meaningful if all four variables are present in the vector, if one of them is not present, the if statements within the function will fill in the missing value with the average value from the respective column in the dataset. For the test diamond, the target price is $15,964 USD. The price_predict function returned a price of $15,895 USD, which is a very close estimate given the imperfect fit of our model.
Although the function works on diamonds from the dataset. It is important to determine if the function is generalizable to diamonds that are not in the dataset. To do this, the ranges for each of the values were determined, and a random value from each was selected. From each of these random values, a random diamond was generated. Then, the random diamond was evaluated with the price_predict function. The random diamond generated a value of $1168 USD, which is well within the range of possible values for diamond prices given the parameters.

```
#vars for function are carat, length, width, depth
#target price = 15964
testdiamond <- c(carat=3.40, length=9.42 , width=9.34 , depth=6.27)

price_predict <- function(x){
  #' returns a price prediction for the values put into the function
  #' @param x is a vector of 1 to 4 values. their position in the vector determines what
their name is
  #' @param price is the output of the price predition function after it all values are
 put into the model
  #' the if statements will assign the mean of the column in place of any missing values

    if(is.na(x[1]))
        x[1] <- 0.7979397
    if(is.na(x[2]))
        x[2] <-5.7311572
    if(is.na(x[3]))
        x[3] <-5.7345260
    if(is.na(x[4]))
        x[4] <-3.5387338

  price <- 1.803406 + sum((-0.645099)*x[1] + 0.988625*x[2] + 0.037068*x[3] + 0.175085*x[
4])
  return(exp(price)/2)
}

price_predict(testdiamond)
```

```
## [1] 15895.13
```

```
#generate random values for carat, length, width, depth
rc <- (0.2:5.01)
rl <- (0:10.74)
rw <- (0:58.9)
rd <- (0:31.8)

#generate a random diamond
randomdiamond <- c(carat=sample(rc,1), length=sample(rl,1), width=sample(rw,1), depth=sa
mple(rd,1))
print(randomdiamond)
```

```
##  carat length  width  depth
##    4.2    8.0   24.0    7.0
```

```
price_predict <- function(x){
  #' returns a price prediction for the values put into the function
  #' @param x is a vector of 1 to 4 values. their position in the vector determines what
their name is
  #' @param price is the output of the price predition function after it all values are
 put into the model
  #' the if statements will assign the mean of the column in place of any missing values
    if(is.na(x[1]))
        x[1] <- 0.7979397
    if(is.na(x[2]))
        x[2] <-5.7311572
    if(is.na(x[3]))
        x[3] <-5.7345260
    if(is.na(x[4]))
        x[4] <-3.5387338

  price <- 1.803406 + sum((-0.645099)*x[1] + 0.988625*x[2] + 0.037068*x[3] + 0.175085*x[
4])
  return(exp(price)/2)
}

price_predict(randomdiamond)
```

```
## [1] 4559.964
```

###Conclusion

Completing this project required competency in several programming areas. First, in orde r to evaluate this dataset, one must know how to clean data and manipulate it in such a way that it is easy to analyze and udnerstand. Next, one must understand how to visualiz e data and conclude which variables are significant for the project. After that, it is i mportant to be able to further visualize data so that the proper model for regression an alysis can be determined. After that, one must be able to generate a regression model to predict price, write a function to utilize that model in price prediction, and simulate data to evaluate the efficacy of that function.

Based on thse findings, it is clear that carat is the most strongly correlated with pric e, and it is closely followed by length, width, and depth. Because carat is the weight o f the diamond and length, width, and depth are are related to the size of the diamond, i t makes sense that these are the strongest predictors of price because large diamonds ar e frequently in high demand. Also, size and weight are closely related, so it makes sens e that they would both be correlated with price. While it was originally hypothesized th at cut, color, or clarity may also have a significant impact on price, the analysis show ed that this is not the case. This may be because, while these are factors one may want to consider when making a purchase, carat is still a more important factor to most consu mers.

While the model has a reasonable fit, it is not perfect, and therefore it is not a perfe ct predictor of price. However, with this in mind, it is able to return reasonable estim ates for diamonds in the dataset and simulated diamonds. Further research into this subj ect could study how to better fit the model. Overall, this analysis was able to solidly and comprehensively examine how various features of diamonds relate to one another and i nfluence price.