Simple Exploratory Data Analysis in R

Marius G. Sampid

29 October 2018

Introduction

In this report, I present a simple exploratory data analysis using the publicly available diamond data set, which can be found in https://vincentarelbundock.github.io/Rdatasets/datasets.html. The analysis are produced using R markdown script so that readers can easily view the R codes, comments, and output results.

Data Description

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)

Reading data into R

```r
# Get the address/path where data is stored:
address <-
"C:/Users/marius/Desktop/deskTopFiles/DataAnalysis_R_Python/SampleData.csv"

# store data in an object called 'myData'
myData <- read.csv(address)
```

Exploring the data set

In this section, I present some basic data visualization to see the overall shape and meaning in the data. That is, view the data in some dofferent perspectives.

```r
dim(myData) # see the dimension of the data

## [1] 53940    10

head(myData) # See the first 6 rows of the data set. A quick snapshot of the
data to make sure that the data was correctly imported into R

##   carat       cut color clarity depth table price    x    y    z
## 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
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48

str(myData) # see the basic structure of the data

## 'data.frame':    53940 obs. of  10 variables:
##  $ 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 ...
```

The output of the str( ) function above shows that the data has 53940 observations and 10 variables. Of the 10 variables, there are 3 factor (categorical) variables of 5, 7, and 8 levels, respectively.

We can easily see the summary statistics of the numeric variables such as the mean, standard deviation, variance, etc with 95% confidence level using the bsicStats( ) function from the fBasics package. This function is also important to see if there are any missing values (NA's):

```r
Install.packages(“fBasics”) # install package
library(fBasics) # Load package

# isolate or subset the numeric variables and calculate summary statistics.

basicStats(myData[,c(1,5:10)])

##                    carat        depth        table        price
## nobs        53940.000000 5.394000e+04 5.394000e+04 5.394000e+04
## NAs             0.000000 0.000000e+00 0.000000e+00 0.000000e+00
## Minimum         0.200000 4.300000e+01 4.300000e+01 3.260000e+02
## Maximum         5.010000 7.900000e+01 9.500000e+01 1.882300e+04
## 1. Quartile     0.400000 6.100000e+01 5.600000e+01 9.500000e+02
## 3. Quartile     1.040000 6.250000e+01 5.900000e+01 5.324250e+03
```

```
## Mean            0.797940  6.174941e+01 5.745718e+01 3.932800e+03
## Median          0.700000  6.180000e+01 5.700000e+01 2.401000e+03
## Sum         43040.870000  3.330763e+06 3.099241e+06 2.121352e+08
## SE Mean         0.002041  6.168000e-03 9.621000e-03 1.717736e+01
## LCL Mean        0.793939  6.173732e+01 5.743833e+01 3.899132e+03
## UCL Mean        0.801940  6.176149e+01 5.747604e+01 3.966467e+03
## Variance        0.224687  2.052404e+00 4.992948e+00 1.591563e+07
## Stdev           0.474011  1.432621e+00 2.234491e+00 3.989440e+03
## Skewness        1.116584 -8.228900e-02 7.968520e-01 1.618305e+00
## Kurtosis        1.256250  5.738447e+00 2.801271e+00 2.177191e+00
##                         x            y            z
## nobs         53940.000000 5.394000e+04 5.394000e+04
## NAs              0.000000 0.000000e+00 0.000000e+00
## Minimum          0.000000 0.000000e+00 0.000000e+00
## Maximum         10.740000 5.890000e+01 3.180000e+01
## 1. Quartile      4.710000 4.720000e+00 2.910000e+00
## 3. Quartile      6.540000 6.540000e+00 4.040000e+00
## Mean             5.731157 5.734526e+00 3.538734e+00
## Median           5.700000 5.710000e+00 3.530000e+00
## Sum         309138.620000 3.093203e+05 1.908793e+05
## SE Mean          0.004830 4.918000e-03 3.039000e-03
## LCL Mean         5.721690 5.724887e+00 3.532778e+00
## UCL Mean         5.740624 5.744165e+00 3.544689e+00
## Variance         1.258347 1.304472e+00 4.980110e-01
## Stdev            1.121761 1.142135e+00 7.056990e-01
## Skewness         0.378655 2.434031e+00 1.522338e+00
## Kurtosis        -0.618303 9.120250e+01 4.708029e+01
```

We can also see from the summary statistics that the distribution of the numeric variables: 'carat', 'table', 'price', 'x', 'y' and 'z' are positively skewed while 'depth' is slightly negatively skewed. Apart from variable 'x', the variables also show significant excess kurtosis. Thus the variables are not normally distributed. A normal distribution has excess kurtosis of 0, is symmetric around the mean with 0 skewness. Asumming normality with this data set to predict or forecast future diamond prices might yield false results because extreme events will be negleted or cut out. Therefore, we need to rely on the standardized residuals for forecasting. We also see from the summany statistics that there are no missing values.

It is also worth noting that a positive kurtosis imply that there are more data points in the tails than the normal distribution and a negative kurtosis imply that there are less data points in the tails than the normal distribution, which is the case for variable 'x'.

We can also use the summary function; summary( ), to see summary statistics of the numeric variables as:

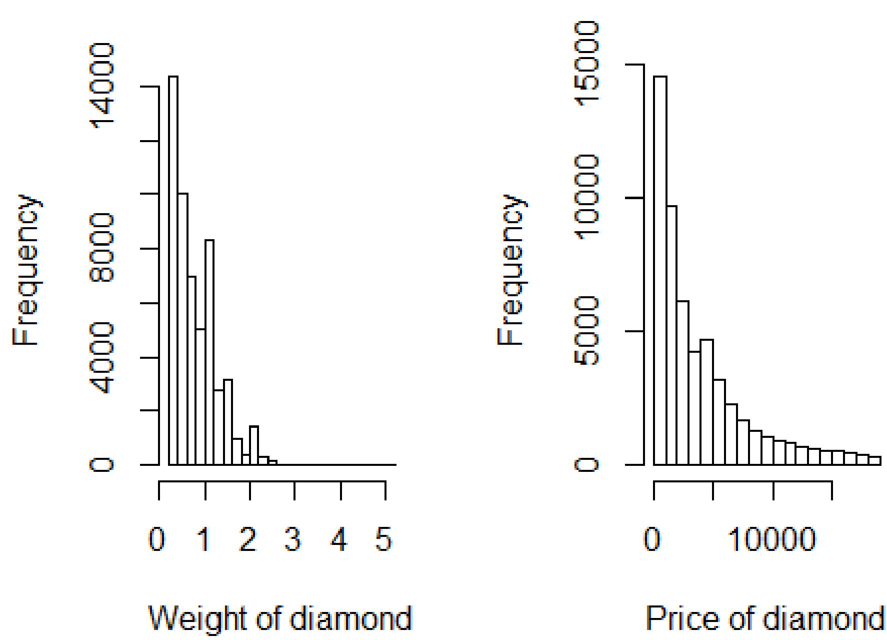
```
summary(myData[,c(1,5:10)]) # give summary of the numeric variables in the
data set.
```

```
##      carat            depth           table           price      
##  Min.   :0.2000   Min.   :43.00   Min.   :43.00   Min.   :  326  
##  1st Qu.:0.4000   1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950  
```

```
##  Median :0.7000   Median :61.80   Median :57.00   Median : 2401
##  Mean   :0.7979   Mean   :61.75   Mean   :57.46   Mean   : 3933
##  3rd Qu.:1.0400   3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324
##  Max.   :5.0100   Max.   :79.00   Max.   :95.00   Max.   :18823
##        x                y                z
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.700   Median : 5.710   Median : 3.530
##  Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :10.740   Max.   :58.900   Max.   :31.800
```

The distribution of each variable can also be viewed with the hist( ) command. For example, let's take a look at variables 'carat' and 'price'; the weights and price of diamond, respectively:

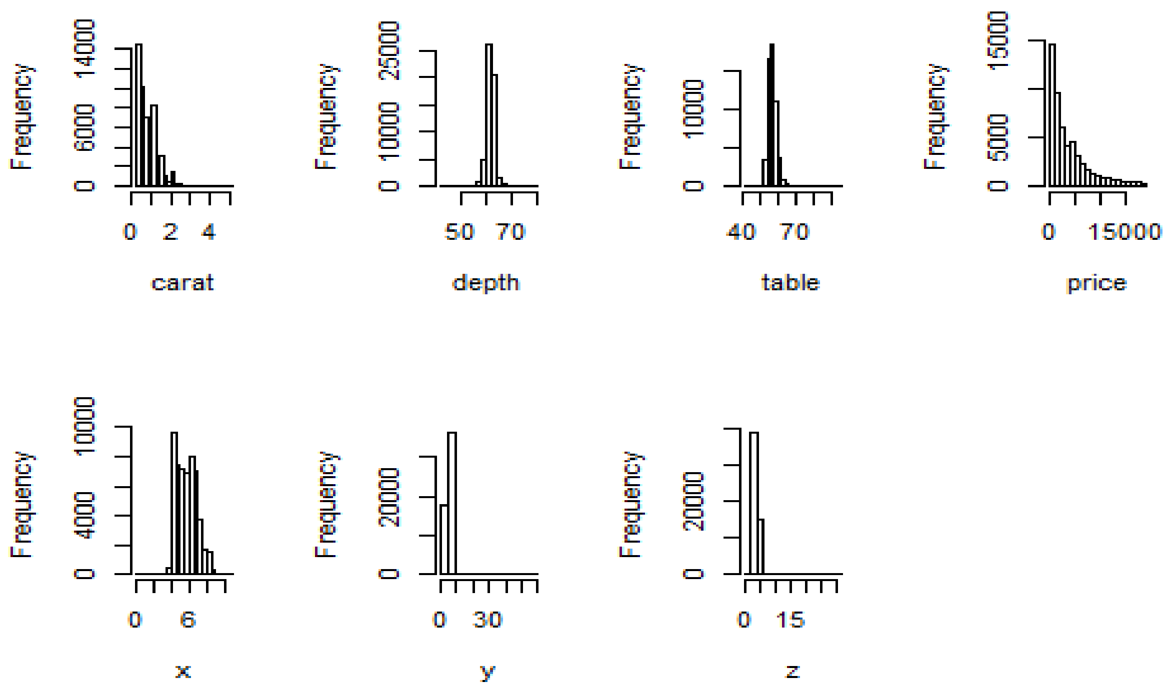
```r
par(mfrow = c(1,2)) # create a matrix of one rows and 2 columns to display figures
diamondWeight = myData[,1]
diamondPrice  = myData[,7]
hist(diamondWeight, breaks = 20, main = "", xlab =  "Weight of diamond")
hist(diamondPrice, breaks = 20, main = "", xlab =  "Price of diamond")
```




```r
par(mfrow = c(1,1)) # bring back to its original layout
```

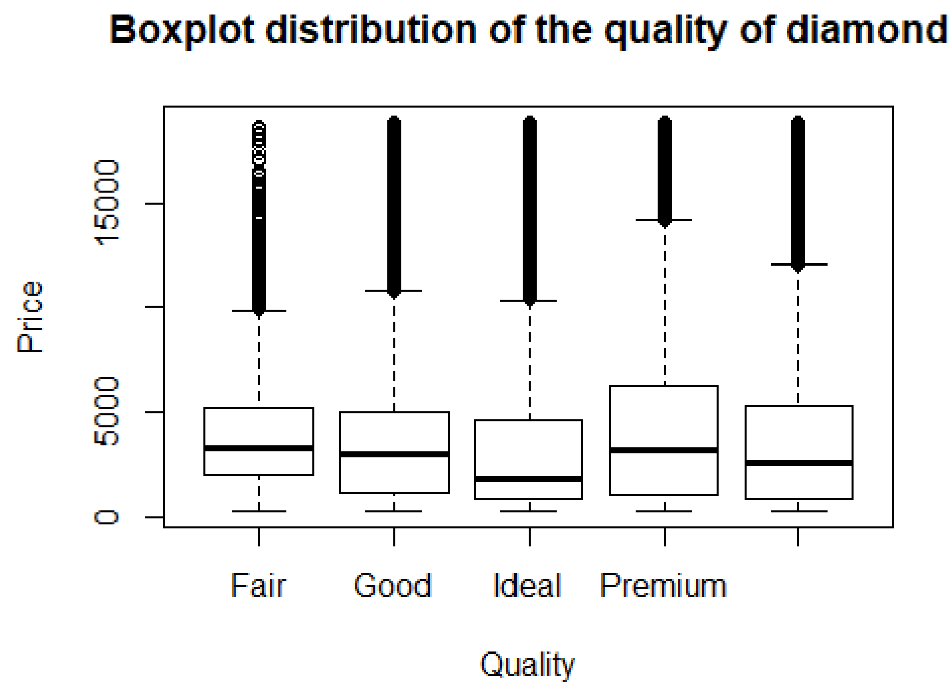
We can see from the above plots that the distributions of the weights and prices of diamond are far from a normal distribution. The rest of the plots are shown below:

```
par(mfrow = c(2,4)) # create a matrix of two rows and 4 columns to display figures
hist(myData[,1], main = "", xlab = "carat", breaks = 20)
hist(myData[,5], main = "", xlab = "depth", breaks = 20)
hist(myData[,6], main = "", xlab =  "table", breaks = 20)
hist(myData[,7], main = "", xlab =  "price", breaks = 20)
hist(myData[,8], main = "", xlab =  "x", breaks = 20)
hist(myData[,9], main = "", xlab =  "y", breaks = 20)
hist(myData[,10], main = "", xlab =  "z", breaks = 20)
par(mfrow = c(1,1)) # bring back to its original layout
```

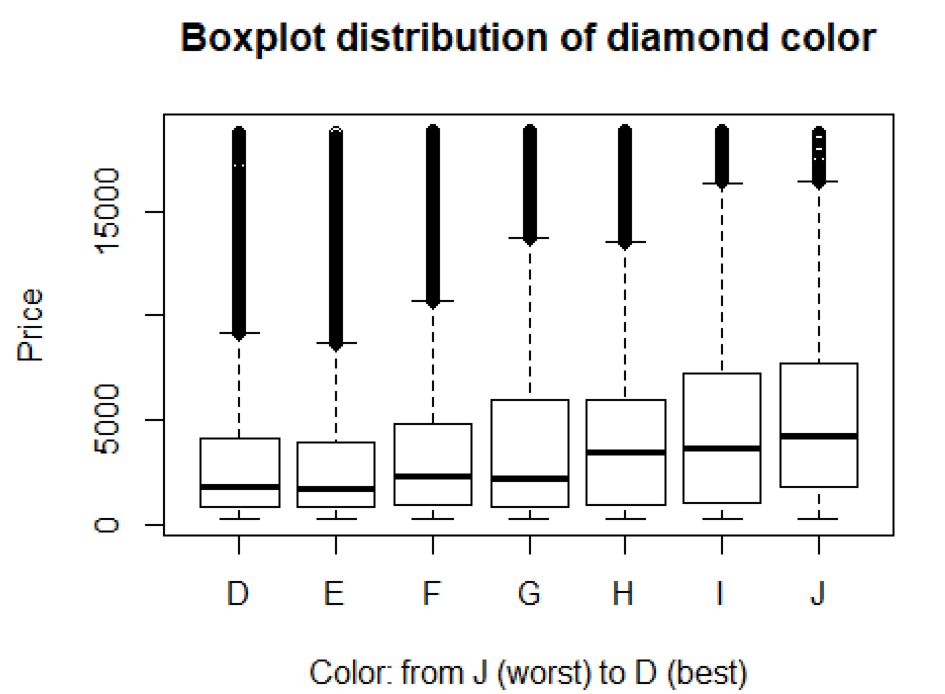


Another good method to visualize the distribution of the data set across the various categories is by using boxplots as follows:

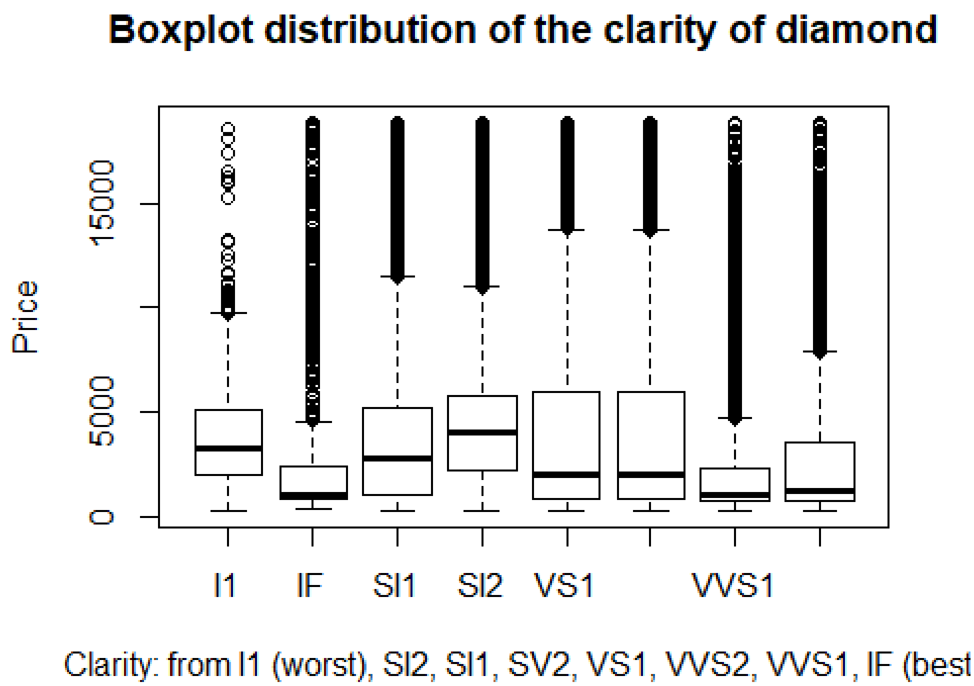
```
boxplot(myData$price~myData$cut, xlab ="Quality", ylab = "Price", main =
"Boxplot distribution of the quality of diamond")
```




```
boxplot(myData$price~myData$color, xlab ="Color: from J (worst) to D (best)",
ylab = "Price", main = "Boxplot distribution of diamond color")
```



```r
boxplot(myData$price~myData$clarity, xlab ="Clarity: from I1 (worst), SI2, SI1, SV2, VS1, VVS2, VVS1, IF (best)", ylab = "Price", main = "Boxplot distribution of the clarity of diamond")
```



We can see from the boxplots that the distribution of the diamond data set with respect to price in the various categories are all skewed with huge outliers.

The variable names in the data set can be viewed with the name( ) function as seen below. This information will help in case we want to subset the data using [ ] or $ sign, so that we can easily refer to the variable names.

```r
names(myData)

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

We now take a closer look at the factor (Categorical) variables

Of the 53940 observations, how many of the diamonds were recorded as Fair, Good, Very Good, Premium, and Ideal quality? The following code gives us the answer:

```r
attach(myData)
table(cut)

## cut
##      Fair      Good     Ideal   Premium Very Good
##      1610      4906     21551     13791     12082
```

```
# Note: To avoid using '$' sign or '[,]' in subsetting the data set, We can use the attach( ) function to attach everything hidden within the data set to the work space. This makes it easy to work directly with the variable names. Remember to detach at the end using the detach( ) function.
```

What was the distribution of the colors from worst (J) to best (D), and the measurement of how clear the diamonds were (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))? The following code gives us this answer:

```
table(color) # distribution of diamond color from worst (J) to best (D)

## color
##     D     E     F     G     H     I     J
##  6775  9797  9542 11292  8304  5422  2808

table(clarity) # distribution of clarity of diamond; I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best)

## clarity
##    I1    IF   SI1   SI2   VS1   VS2  VVS1  VVS2
##   741  1790 13065  9194  8171 12258  3655  5066
```

To see the total price (in US dollars) on each category of the quality of diamond, diamond color and diamond clarity, we use the following codes:

```
# Calculates the total price (in US dollars) on each category of the quality of diamond (Fair, Good, Very Good, Premium, Ideal).
TotalPrice_cut <- aggregate(price,by=list(cut),sum)
colnames(TotalPrice_cut) <- c("Quality of Diamond ", "  Total Price")
TotalPrice_cut

##   Quality of Diamond    Total Price
## 1                Fair       7017600
## 2                Good      19275009
## 3               Ideal      74513487
## 4             Premium      63221498
## 5           Very Good      48107623

# Calculates total price (in US dollars) on each category of the color of diamond from J (worst) to D (best).
TotalPrice_color <- aggregate(price,by=list(color),sum)
colnames(TotalPrice_color) <- c("Diamond Color ", "  Total Pricet")
TotalPrice_color

##   Diamond Color    Total Pricet
## 1             D        21476439
## 2             E        30142944
## 3             F        35542866
## 4             G        45158240
## 5             H        37257301
## 6             I        27608146
## 7             J        14949281
```

```
# Calculates the total price (in US dollars) on each category of the clarity
of diamond (Fair, Good, Very Good, Premium, Ideal).
TotalPrice_clarity <- aggregate(price,by=list(clarity),sum)
colnames(TotalPrice_clarity) <- c("Clarity of Diamond ", "  Total Price")
TotalPrice_clarity

##   Clarity of Diamond    Total Price
## 1                  I1       2907809
## 2                  IF       5128062
## 3                 SI1      52207755
## 4                 SI2      46549485
## 5                 VS1      31372190
## 6                 VS2      48112520
## 7                VVS1       9221984
## 8                VVS2      16635412
```

To see the total weights, in carats, on each category based on the color of diamond, we employ the following code:

```
# Calculates total weight, in carats, of diamond on each category of the
color of diamond from J (worst) to D (best).
TotalWeight_color <- aggregate(carat,by=list(color),sum)
colnames(TotalWeight_color) <- c("Diamond Color ", "  Total Weight")
TotalWeight_color

##   Diamond Color    Total Weight
## 1             D         4456.56
## 2             E         6445.12
## 3             F         7028.05
## 4             G         8708.28
## 5             H         7571.58
## 6             I         5568.00
## 7             J         3263.28
```

To see the total weights, in carats, of each category based on the quality of diamond, we use the following code:

```
TotalWeightCarat <- aggregate(carat,by=list(cut),sum) # calculates total
weight in each category of quality of diamond.
colnames(TotalWeightCarat) <- c("Quality ", "  Total weight of diamond
(carats)")
TotalWeightCarat

##    Quality    Total weight of diamond (carats)
## 1      Fair                            1684.28
## 2      Good                            4166.10
## 3     Ideal                           15146.84
## 4   Premium                           12300.95
## 5 Very Good                            9742.70
```

Note that we can also use other aggregate functions to get information such as summary statistics, max, mean, median, and counts values, etc. For example, the following codes show summary statistics of price based on quality of diamond, color of diamond and clarity of diamond:

```
aggregate(price,by=list(cut),summary) # summary statistics for the quality of diamond based on price

##     Group.1 x.Min. x.1st Qu. x.Median x.Mean x.3rd Qu. x.Max.
## 1      Fair    337      2050     3282   4359      5206  18570
## 2      Good    327      1145     3050   3929      5028  18790
## 3     Ideal    326       878     1810   3458      4678  18810
## 4   Premium    326      1046     3185   4584      6296  18820
## 5 Very Good    336       912     2648   3982      5373  18820

aggregate(price,by=list(color),summary) #  summary statistics for the color of diamond based on price

##   Group.1 x.Min. x.1st Qu. x.Median x.Mean x.3rd Qu. x.Max.
## 1       D    357       911     1838   3170      4214  18690
## 2       E    326       882     1739   3077      4003  18730
## 3       F    342       982     2344   3725      4868  18790
## 4       G    354       931     2242   3999      6048  18820
## 5       H    337       984     3460   4487      5980  18800
## 6       I    334      1120     3730   5092      7202  18820
## 7       J    335      1860     4234   5324      7695  18710

aggregate(price,by=list(clarity),summary) # summary statistics for the clarity of diamond based on price

##   Group.1  x.Min. x.1st Qu. x.Median  x.Mean x.3rd Qu.  x.Max.
## 1      I1   345.0    2080.0   3344.0  3924.0    5161.0 18530.0
## 2      IF   369.0     895.0   1080.0  2865.0    2388.0 18810.0
## 3     SI1   326.0    1089.0   2822.0  3996.0    5250.0 18820.0
## 4     SI2   326.0    2264.0   4072.0  5063.0    5777.0 18800.0
## 5     VS1   327.0     876.0   2005.0  3839.0    6023.0 18800.0
## 6     VS2   334.0     900.0   2054.0  3925.0    6024.0 18820.0
## 7    VVS1   336.0     816.0   1093.0  2523.0    2379.0 18780.0
## 8    VVS2   336.0     794.2   1311.0  3284.0    3638.0 18770.0
```

A simple t-test

Is the average price per diamond different depending on the quality of the diamond? This question can be answered by conducting a simple t-tests. Let’s conduct two simple t-tests; for the first test, we use qualities: “Fair” and “Good”. For the second test we used qualities “Fair” and “Ideal” as follows:

```
t.test(price[cut=="Fair"], price[cut=="Good"]) # first t-test
```

```
##
##  Welch Two Sample t-test
##
## data:  price[cut == "Fair"] and price[cut == "Good"]
## t = 4.1684, df = 2822.3, p-value = 3.16e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  227.6710 632.1156
## sample estimates:
## mean of x mean of y
##  4358.758  3928.864

t.test(price[cut=="Fair"], price[cut=="Ideal"]) # second t-test

##
##  Welch Two Sample t-test
##
## data:  price[cut == "Fair"] and price[cut == "Ideal"]
## t = 9.7484, df = 1894.8, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   719.9065 1082.5251
## sample estimates:
## mean of x mean of y
##  4358.758  3457.542

detach(myData)
```

Looking at the output of the t-test, for example the first t-test, when the diamond quality is 'Fair', the average price is 4358.758 and when the diamond quality is 'Good', the average price is 3928.864. The p-value for both t-tests suggest that there is significant difference. Therefore, at 95% confidence level, we should reject the null hypothesis and conclude that the average price per diamond is dependent on the quality of the diamond.

Note: p-value is the smallest level of significance in which we will reject the null hypothesis in favour of the alternative hypotheisis. At 95% confidence, we should reject the null hypothesis if p-value < 5% significance level.