Assignment1 - Diamond data set

PC

13 November 2016

Data Set

```
install.packages("ggplot2", dependencies = T, repos = "http://cran.us.r-project.org")
```

```
## also installing the dependencies 'plyr', 'scales'
```

```
## package 'plyr' successfully unpacked and MD5 sums checked
## package 'scales' successfully unpacked and MD5 sums checked
## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
data("diamonds")
```

Basic summary

```
str(diamonds)
```

```
## Classes 'tbl_df', 'tbl' and '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    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 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 ...
```

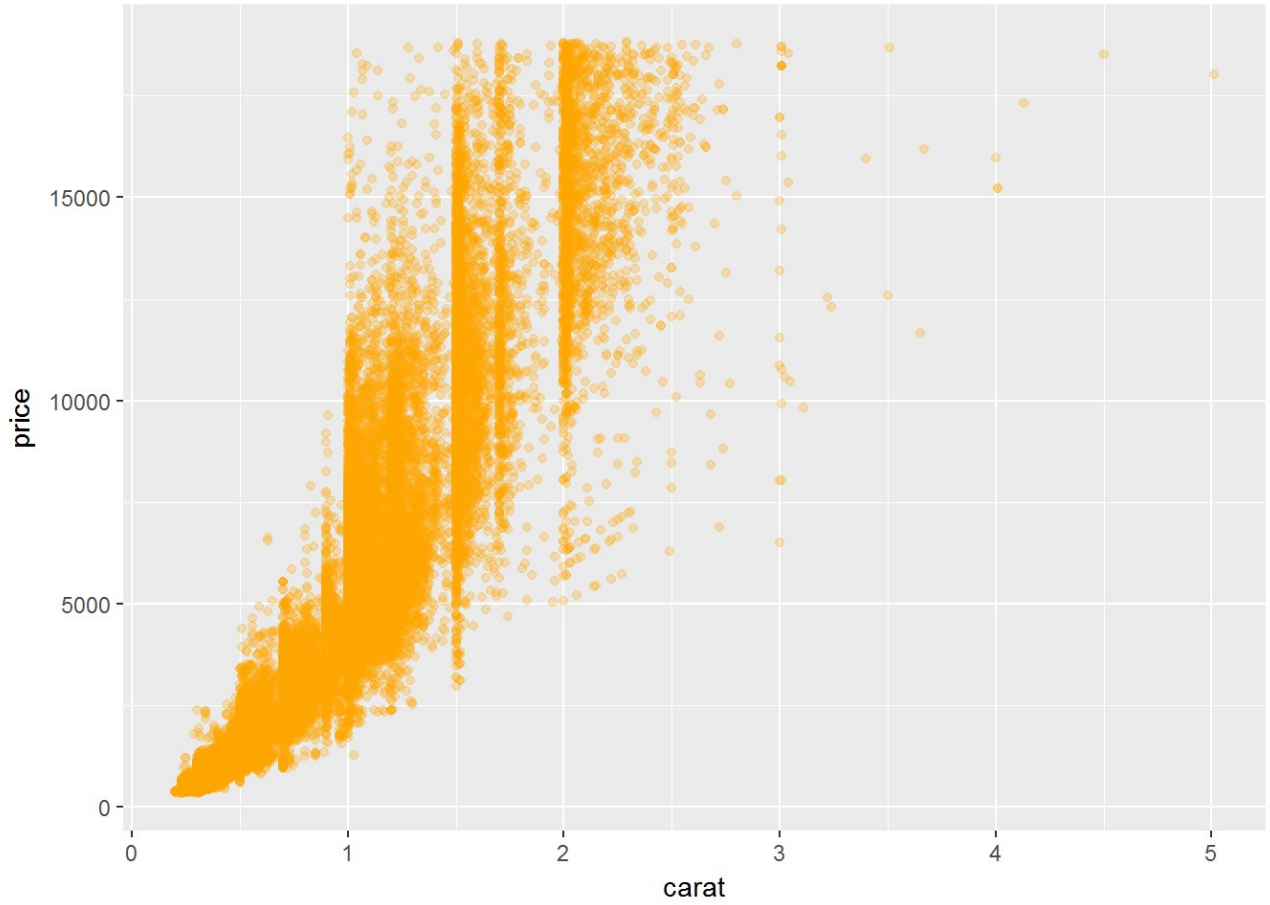
The `diamonds` dataset chips with `ggplot2` and contains the prices and specs for more than 50 thousand diamonds collected from diamondse.info (http://www.diamondse.info/) between the years of 2008 to 2014. It consists of 53940 observations of 10 variables.

Task 1

Begin your analysis with the summary of the variables and basic numerical statistics. Pick 2 variables that you find interesting and express your thoughts based only on the numbers you see.

As our data set contains observations about characteristics of diamonds, we can start the analysis with two variables which can be `price` and `carat`. So first we create a scatterplot for these two variables.

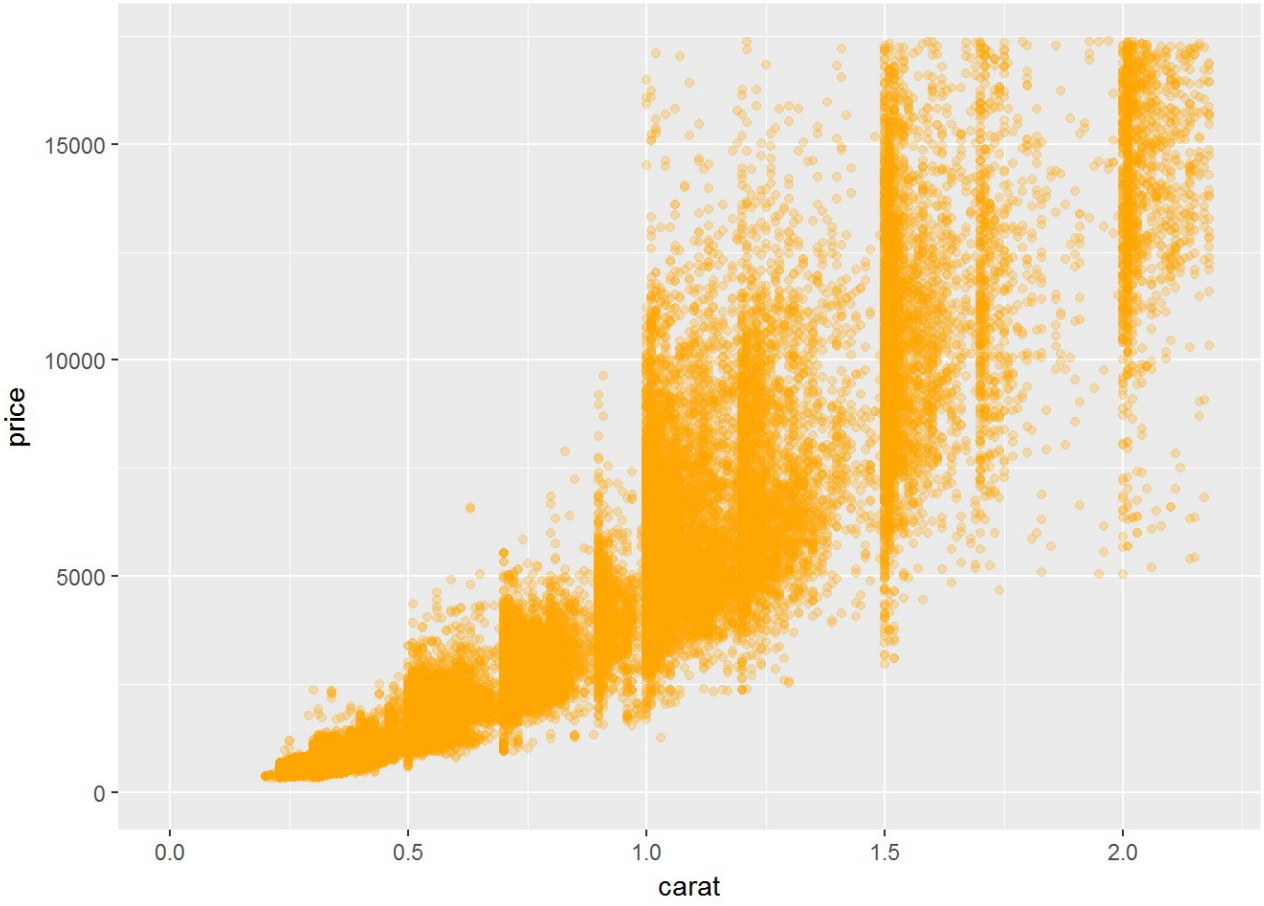
```
ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point( color = 'orange', alpha = 1/4)
```



We can see that our data has outliers, thus making it a long tailed data. What we can do to remove these outliers is we can plot our scatterplot and remove the last 1% of data from our plot. This can be done as follows:

```
ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point( color = 'orange', alpha = 1/4) +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99))) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99)))
```

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



Some observations from the plot are - * We have a more clear view of our data now. But we cannot tell what kind of relationship does the two variables hold. * It is a non-linear relationship but it can be or cannot be exponential.

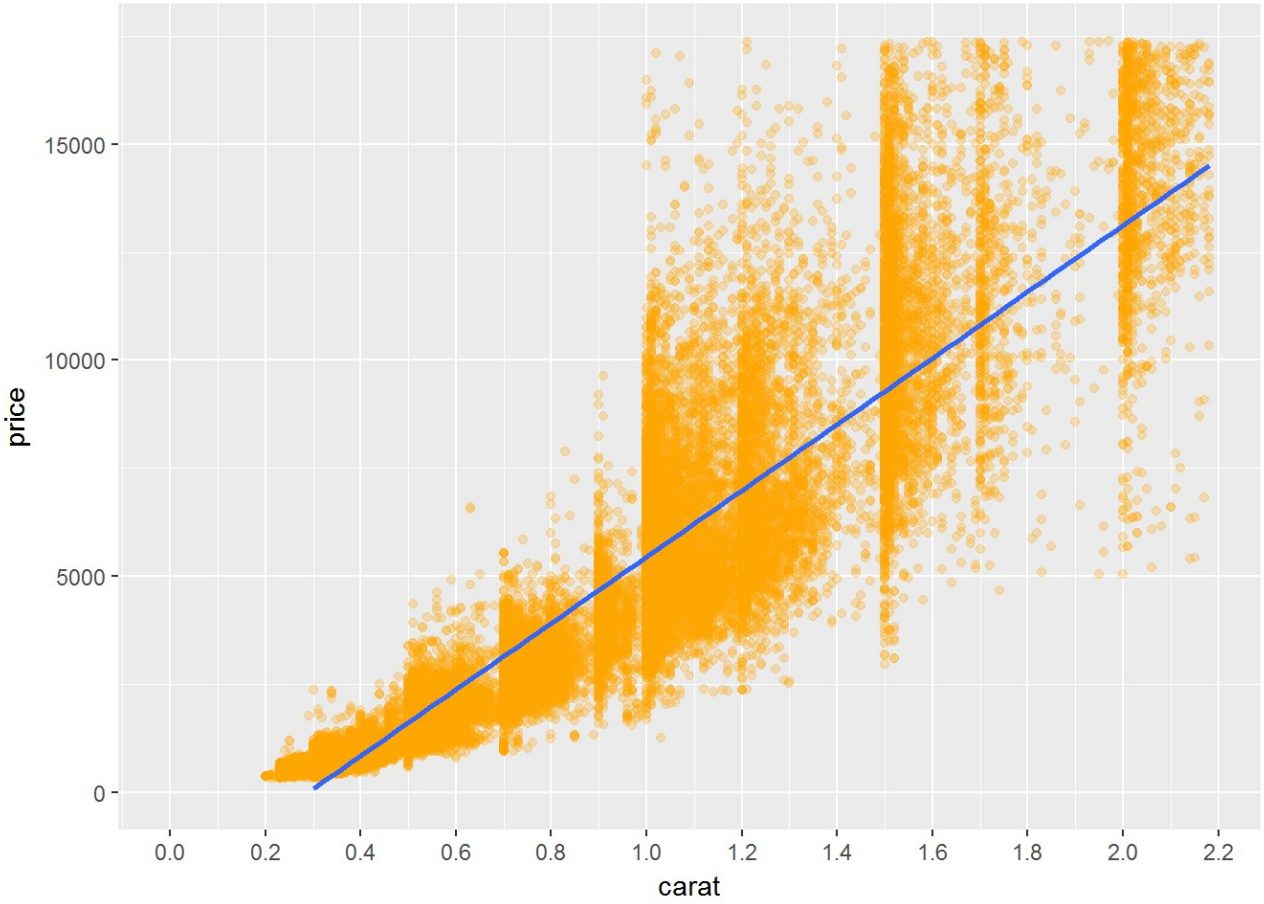
We can add a linear line to our plot.

```
ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point( color = 'orange', alpha = 1/4) +
  stat_smooth(method = 'lm') +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99)), breaks = seq(0,
2.5, 0.2)) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99)))
```

```
## 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).
```



Changing the markings on x-axis and adding our linear line, we can see that it has missed out some keyplaces in the scatterplot, such as - * The line has not covered the area where most of our data is in the beginning of the x-axis i.e. the line has not started fomr 0.2, instead it has started from 0.3 and has left out number of observations. * Between the carat values of 1.0 - 1.4, there shoud have been a curve to cover the points in our plot, but the line has left that area too. * After the carat value of 1.8 the line should have changed its slope, and the curve should have gone upwards to cover the points where price is greater than 15000 and carat is greater than 2.0

If we make predictions for our data based on this model, we will surely miss out on relevant information.

Task 2

Prepare 4 plots, each of different type: pie chart, bar chart, histogram, scatter plot. Each plot should present different information (different combinations of variables), do NOT use price variable at this time. Each plot should have a title, meaningful labels and must be commented. If possible, try to put more than 2 variables on one plot.

Pie chart

We can make some simple pie charts for our variables such as - * cut * color * clarity

This will help us to know how much of our data has a particular cut or the variety in color or the how many diamonds fall into different categories of clarity.

```
install.packages("plotly", dependencies = T, repos = "http://cran.us.r-project.or
g")
```

```
## package 'plotly' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 3.3.2
```

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

```
## 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(dplyr)
```

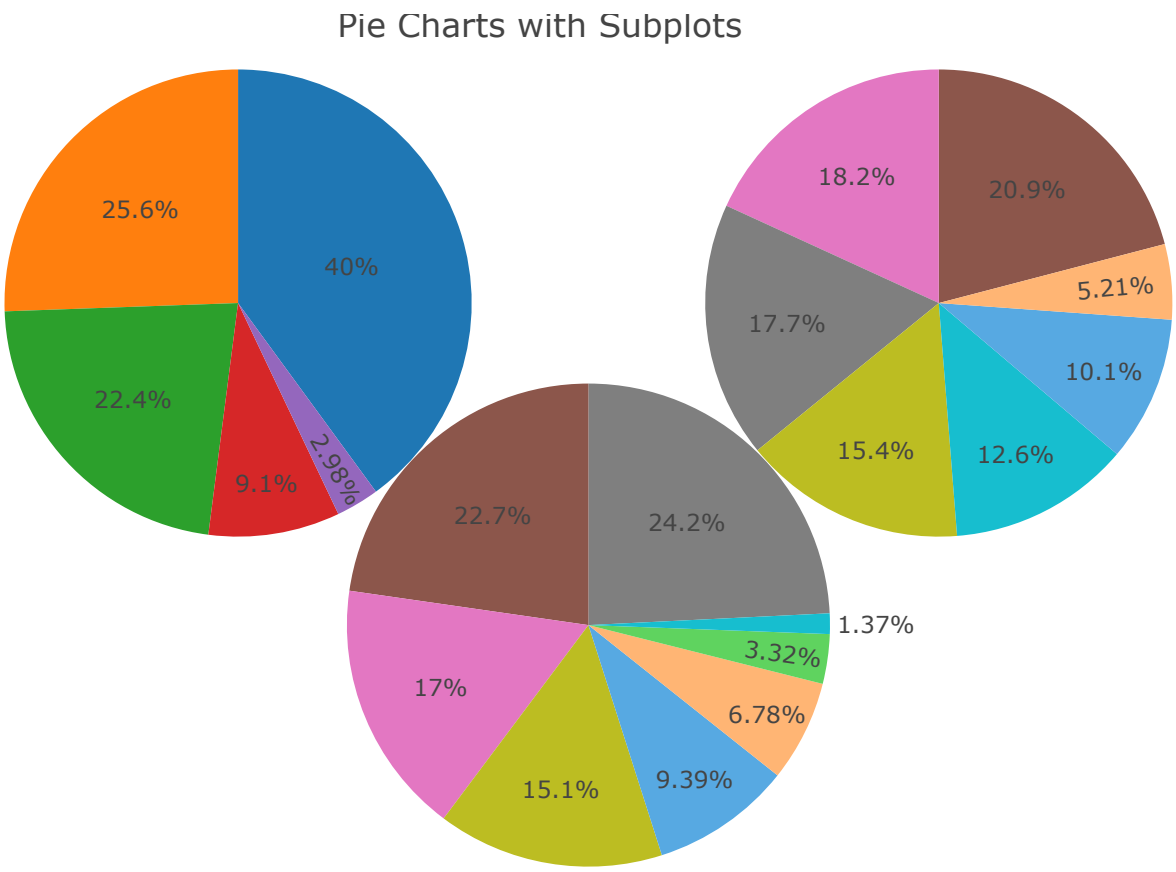
```
## Warning: package 'dplyr' was built under R version 3.3.2
```

```
##
## 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
```

```
plot_ly() %>%
  add_pie(data = count(diamonds, cut), labels = ~cut, values = ~n,
          name = "Cut", domain = list(x = c(0, 0.4), y = c(0.4, 1))) %>%
  add_pie(data = count(diamonds, color), labels = ~cut, values = ~n,
          name = "Color", domain = list(x = c(0.6, 1), y = c(0.4, 1))) %>%
  add_pie(data = count(diamonds, clarity), labels = ~cut, values = ~n,
          name = "Clarity", domain = list(x = c(0.25, 0.75), y = c(0, 0.6))) %>%
  layout(title = "Pie Charts with Subplots", showlegend = F,
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))
```



- First let's observe the pie chart for cut. We can see that 40% of the diamonds in our data set have an ideal cut, and the rest 60% remaining diamonds are divided into 4 categories which means that most of the diamonds in our data set have an ideal cut.
- Second, if we look at our second pie chart that represents color, we can see that the data set is somewhat divided and there is no one color that the diamonds in our data set belong to.
- Third, the last pie chart is of clarity. We can see that around 50 percent ie close to half of the diamonds in our data set belong only to two categories of clarity. This we can also see by summarizing clarity

```
summary(diamonds$clarity)
```

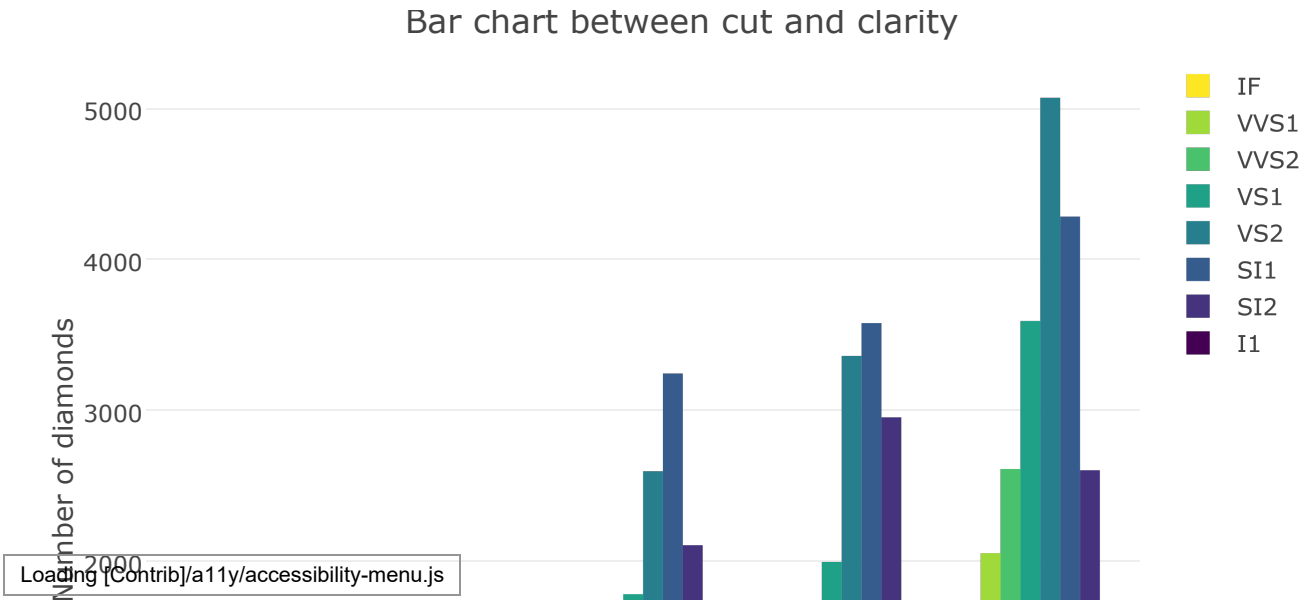
```
##    I1   SI2   SI1   VS2   VS1  VVS2  VVS1    IF
##   741  9194 13065 12258  8171  5066  3655  1790
```

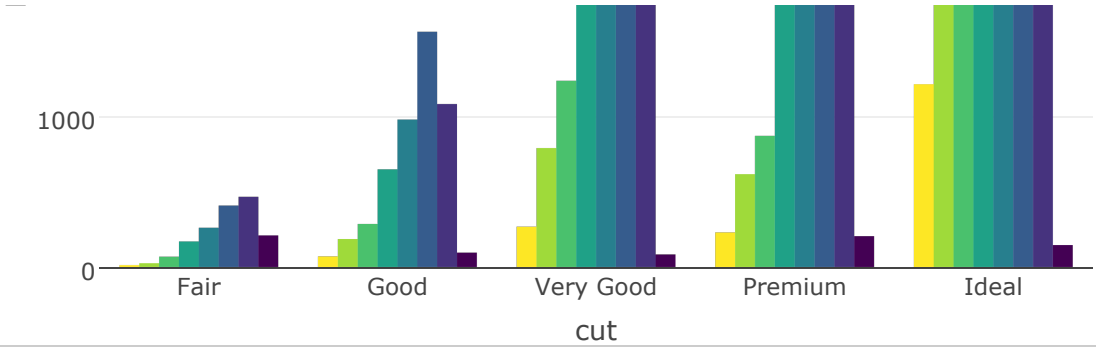
As we can see that the categories SI1 and VS2 have majority of our data thus the pie chart represents these two categories as 24.2% and 22.7% respectively.

Bar charts

```
ggplot2::diamonds %>%
  count(cut,clarity) %>%
  plot_ly(x = ~cut, y = ~n, color = ~clarity) %>%
  layout(title = 'Bar chart between cut and clarity',
         yaxis = list(title = 'Number of diamonds'))
```

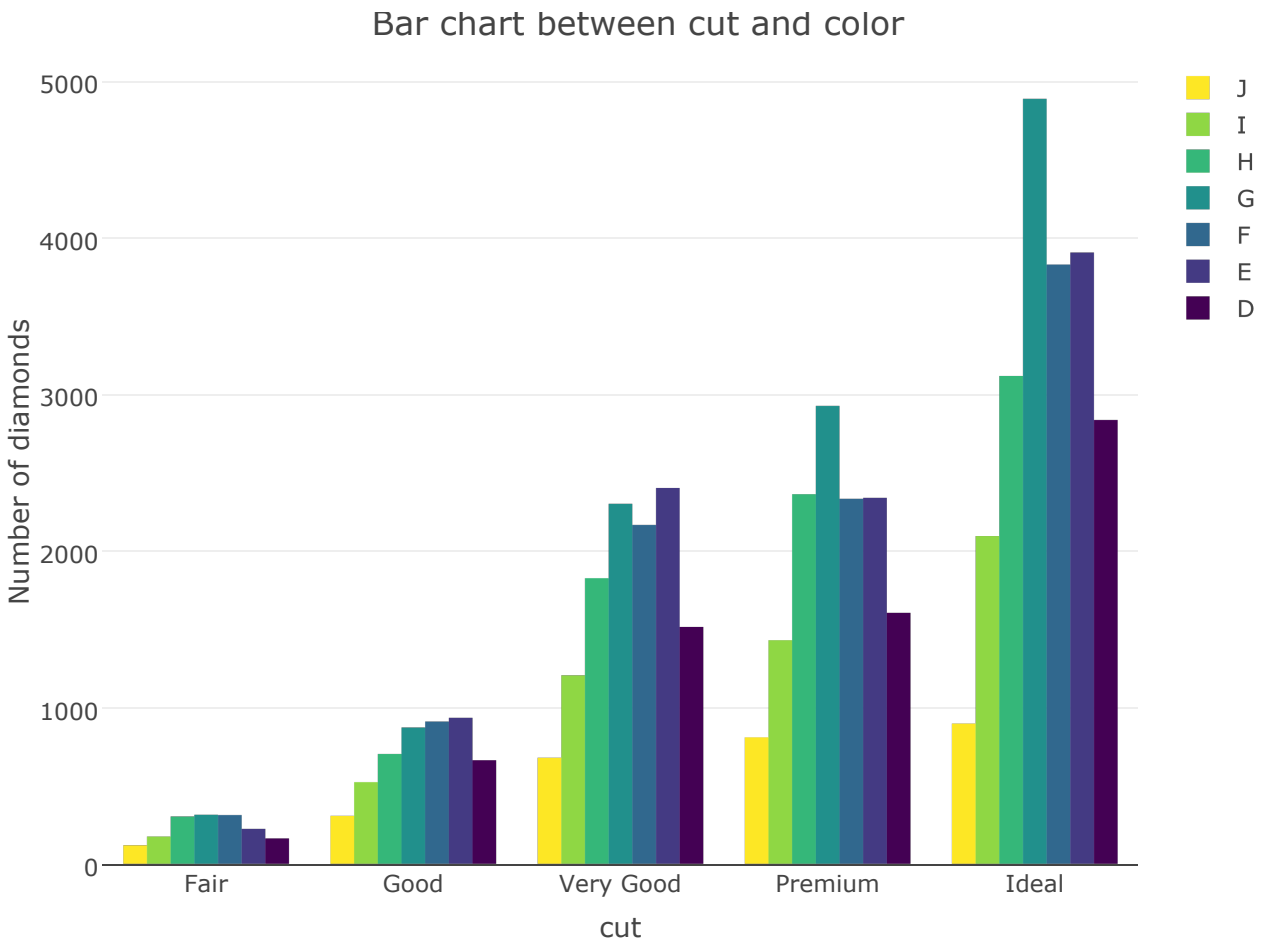
```
## No trace type specified:
##   Based on info supplied, a 'bar' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#bar
```






```
ggplot2::diamonds %>%
  count(cut,color) %>%
  plot_ly(x = ~cut, y = ~n, color = ~color) %>%
  layout(title = 'Bar chart between cut and color',
         yaxis = list(title = 'Number of diamonds'))
```

```
## No trace type specified:
##   Based on info supplied, a 'bar' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#bar
```

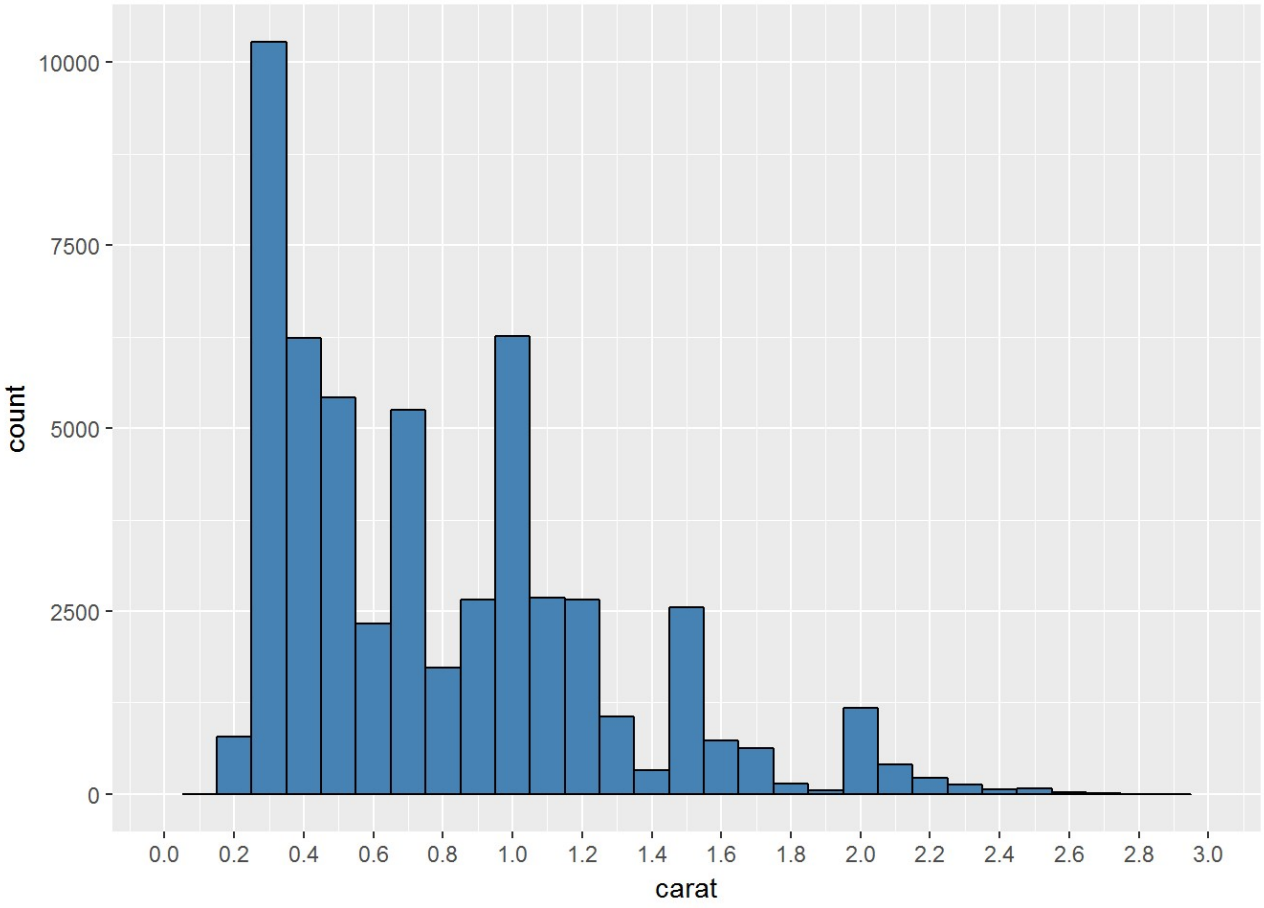


These two bar graphs that we have just plotted shows us the number of diamonds grouped by quantitative - quantitative variables. The first bar graph shows a plot between Diamonds accordint to cut and clarity. The second bar graph shows diamonds according to cur and color. Hover over these bars to get the values.

Histograms

```
ggplot(aes(x = carat), data = diamonds) +
  geom_histogram(binwidth = 0.1,
                 color = 'black',
                 fill = 'steelblue') +
  scale_x_continuous(limits = c(0,3), breaks = seq(0, 3, 0.2))
```

```
## Warning: Removed 32 rows containing non-finite values (stat_bin).
```

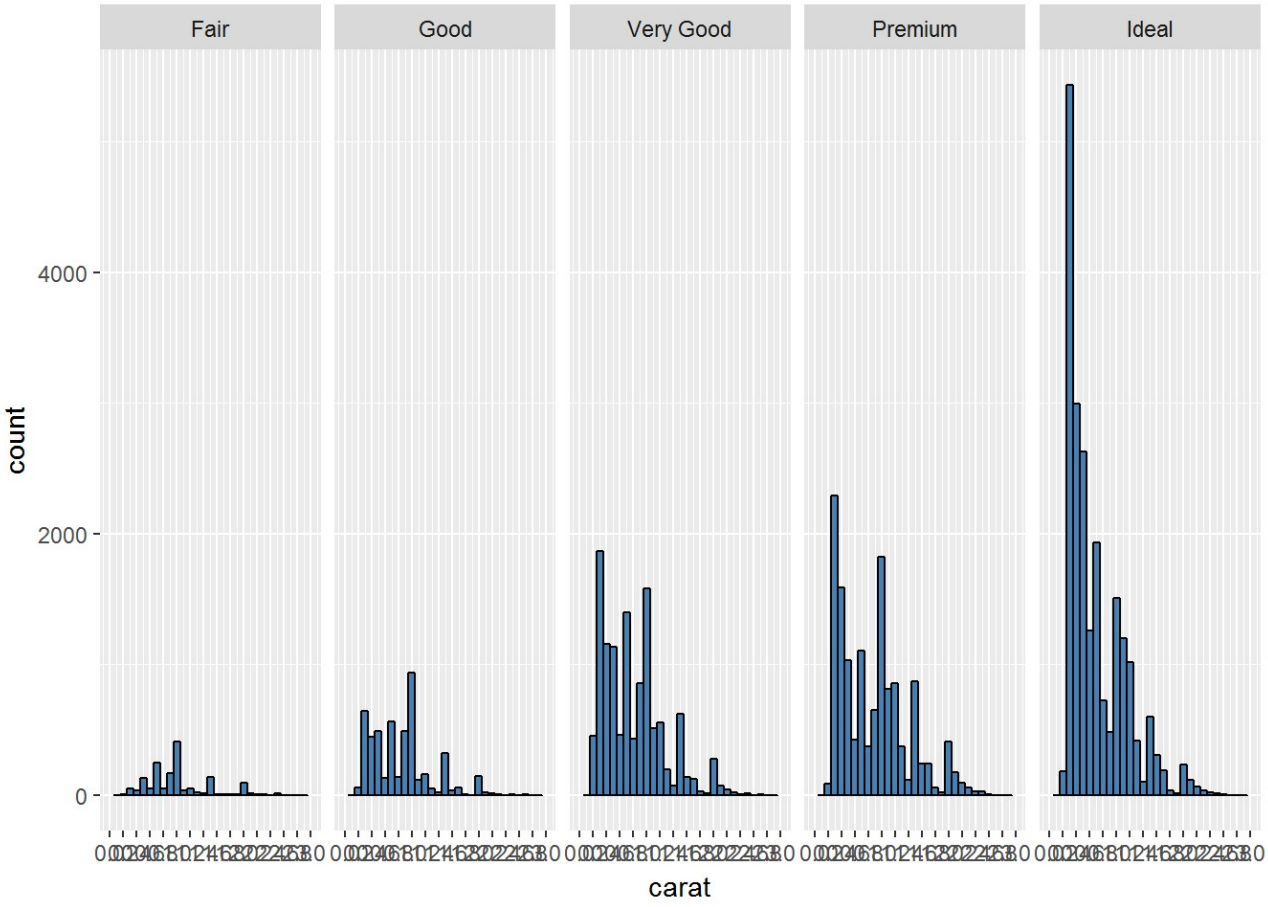


The histogram we plotted is a simple histogram giving us the frequency of diamonds against carats.

Now we can one more variable in this histogram just by making a small change

```
ggplot(aes(x = carat), data = diamonds) +
  geom_histogram(binwidth = 0.1,
                 color = 'black',
                 fill = 'steelblue') +
  scale_x_continuous(limits = c(0,3), breaks = seq(0, 3, 0.2)) +
  facet_wrap(~cut, ncol = 5)
```

```
## Warning: Removed 32 rows containing non-finite values (stat_bin).
```

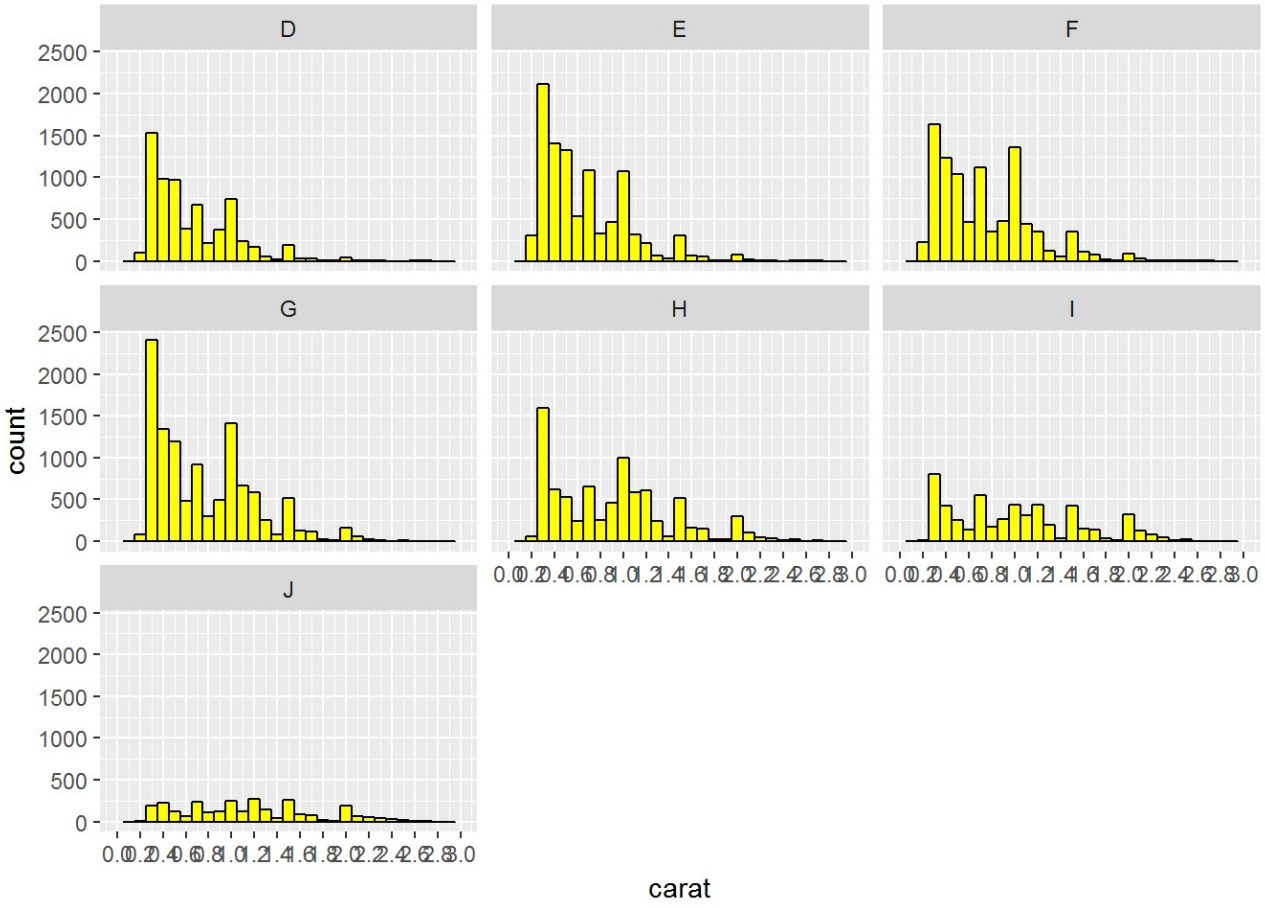


We used the facet wrap and divided our one histogram into several other histograms based on the categories in cut variable.

We can do the same with our other two categorical variables i.e. color and clarity.

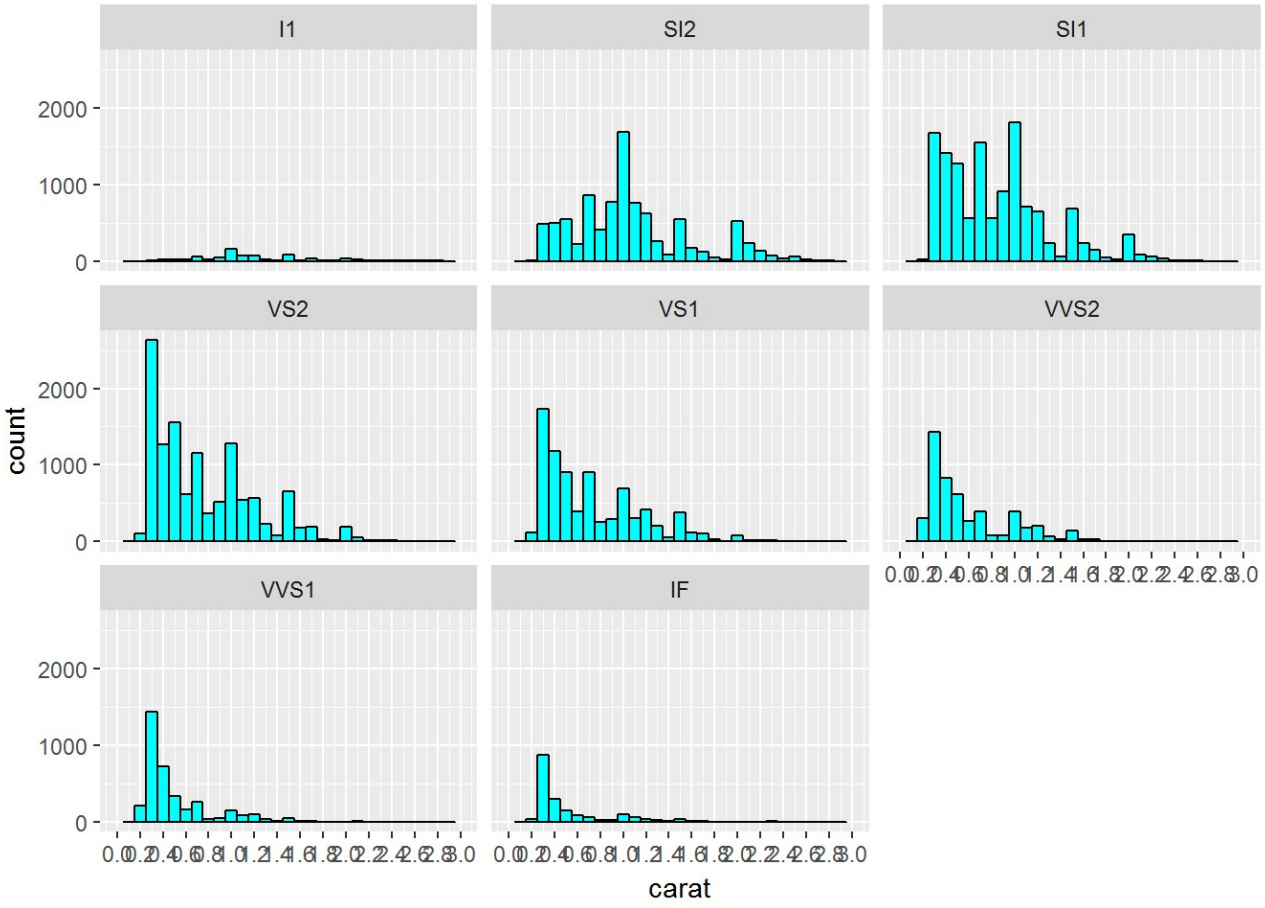
```
ggplot(aes(x = carat), data = diamonds) +
  geom_histogram(binwidth = 0.1,
                 color = 'black',
                 fill = 'yellow') +
  scale_x_continuous(limits = c(0,3), breaks = seq(0, 3, 0.2)) +
  facet_wrap(~color)
```

```
## Warning: Removed 32 rows containing non-finite values (stat_bin).
```




```
ggplot(aes(x = carat), data = diamonds) +
  geom_histogram(binwidth = 0.1,
                 color = 'black',
                 fill = 'cyan') +
  scale_x_continuous(limits = c(0,3), breaks = seq(0, 3, 0.2)) +
  facet_wrap(~clarity)
```

```
## Warning: Removed 32 rows containing non-finite values (stat_bin).
```



Note: The findings from these histograms closely relate to what we got from Bar charts and Pie Charts

---

Scatterplot

```
install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## package 'gridExtra' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

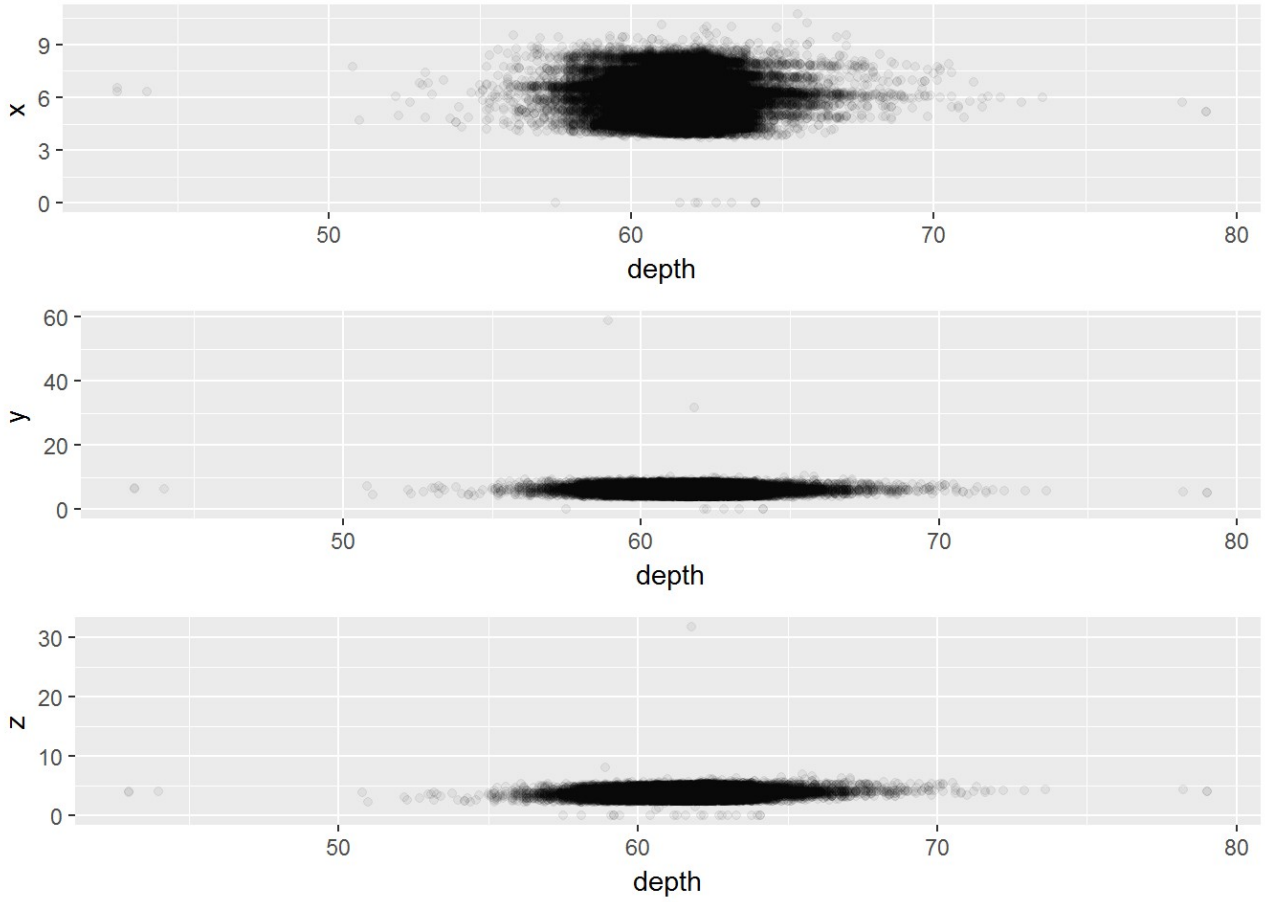
```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.3.2
```

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

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

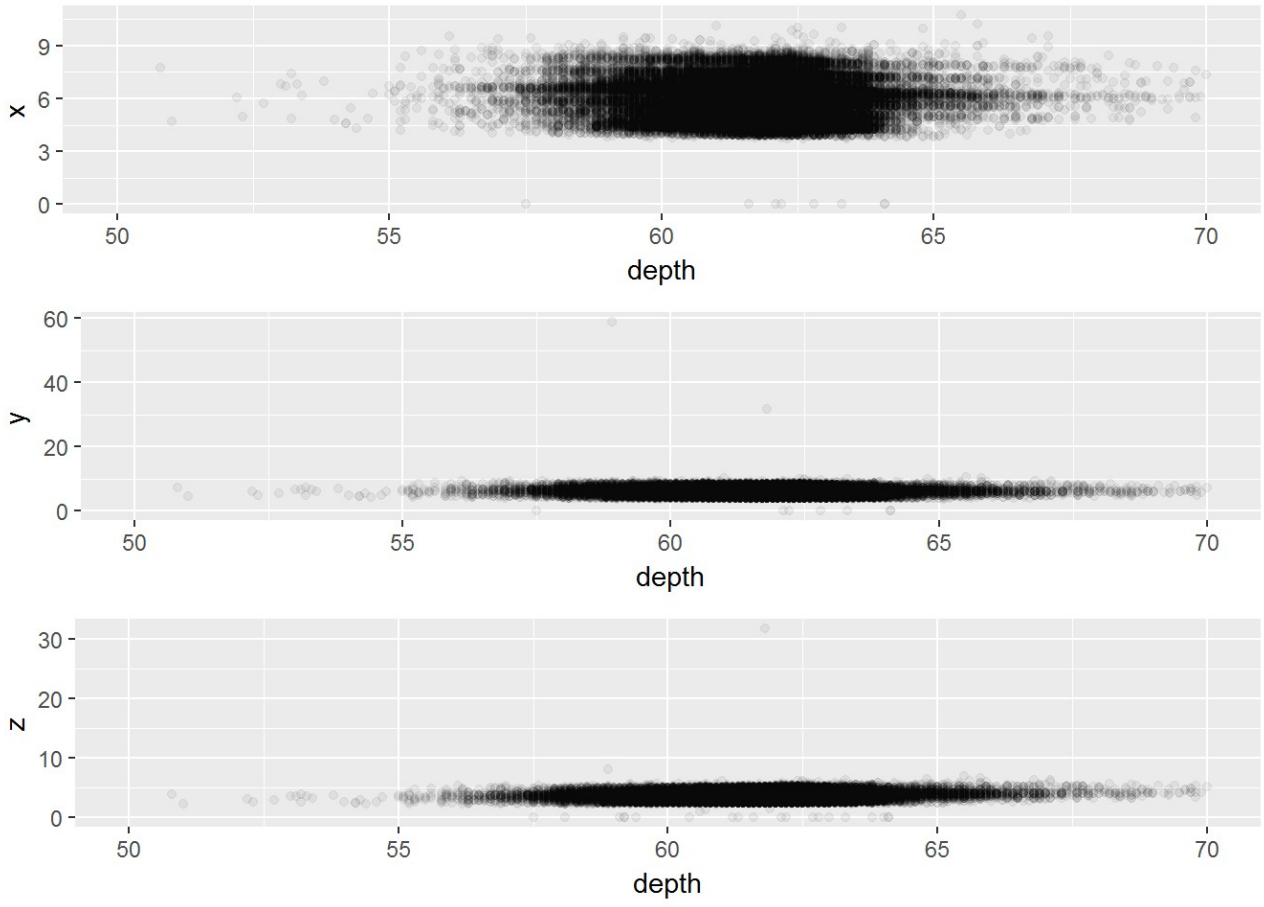
```
p1 <- ggplot(aes(x = depth, y = x), data = diamonds) +
  geom_point(alpha = 1/20)

p2 <- ggplot(aes(x = depth, y = y), data = diamonds) +
  geom_point(alpha = 1/20)

p3 <- ggplot(aes(x = depth, y = z), data = diamonds) +
  geom_point(alpha = 1/20)

grid.arrange(p1,p2,p3, ncol = 1)
```



The scatterplot we have just made shows us the relationship between depth and dimensions of a diamond i.e. x, y and z. An interesting thing to notice is that the values of x, y and z vary in the graphs but the values of depth lie between 50 to 70. We can remove other values by limting our x-axis in each graph.

```
p1 <- ggplot(aes(x = depth, y = x), data = diamonds) +
  geom_point(alpha = 1/20) +
  scale_x_continuous(limits = c(50,70), breaks = seq(50,70,5))

p2 <- ggplot(aes(x = depth, y = y), data = diamonds) +
  geom_point(alpha = 1/20)+
  scale_x_continuous(limits = c(50,70), breaks = seq(50,70,5))

p3 <- ggplot(aes(x = depth, y = z), data = diamonds) +
  geom_point(alpha = 1/20)+
  scale_x_continuous(limits = c(50,70), breaks = seq(50,70,5))

grid.arrange(p1,p2,p3, ncol = 1)
```

```
## Warning: Removed 26 rows containing missing values (geom_point).

## Warning: Removed 26 rows containing missing values (geom_point).

## Warning: Removed 26 rows containing missing values (geom_point).
```



Now, when we talk about the relationship between these variables it is not wise to calculate the correlation between them as depth is a super set of x, y or z. So, either they may be strongly correlated or not correlated at all.

We can see this by using the cor.test() formula -

```
cor.test(diamonds$depth,diamonds$x)
```

```
##
##  Pearson's product-moment correlation
##
## data:  diamonds$depth and diamonds$x
## t = -5.8752, df = 53938, p-value = 4.248e-09
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.03372112 -0.01685378
## sample estimates:
##         cor
## -0.02528925
```

```
cor.test(diamonds$depth,diamonds$y)
```

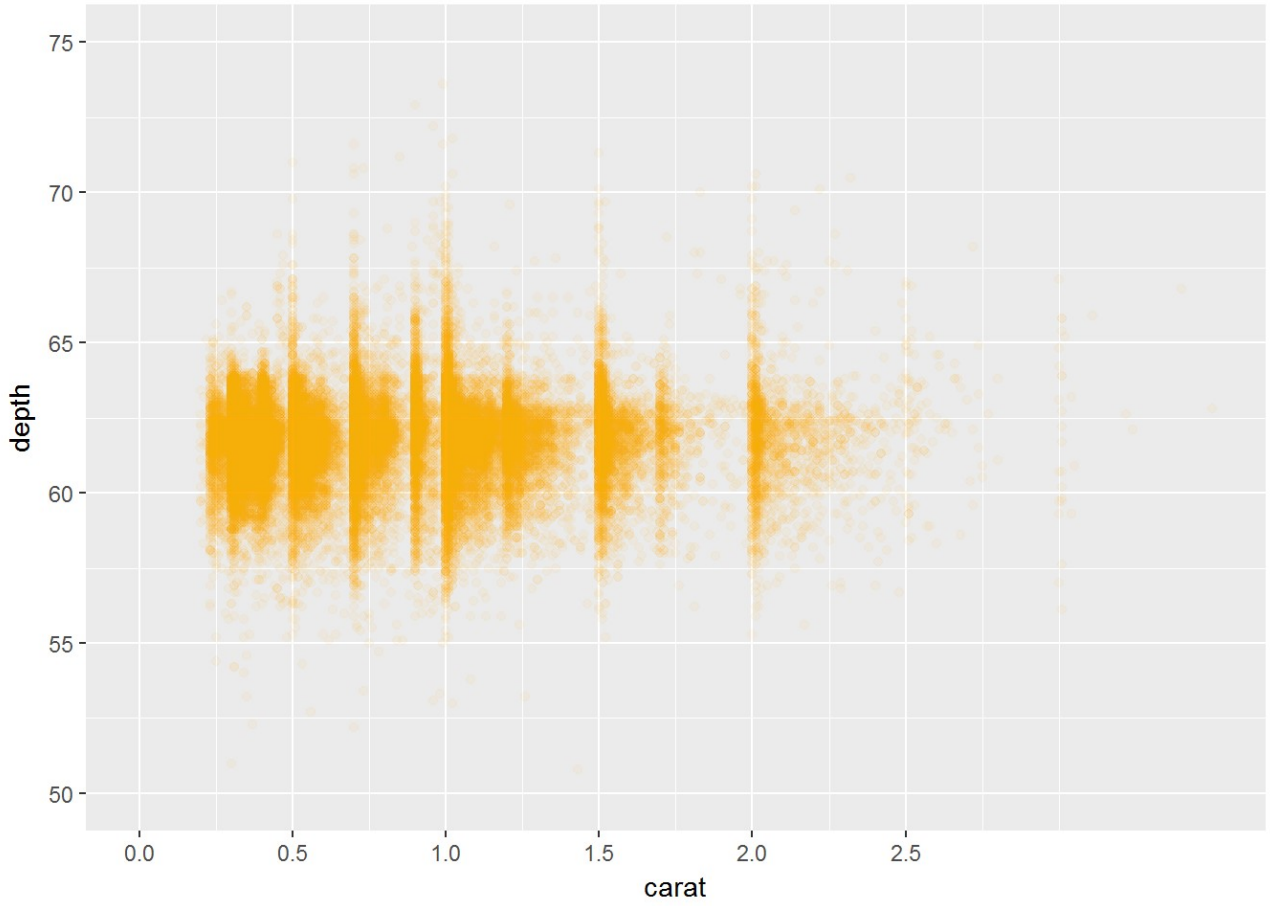
```
##
##  Pearson's product-moment correlation
##
## data:  diamonds$depth and diamonds$y
## t = -6.8172, df = 53938, p-value = 9.382e-12
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.03777038 -0.02090678
## sample estimates:
##         cor
## -0.02934067
```

```
cor.test(diamonds$depth,diamonds$z)
```

```
##
##  Pearson's product-moment correlation
##
## data:  diamonds$depth and diamonds$z
## t = 22.146, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.08655415 0.10328021
## sample estimates:
##        cor
## 0.09492388
```

Now another scatterplot for which we can talk about is between carat and depth

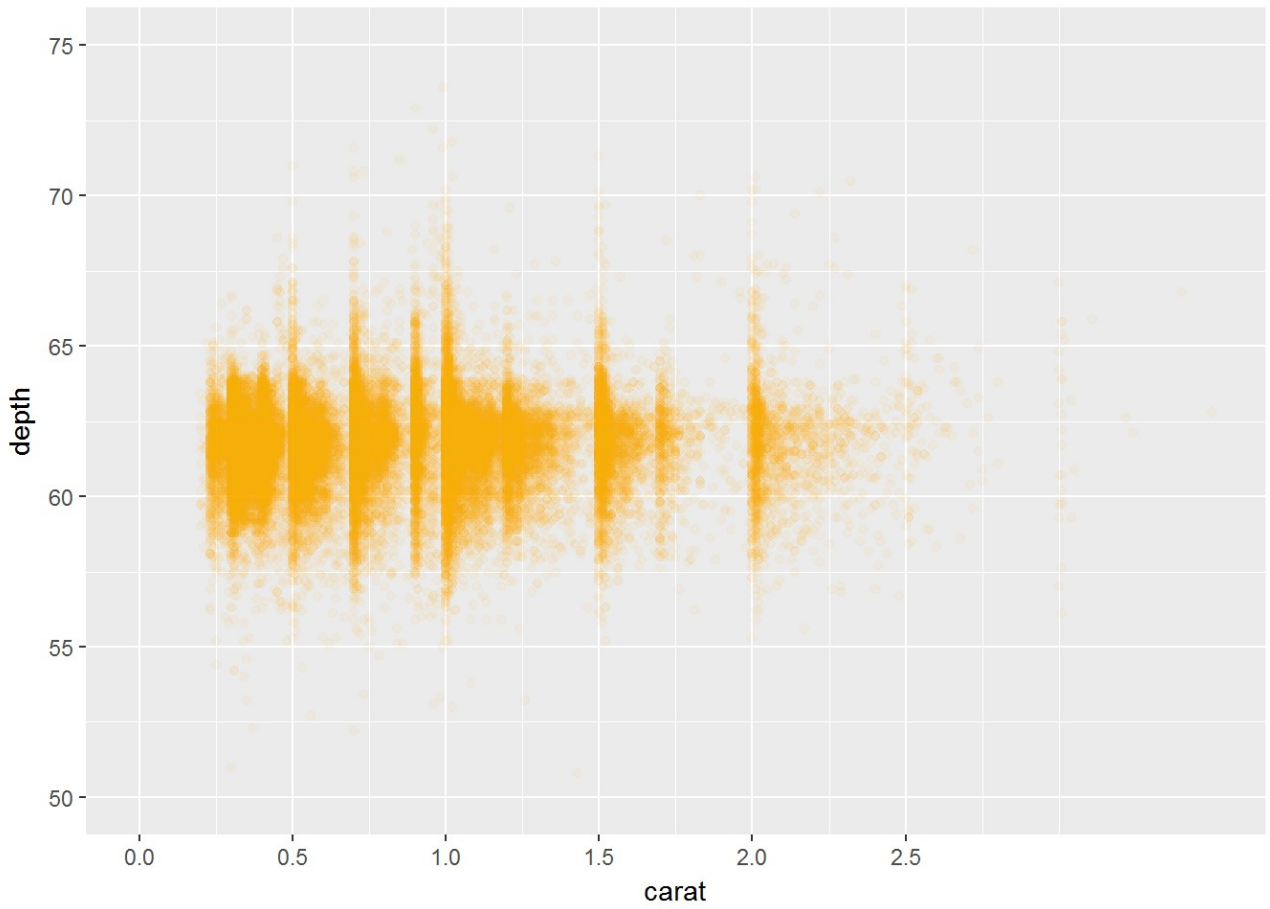
```
ggplot(aes(x = carat, y = depth), data = diamonds) +
  geom_point()
```



Again, we have to remove the outliers from our plot. Tuning the plot more, we get -

```
ggplot(aes(x = carat, y = depth), data = diamonds) +
  geom_point(alpha = 1/20,
             color = 'orange') +
  scale_x_continuous(limits = c(0,3.5), breaks = seq(0,2.5, 0.5)) +
  scale_y_continuous(limits = c(50,75))
```

```
## Warning: Removed 15 rows containing missing values (geom_point).
```

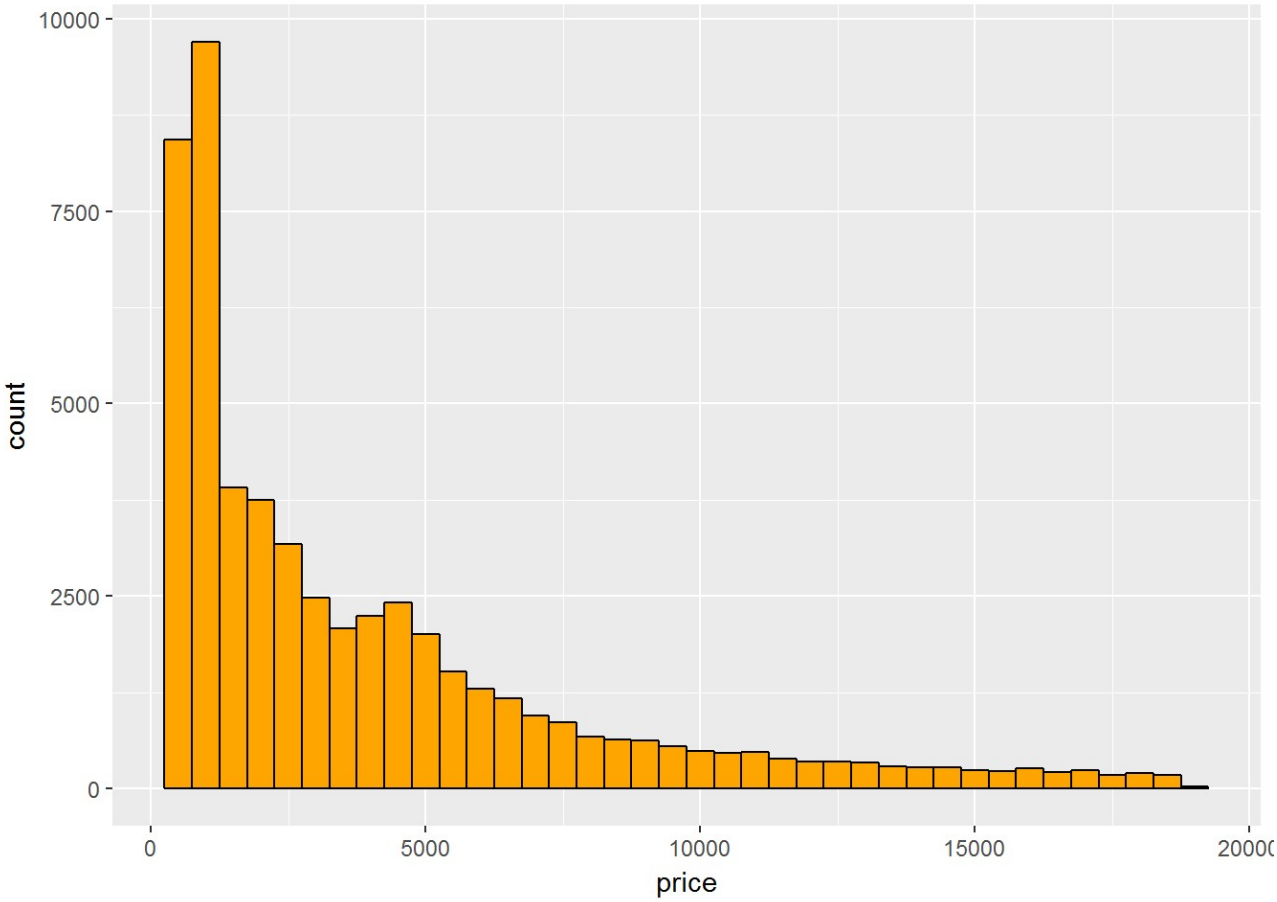


Now we can see that the scatterplot has higher density at certain areas as comapred to others. This would be true because the carat of a diamond is mostly measures as 0.5, 1.0, 1.5. Therefore, we see the spikes in our plot.

Task 3

Focus your analysis on the price variable: (a) Show the histogram of the price variable. Describe it. Include summary statistics like mean or median.

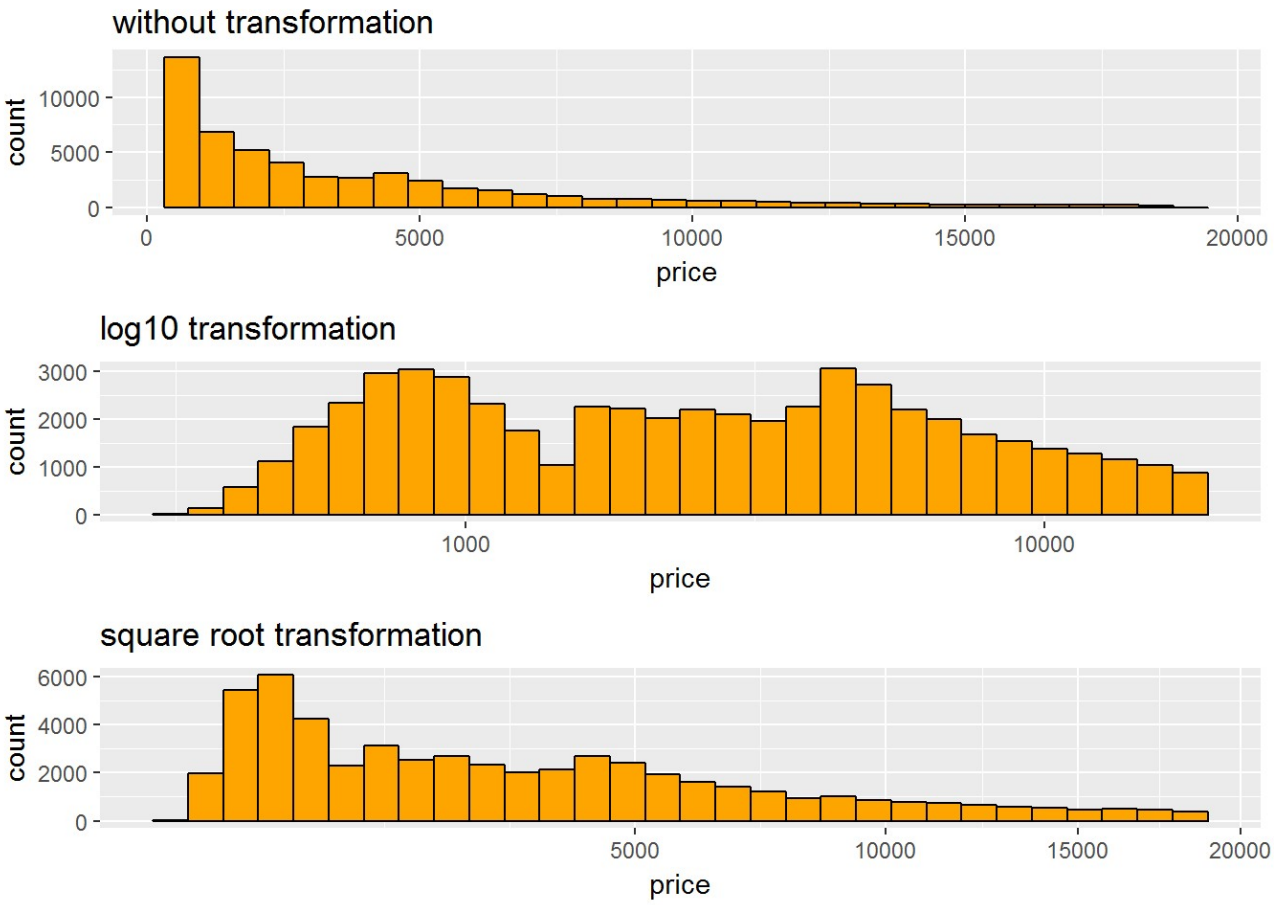
```
ggplot(aes(price), data = diamonds) +
  geom_histogram(color = 'black', fill = 'orange', binwidth = 500)
```



The shape tells us that the number of diamonds having a price range of 0 to 5000 is way more than the price greater than 5000. Also we see a spike at around 1000 to 2000. But we do not know the distribution of our data, for this we have to transform our data.

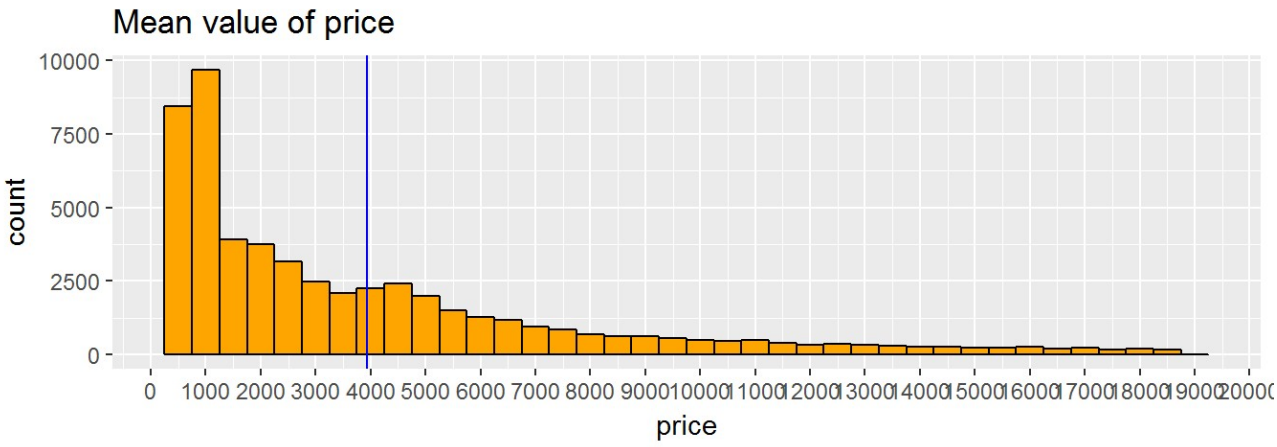
```
p1 <- ggplot(aes(price), data = diamonds) +
  geom_histogram(color = 'black', fill = 'orange') +
  scale_x_continuous() +
  labs(title = 'without transformation')

p2 <- ggplot(aes(price), data = diamonds) +
  geom_histogram(color = 'black', fill = 'orange') +
  scale_x_log10()+
  labs(title = 'log10 transformation')

p3 <- ggplot(aes(price), data = diamonds) +
  geom_histogram(color = 'black', fill = 'orange') +
  scale_x_sqrt() +
  labs(title = 'square root transformation')

grid.arrange(p1, p2, p3, ncol = 1)
```

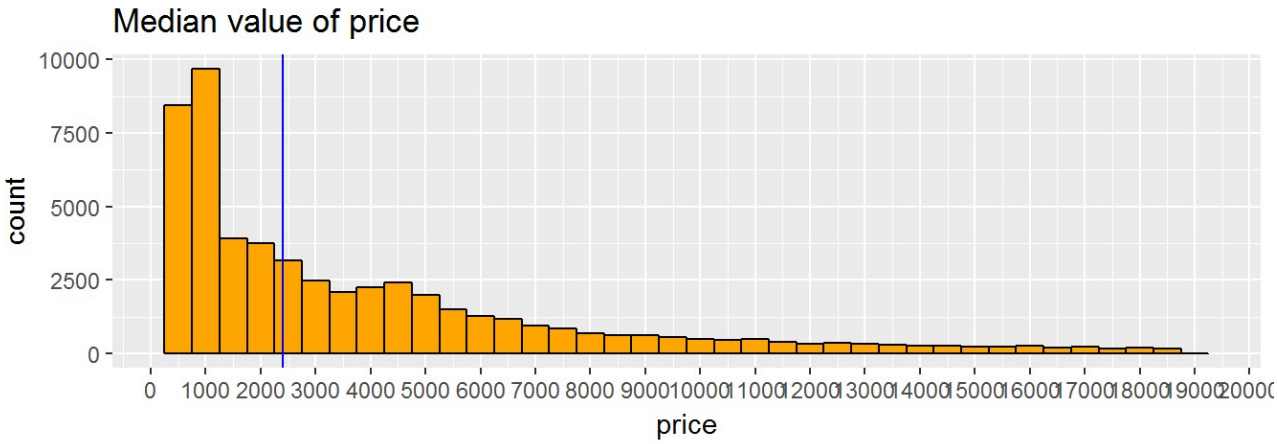
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



After transforming our data, we can see that the second histogram has a curvature somewhat related to normal distribution but still we do not have an exact idea aboout the model. We can include mean and median as follows :

```
p1 <- ggplot(aes(price), data = diamonds) +
  geom_histogram(color = 'black', fill = 'orange', binwidth = 500) +
  scale_x_continuous(breaks = seq(0,20000, 1000)) +
  geom_vline(xintercept = mean(diamonds$price), color = 'blue') +
  labs(title = 'Mean value of price')

p2 <- ggplot(aes(price), data = diamonds) +
  geom_histogram(color = 'black', fill = 'orange', binwidth = 500) +
  scale_x_continuous(breaks = seq(0,20000, 1000)) +
  geom_vline(xintercept = median(diamonds$price), color = 'blue')+
  labs(title = 'Median value of price')

grid.arrange(p1, p2, ncol = 1)
```





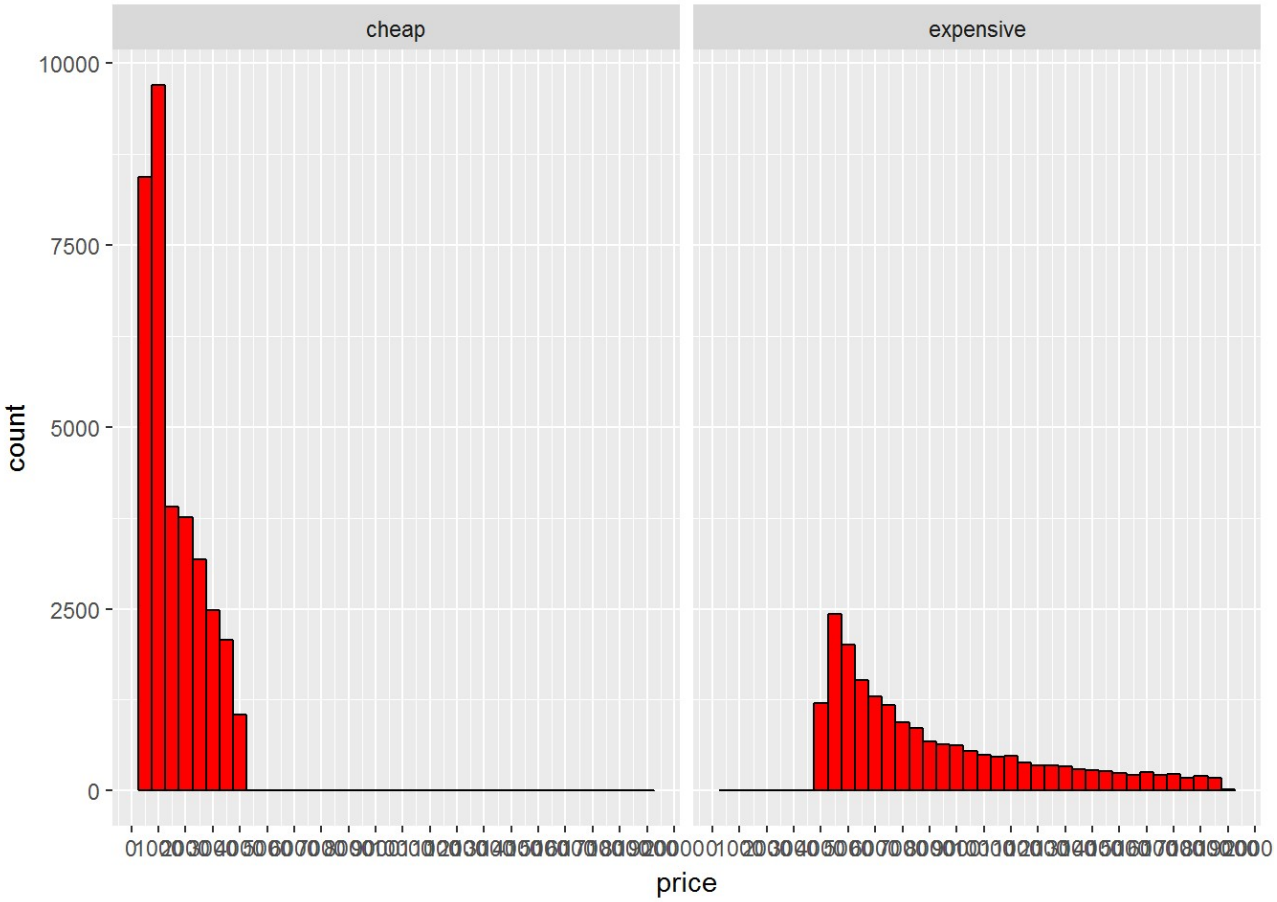
b. Group diamonds by some price ranges (like cheap, expensive, etc.) and summarise those groups separately.

Here, what we can do, is to create another variable in our data set called `category` and categorize it as follows: * cheap: if the price of diamond is less than $4000. * expensive: if the price is greater than $4000

```
category <- NA
diamonds$category <- ifelse(diamonds$price > 4000, 1, 0)
diamonds$category <- factor(diamonds$category)
levels(diamonds$category) <- c("cheap", "expensive")
summary(diamonds$category)
```

```
##     cheap expensive
##     34561     19379
```

```
ggplot(aes(price), data = diamonds) +
  geom_histogram(color = 'black', fill = 'red', binwidth = 500)  +
  scale_x_continuous(breaks = seq(0,20000, 1000)) +
  facet_wrap(~diamonds$category)
```

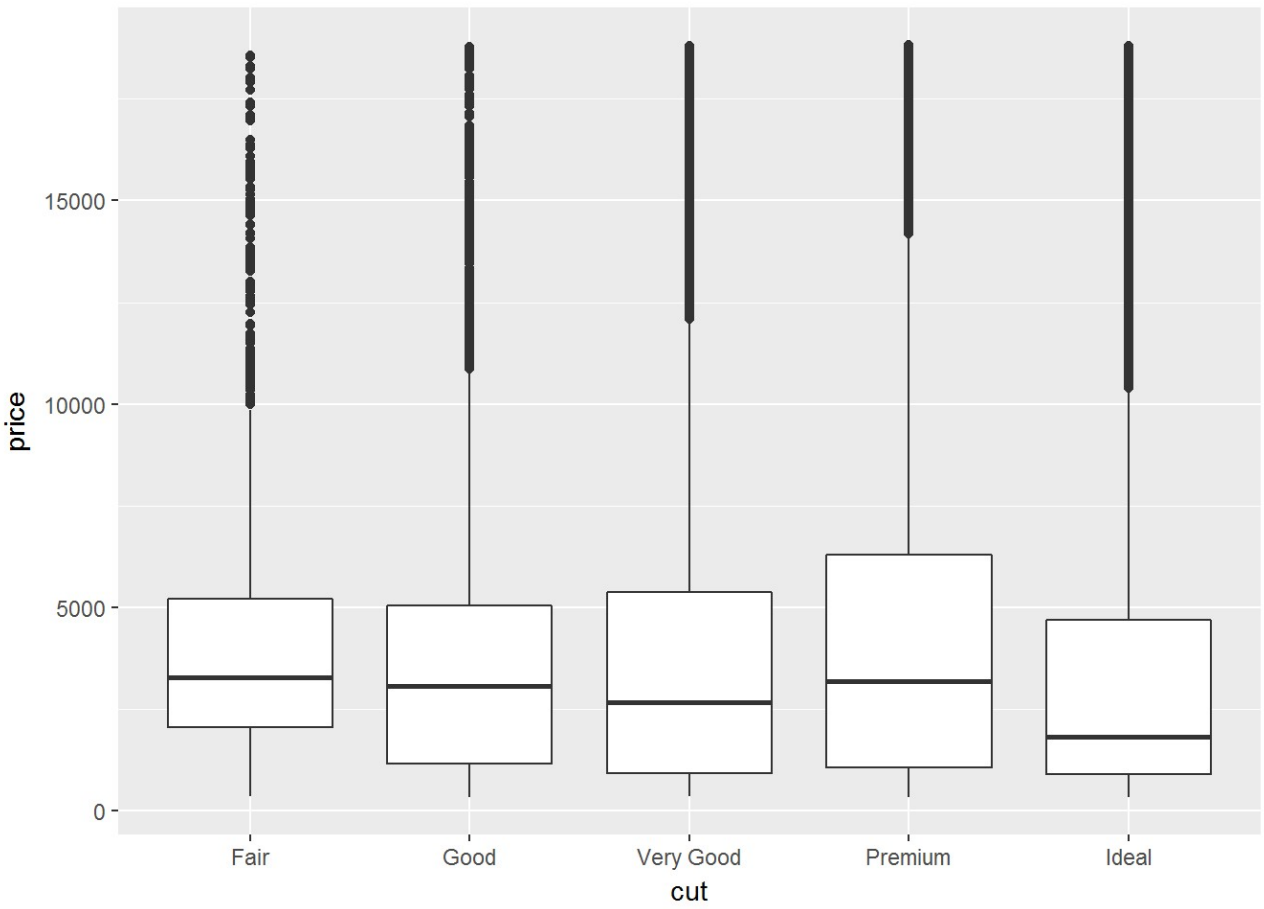


Thus, we can see that by categorizing our diamonds and faceting them, the histogram is divided into 2, one showing plot for cheap diamonds whose price is less than $4000 and other expensive diamonds whose price is greater than or equal to $4000.

---

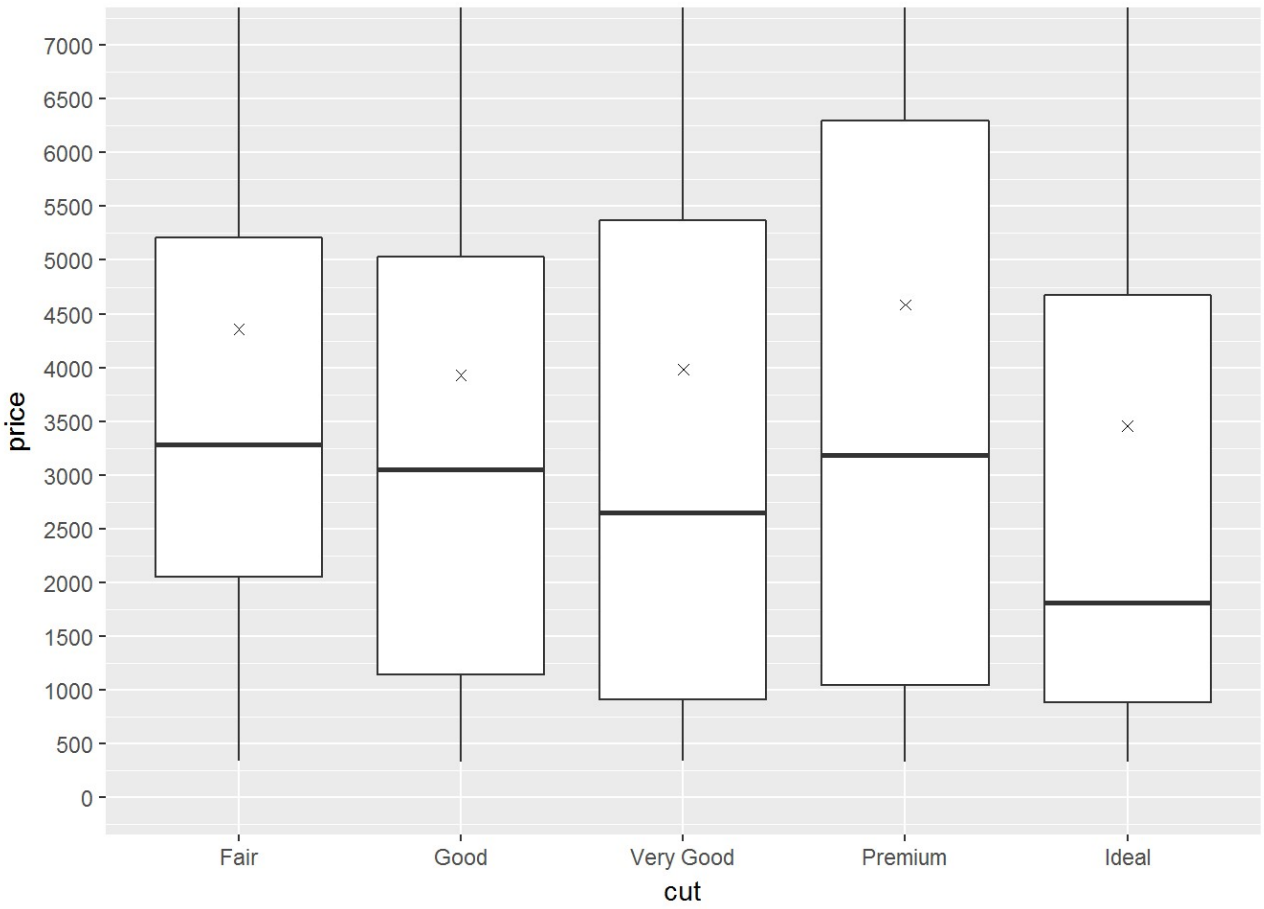
c. Explore prices for different cut types. You might want to use the boxplot.

```
ggplot(aes(y = price, x = cut), data = diamonds) +
  geom_boxplot()
```



Now we have plotted our boxplot for price against cut. As we can see that the data is long tailed, we have to tune our y-axis to remove it.

```
ggplot(aes(y = price, x = cut), data = diamonds) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0,7000)) +
  stat_summary(fun.y = mean, geom = 'point', shape = 4) +
  scale_y_continuous( breaks = seq(0, 7000, 500))
```



We have tuned our boxplot by removing the long tailed data. Also we have plotted the mean price of every cut in the box plot using a cross. We can cross reference this using summary on this data

```
by(diamonds$price, diamonds$cut, summary)
```

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

We can take the case of `fair` cut: * The first quartile is: 2050 * The third quartile is: 5206 * The second quartile (median) is: 3282 * The mean is: 4359

Now in the boxplot, we can see that for `fair` cut, the box lies between the first quartile and the third quartile. Also the median is drawn in the boxplot.

From this we can inference that 75% of diamonds having a fair cut have price $5206 or less. We can have these inferences for other cuts as well.

---

d. How different attributes are correlated with the price? Which 2 are correlated the most?

For this task, we can simply create a correlation matrix and make our observations from that matrix

```
install.packages('GGally', repos = "http://cran.us.r-project.org")
```

```
##
##   There is a binary version available but the source version is
##   later:
##        binary source needs_compilation
## GGally  1.2.0  1.3.0             FALSE
```

```
## installing the source package 'GGally'
```

```
install.packages('scales', repos = "http://cran.us.r-project.org", dependencies =
T)
```

```
## package 'scales' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'scales'
```

```
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
install.packages('memisc', repos = "http://cran.us.r-project.org")
```

```
## package 'memisc' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
install.packages('lattice', repos = "http://cran.us.r-project.org")
```

```
## package 'lattice' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
install.packages('MASS', repos = "http://cran.us.r-project.org")
```

```
## package 'MASS' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
install.packages('car', repos = "http://cran.us.r-project.org")
```

```
## package 'car' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
install.packages('reshape', repos = "http://cran.us.r-project.org")
```

```
## package 'reshape' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
install.packages('plyr', repos = "http://cran.us.r-project.org")
```

```
## package 'plyr' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'plyr'
```

```
##
## The downloaded binary packages are in
##  C:\Users\Pc\AppData\Local\Temp\RtmpqKRql4\downloaded_packages
```

```
library(ggplot2)
library(GGally)
```

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

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

```
#library(scales)
library(memisc)
```

```
## Warning: package 'memisc' was built under R version 3.3.2
```

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

```
## Warning: package 'lattice' was built under R version 3.3.2
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.3.2
```

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

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

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

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

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

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

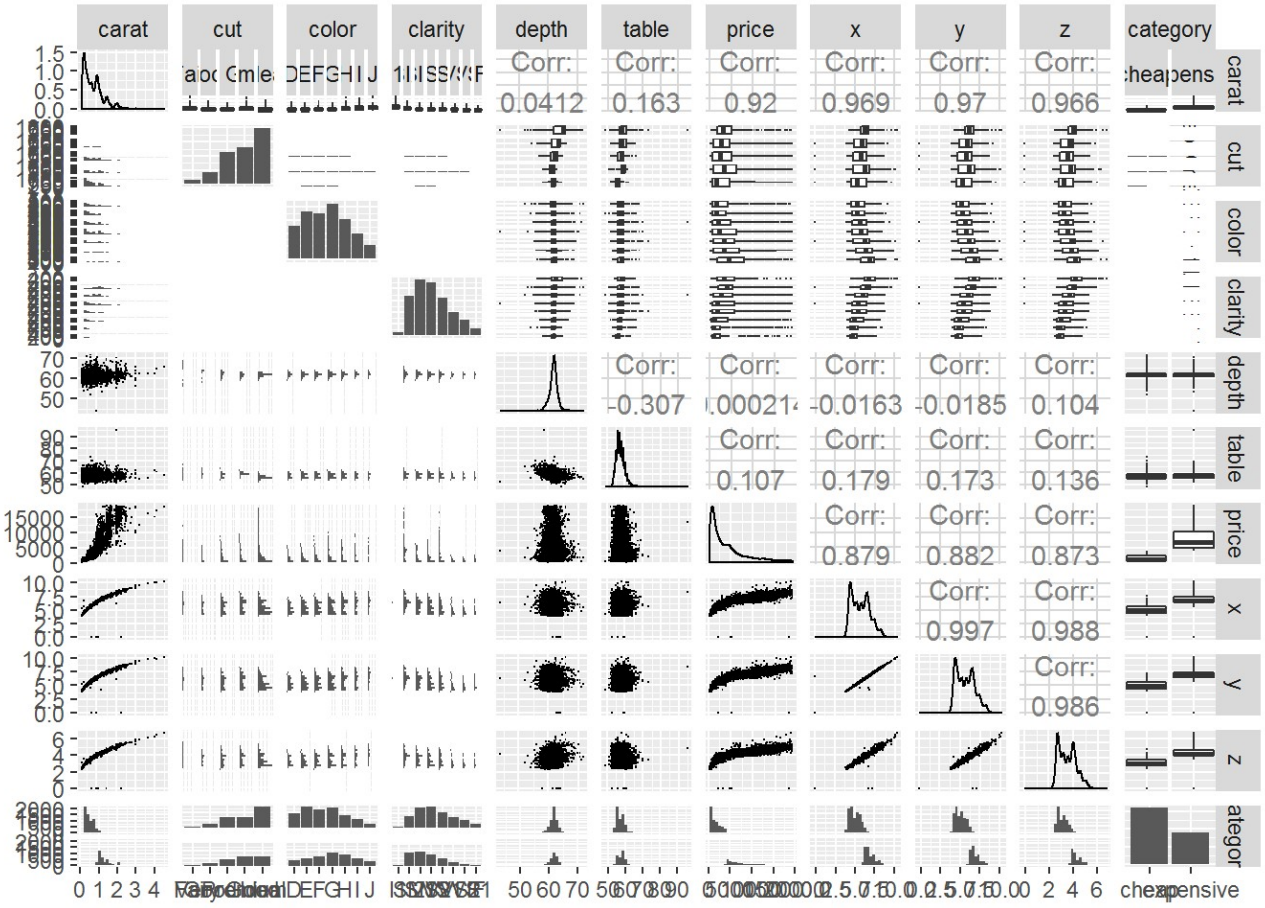
```
## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts
```

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

```
set.seed(20082014)
diamond_samp <- diamonds[sample(1:length(diamonds$price), 10000), ]
ggpairs(diamond_samp,
  lower = list(continuous = wrap("points", shape = I('.'))),
  upper = list(combo = wrap("box", outlier.shape = I('.'))))
```

```
## `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`.
## `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`.
```

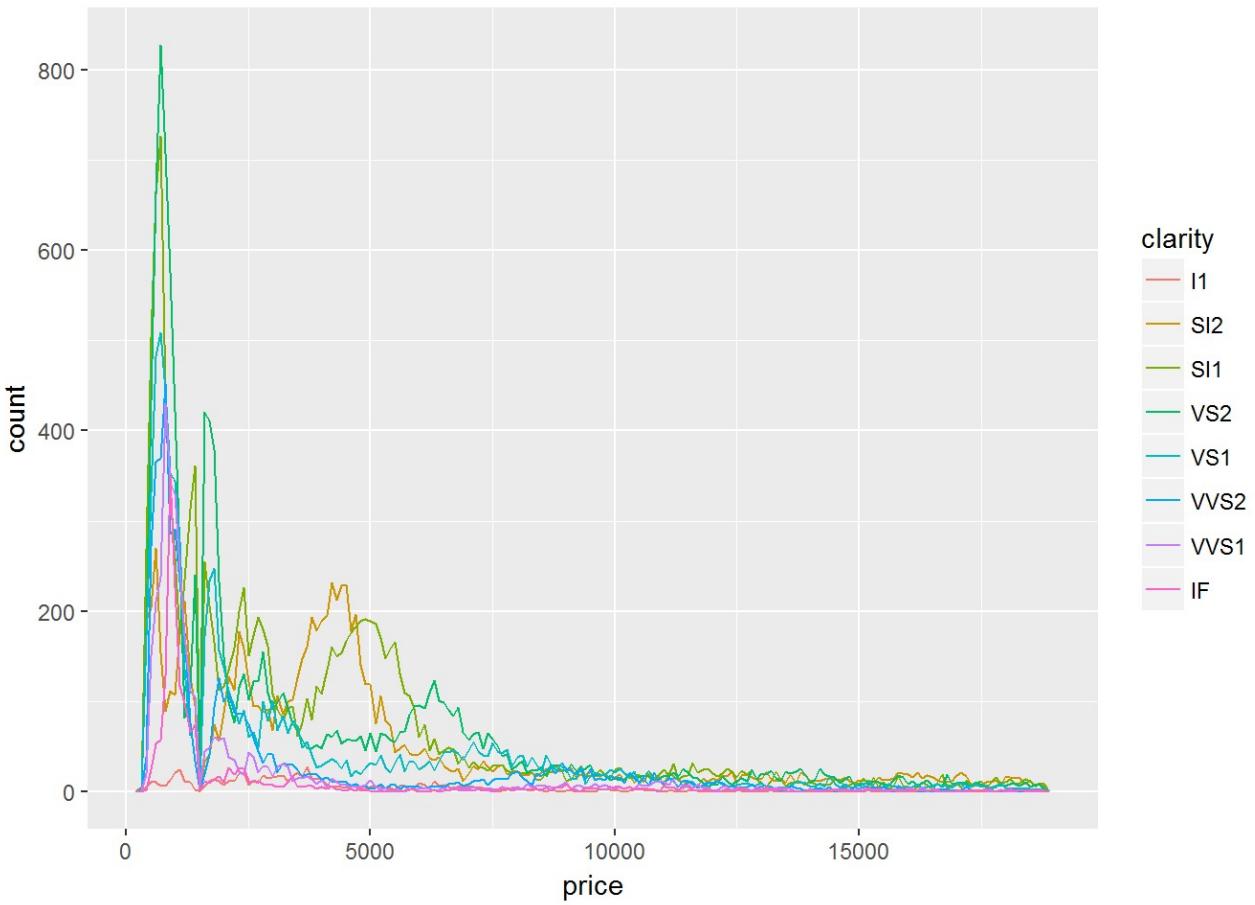


From this matrix we can see the correlation between price and other variables - * Correlation between price and carat: 0.92 * Correlation between price and cut, color & clarity: boxplots as we are comparing quantitative variable with qualitative variable * Correlation between price and depth: 0.000214 which means that they are not correlated * Correlation between price and table: 0.107 which means that they are not correlated * Correlation between price and x y z: is a scatterplot as we are comparing quantitative variable with quantitative variable.

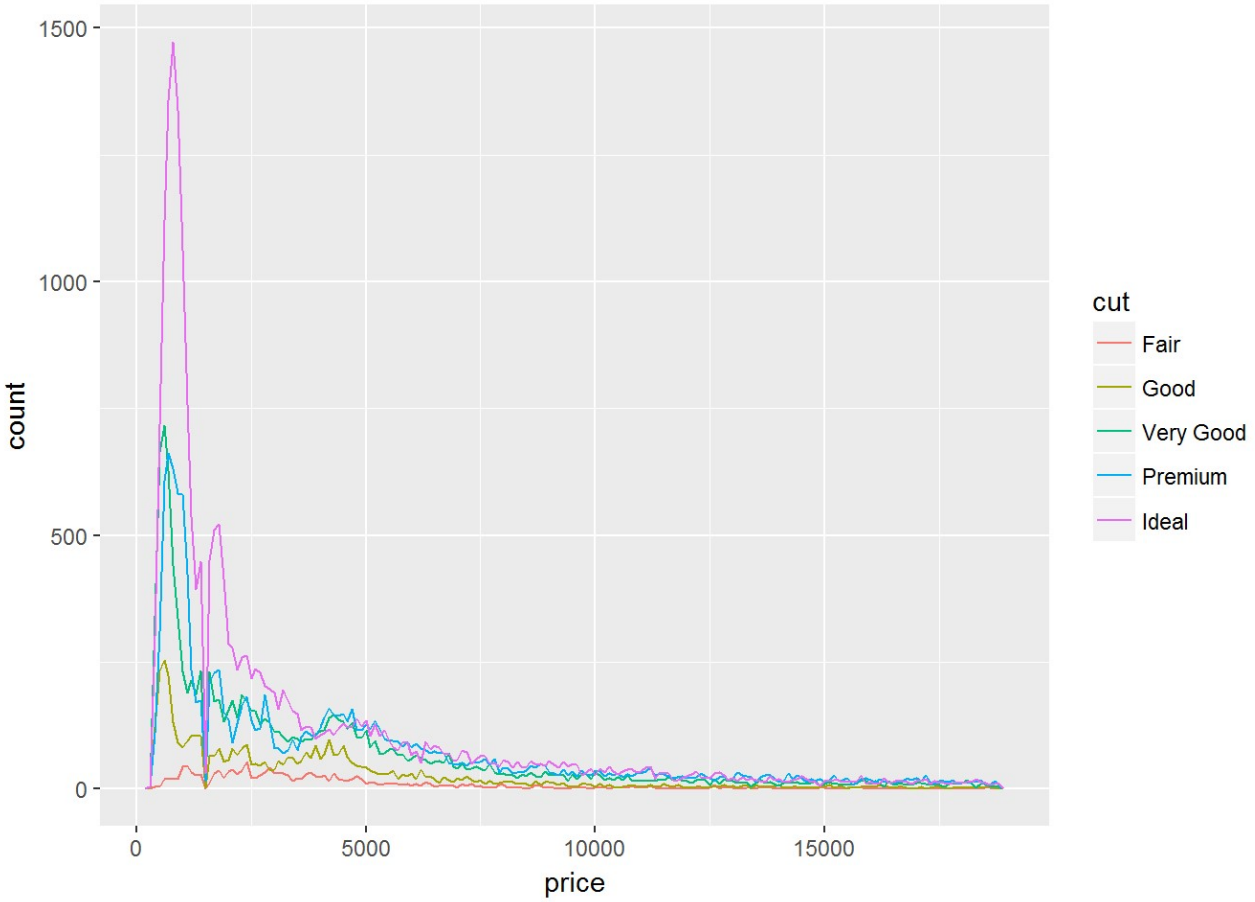
Task 4

Check the frequencies of diamonds for different clarity levels and cuts. Describe the results.

```
qplot(x = price,
      data = diamonds, binwidth = 100,
      geom = 'freqpoly',
      color = clarity)
```




```
qplot(x = price,
      data = diamonds, binwidth = 100,
      geom = 'freqpoly',
      color = cut)
```



We have plotted 2 frequency polygon curves. The first one, as we can see is for different levels of clarity and the second is for different levels of cut.

Again, what we can see in these plots is that the number of diamonds having a price less than $5000 is much greater for any cut or clarity categories than the rest, as seen in the plot.

And again we can see that the number of diamonds having an `ideal` cut is far greater than others. For clarity `SI1` and `VS2` have somewhat same number of diamonds as compared to others.

We can tune our x-axis to take a closer look.

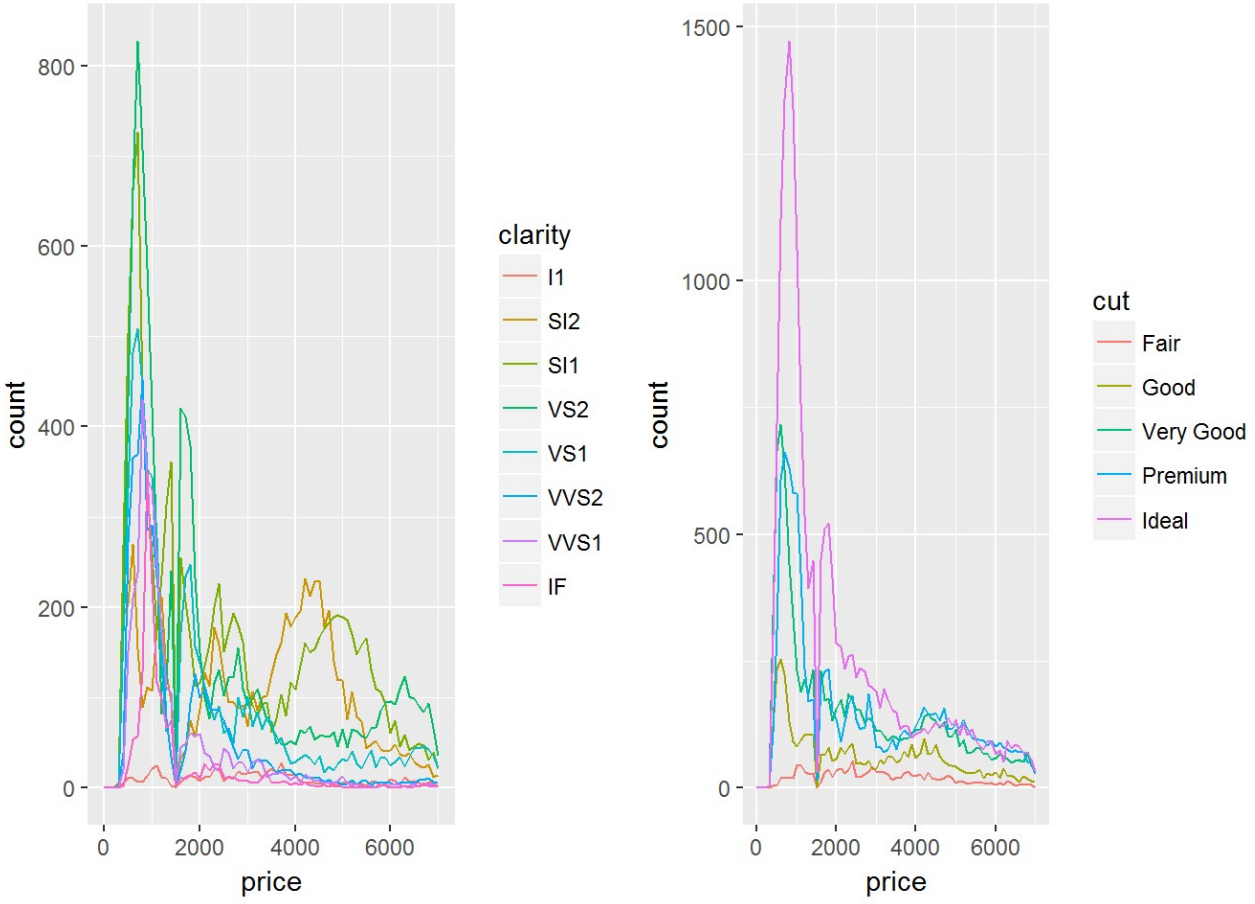
```
p1 <- qplot(x = price,
      data = diamonds, binwidth = 100,
      geom = 'freqpoly',
      color = clarity, xlim = c(0,7000))

p2 <- qplot(x = price,
      data = diamonds, binwidth = 100,
      geom = 'freqpoly',
      color = cut, xlim = c(0,7000))

library(gridExtra)
grid.arrange(p1, p2, ncol = 2)
```

```
## Warning: Removed 9273 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 16 rows containing missing values (geom_path).
```

```
## Warning: Removed 9273 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 10 rows containing missing values (geom_path).
```



Task 5

Focus your analysis on the carat, depth, table and dimensions:

```
summary(diamonds$carat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2000  0.4000  0.7000  0.7979  1.0400  5.0100
```

```
summary(diamonds$depth)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   43.00   61.00   61.80   61.75   62.50   79.00
```

```
summary(diamonds$table)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   43.00   56.00   57.00   57.46   59.00   95.00
```

```
summary(diamonds$x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.710   5.700   5.731   6.540  10.740
```

```
summary(diamonds$y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.720   5.710   5.735   6.540  58.900
```

```
summary(diamonds$z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.910   3.530   3.539   4.040  31.800
```

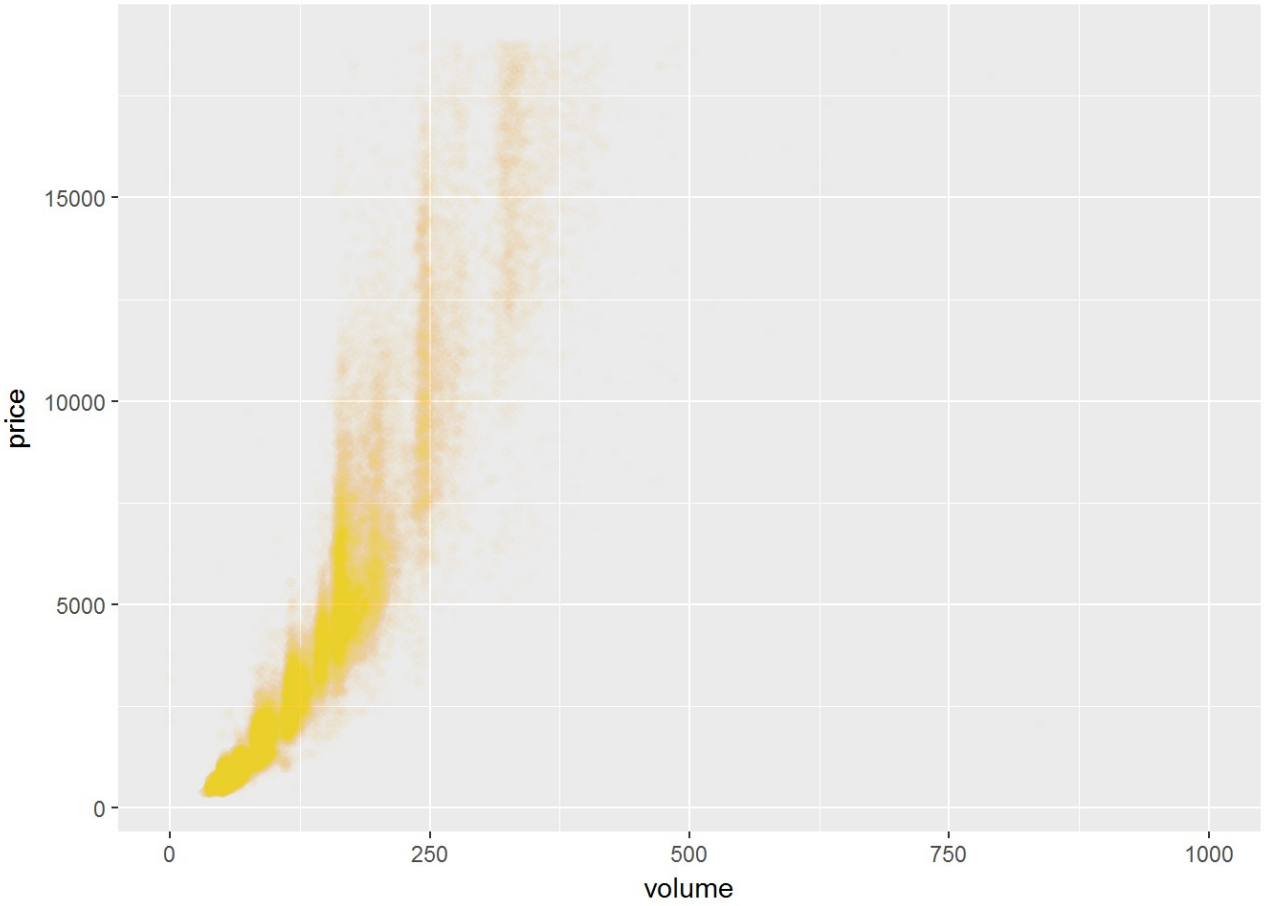
Here are the summaries for the variables required in this task.

---

a. Create volume variable - plot it against the price. Describe findings.

```
volume <- diamonds$x *diamonds$y * diamonds$z
diamonds$volume <- volume

ggplot(aes(x = volume, y = price), data = diamonds) +
  geom_point(alpha = 1/100,
             color = 'orange')+
  xlim(0, 1000)
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



We created a new variable in our data set, volume. And we ploted it against price variable. The findings were that the data was again, long tailed so we had to tune our x-axis.

Now we want to see the correlation between these two variables.

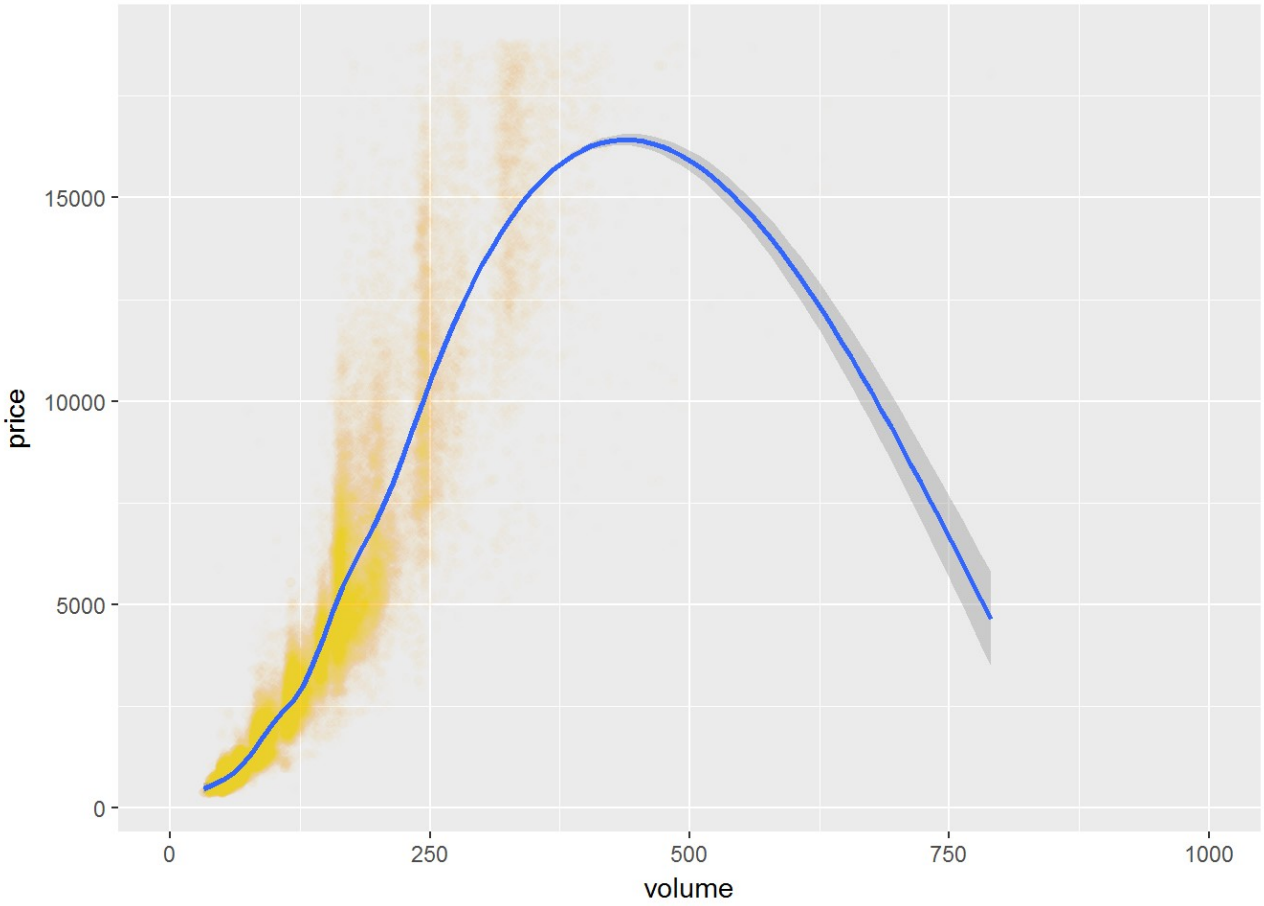
```
diamondsSubset <- subset(diamonds, diamonds$volume > 0 & diamonds$volume <= 800)
cor(x = diamondsSubset$price, y = diamondsSubset$volume)
```

```
## [1] 0.9235455
```

This shows us that these two variables are strongly correlated to each other. We can add a smooth layer to our plot to see the model it generates.

```
ggplot(aes(x = volume, y = price), data = diamondsSubset) +
  geom_point(color = 'orange',
             alpha = 1/100) +
  geom_smooth() +
  xlim(0, 1000)
```

```
## `geom_smooth()` using method = 'gam'
```



b. Are carat and volume attribute correlated? Is that relationship stronger then relationships with dimensions separately?

For this we can check the correlation of these variables

```
cor(x = diamondsSubset$carat, y = diamondsSubset$volume)
```

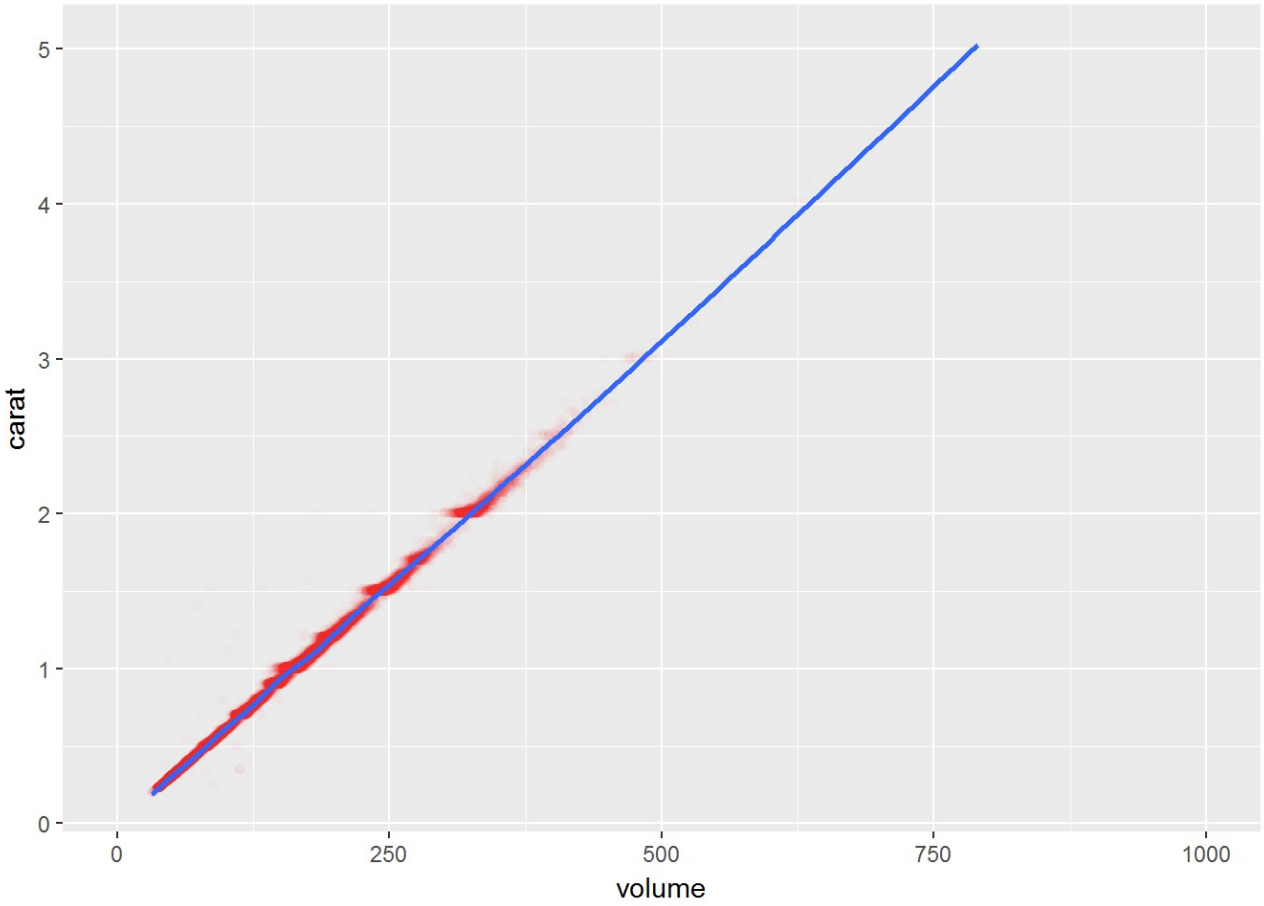
```
## [1] 0.9989412
```

Ohh! These two are strongly correlated. And this makes sense as volume is strongly correlated to price and price is correlated to carat, therefore volume is also strongly correlated to carat.

We can plot a scatterplot between them and add a smooth layer to it to see the correaltion.

```
ggplot(aes(x = volume, y = carat), data = diamondsSubset) +
  geom_point(color = 'red',
             alpha = 1/100) +
  geom_smooth() +
  xlim(0, 1000)
```

```
## `geom_smooth()` using method = 'gam'
```



We can see that the line of regression lies on the plot itself. Strong correlation, it is (Yoda reference!)

Now to see the correlations between volumes and its dimensions we need to do the correlation test first.

```
cor(x = diamondsSubset$x, y = diamondsSubset$volume)
```

```
## [1] 0.9790779
```

```
cor(x = diamondsSubset$y, y = diamondsSubset$volume)
```

```
## [1] 0.9786284
```

```
cor(x = diamondsSubset$z, y = diamondsSubset$volume)
```

```
## [1] 0.9760615
```

And surprisingly, the correlation between volume and carat is stronger than the correlation between volume and dimensions even though volume is basically a superset of dimensions. Thus, we always should be cynical about our findings because what we may assume about our data may not always be true.

c. Explore the relationships between table and depth variables, and other variables.

First we can look for the correlation between table and depth.

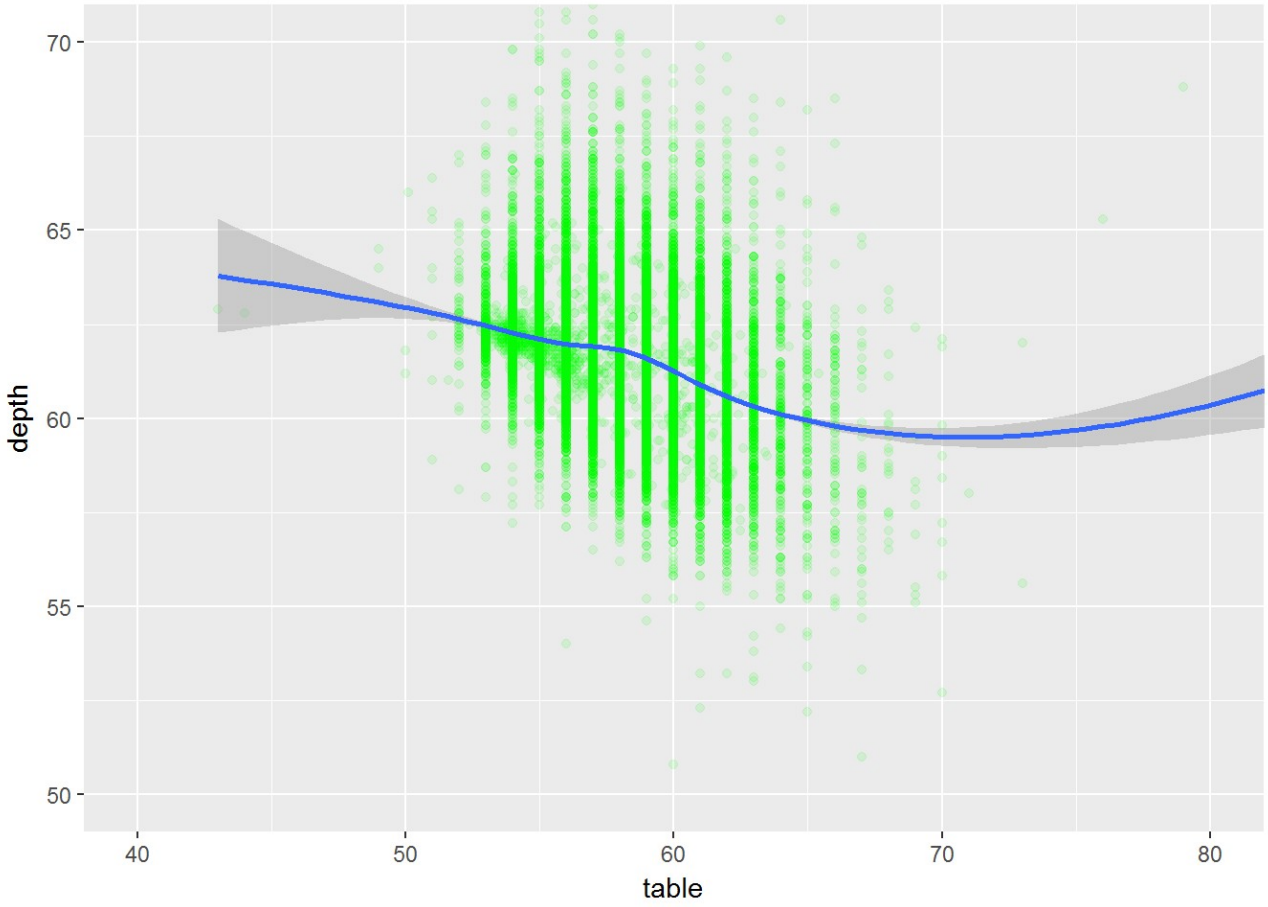
```
cor(x = diamondsSubset$table, y = diamondsSubset$depth)
```

```
## [1] -0.295758
```

The correlation came out to be around -0.3, which can be termed as not a decent correlation. Now we can plot these two variables and see how does the model turns out.

```
ggplot(aes(x = table, y = depth), data = diamondsSubset) +
  geom_point(color = 'green',
             alpha = 1/10) +
  geom_smooth() +
  coord_cartesian(xlim = c(40, 80), ylim = c(50,70))
```

```
## `geom_smooth()` using method = 'gam'
```



Now we can see that the data is scattered around the regression line. Also the line is going downwards thus proving the negative in correlation.

We have a new variable volume. We can check its relationship with table and depth and see how does volume effects these variables.

```
cor(x = diamondsSubset$table, y = diamondsSubset$volume)
```

```
## [1] 0.172287
```

```
cor(x = diamondsSubset$depth, y = diamondsSubset$volume)
```

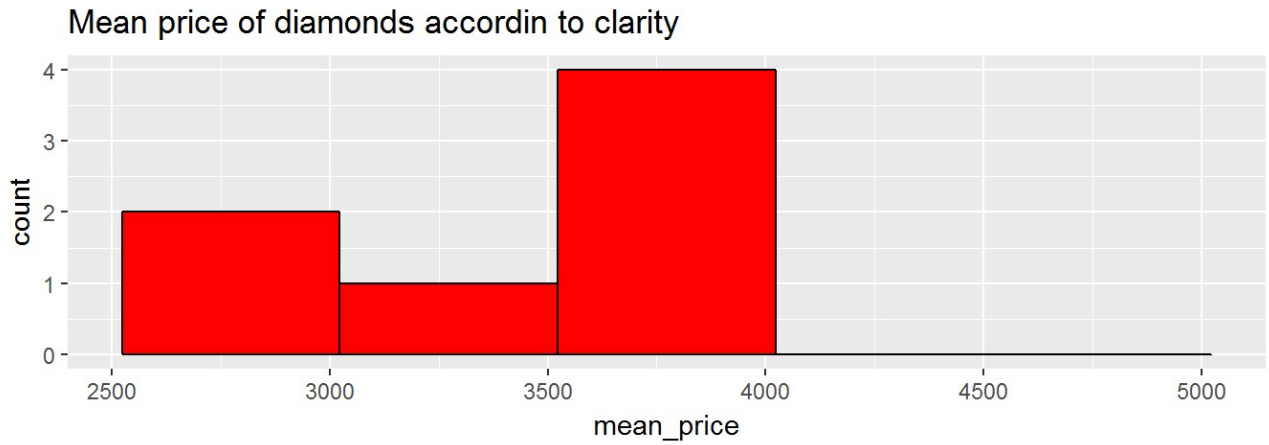
```
## [1] 0.01102758
```

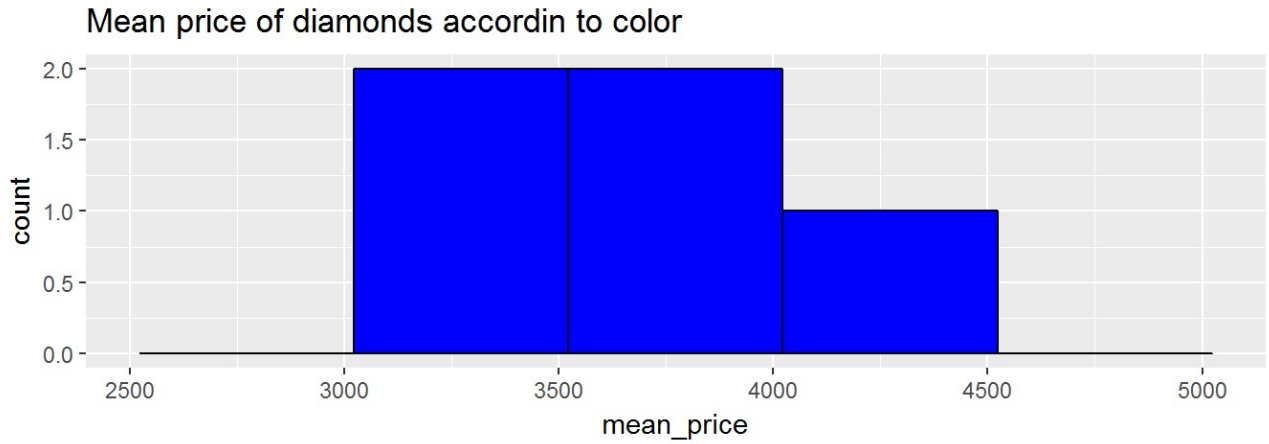
The correlation came out to be not that great. That means volume does not affects table and depth variable that much.

Some extra analysis

Bar charts of mean prices according to clarity

```
library(dplyr)
diamonds_by_clarity <- group_by(diamonds, clarity)
diamonds_mp_by_clarity <- summarise(diamonds_by_clarity, mean_price = mean(price))

diamonds_by_color <- group_by(diamonds, color)
diamonds_mp_by_color <- summarise(diamonds_by_color, mean_price = mean(price))

p1 <- ggplot(aes(x = mean_price), data = diamonds_mp_by_clarity) +
  geom_bar(stat = 'bin',
           breaks = seq(2523, 5324, 500),
           color = 'black',
           fill = 'red') +
  labs(title = 'Mean price of diamonds accordin to clarity')

p2 <- ggplot(aes(x = mean_price), data = diamonds_mp_by_color) +
  geom_bar(stat = 'bin',
           breaks = seq(2523, 5324, 500),
           color = 'black',
           fill = 'blue')+
  labs(title = 'Mean price of diamonds accordin to color')

library(gridExtra)
grid.arrange(p1, p2, ncol = 1)
```





Here we have plotter two bar graphs for mean_price using the subsetted data which is done by subsetting `diamonds` dataset on the basis of clarity and cut.

We have used these two datasets to see the mean price of diamonds according to cut and according to clarity.