#Name: Ajay Joshi

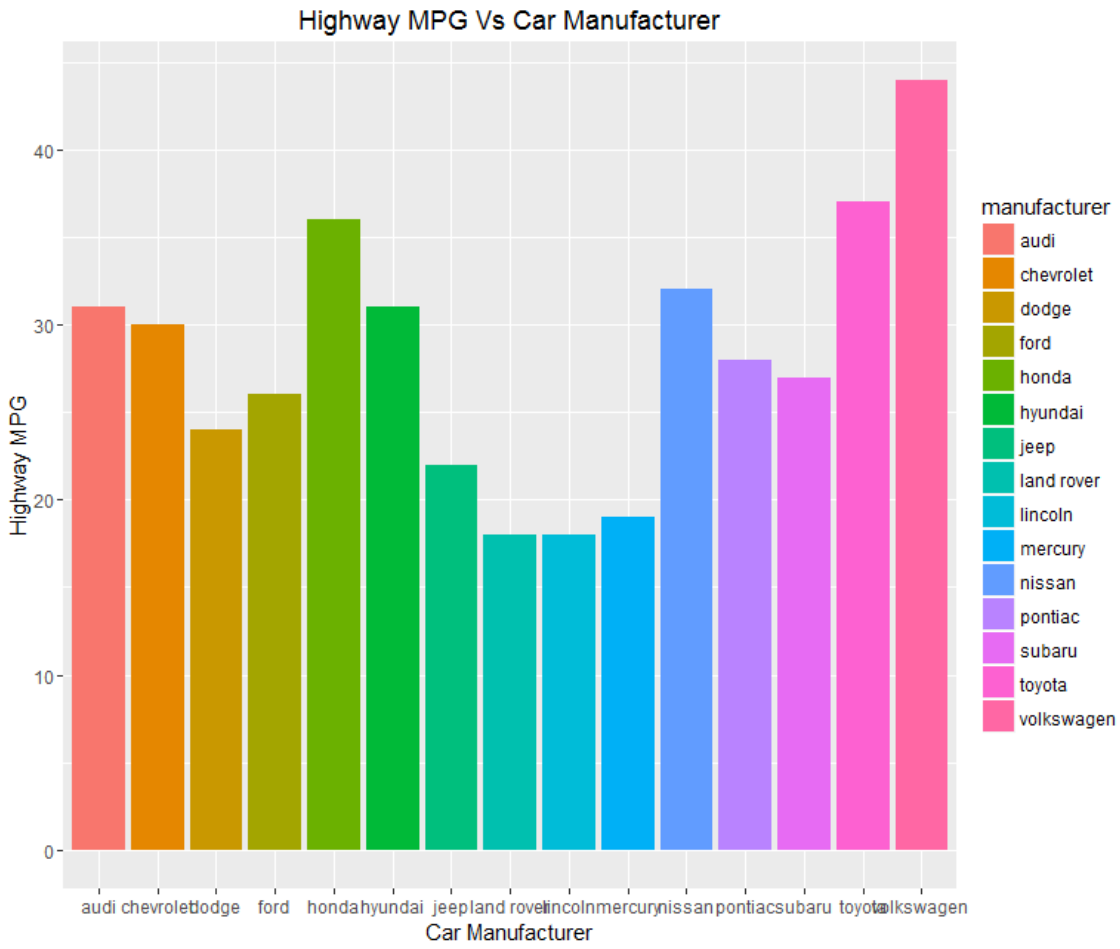
#----------

#**question 1**: Using the mpg data, describe the relationship between highway mpg and

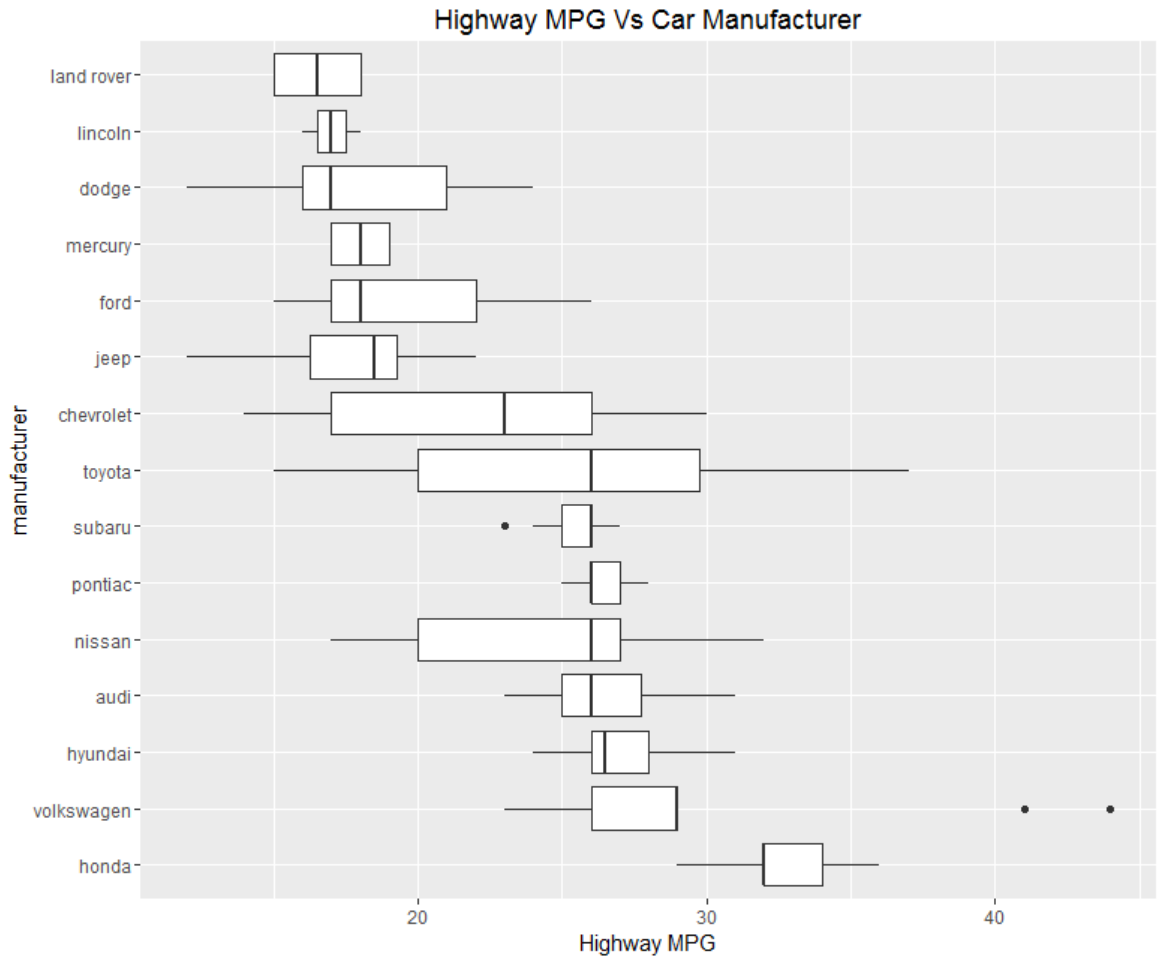
#car manufacturer. Describe which companies produce the most and least

#fuel efficient cars, and display a graph supporting your conclusion.

```
# Use bar plot
ggplot(mpg, aes(x = mpg$manufacturer, y = mpg$hwy, fill=manufacturer )) +
  geom_bar(stat="identity", position=position_dodge()) +
      labs(x="Car Manufacturer",y="Highway MPG") +
      ggtitle("Highway MPG Vs Car Manufacturer")
```



#Use Box plot graph




```
ggplot(mpg, aes(reorder(manufacturer, -hwy, median), hwy)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete("manufacturer") +
  labs(x="Car Manufacturer",y="Highway MPG") +
  ggtitle("Highway MPG Vs Car Manufacturer")
```

We can conclude from the above plot, the most fuel efficient cars are Volkswagen, toyota, and honda

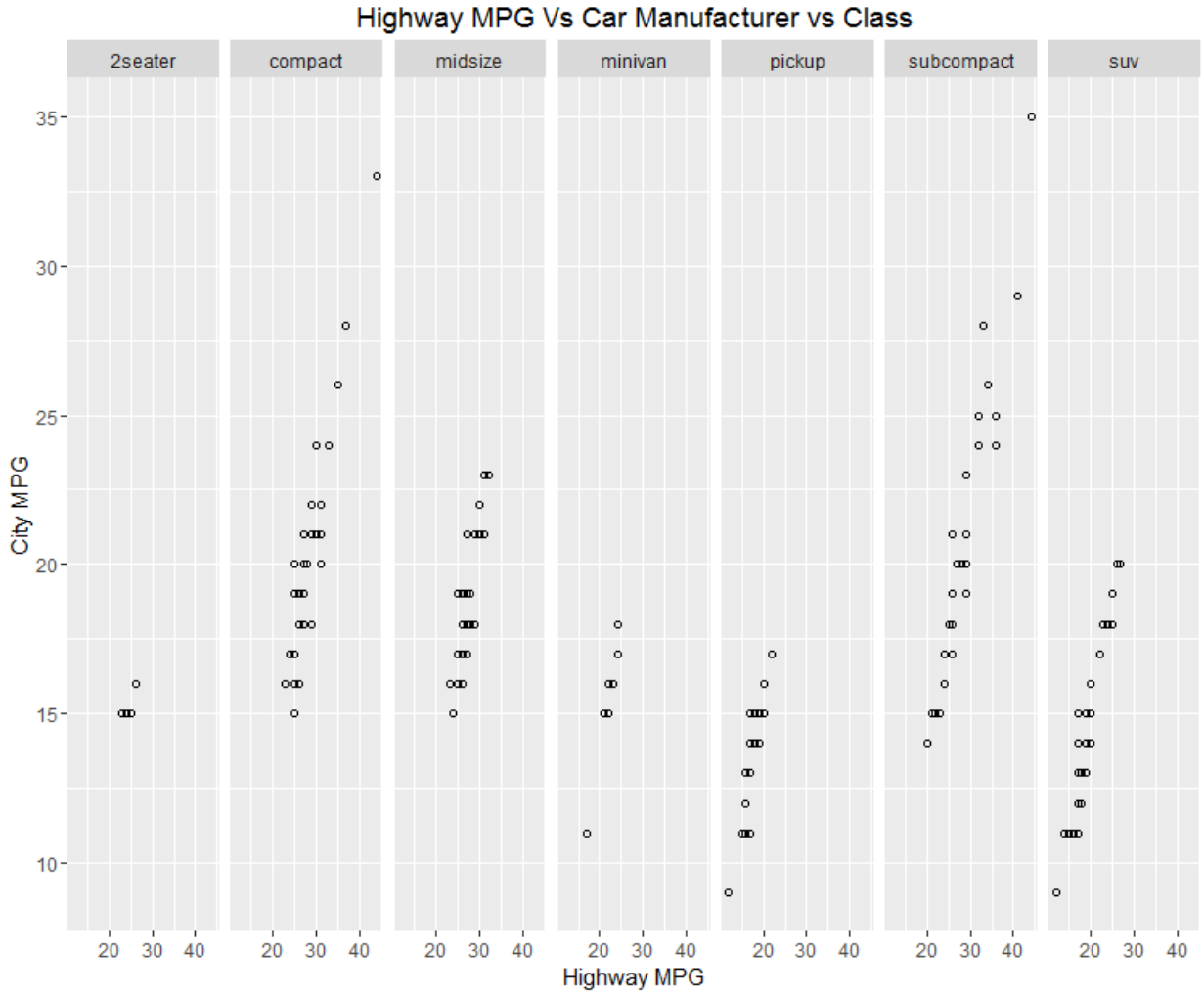
The least fuel efficient cars models are dodge, jeep, and chevrolet.

#----------

#**question 2** : Using the mpg data, explore the three-way relationship between highway

#mpg, city mpg, and model class. What are your observations? Display a graph supporting these observations.

```
# plot the graph - three-way relationship between highway mpg, city mpg, and class
ggplot(mpg, aes(x=hwy, y=cty))+
    facet_grid(. ~ class) +
    geom_point(shape =1 ) +
    labs(x="Highway MPG",y="City MPG") +
    ggtitle("Highway MPG Vs Car Manufacturer vs Class")
```

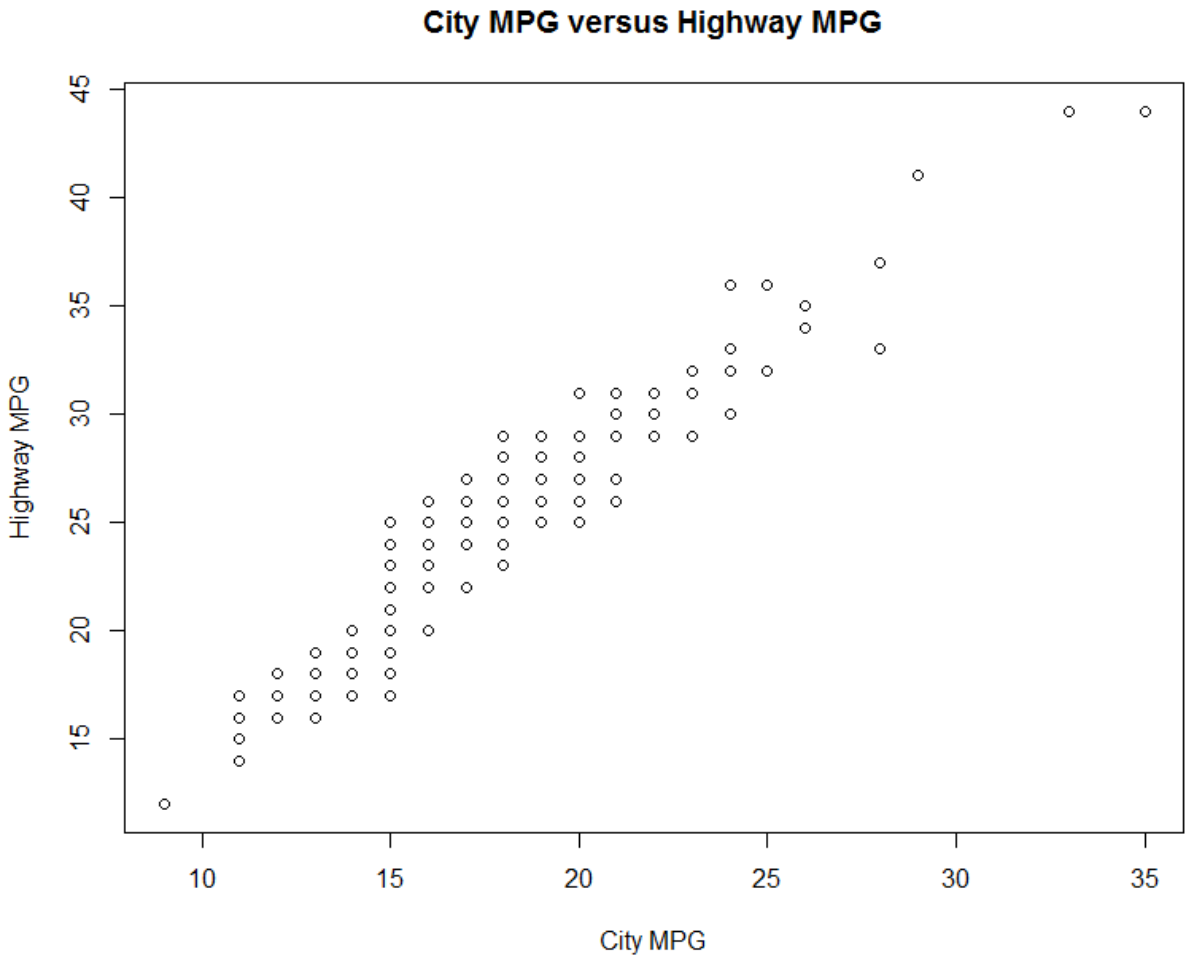


-> By looking at the above graph, subcompact and compact classes are the most fuel efficient vehicles,

#whereas pickup and SUV classes are the worst fuel efficient vehicles

Scatter plot- City mpg vs Highway mpg graph

plot(mpgcty, mpghwy, data=mpg, xlab = "City MPG", ylab="Highway MPG", main="City MPG versus Highway MPG")



calculate Correlation coefficients

cor(mpgcty, mpghwy)

the value is 0.9559159, which is positive correlation

#means that if one city mpg value gets bigger, the highway mpg tends to get bigger.

#----------

#Question 3: What are the pros and cons of using a histogram vs a box plot? Which one

#will you prefer for what purpose?

#Histogram

#Pros

1. It divides the numeric data into uniform intervals and displays the number of data values falling within each bin.

2. They group data into a small chunk. They are useful for summarizing numeric data in that they show the

rough distribution of values

#cons

1. The histogram doesn't show information about what is happening within each bin of the graph.

2. It shows the number of values within an interval but not the actual values

#Box Plot

#Pros

1. It is a good way to summarize large amounts of data.

2. It is easier to read minimum value, median, outliers, quantiles, and maximum value.

#cons

1. It’s hard to identify the original data

I will use boxplot if I have display the range and distribution of data whereas

#histogram will be used to displays the number of values within an interval

#----------

Question 4: Generate two sets of N random points using the function runif and display

a corresponding scatter plot. If you save the file to disk, what is the

```
# resulting file size for the following file formats: ps, pdf, jpeg, png? How do
# these values scale with increasing N?

Generate.RandomDeviates <- function(n) {
  X <- runif(n)
  Y <- runif(n)
  Generate.File(X,Y)
}

Generate.File <- function(X, Y) {
  dev.new()
  postscript("Plot.ps")
  plot(X,Y)
  dev.off()

  pdf("Plot.pdf")
  plot(X,Y)
  dev.off()

  jpeg("Plot.jpeg")
  plot(X,Y)
  dev.off()

  png("Plot.png")
  plot(X,Y)
  dev.off()
}

##########################Main Function#######################
```

#set1, N=200

Generate.RandomDeviates(200)

#set2, N=40

Generate.RandomDeviates(400)

#set3, N=600

Generate.RandomDeviates(600)

#set4, N=1200

Generate.RandomDeviates(1200)

#set5, N=12000

Generate.RandomDeviates(12000)

#set5, N=1000000

Generate.RandomDeviates(1000000)

N	Jpeg (KB)	Png (KB)	Pdf (KB)	Ps (KB)
200	23	4	6	11
400	35	7	8	17
600	42	8	9	22
1200	61	12	14	36
12000	87	21	91	300
12000000	13	3	7121	24421

#By looking at the above table, the value of N increases to from 200 1 million,

#All the jpeg and png files value decreased whereas pdf and PS values increased as the value of N increased. All the images are attached as part of this assignment.

Question 5: The diamonds dataset within ggplot2 contains 10 columns (price, carat,cut, color, etc.)

for 53940 diferent diamonds. Type help(diamonds) for

more information. Plot histograms for color, carat, and price, and comment

on their shapes. Investigate the three-way relationship between price, carat,

and cut. What are your conclusions? Provide graphs that support your

conclusions. If you encounter computational di!culties, consider using a

smaller dataframe whose rows are sampled from the original diamonds

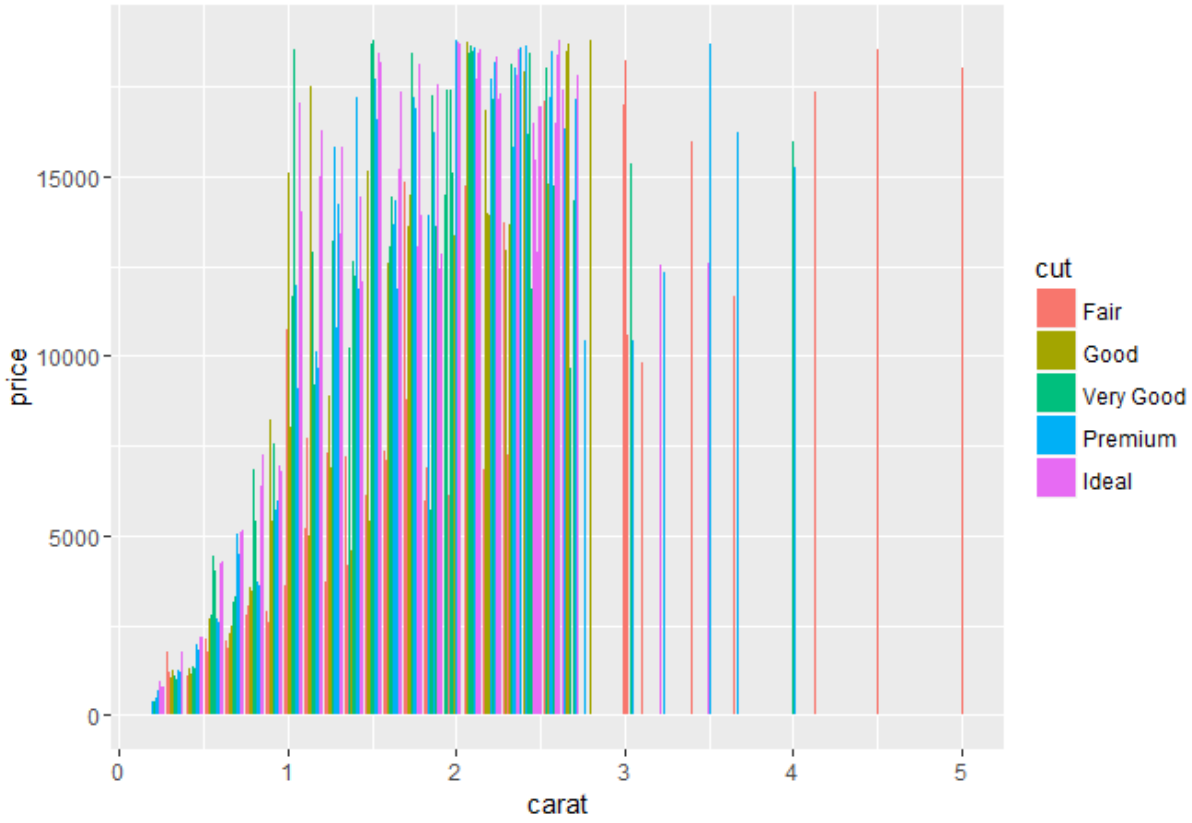
dataframe. Use the function sample to create a subset of indices that

may be used to create the smaller dataframe.

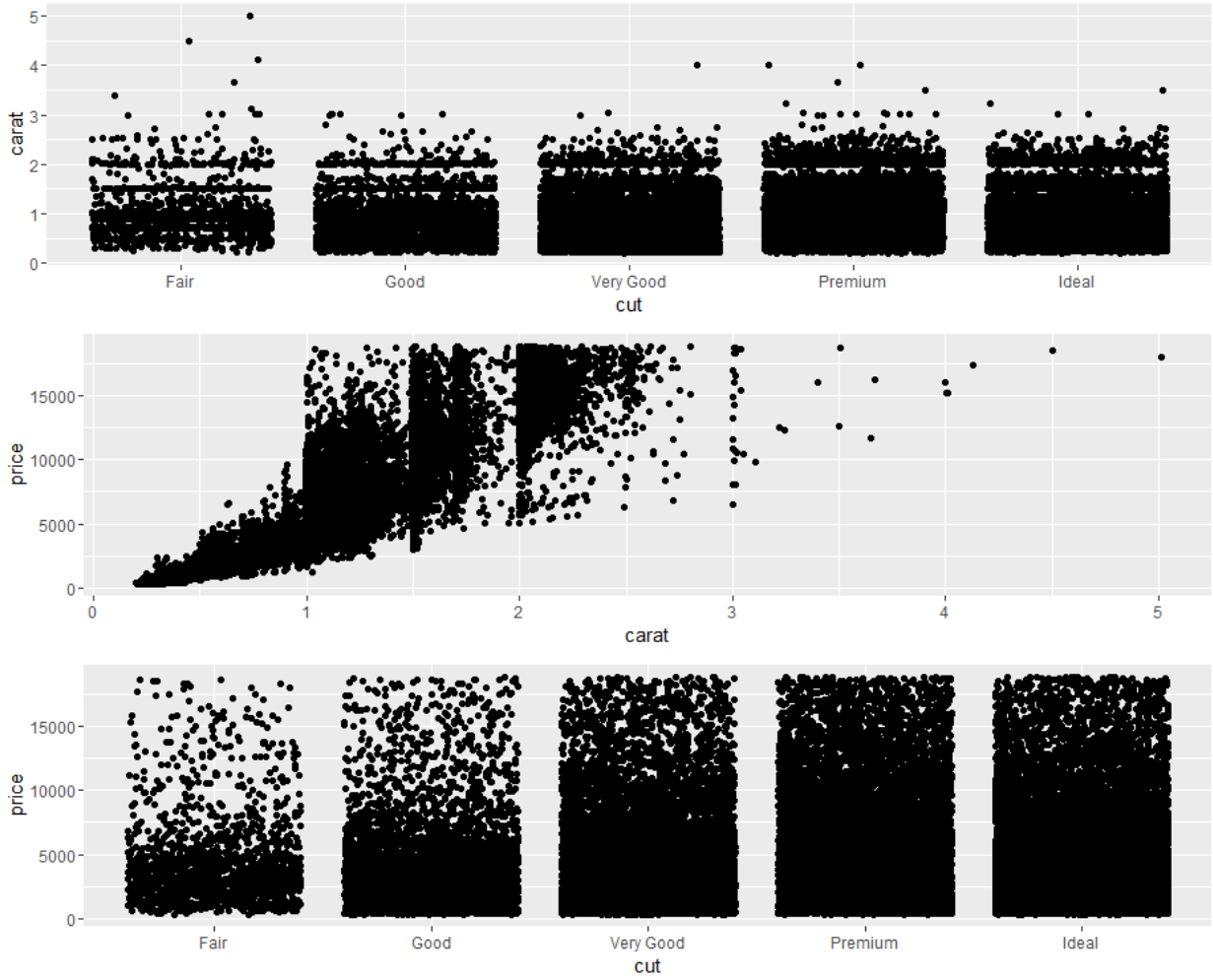
Investigation about the three-way relationship

Use bar graph to plot three-way relationship between cut, carat, and price

```
ggplot(diamonds, aes(x = carat, y = price, fill=cut)) +
  geom_bar(stat="identity", position=position_dodge())
```



Cut Vs Carat, Carat Vs Price, and Cut Vs Price graphs are used to investigate their relationship.



Cut Vs Carat

```
p1 <- ggplot(diamonds, aes(cut, carat)) +
  geom_jitter()
```

->By looking at the graph, we can see that there are more demands for ideal, permium,

and very good cut than good and fair ones. 0.3 ro 1.5 carat diamonds are most demanding than others.

#Carat Vs Price

```
p2 <- ggplot(diamonds, aes(carat,price)) +
  geom_jitter()
```

-> Price increases from as the carat increase 0.3 to 1.0.

The most expensive diamonds are 1.2 to 2.7 carat.

Cut Vs Price

```
p3 <- ggplot(diamonds, aes(cut, price)) +
  geom_jitter()
```

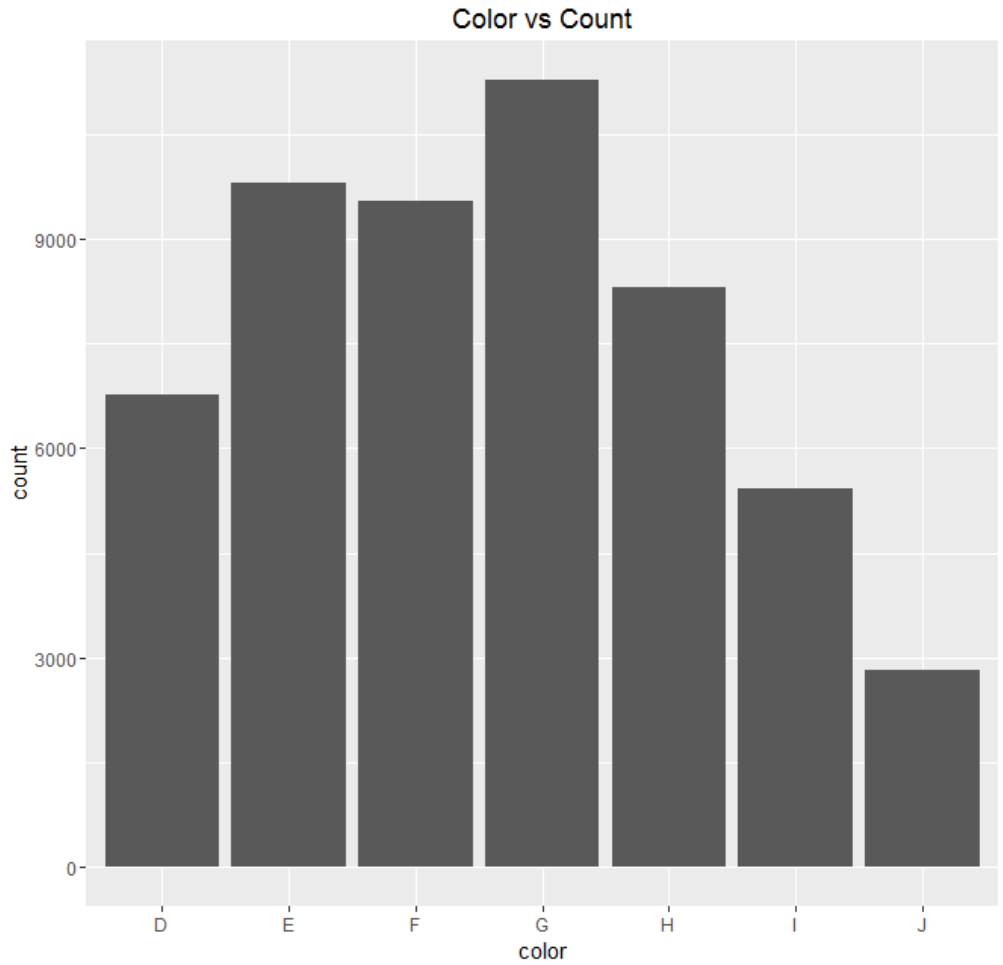
->Usually, the ideal cut diamonds tends to cost more money wheareas the fair cuts are cheaper among all.

```
# combine three graphs in one page
multiplot(p1, p2, p3, cols=1)
```

#++

#plots histograms for color, carats, and price

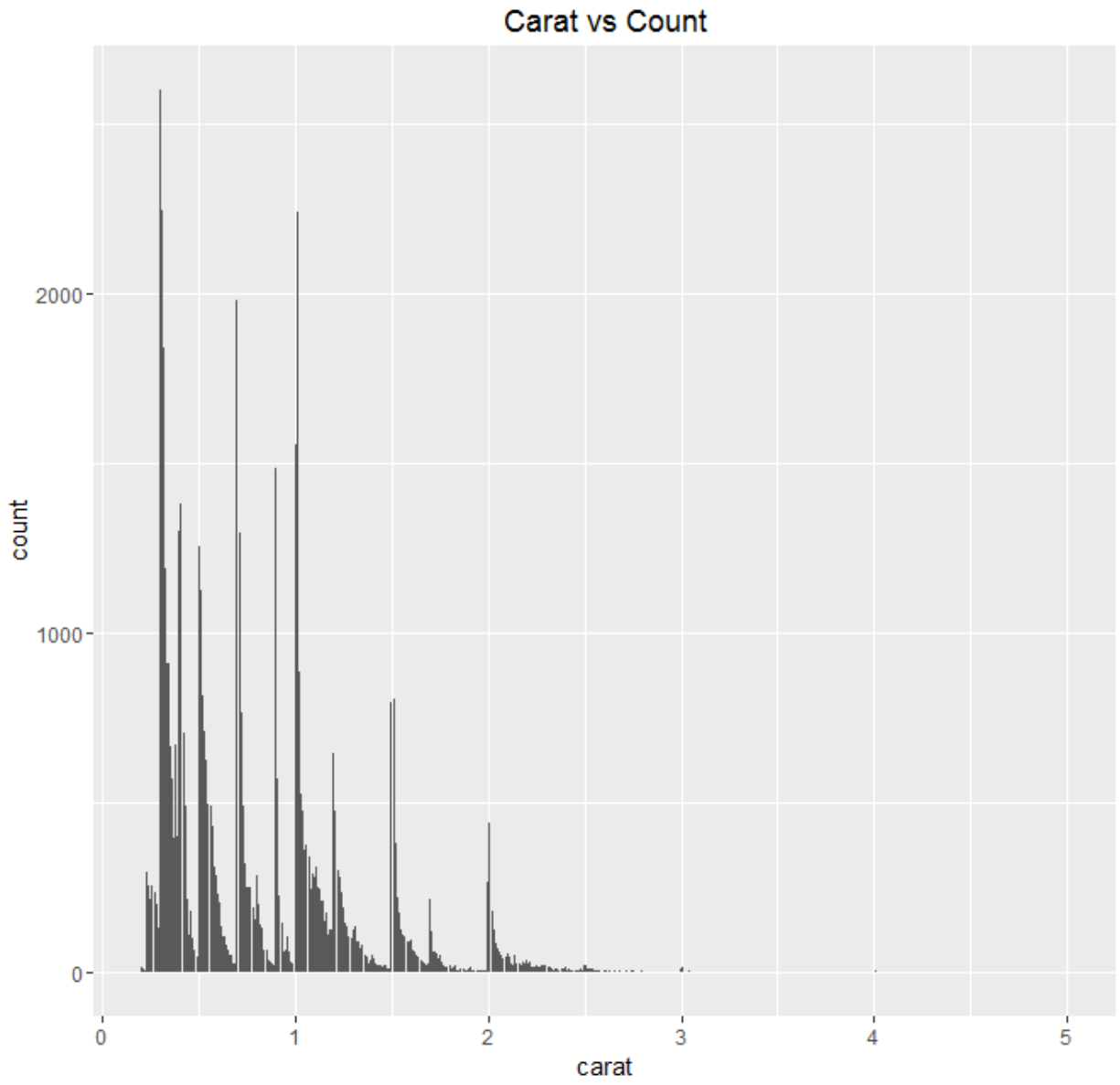
Plot histograms for color




```
ggplot(diamonds, aes(color)) + geom_bar() +
  ggtitle("Color vs Count")
```

#-> From the highest to lowest demanding diamond color: G, E, R, H D,I,J

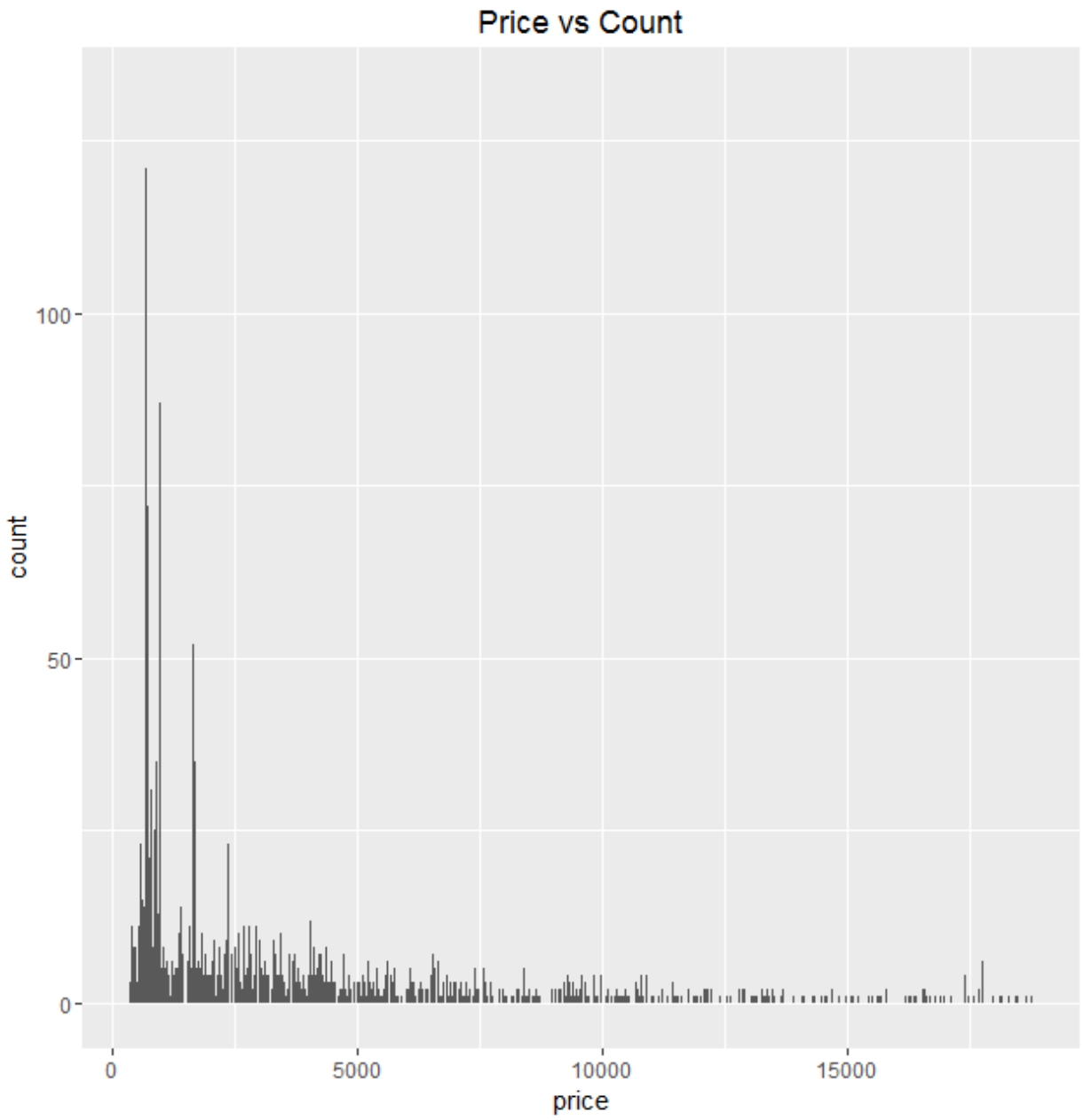
Plot histograms for carat




```
ggplot(diamonds, aes(carat)) + geom_bar()+
  ggtitle("Carat vs Count")
```

#-> The highest count is from 0.3 to 0.4 carat diamonds.

Plot histograms for price




```
ggplot(diamonds, aes(price)) +  geom_histogram(binwidth = 1) +
  ggtitle("Price vs Count")
```

#-> By looking at the graph, the count reaches to the maz which is 80 when the price is around 750 to 900.

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

# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }
```

```
 if (numPlots==1) {
   print(plots[[1]])

 } else {
   # Set up the page
   grid.newpage()
   pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

   # Make each plot, in the correct location
   for (i in 1:numPlots) {
     # Get the i,j matrix positions of the regions that contain this subplot
     matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

     print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                     layout.pos.col = matchidx$col))
   }
 }
}
```