INFX 573: Problem Set 2 - Data Wrangling

Rajendran Seetharaman

Due: Thursday, October 19, 2017

Collaborators:

Anuj Anand

Instructions:

Before beginning this assignment, please ensure you have access to R and RStudio.

1. Download the `problemset2.Rmd` file from Canvas. Open `problemset2.Rmd` in RStudio and supply your solutions to the assignment by editing `problemset2.Rmd`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment.
3. Be sure to include well-documented (e.g. commented) code chucks, figures and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text.
4. Collaboration on problem sets is acceptable, and even encouraged, but each student must turn in an individual write-up in his or her own words and his or her own work. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
5. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF`, rename the R Markdown file to `YourLastName_YourFirstName_ps2.Rmd`, knit a PDF and submit the PDF file on Canvas.

Setup:

In this problem set you will need, at minimum, the following R packages.

```
# Load standard libraries
library(tidyverse)
library(nycflights13)
library(jsonlite)
library(tidyr)
```

Problem 1: Open Government Data

Use the following code to obtain data on the Seattle Police Department Police Report Incidents.

```
police_incidents <- fromJSON("https://data.seattle.gov/resource/policereport.json")
```

(a) Describe, in detail, what the data represents.

The police incidents data is based on initial police reports taken by officers when responding to incidents around the city of Seattle. The information The police Records Management System (RMS) and is then transmitted out to data.seattle.gov and published within 6 to 12 hours after the report is filed into the system.

The original dataset has 742000 rows and 19 columns. However, the downloaded dataset has only a 1000 rows of the data.

(b) Describe each variable and what it measures. Be sure to note when data is missing. Confirm that each variable is appropriately cast - it has the correct data type. If any are incorrect, recast them to be in the appropriate format.

year: Displays the year when the incident was reported. Recasting this variable from character to integer to perform data manipulation such as filtering.

zone_beat: Displays the beat where the incident took place. Seattle PD has divided the city into 5 precints which is has a total of 17 sectors. These sectors are then divided into three beats and each beat has a code.

latitude, longitude: Displays the latitude and longitude of the location where the criminal incident took place. Recasting this variable from character to numeric to perform data manipulations.

offense_code_extension: Displays the offense code extension number. Recasting this variable from character to integer to perform data manipulation.

summarized_offense_description: Displays the description of the offense reported.

date_reported: Displays the date on which the offense was reported. Recasting this variable to POSIXct datatype.

offense_type: Displays the type of offense.

occurred_date_or_date_range_start: If the offense happened for a long time, it displays the start time of the offense. Recasting this variable to POSIXct type.

summary_offense_code: Displays a code to identify the offense summary. Recasting this variable from character to integer to perform data manipulation.

month: Displays the month when the incident was reported. Recasting this variable from character to integer to perform data manipulations.

general_offense_number: Displays a number used to identify the offense on the manual recordsr. Recasting this variable from character to integer to perform data manipulation.

census_tract_2000: Displays the details about the census tract or census area where the crime occured.

location: Displays the information about the crime location. It is a dataframe consisting of three variables namely longitude,latitude and needs_recoding which helps in understanding the crime location.

offense_code: Displays a code used to identify the offense. Recasting this variable from character to integer to perform data manipulation.

hundred_block_location: Displays the block location where the crime took place.

rms_cdw_id: Displays the unique identification number for Seattle PD's Records Management System(RMS). Recasting this variable from character to integer to perform data manipulation to filter for records with rms_cdw_id greater than a number.

district_sector: Displays the sector of Seattle in which the crime occured.

occurred_date_range_end: Displays the end time of the crime. Recasting this variable to POSIXct datatype.

```
police_incidents$year = as.integer(as.character(police_incidents$year))
police_incidents$month = as.integer(as.character(police_incidents$month))
police_incidents$latitude = as.numeric(as.character(police_incidents$latitude))
police_incidents$longitude = as.numeric(as.character(police_incidents$longitude))
police_incidents$offense_code[police_incidents$offense_code == "X"] <- "0"
police_incidents$offense_code =
  as.integer(as.character(police_incidents$offense_code))
police_incidents$offense_code_extension =
  as.integer(as.character(police_incidents$offense_code_extension))
police_incidents$summary_offense_code[police_incidents$summary_offense_code
```

```
                                        == "X"] <- "0"
police_incidents$summary_offense_code =
  as.integer(as.character(police_incidents$summary_offense_code))
police_incidents$general_offense_number =
  as.integer(as.character(police_incidents$general_offense_number))
police_incidents$rms_cdw_id =
  as.integer(as.character(police_incidents$rms_cdw_id))
police_incidents$date_reported =
  as.POSIXct(police_incidents$date_reported,format="%Y-%m-%dT%H:%M:%OS")
police_incidents$occurred_date_or_date_range_start =
  as.POSIXct(police_incidents$occurred_date_or_date_range_start,
                      format="%Y-%m-%dT%H:%M:%OS")
police_incidents$occurred_date_range_end =
  as.POSIXct(police_incidents$occurred_date_range_end,
             format="%Y-%m-%dT%H:%M:%OS")
police_incidents$needs_recoding <- police_incidents$location$needs_recoding
```

(c) Produce a clean dataset, according to the rules of tidy data discussed in class. Export the data for future analysis using the Rdata format.

Answer: To clean the dataset I wish to separate the datetime fields in the into date and time fields.This would help us to analyze the time and dates when more crimes tend to happen.

```
police_incidents <- separate(police_incidents,occurred_date_or_date_range_start,
                             c("occurred_date_or_date_range_start",
                               "occurred_time_or_date_range_start_time"),
                             sep = " ")

police_incidents <- separate(police_incidents,occurred_date_range_end,
                             c("occurred_date_range_end",
                               "occurred_date_range_end_time"),sep = " ")

police_incidents <- separate(police_incidents,date_reported,
                             c("date_reported","time_reported"),sep = " ")
```

Also, the current dataset has two data units. One which contains the crime info like offense code while the other contains information about the offense report like location and date. I wish to seperate the two into variables offense and report containing the appropriate information.

```
offense <- police_incidents %>%
  select(offense_code, offense_type, summary_offense_code,summarized_offense_description,
         offense_code_extension) %>%
  unique()

reports <- police_incidents %>%
  select(rms_cdw_id, general_offense_number, year,
         month,date_reported, time_reported, offense_code,
         district_sector,zone_beat,census_tract_2000,
         latitude, longitude, needs_recoding,
         occurred_date_or_date_range_start,
         occurred_time_or_date_range_start_time,
         occurred_date_range_end, occurred_date_range_end_time,
         hundred_block_location)
```

Saving the datafiles for further analysis.

```
save(offense, reports, file = "police_incidents.Rdata")
```

(d) Describe any concerns you might have about this data. This may include biases, missing data, or ethical concerns.

Ans. The data contains information abour incident being reported but the information about if or not an investigation was conducted for the report is missing. Also the data about the person reporting the incident is missing, probably purposely with the intentions of protecting private information about an individual. Also the block address is a very broad indicator of where the crime actually happened.

Problem 2: Wrangling the NYC Flights Data

In this problem set we will use the data on all flights that departed NYC (i.e. JFK, LGA or EWR) in 2013. You can find this data in the `nycflights13` R package.

(a) Importing Data:

Load the data.

```
#loading the nycflights13 library automatically loads the flights dataset
library(nycflights13)
```

(b) Data Manipulation:

Use the flights data to answer each of the following questions. Be sure to answer each question with a written response and supporting analysis.

- How many flights were there from NYC airports to LAX in 2013? Ans. There were 16,174 flights from NYC airports to LAX in 2013 as seen from the below analysis.

```
#filter flights with destination LAX and year as 2013
Lax_dest <- flights %>% filter(dest=='LAX', year==2013)
#count the flights
Lax_dest %>% summarise(Lax_flights_count=n())
```

```
## # A tibble: 1 x 1
##   Lax_flights_count
##               <int>
## 1             16174
```

- How many airlines fly from NYC to LAX? Ans. 5 airlines fly from NYC airports to LAX as seen from the below analysis.

```
#Count the distinct carriers for flights which fly to LAX
Lax_dest %>% summarize(carrier_count=n_distinct(carrier))
```

```
## # A tibble: 1 x 1
##   carrier_count
##           <int>
## 1             5
```

- How many unique air planes fly from NYC to LAX? Ans.992 unique air planes fly from NYC airports to LAX as seen from the below analysis.

```
#count of distinct tail numbers yields unique plane count
Lax_dest %>% summarize(unique_plane_count=n_distinct(tailnum))
```

```
## # A tibble: 1 x 1
##   unique_plane_count
##                <int>
## 1                992
```

- What is the average arrival delay for flights from NYC to LAX? Ans. The average arrival delay for flights from NYC airports to LAX is 0.54711 mins as seen from the below analysis.

```
#take the mean of arrival delays of flights to LAX
Lax_dest %>% summarize(avg_arrival_delay=mean(arr_delay,na.rm=TRUE))
```

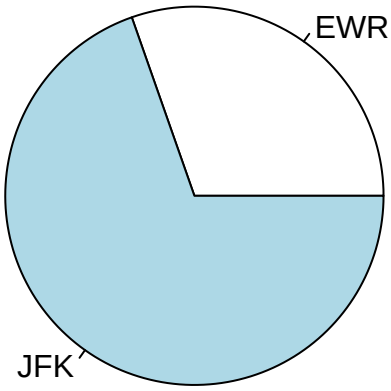
```
## # A tibble: 1 x 1
##   avg_arrival_delay
##               <dbl>
## 1         0.5471109
```

- What proportion of flights to LAX come from each NYC airport? Ans. Of all the flights to NYC, 30.36% of flights to LAX come from EWR and the remaining 69.63% come from JFK.

```
#take a count of all flights to LAX
totalflightcount <- Lax_dest %>% summarise(n()) %>% as.integer()
#group flights by origin and compute count of flights from origin as a proportion of total flights
flightprop <- Lax_dest %>% group_by(origin) %>%
  summarize(flightcountproportion=(n()/totalflightcount)*100)
print(flightprop)
```

```
## # A tibble: 2 x 2
##   origin flightcountproportion
##    <chr>                 <dbl>
## 1    EWR              30.36973
## 2    JFK              69.63027
```

```
#draw pie to represent the information
pie(flightprop$flightcountproportion,flightprop$origin)
```



Problem 3: Pipes and Diamonds

The following questions relate to the "diamonds" dataset included in the ggplot2 package.

(a) Importing Data:

Load and describe the data Ans. It is a dataset containing the prices and other attributes of almost 54,000 diamonds. It has 53940 rows and 10 variables. The variables in the dataset are-

price: price in US dollars carat: weight of the diamond 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)) x: length in mm y: width in mm z: depth in mm depth: total depth percentage = z / mean(x, y) = 2 * z / (x + y) table: width of top of diamond relative to widest point

Source: http://ggplot2.tidyverse.org/reference/diamonds.html

```
#loading the ggplot2 library loads the diamonds dataset automatically
library(ggplot2)
```

(b) Exploring the price column:

Create a new variable which contains the average `price` for each `cut` of diamond in decreasing order. Does this follow the ordering what you would expect? Ans. The price for each cut does not follow the ordering that I expected to see. The price ordering that I expected to see was Ideal > Premium > Very Good > Good >Fair. The ordering that I see might probably be because some other variables like carat might also be impacting the price.

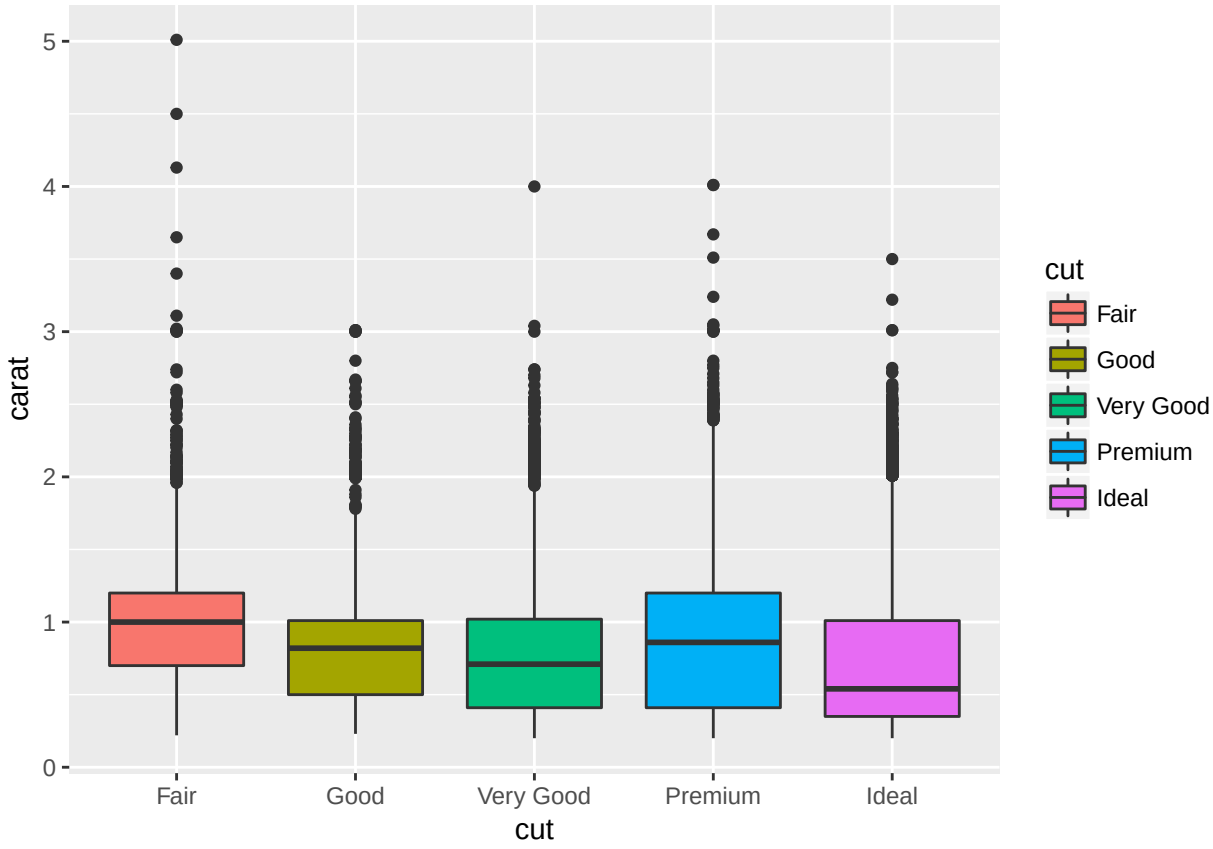
```r
#compute average price for each cut of diamond
price_by_cut <- diamonds %>% group_by(cut) %>% summarise(Avg_price=mean(price,na.rm = TRUE))
#arrange cuts according to average price in descending order
price_by_cut_ordered <- price_by_cut %>% arrange(desc(Avg_price))
price_by_cut_ordered
```

```
## # A tibble: 5 x 2
##         cut Avg_price
##       <ord>     <dbl>
## 1   Premium  4584.258
## 2      Fair  4358.758
## 3 Very Good  3981.760
## 4      Good  3928.864
## 5     Ideal  3457.542
```

(c) Correcting via carats:

One possible explaination for this is that the `carat` values might not be distributed evenly across different cuts. To check this, create a chart with a boxplot of the `carat` distribution for each `cut`. Ans. The median carats or weight for the fair cut of diamonds is higher than all other cut of diamonds. Its interquartile range is also small. On the other end, the median carats or weight for the ideal cut of diamonds is lower than all other cut of diamonds. The 75th percentile of the ideal cut diamonds almost equals the median of fair cut diamonds.

```r
#create boxplot
ggplot(diamonds,aes(x=cut,y=carat))+geom_boxplot(aes(fill=cut))
```



With this in mind, update the variable created in part (b) to weight the `price` by the `carat` value. Did this solve the issue? Ans. Weighting the price by carat (Computing price/carat) did have some effect on the above ordering of average price per cut of diamond. Fair moved to the bottom af the list and Ideal started moving up, indicating that the linear relationship between the weight and price of the diamond was skewing our analysis. However, the issue is not completely resolved as Ideal is displayed below Premium and Very good. It should be at the top of the list.

```
#compute mean of price per carat for each cut of diamond
price_by_cut_wt <- diamonds %>%
  mutate(pricepercarat=price/(carat))%>% group_by(cut) %>% summarise(Avg_w_price=mean(pricepercarat))
#Arrange cuts in descending order of Average weighted price
price_by_cut_wt_ordered <- price_by_cut_wt %>% arrange(desc(Avg_w_price))
price_by_cut_wt_ordered
```

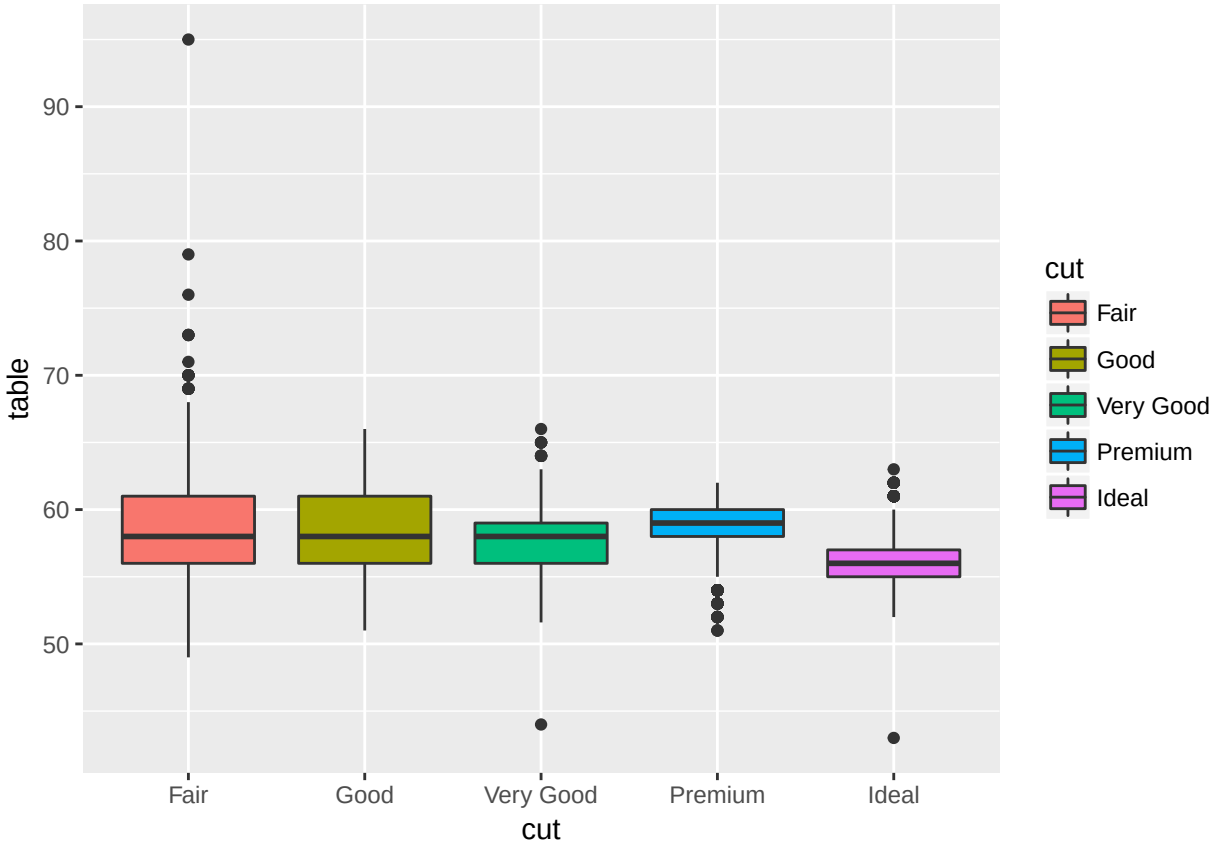
```
## # A tibble: 5 x 2
##         cut Avg_w_price
##       <ord>       <dbl>
## 1   Premium    4222.905
## 2 Very Good    4014.128
## 3     Ideal    3919.700
## 4      Good    3860.028
## 5      Fair    3767.256
```

(d) Further exploration:

Further refine your analysis by including at least one other varaible in the dataset. Does this improve your results? What factors might still be skewing them? Feel free to use visualizations, but be sure to include dplyr manipulations in your analysis. Ans. A variable which could have a possible effect on the price could be table size. Creating a boxplot of table sizes by cut type shown below strengthens this hypothesis. Adding the table size to the analysis results in ideal moving a spot up in the list of average pricing. But it is still not at the top of the list indicating that other variables need to be included as well.

Another variable which could be included in the analysis is the clarity of the diamonds. The clarity of a diamond should ideally have a correlation with the price. Greater the clarity, more the price. The ideal cut diamonds in our dataset are more skewed towards diamonds which have lower clarity as seen in the visualization below. There are a large number of S1, VS1, and VS2 clarity diamonds in our ideal cut diamond set This could have possibly brought down the price per carat value of ideal cut diamonds.

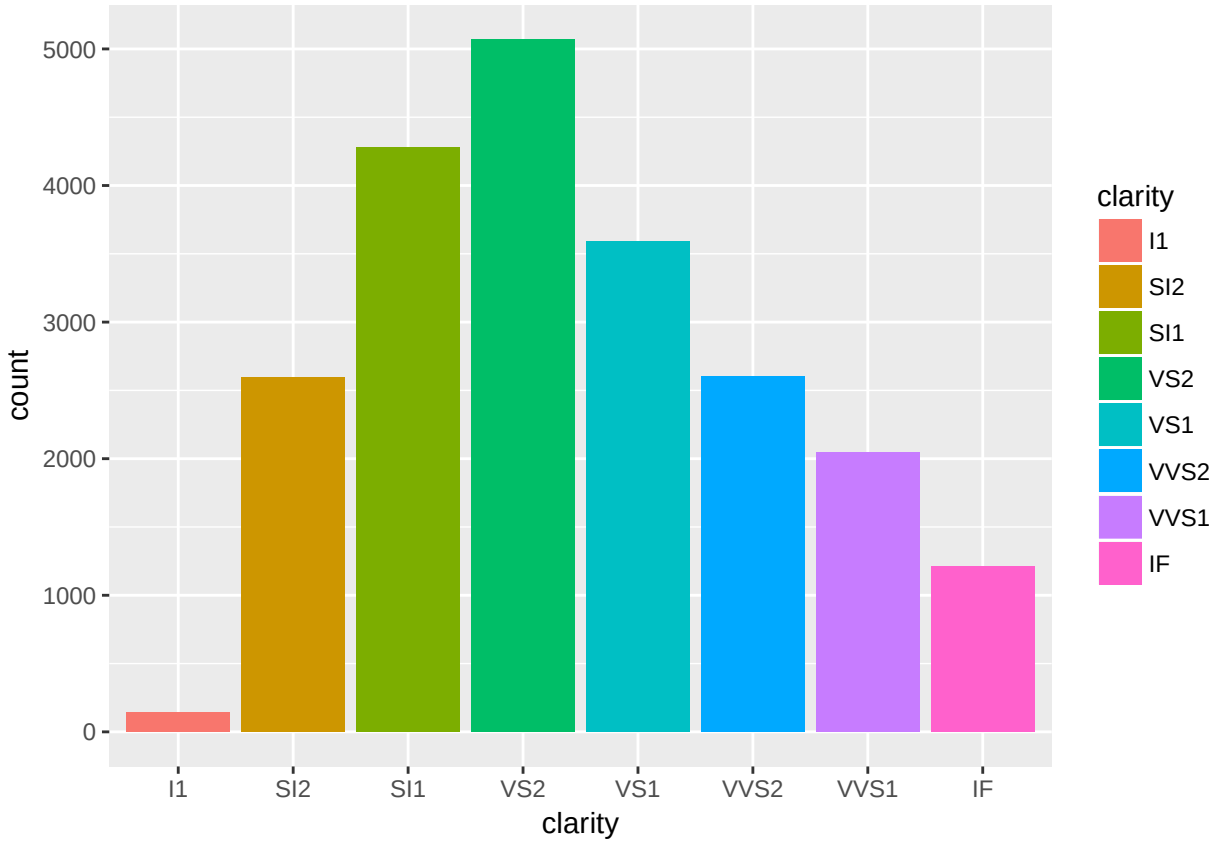
```
#Analyzing the impact of the varible table size
#create boxplot
ggplot(diamonds,aes(x=cut,y=table))+geom_boxplot(aes(fill=cut))
```



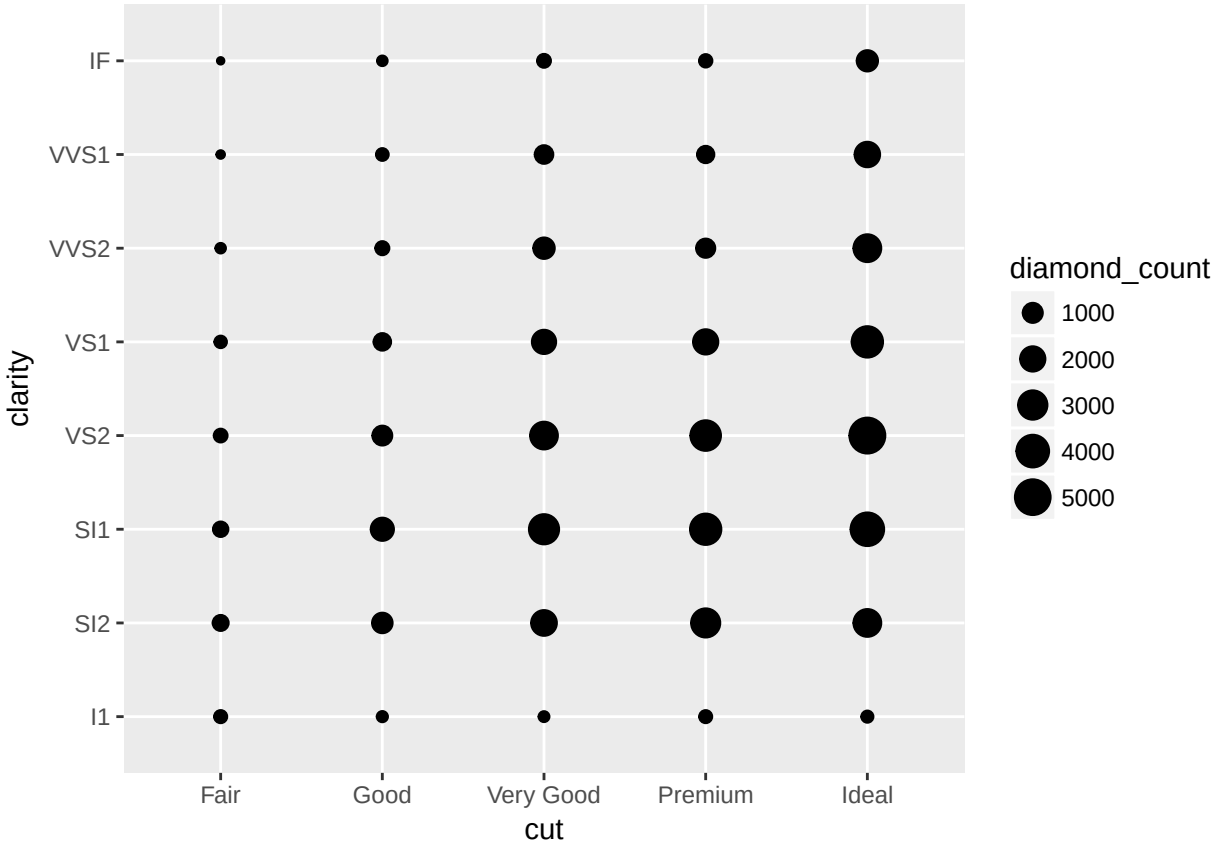

```
#compute mean of price per carat table for each cut of diamond
price_by_cut_wt2 <- diamonds %>%
  mutate(pricepercarattable=price/(carat*table))%>%
  group_by(cut) %>% summarise(Avg_w2_price=mean(pricepercarattable))
#Arrange cuts in descending order of Average weighted price
price_by_cut_wt2_ordered <- price_by_cut_wt2 %>%
  arrange(desc(Avg_w2_price))
print(price_by_cut_wt2_ordered)
```

```
## # A tibble: 5 x 2
##         cut Avg_w2_price
##       <ord>        <dbl>
## 1   Premium     71.94275
## 2     Ideal     69.98010
## 3 Very Good     69.25773
## 4      Good     65.65531
## 5      Fair     64.00536
```

```
#Analyzing impact of variable clarity
#select data with ideal cuts and plot graph of number of diamonds at each clarity level
ggplot(diamonds %>% filter(cut=="Ideal"),aes(x=clarity))+geom_bar(aes(fill=clarity))
```




```
#For different cuts, plot count of diamonds at each clarity level
#Greater size indicates greater count
ggplot(diamonds %>% group_by(cut,clarity) %>% summarize(diamond_count=n()),
       aes(x=cut,y=clarity))+geom_point(aes(size=diamond_count))
```


clarity
IF
VVS1
VVS2
VS1
VS2
SI1
SI2
I1
Fair
Good
Very Good
Premium
Ideal
cut
diamond_count
1000
2000
3000
4000
5000