EDA Technique Summary

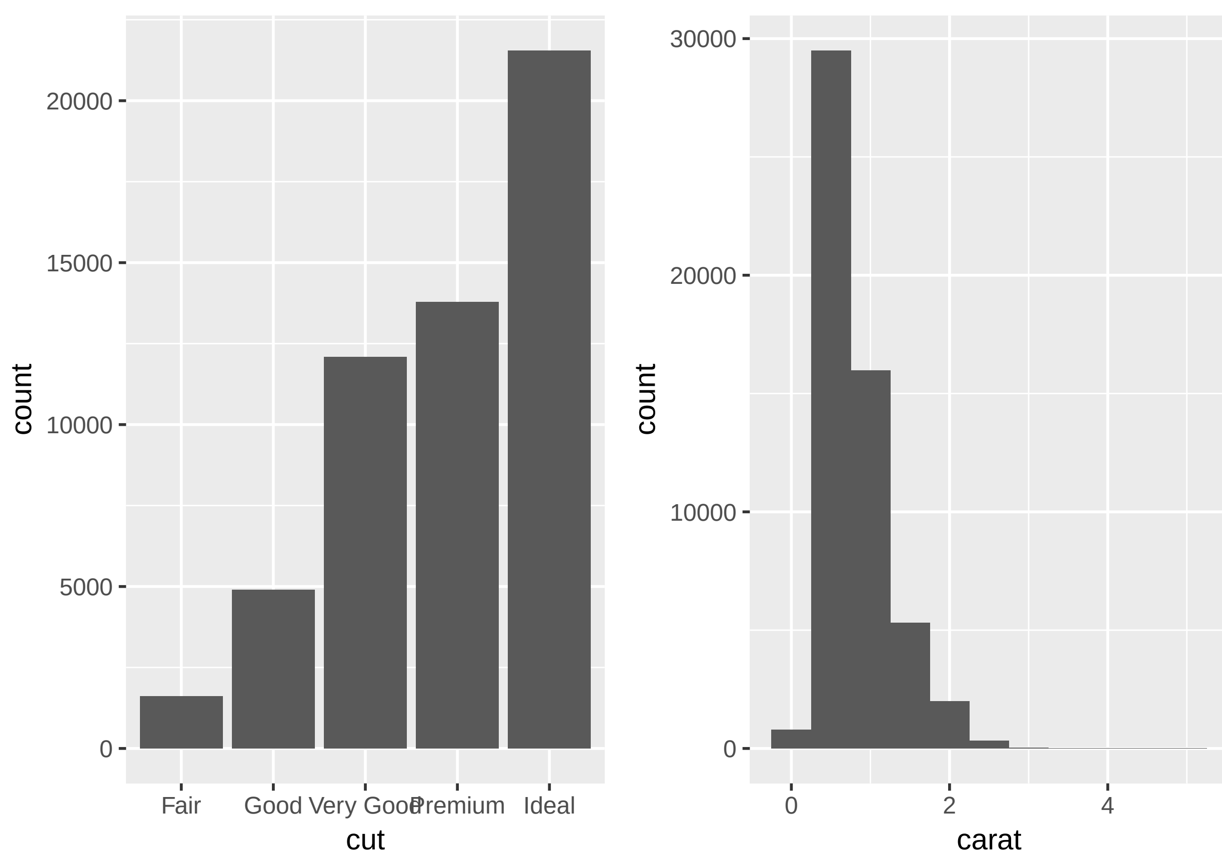
Group 3

May 15, 2018

Technique 1: The Variation of Continious and Categorical Variables (Alireza Mostafizi)

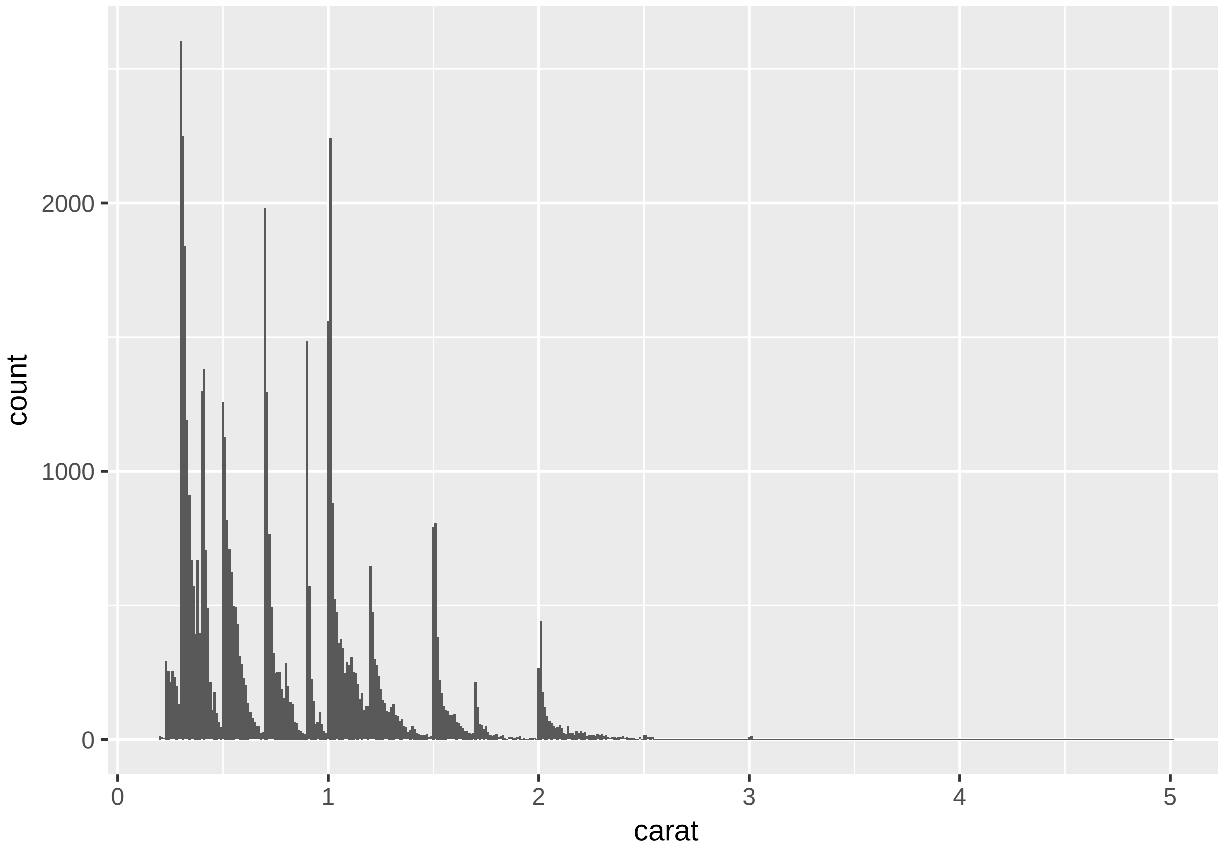
This chapter briefly explaines a few tehcniques to visualized the distribution of the categorical and continious variables. *Categorical Values* are normally visualized with bar plots (*geom_bar*) and *continious values* are typically visualized with histograms (*geom_histogram* or *geom_freqpoly*). The following code plots the distribution of *cut* (categorical) and *carat* (continious) variables from *dimonds* dataset in *ggplot2*.

```
library(ggplot2, tidyverse)
library(gridExtra)
library(dplyr)
# Categorical Variable
cut_dist <- ggplot(data = diamonds) +
            geom_bar(mapping = aes(x = cut))
# Continious Variable
carat_dist <- ggplot(data = diamonds) +
              geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
grid.arrange(cut_dist, carat_dist, ncol = 2)
```



Please note the use of *grid.arrange()* function from the package *ggExtra*. Also, I recommend always trying different *bin_width*. Small *bin_disth* helps you find the most common values in a better way as folllwing,

```r
ggplot(data = diamonds, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.01) # Low bin_width reveals the common values that
```




```r
                                  # were cluttered in the original histogram
```

There might be some small details hidden in large *bin_widths*. In addition, if, for some reason, you rather using any other type of plot, you can generate the dataset with the count values with the following code for both categorical and continious variables.

```r
library(dplyr) ## Needed for the pip function
# categorical variable
diamonds %>% ## %>% is the pipe function. Similar to | in unix systems
  count(cut)
```

```
## # A tibble: 5 x 2
##   cut           n
##   <ord>     <int>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

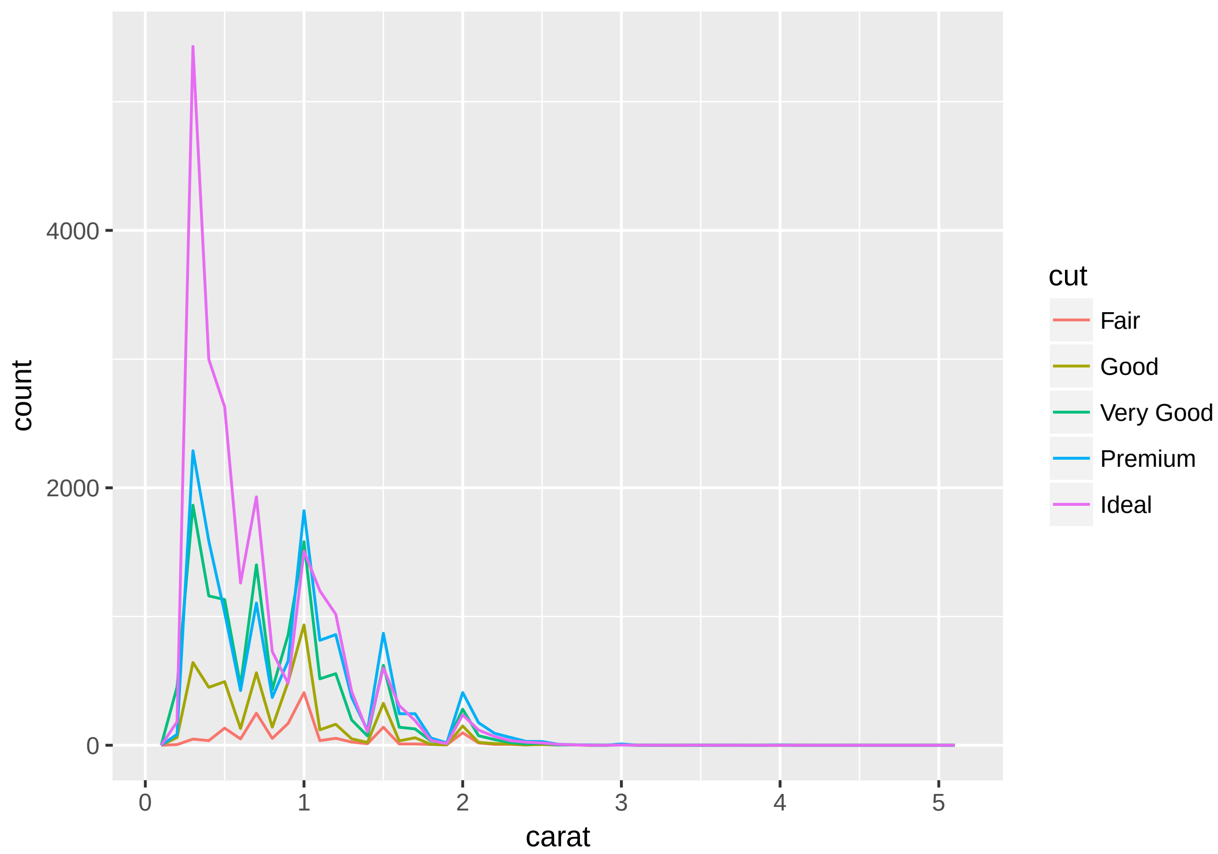
```r
# continous variable
diamonds %>%
  count(cut_width(carat, 0.5)) # binwidth of 0.5
```

```
## # A tibble: 11 x 2
```

```
##    `cut_width(carat, 0.5)`      n
##    <fct>                    <int>
##  1 [-0.25,0.25]               785
##  2 (0.25,0.75]              29498
##  3 (0.75,1.25]              15977
##  4 (1.25,1.75]               5313
##  5 (1.75,2.25]               2002
##  6 (2.25,2.75]                322
##  7 (2.75,3.25]                 32
##  8 (3.25,3.75]                  5
##  9 (3.75,4.25]                  4
## 10 (4.25,4.75]                  1
## 11 (4.75,5.25]                  1
```

Anothr usefull technique for overlaying different historgrams is to use *geom_freqpoly()* instead of *geom_histogram()*. It is exactly the same but instead of bars, a poly line represents the histogram.

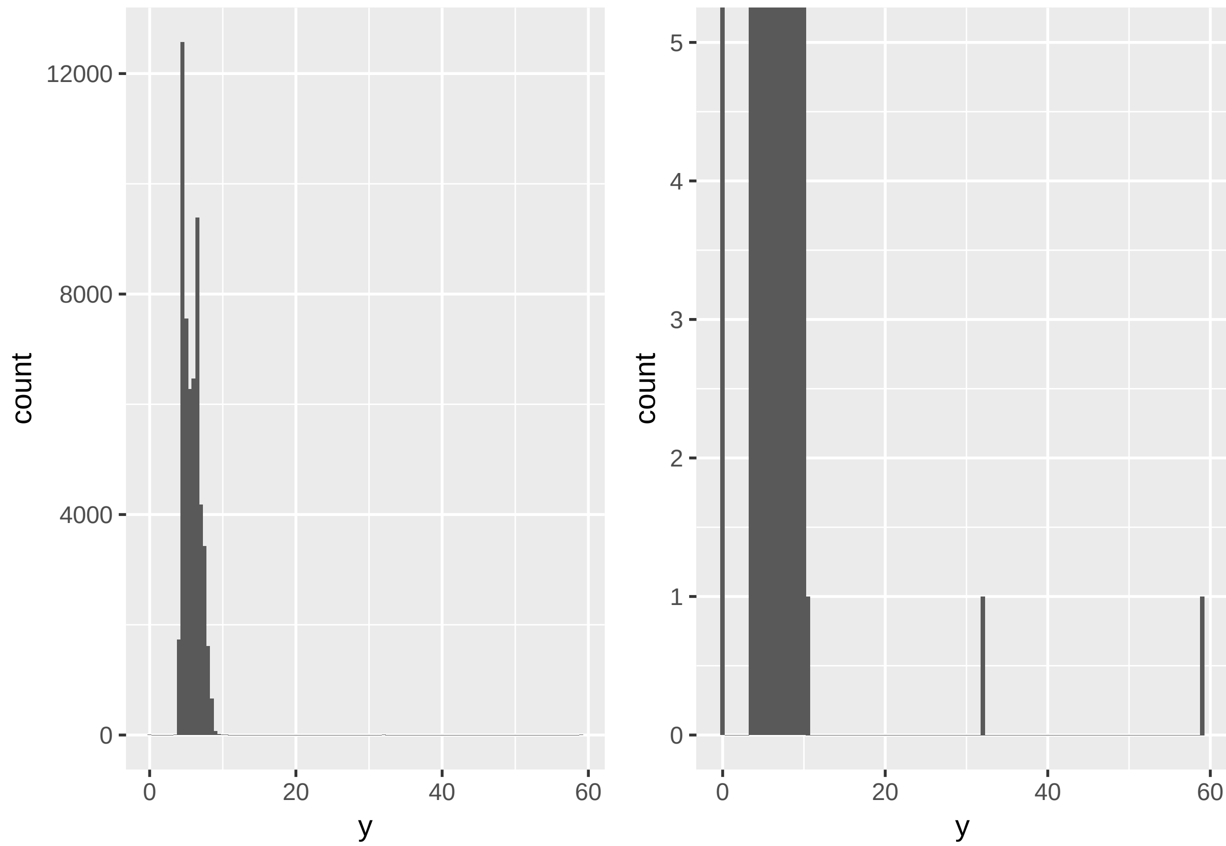
```r
ggplot(data = diamonds, mapping = aes(x = carat, colour = cut)) +
  geom_freqpoly(binwidth = 0.1)
```



Another good technique is zooming into a plot to find the unusual values in a distribution. *coord_cartesian()* function and its parameters, *xlim* and *ylim* come in handy for this purpose. Let's try it on the histogram of *y*, the width of the dimonds.

```r
# Normal histogram
y_dist <- ggplot(data = diamonds) +
          geom_histogram(mapping = aes(x = y), binwidth = 0.5)
# Zoomed in histogran
```

```
y_dist_zoomed <- ggplot(data = diamonds) +
                 geom_histogram(mapping = aes(x = y), binwidth = 0.5) +
                 coord_cartesian(ylim = c(0,5))
grid.arrange(y_dist, y_dist_zoomed, ncol = 2)
```



That clears the unusual values in 0, around 30 and around 60. Alternatively, you can found these observations with *filter()* function from *dplyr* package.

```
diamonds %>%
  filter(y < 3 | y > 20) %>%
  select(price, x, y, z) %>%
  arrange(y)
```

```
## # A tibble: 9 x 4
##   price     x     y     z
##   <int> <dbl> <dbl> <dbl>
## 1  5139  0      0    0
## 2  6381  0      0    0
## 3 12800  0      0    0
## 4 15686  0      0    0
## 5 18034  0      0    0
## 6  2130  0      0    0
## 7  2130  0      0    0
## 8  2075  5.15  31.8  5.12
## 9 12210  8.09  58.9  8.06
```