Exploring a data set in R

Exploratory Data Analysis of the diamonds dataset in ggplot

Initializing the *ggplot* library.

```
library(ggplot2)
library(dplyr)
```

```
##
## 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
```

Visualing the distribution of variables

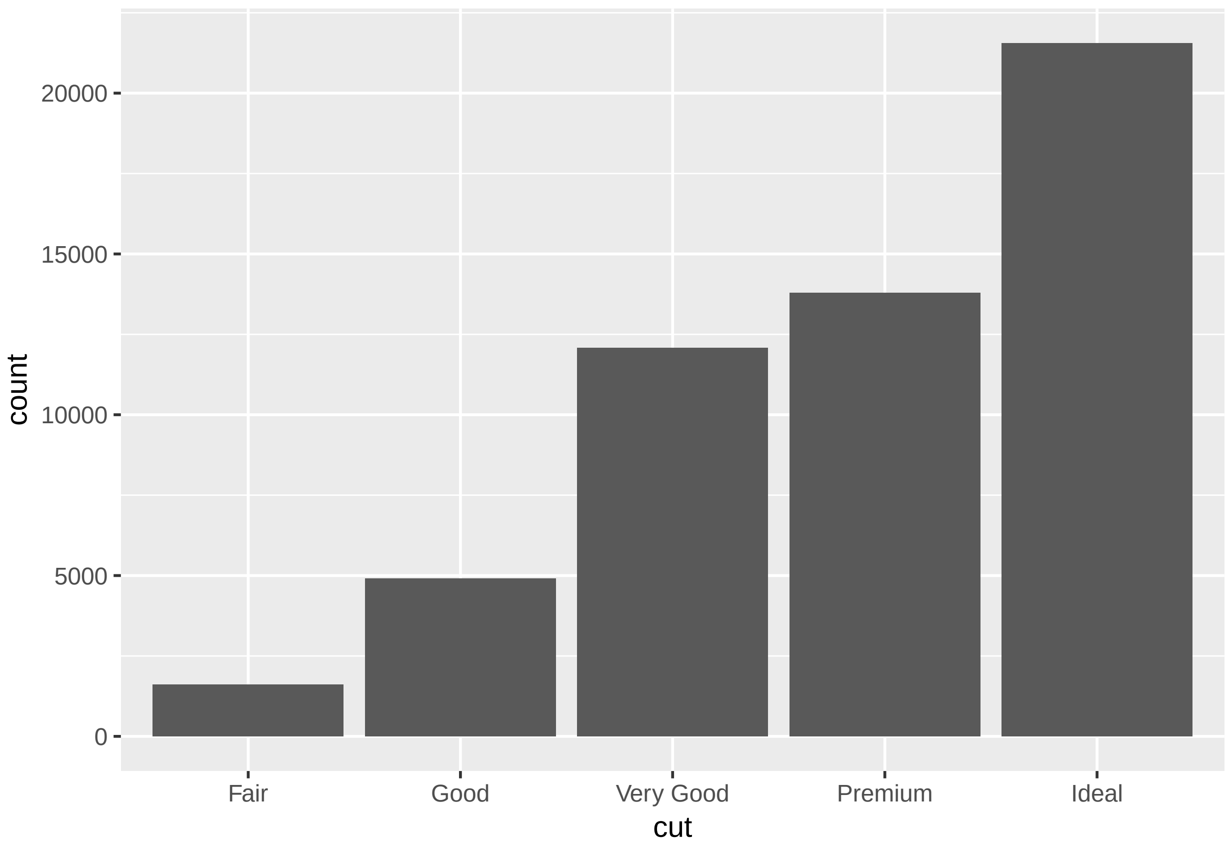
Let's start with understanding the distribution of variables. In order to understand the distribution of variables we can visualize them.

I'll start by understanding the distribution of the categorical variable 'cut' in the data set using a bar plot.

*Visualizing the distribution of a variable depends on the type of the variable. A **categorical** variable can be visualized using **bar plot** and a **continuous** variable can ve visualized using a **histogram***

Visualizing a categorical variable

```
ggplot(data = diamonds) + geom_bar(mapping = aes(x=cut))
```



The height of the bars display how many observations occurred under each category in the variable 'cut'. This can also be done using the count command of dplyr.

```
diamonds %>%
  count(cut)
```

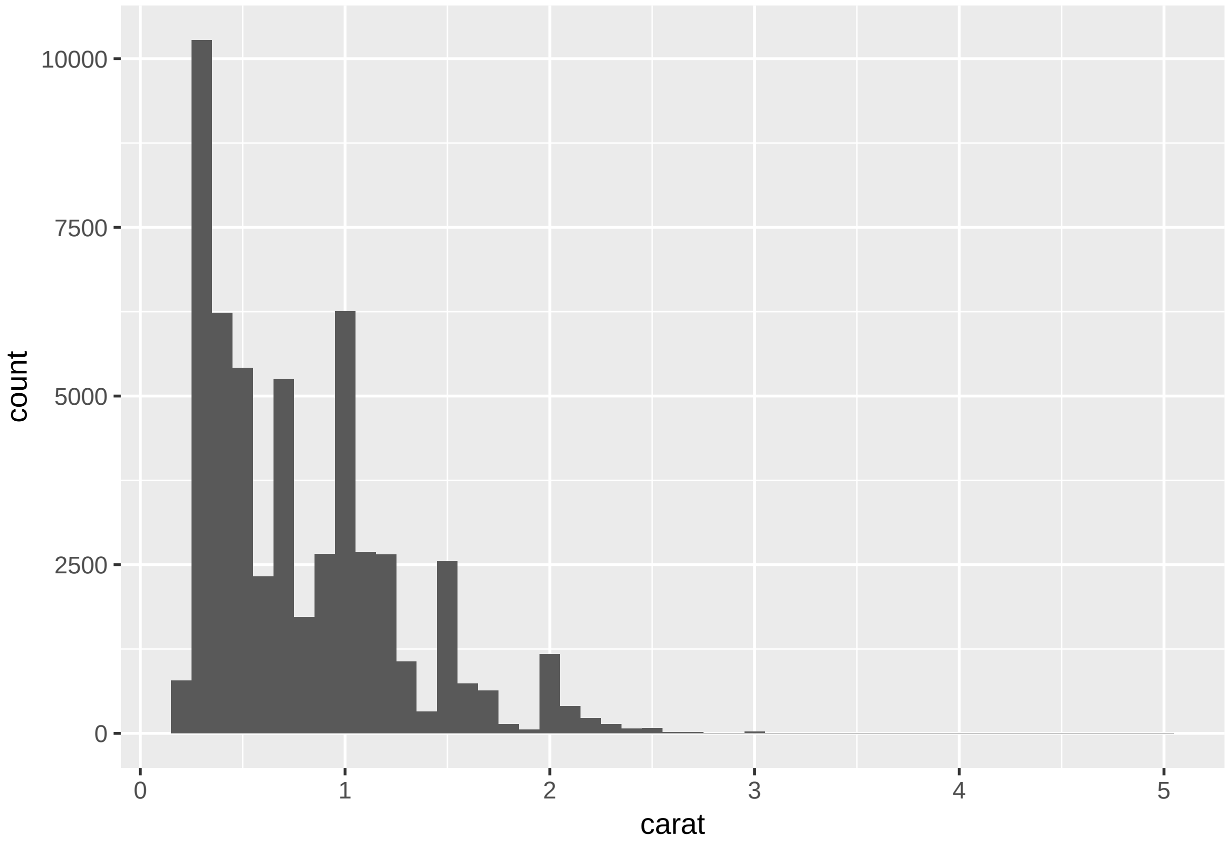
```
## # A tibble: 5 x 2
##   cut           n
##   <ord>     <int>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

From the results above, we can notice that maximum number of diamonds were ideally cut.

Visualizing a continuous variable

A continuous variable can be visualized using using an histogram. A continuous variable can take any value from negative to positive infinity.

```
ggplot(diamonds) + geom_histogram(mapping = aes(x=carat), binwidth = 0.1)
```



A histogram divides the x-axis into equally spaced bins and uses a bar to indicate how many times values from that bin has occurred in the data set. Here, width of the bins are defined by me as 0.1. Different bin width will result in different visualizations.

Frequency of occurrence in each bin can also be seen using the cut_width and count commands in dplyr.

```
diamonds %>%
  count(cut_width(carat, 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
```

From the tibble above, we can see that diamonds within the carat range 0.25 to 0.75 were present the maximum number of times in the data set. Also, maximum number of diamonds are below 3.25 carats, so let's slice the data set for all diamonds below 3 carats and create a smaller data set.

```
smaller_diamonds = diamonds %>%
  filter(carat < 3.25)
```

Let's see the dimensions of the original diamonds data set and the smaller diamonds data set.

```
cat("dimension of the original dataset: ", dim(diamonds))
```

```
## dimension of the original dataset:  53940 10
```

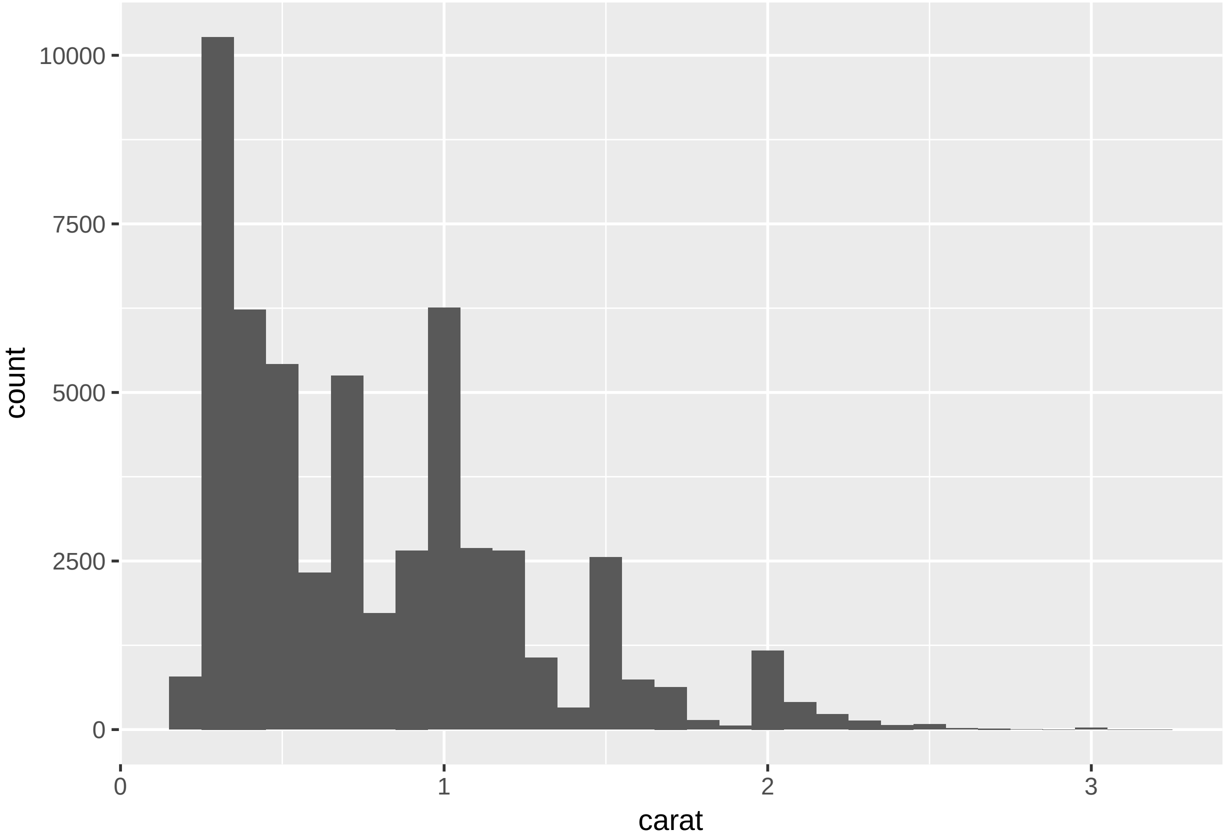
```
cat("\n dimension of the smaller dataset: ", dim(smaller_diamonds))
```

```
##
##  dimension of the smaller dataset:  53929 10
```

So, in the smaller data set we have 11 samples lesser compared to the original data set.

Let's see how the histogram of this data set will look like

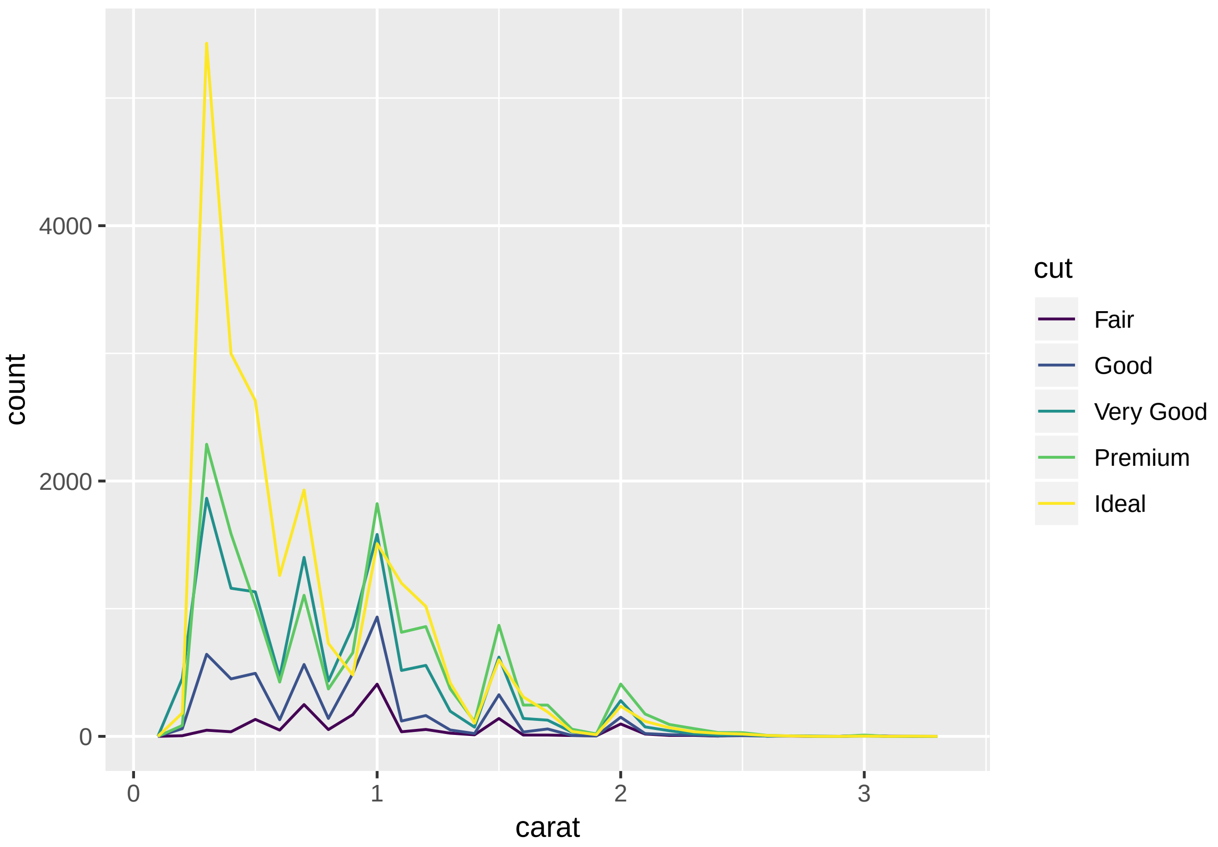
```
ggplot(data = smaller_diamonds) + geom_histogram(mapping = aes(x= carat), binwidth = 0.1)
```



This looks like a zoomed in version of the previous histogram because we have excluded all diamonds below carat value of 3.25.

Let's see the histogram of the carats of the diamonds according to their cut. This can be done by freqpoly. Freqpoly does the same operation as histogram but instead of displaying the counts with bars, it uses lines instead.

```
ggplot(data = smaller_diamonds) + geom_freqpoly(mapping = aes(x= carat, color=cut), binwidth = 0.1)
```

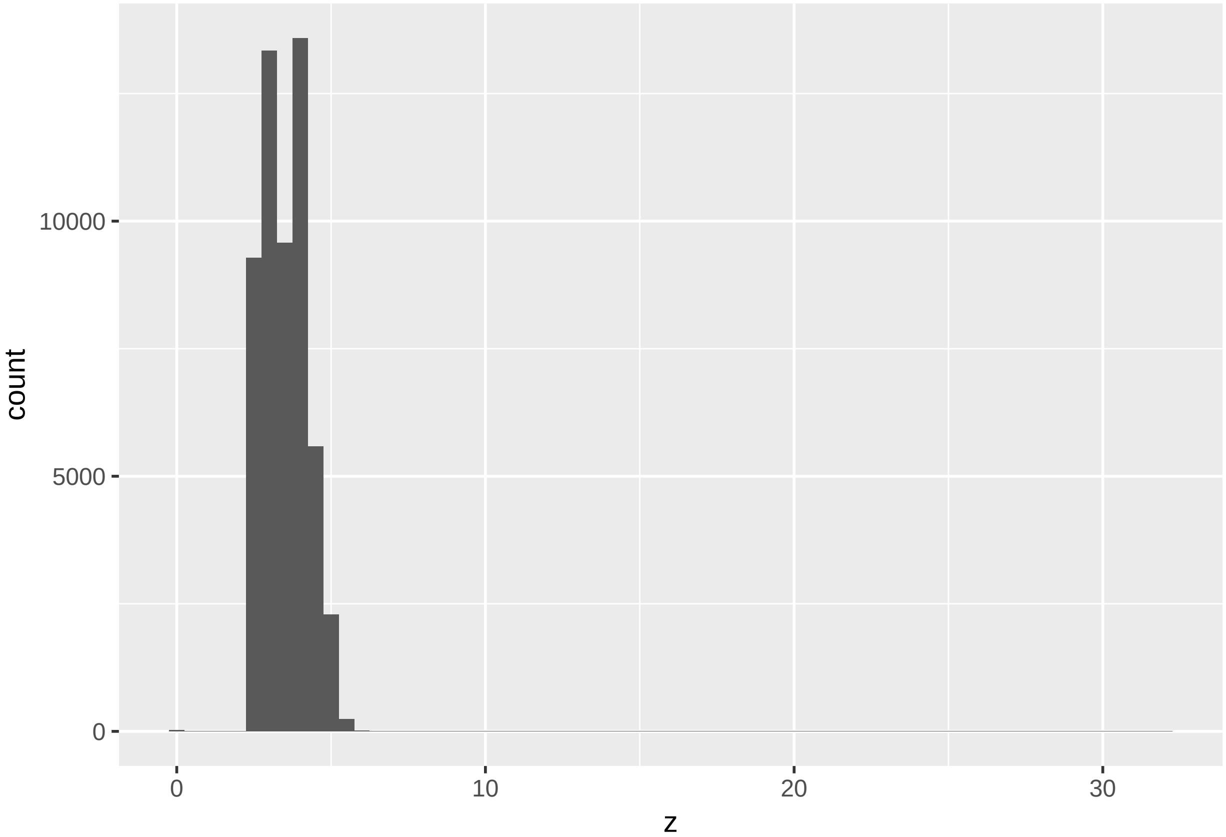


Detecting outliers

Outliers are data points in our data set that doesn't seem to fit a pattern. Sometimes outliers may indicate error in data collection but not all outliers are errors. Sometimes outliers might indicate an important new information. So, before eliminating an outlier we need to justify the reason for elimination.

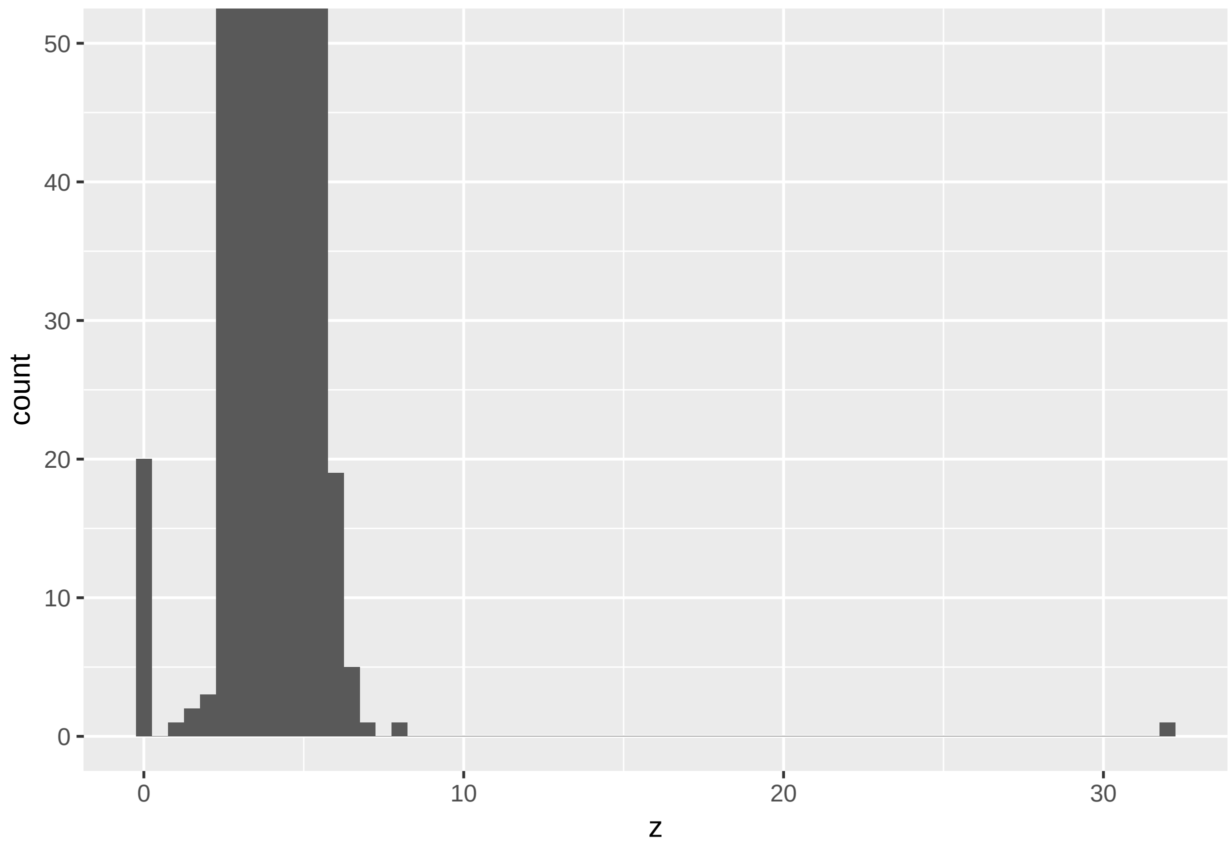
Let's take the variable 'z' from the data set which measures the diamond along the z axis. Let's visualize the distribution of this variable.

```r
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = z), binwidth = 0.5)
```



In the plot above, even though the typical value of z varies from 2 to 6 in the x-axis, the x-axis coordinates in the image is from 0 to 30. This is one indication that there might be outliers which are not visible in the image. So, let's use the coord_cartesian command to zoom into the image along y-axis.

```
ggplot(diamonds) +
  geom_histogram(mapping = aes(x= z), binwidth = 0.5) +
  coord_cartesian(ylim = c(0,50))
```



In the previous image, the count in y-axis was around 12500, so we were not able to see the data point after 30. Now after zooming, we can clearly see two outliers, one around 8 and another after 30 in the x-axis.

Let's see where those points are actually located.

```
diamonds %>%
  filter(z >=6)
```

```
## # A tibble: 14 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  3.65 Fair      H     I1       67.1  53   11668  9.53  9.48  6.38
##  2  2    Premium   H     SI2      58.9  57   12210  8.09 58.9   8.06
##  3  3.5  Ideal     H     I1       62.8  57   12587  9.65  9.59  6.03
##  4  4.01 Premium   I     I1       61    61   15223 10.1  10.1   6.17
##  5  4.01 Premium   J     I1       62.5  62   15223 10.0   9.94  6.24
##  6  2.01 Fair      G     SI2      65.6  56   15562  7.89  7.84  6.16
##  7  3.4  Fair      D     I1       66.8  52   15964  9.42  9.34  6.27
##  8  4    Very Good I     I1       63.3  58   15984 10.0   9.94  6.31
##  9  3.67 Premium   I     I1       62.4  56   16193  9.86  9.81  6.13
## 10  4.13 Fair      H     I1       64.8  61   17329 10     9.85  6.43
## 11  5.01 Fair      J     I1       65.5  59   18018 10.7  10.5   6.98
## 12  4.5  Fair      J     I1       65.8  58   18531 10.2  10.2   6.72
## 13  3.51 Premium   J     VS2      62.5  59   18701  9.66  9.63  6.03
## 14  0.51 Very Good E     VS1      61.8  54.7  1970  5.12  5.15 31.8
```

This seems to be a lot of data. But, we actually only see 2 points on the x-axis. So, let's refine it even further.

```r
diamonds %>%
  filter(z >= 7)
```

```
## # A tibble: 2 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  2    Premium   H     SI2      58.9  57   12210  8.09 58.9   8.06
## 2  0.51 Very Good E     VS1      61.8  54.7  1970  5.12  5.15 31.8
```

YES!! Now we got only two points and they match with the visual information also. The value of the z variable of the first point is around 8 and the value of the second point is above 30.

x, y and z variables in the data set indicates the x, y and z axis dimensions of the diamond. Let's just assume that x is the length, y is the breadth and z is the height of the diamond. (This is just an assumption!)

In order to see how different these two values of z are from the other values of z let's compare these two values (8.06 and 31.8) with the mean and median values of the variable z.

```r
cat("mean of the variable z: ", mean(diamonds$z))
```

```
## mean of the variable z:  3.538734
```

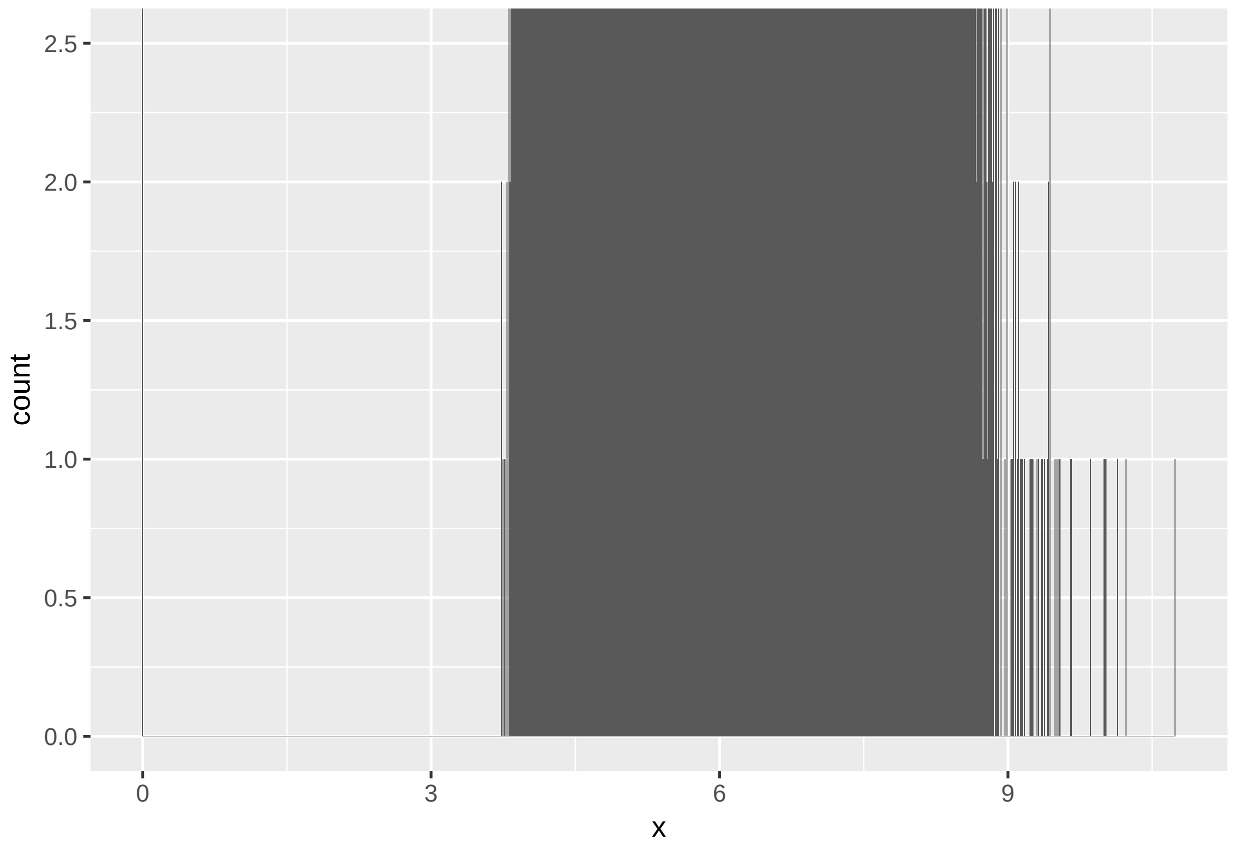
```r
cat("\n median of the variable z: ", median(diamonds$z))
```

```
##
##  median of the variable z:  3.53
```

As we see both the mean and median values of the variable is around 3.5. Now the 2 values we detected as outlier are 8.06 and 31.8 which are definitely very different from the mean and median value of the distribution. So, it's safe to consider these two values as errors. This is the reason why we eliminate these two values as outliers.

In the same way let's investigate the variable x and y also

```r
ggplot(data = diamonds) +
  geom_histogram(mapping = aes(x=x), binwidth = 0.01) +
  coord_cartesian(ylim = c(0, 2.5))
```



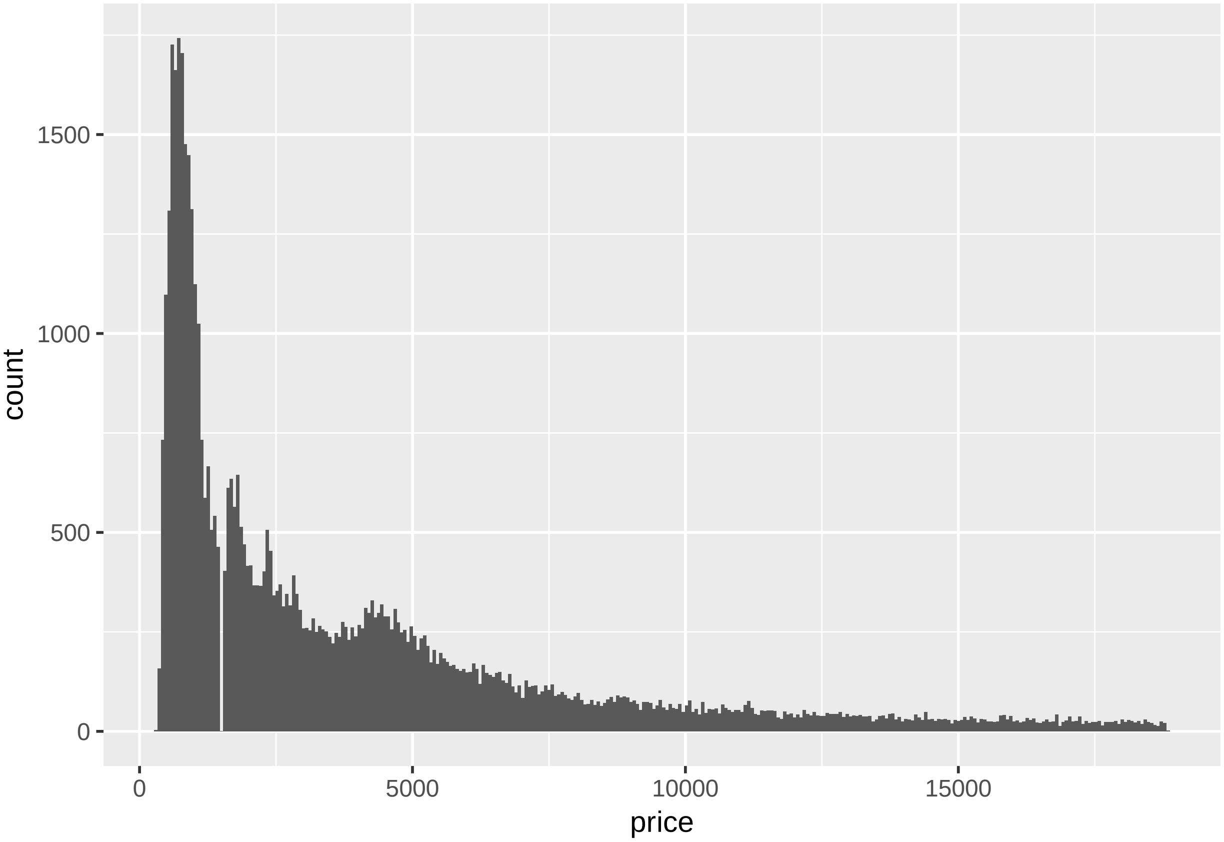
Here, we have a set of values in x which is either 0 or a little less. We can't see it clearly in visualization so let's filter those data points and see them as a table.

```
diamonds %>%
  filter(x < 3)
```

```
## # A tibble: 8 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  1.07 Ideal     F     SI2      61.6    56  4954     0  6.62     0
## 2  1    Very Good H     VS2      63.3    53  5139     0  0        0
## 3  1.14 Fair      G     VS1      57.5    67  6381     0  0        0
## 4  1.56 Ideal     G     VS2      62.2    54 12800     0  0        0
## 5  1.2  Premium   D     VVS1     62.1    59 15686     0  0        0
## 6  2.25 Premium   H     SI2      62.8    59 18034     0  0        0
## 7  0.71 Good      F     SI2      64.1    60  2130     0  0        0
## 8  0.71 Good      F     SI2      64.1    60  2130     0  0        0
```

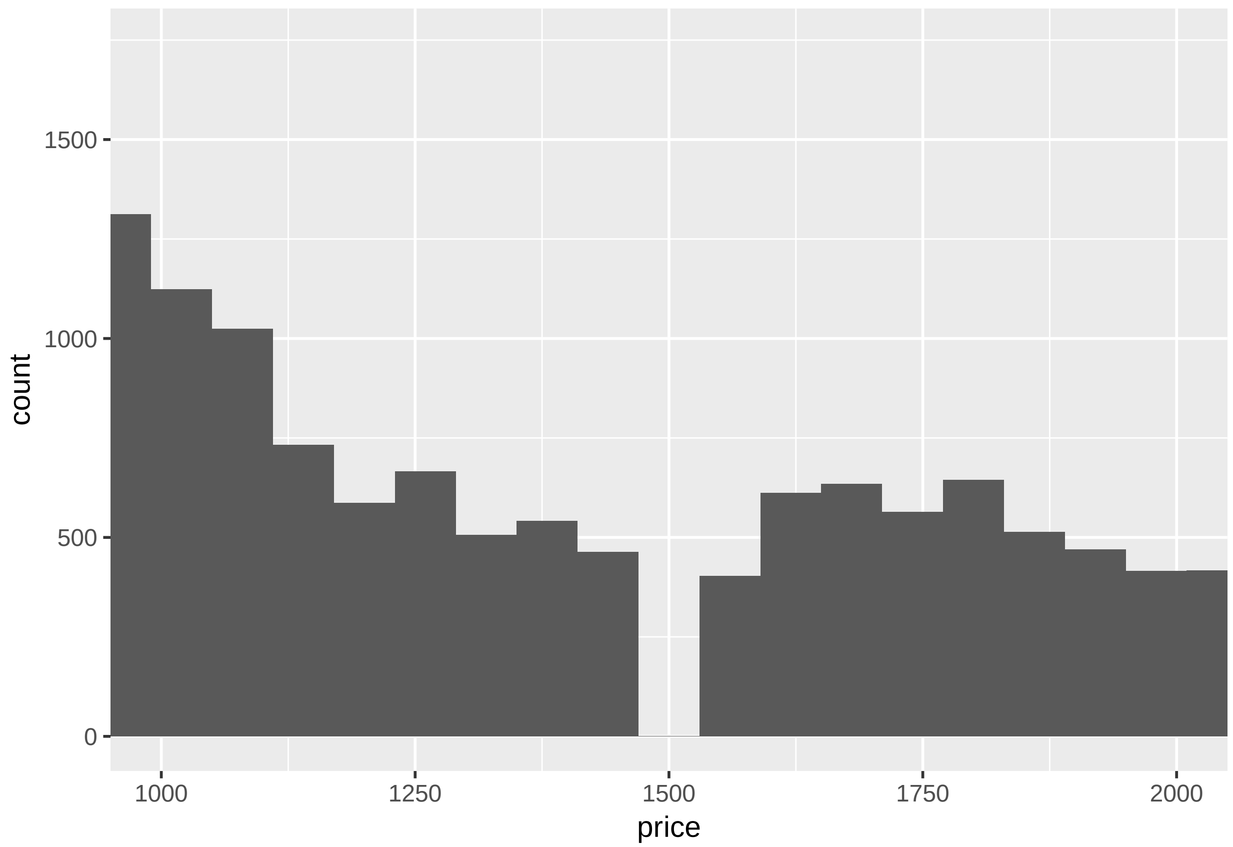
As we can see, all the points of the x variable which is less than 3 is 0. Except one of these points, the values of the other two variables y and z are also 0. Since these 3 variables are the dimension measurements of the diamond and dimensions cannot be 0, we consider these 3 data points to be erroneous entries and that's the reason why we eliminate these as outliers.

```
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = price), binwidth = 60)
```



As we can see in the image above, there are certainly 2 clusters of diamonds. The first cluster on the left has low cost and there are more of those diamonds in our dataset. The second cluster on the right has high and wide price range but the number of diamonds in those clusters are low. Let's zoom the image a little bit more to clearly visualize both those clusters.

```
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = price), binwidth =60) +
  coord_cartesian(xlim = c(1000, 2000))
```



There it is!, there are no diamonds in the price range 1400 to 1600.

Another interesting question that I found about this data set on the internet is, why is there a huge difference between the number of diamonds which are 0.99 carat and 1 carat.

Let's analyze a little bit more about carats now.

```
diamonds %>%
  filter(carat == 0.99)
```

```
## # A tibble: 23 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.99 Fair      I     SI2      68.1    56  2811  6.21  6.06  4.18
##  2  0.99 Fair      J     SI1      55      61  2812  6.72  6.67  3.68
##  3  0.99 Fair      J     SI1      58      67  2949  6.57  6.5   3.79
##  4  0.99 Fair      I     SI1      60.7    66  3337  6.42  6.34  3.87
##  5  0.99 Fair      H     VS2      71.6    57  3593  5.94  5.8   4.2
##  6  0.99 Very Good J     SI1      60.3    57  4002  6.44  6.49  3.9
##  7  0.99 Good      F     SI2      63.3    54  4052  6.36  6.43  4.05
##  8  0.99 Premium   F     SI2      60.6    61  4075  6.45  6.38  3.89
##  9  0.99 Ideal     I     SI1      61.8    57  4763  6.4   6.42  3.96
## 10  0.99 Very Good E     SI2      61.8    59  4780  6.3   6.33  3.9
## # ... with 13 more rows
```

There are about 23 diamonds which are 0.99 carats

```
diamonds %>%
  filter(carat == 1)
```

```
## # A tibble: 1,558 x 10
##    carat cut     color clarity depth table price     x     y     z
##    <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1     1 Premium I     SI2      58.2    60  2795  6.61  6.55  3.83
##  2     1 Premium J     SI2      62.3    58  2801  6.45  6.34  3.98
##  3     1 Fair    G     I1       66.4    59  2808  6.16  6.09  4.07
##  4     1 Fair    J     VS2      65.7    59  2811  6.14  6.07  4.01
##  5     1 Premium H     I1       61.3    60  2818  6.43  6.39  3.93
##  6     1 Fair    H     SI2      65.3    62  2818  6.34  6.12  4.08
##  7     1 Premium F     I1       58.9    60  2841  6.6   6.55  3.87
##  8     1 Fair    G     SI2      67.8    61  2856  5.96  5.9   4.02
##  9     1 Fair    I     SI1      67.9    62  2856  6.19  6.03  4.15
## 10     1 Fair    H     SI2      66.1    56  2856  6.21  5.97  4.04
## # ... with 1,548 more rows
```

There are 1558 diamonds which are 1 carat in the data set.

In order to know a little more, let's summarize all the variables for diamonds of 0.99 and 1 carats.

```
diamonds %>%
  group_by(carat) %>%
  filter(carat == 0.99 | carat == 1) %>%
  summarise(average_price = mean(price), num_of_diamonds = n(), how_many_colors = n_distinct(color),
            how_many_cuts = n_distinct(cut), no_of_clarity = n_distinct(clarity))
```

```
## # A tibble: 2 x 6
##   carat average_price num_of_diamonds how_many_colors how_many_cuts
##   <dbl>         <dbl>           <int>           <int>         <int>
## 1  0.99         4406.              23               7             5
## 2  1            5242.            1558               7             5
## # ... with 1 more variable: no_of_clarity <int>
```

```
diamonds %>%
  group_by(carat) %>%
  filter(carat == 0.99 | carat == 1) %>%
  summarise(avg_depth = mean(depth), avg_x = mean(x),
            avg_y = mean(y), avg_z = mean(z), avg_table = mean(table))
```

```
## # A tibble: 2 x 6
##   carat avg_depth avg_x avg_y avg_z avg_table
##   <dbl>     <dbl> <dbl> <dbl> <dbl>     <dbl>
## 1  0.99      62.6  6.36  6.34  3.97      58.0
## 2  1         62.0  6.38  6.36  3.95      58.6
```

From the 2 results above, we can notice that there are no significant differences between any two variables except average_price and no_of_clarity.

From a quick internet search I realized that 0.99 carat diamonds are rarely made than 1 carat diamonds. Maybe that could be a reason why there are more 1 carat diamonds and less 0.99 carat diamonds.

How to deal with strange values aka outliers

The best way to deal with outliers in a data set is to understand why those values are outliers. This option may not be easy in many cases, so, we can do either one of the two options below.

1. Drop the entire row with strange values, however, just because one value is strange, it doesn't mean that other values in the row are faulty also. If we have a really faulty data set, by the time we end up

dropping outliers, the entire data set will be empty.

First, let's display a set of stange values which needs to be eliminated.

```
diamonds %>%
  filter(y <3 | y > 20)
```

```
## # A tibble: 9 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  1    Very Good H     VS2      63.3    53  5139  0      0     0
## 2  1.14 Fair      G     VS1      57.5    67  6381  0      0     0
## 3  2    Premium   H     SI2      58.9    57 12210  8.09  58.9  8.06
## 4  1.56 Ideal     G     VS2      62.2    54 12800  0      0     0
## 5  1.2  Premium   D     VVS1     62.1    59 15686  0      0     0
## 6  2.25 Premium   H     SI2      62.8    59 18034  0      0     0
## 7  0.51 Ideal     E     VS1      61.8    55  2075  5.15  31.8  5.12
## 8  0.71 Good      F     SI2      64.1    60  2130  0      0     0
## 9  0.71 Good      F     SI2      64.1    60  2130  0      0     0
```

As displayed in the table above, when y value is less than 3 the value of both x and z are 0 also. These 3 variables indicate the dimensions of the diamond. As we all know dimensions cannot be 0. So they need to be removed.

When the value of y is above 20, the value is really huge. Let's see the median value of y first and then check if these 2 values are very different from the median.

```
diamonds %>%
 summarise(Median_Y= median(y))
```

```
## # A tibble: 1 x 1
##   Median_Y
##      <dbl>
## 1     5.71
```

As we see the median value of y is 5.71 and the two values of y which are greater than 20 are 58.9 and 31.8 which are very different from the median value.

So, in conclusion, let's filter out all the values of y below 3 and above 20.

```
diamonds2 <- diamonds %>%
  filter(between(y, 3, 20))
```

2. Instead of dropping the outlier values, we can replace them with NA. NA values are easily usable in R, GGPLOT for example, doesn't include NAs while plotting, but it will warn after the plot that some NA values were dropped.

```
diamonds2 <- diamonds %>%
  mutate(y = ifelse(y <3 | y > 20, NA, y))
```

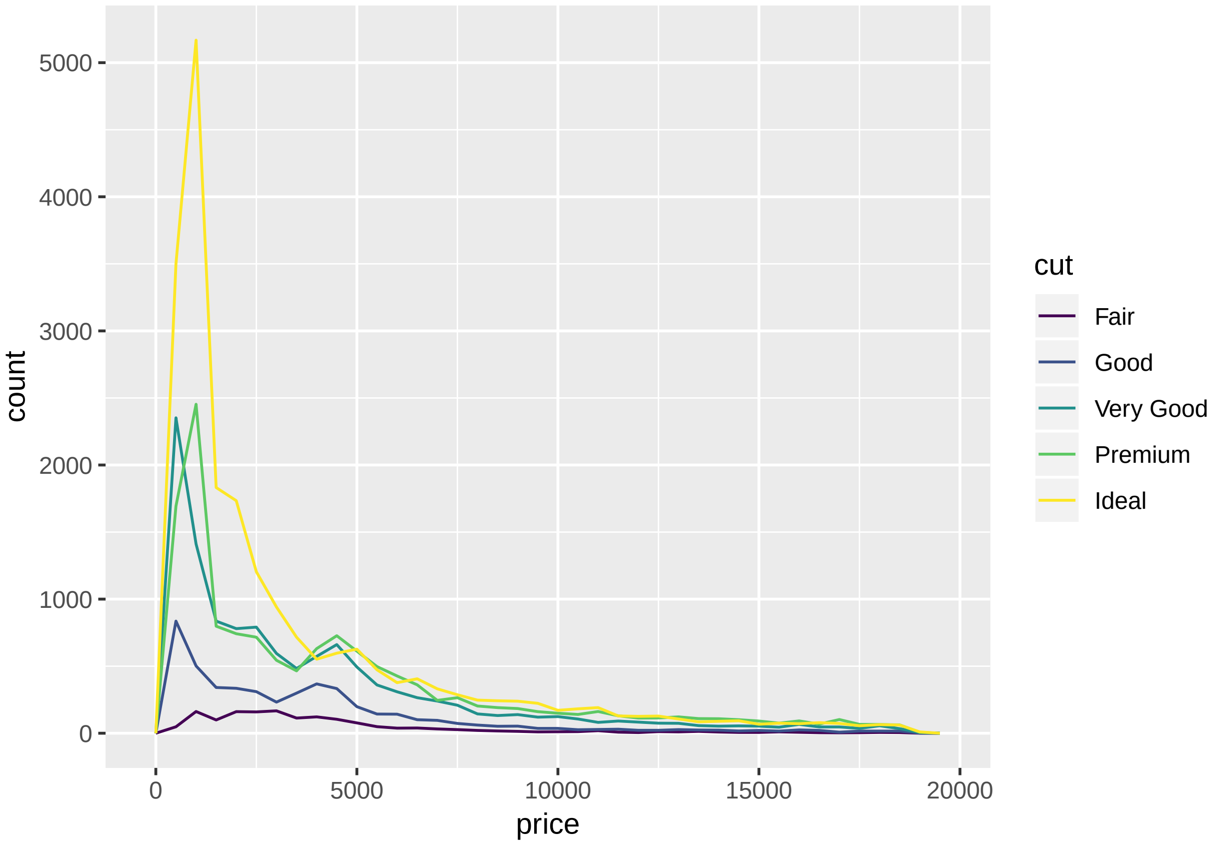
Covariation

By analysing the distributions of variables we understood the behaviour within a variable. By covariation let's understand how two or more variables vary together.

Variation between a categorical and a continuous variable

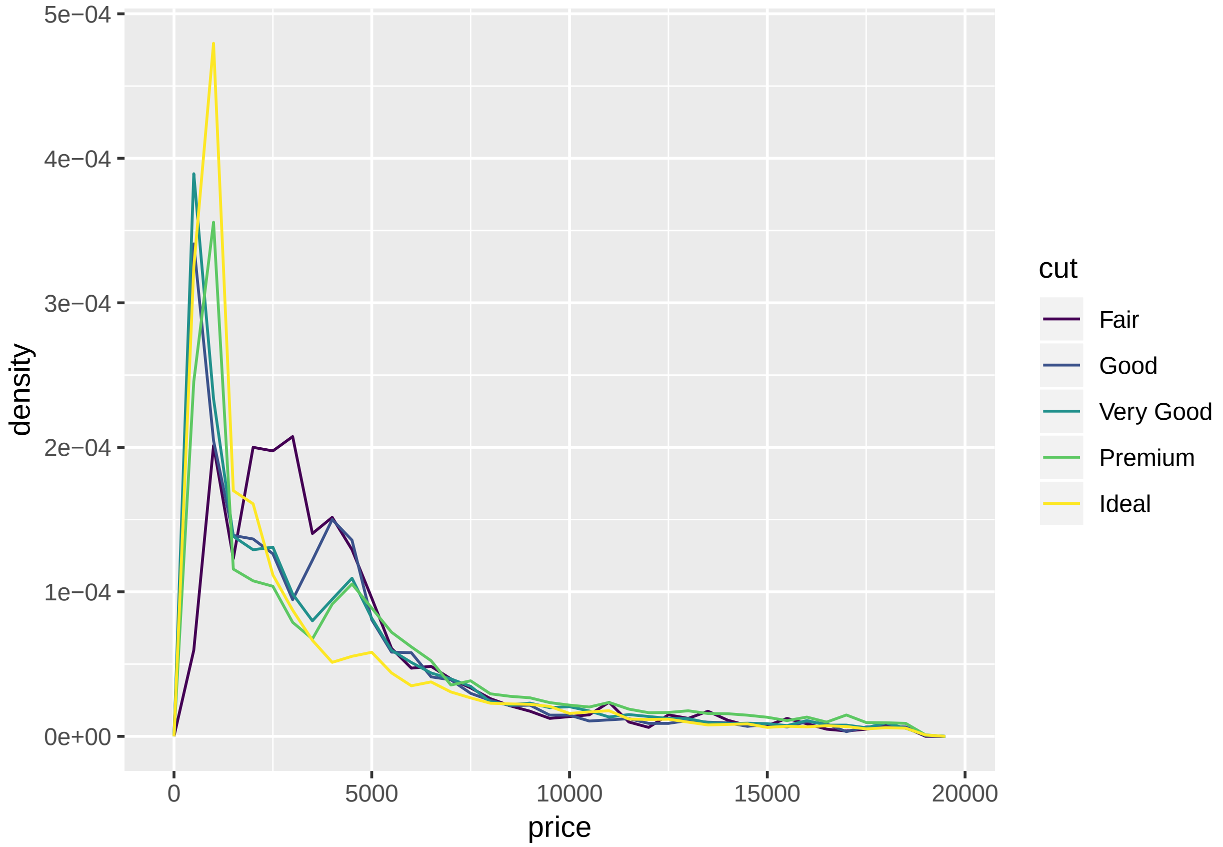
Let's visualize how a categorical variable (cut) and a continuous variable (price) vary together.

```
ggplot(data = diamonds, mapping = aes(x = price)) + geom_freqpoly(mapping = aes(color = cut), binwidth =
```



Since diamonds of some cut are very large in number in the data set compared to others, it's hard to see the shape of diamonds of lower numbers. Let's try something else.

```
ggplot(data = diamonds, mapping = aes(x=price, y = ..density..)) +
  geom_freqpoly(mapping = aes(color = cut), binwidth = 500)
```

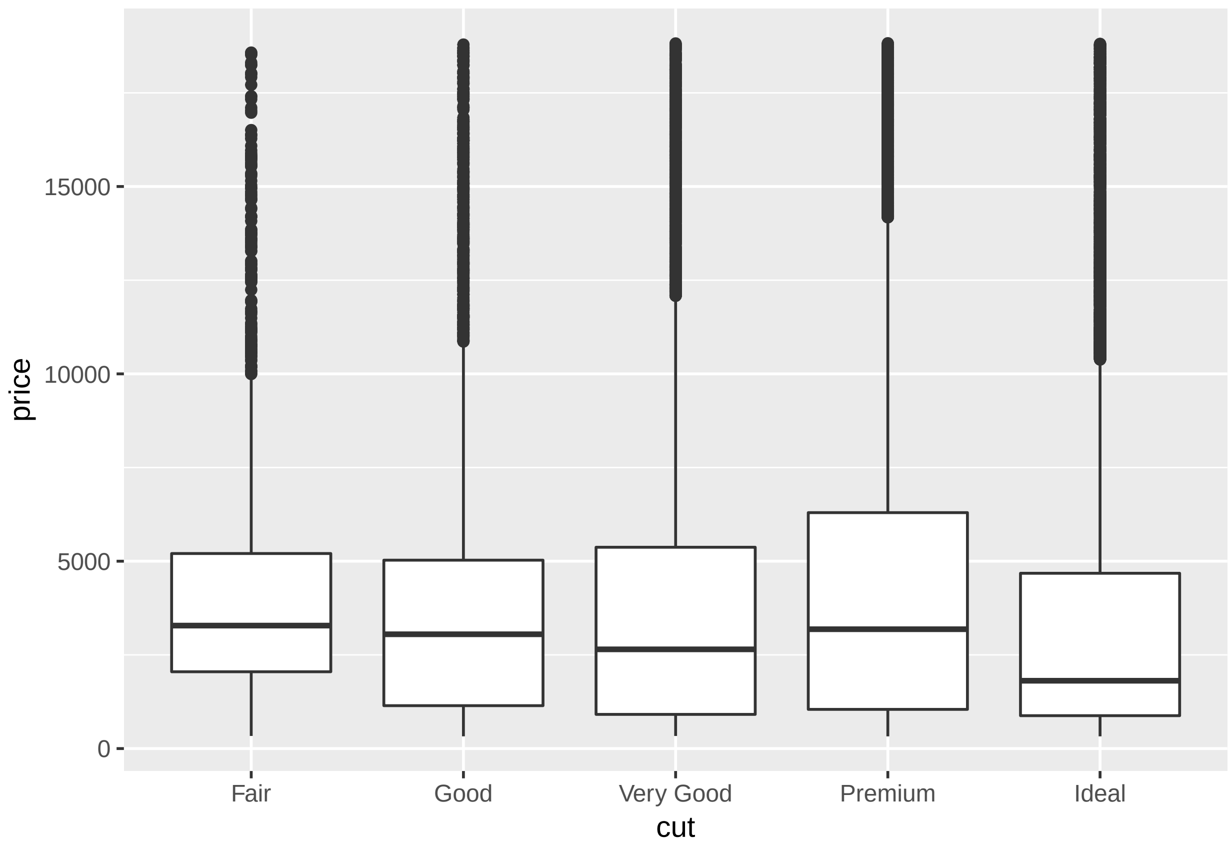


An interesting information that got revelead in this plot is that the price of fair cut diamonds whose quality is lower than ideal cut diamonds has a higher price than ideal cut diamonds.

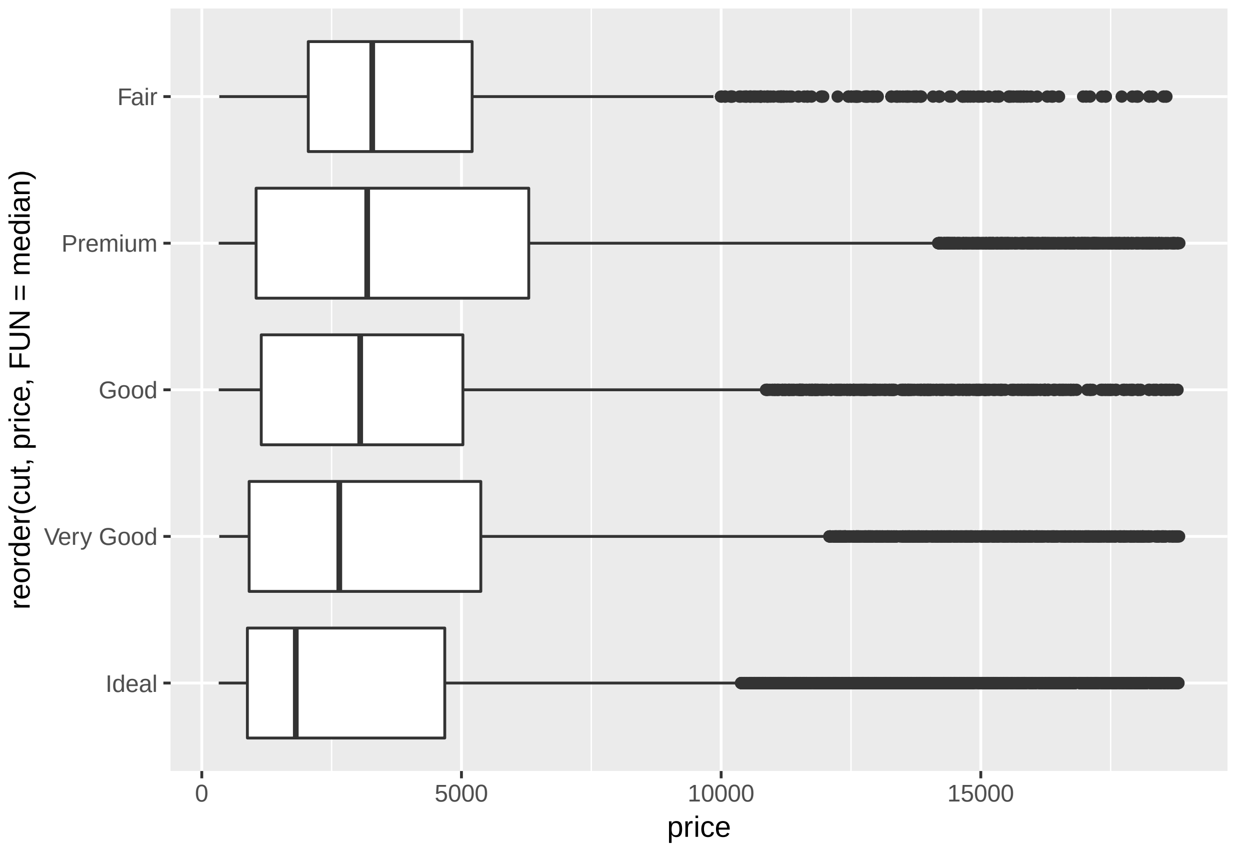
Boxplots

Boxplots are another useful method to compare the distribution of a variable.

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



```
ggplot(data = diamonds,
       mapping = aes(x= reorder(cut, price, FUN = median) , y = price)) + geom_boxplot() +
  coord_flip()
```



Now, it becomes a bit more clear, the median price of fair diamonds is higher than ideal and even very good diamonds.

Variation between two categorical variables

One method to find covariation between continuous variables is to count how many values are common between those two variables. This can be done through dplyr and also through ggplot visualizations.

1. dplyr method

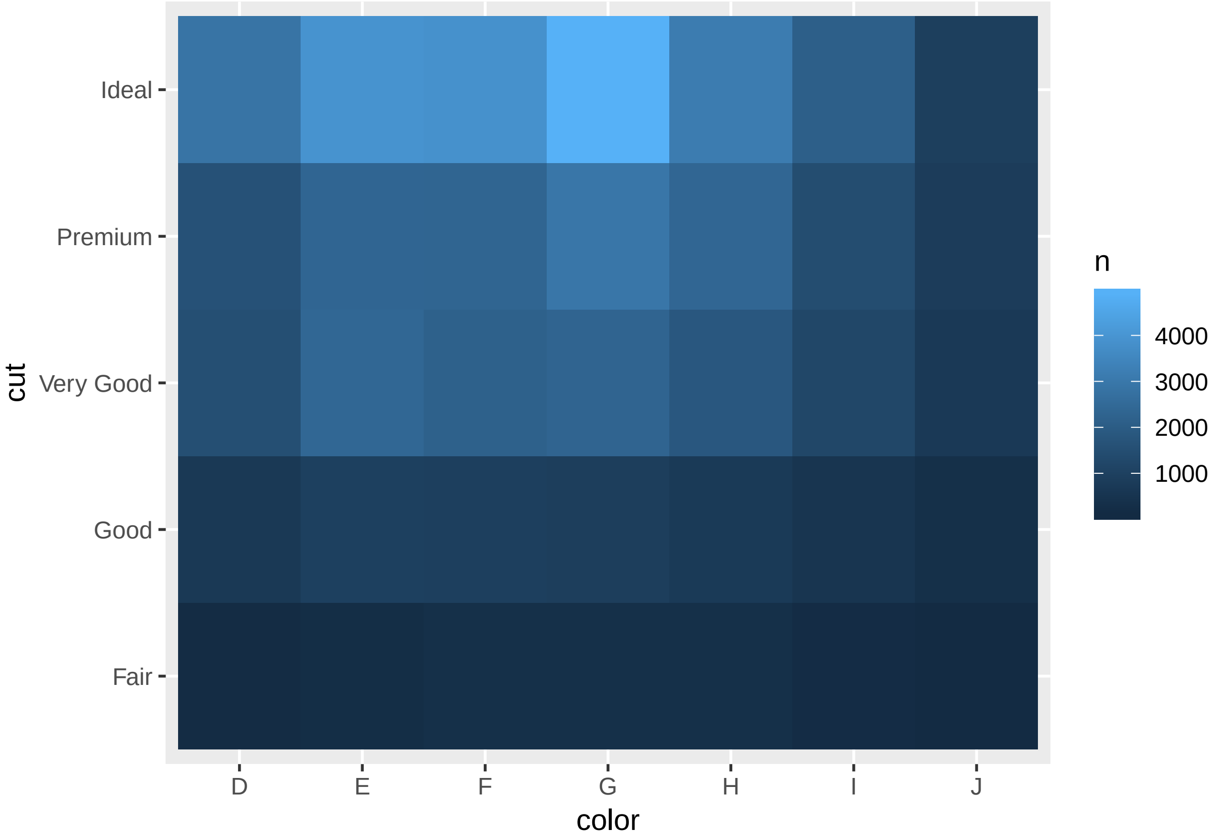
```
diamonds %>%
  count(cut, color)
```

```
## # A tibble: 35 x 3
##    cut   color     n
##    <ord> <ord> <int>
##  1 Fair  D       163
##  2 Fair  E       224
##  3 Fair  F       312
##  4 Fair  G       314
##  5 Fair  H       303
##  6 Fair  I       175
##  7 Fair  J       119
##  8 Good  D       662
##  9 Good  E       933
## 10 Good  F       909
## # ... with 25 more rows
```

In the table above, we can see the number of diamonds which has a certain cut and color.

This table could be further visualized.

```
diamonds %>%
  count(color, cut) %>%
  ggplot(mapping = aes(x = color, y = cut)) +
  geom_tile(mapping = aes(fill = n))
```

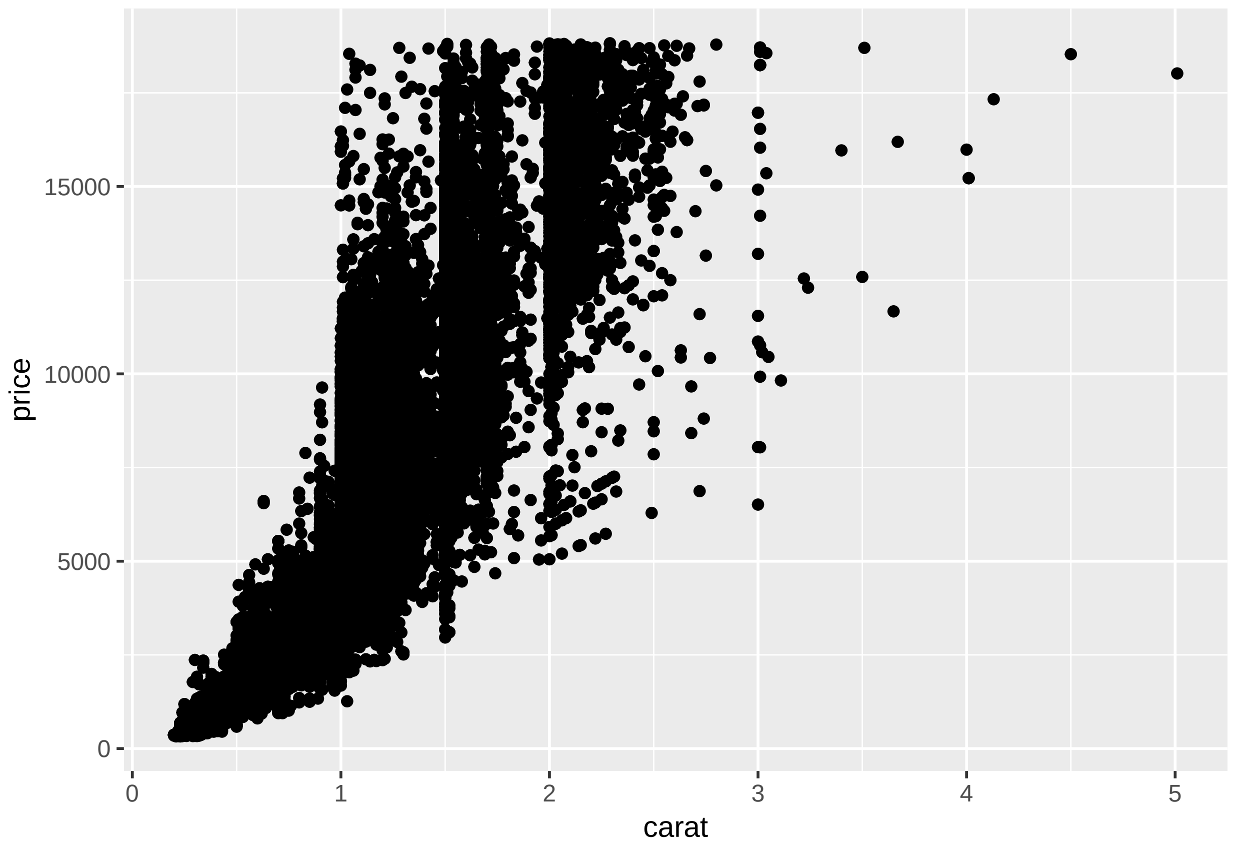


From the visulatization above, we can see that ideally cut diamonds with color G were maximum in number. There are very less number of fair diamonds in all colors.

Variation between two continuous variables

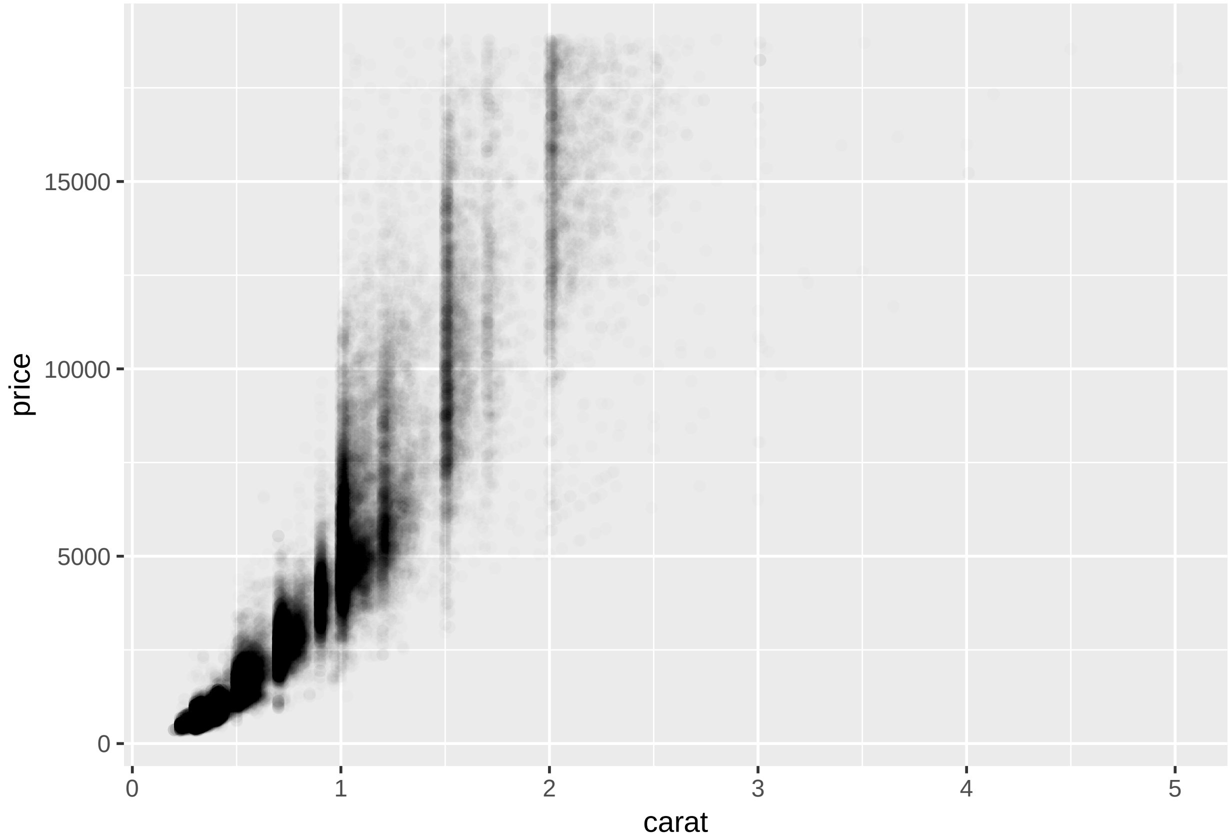
The simplest method to visualize the variation between two continuous variables is the scatter plot.

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



Let’s reduce the point size so we can see clusters clearly.

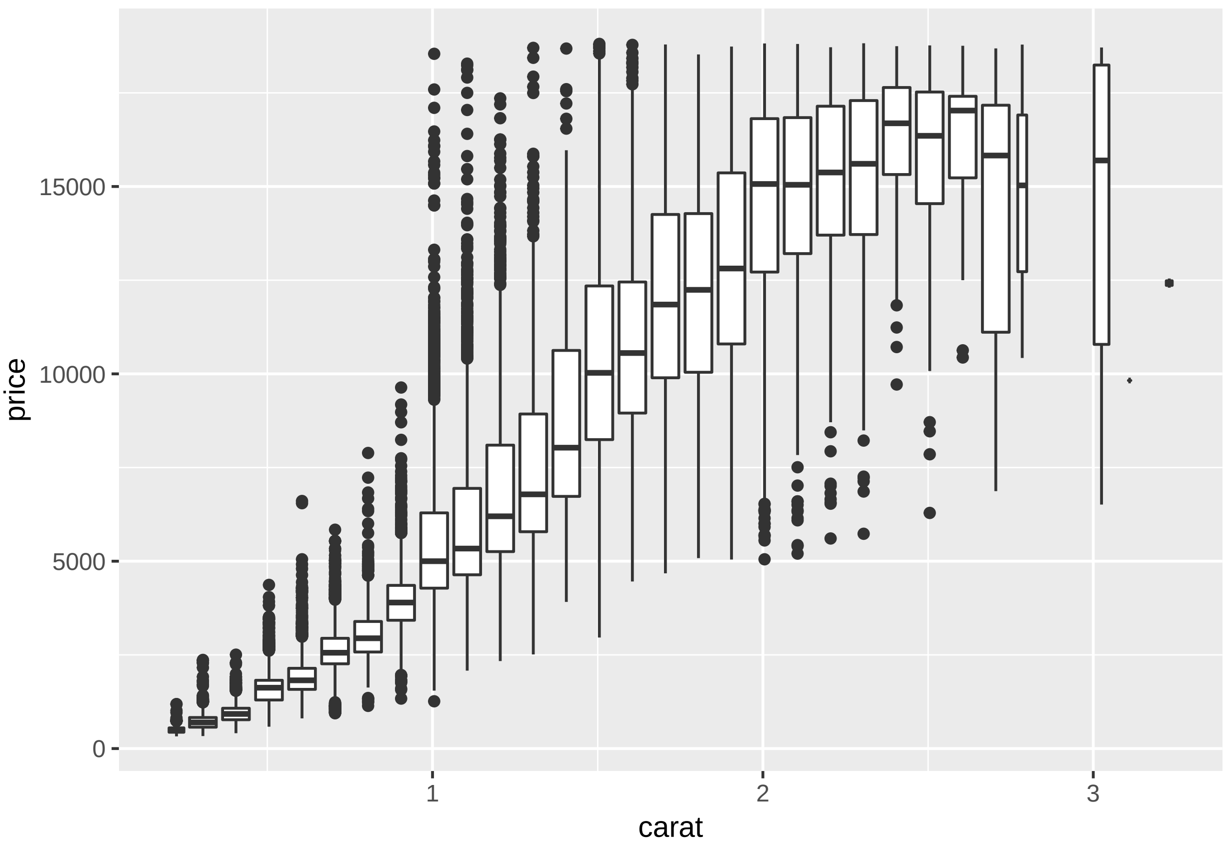
```
ggplot(data = diamonds)  + geom_point(mapping = aes(x = carat, y = price), alpha = 1/100)
```



Now we can see the clusters even more clearly.

Another option is to group one of the continuous variables into bins so it becomes like categorical variable and then we can use one of the methods we used above to find variation between categorical and continuous variables.

```
ggplot(data = smaller_diamonds, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_width(carat, 0.1)))
```


price
15000
10000
5000
0
1
2
3
carat