Using R with Single Variables

Working with Diamonds Dataset

Loading the diamonds data set.

I noticed that the diamonds data set will be listed as a 'Promise' in the workspace. This is a special object in R, and I need to run a command on the data to fully load the data set. I need to run summary in order to get the data appear in the environments.

```
#must load the ggplot package first
library(ggplot2)

#loads the diamonds data set since it comes with the ggplot package
data(diamonds)

summary(diamonds)
```

```
##      carat               cut        color       clarity     
##  Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065  
##  1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258  
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194  
##  Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171  
##  3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066  
##  Max.   :5.0100                     I: 5422   VVS1   : 3655  
##                                     J: 2808   (Other): 2531  
##      depth           table           price             x         
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000  
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710  
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700  
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731  
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540  
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740  
##                                                                  
##        y                z         
##  Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 4.720   1st Qu.: 2.910  
##  Median : 5.710   Median : 3.530  
##  Mean   : 5.735   Mean   : 3.539  
##  3rd Qu.: 6.540   3rd Qu.: 4.040  
##  Max.   :58.900   Max.   :31.800  
## 
```

There are 53940 observations and 10 variables in the dataset. There are 3 ordered factors in the dataset: cut, color, clarity

```
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 ...
```

```
levels(diamonds$color)
```

```
## [1] "D" "E" "F" "G" "H" "I" "J"
```

According to the documents, level 'D' is the best level for the diamonds.

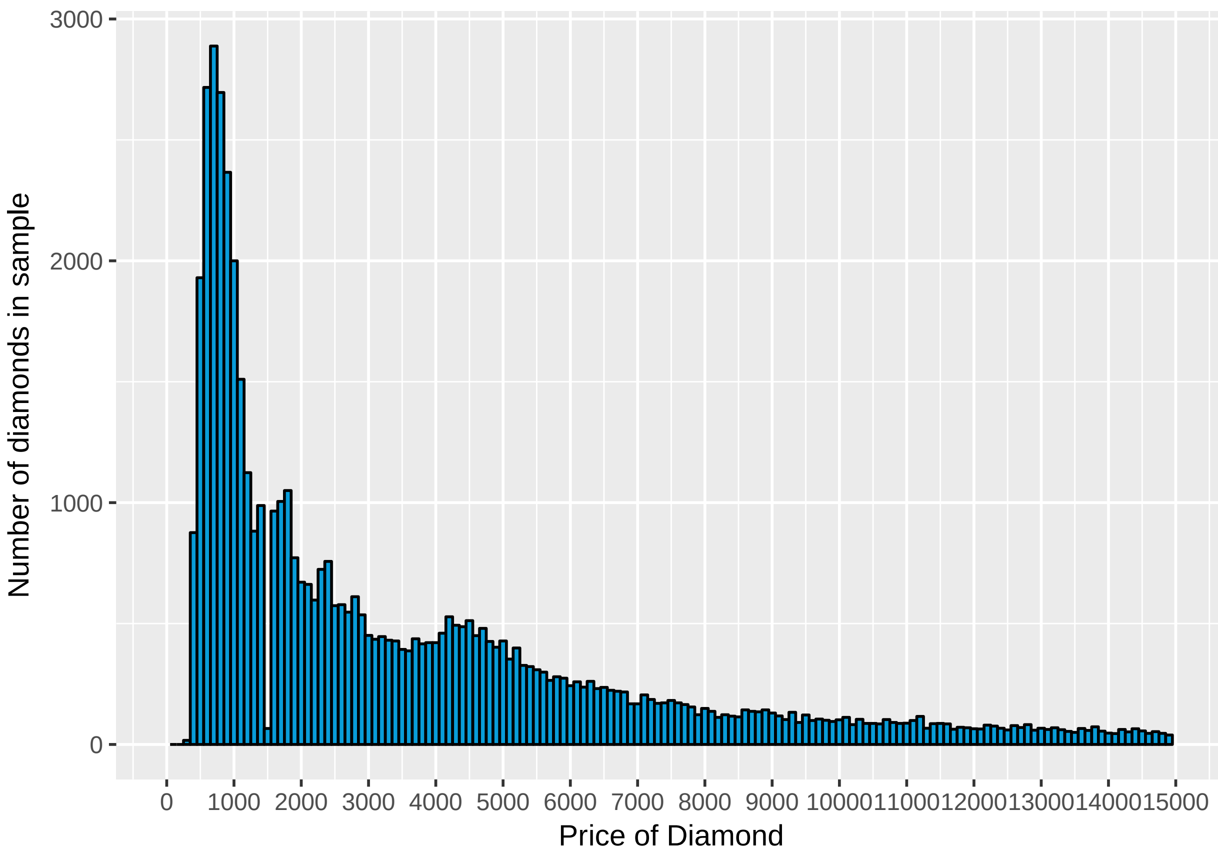
Price Histogram

Getting a histogram of the price shows a long-tailed distribution.

```
qplot(data = diamonds, x = price,
      color = I('black'), fill = I('#099DD9'),
      binwidth = 100) +
  xlab('Price of Diamond') +
  ylab('Number of diamonds in sample') +
  scale_x_log10() +
  scale_x_continuous(limits = c(0,15000), breaks = seq(0, 15000, 1000))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

```
## Warning: Removed 1655 rows containing non-finite values (stat_bin).
```



The shape of the price distribution is long-tailed.This means a large portion of the distribution is far from the 'head' or center of the distribution.

Running summary on price, we'll see:

```r
summary(diamonds$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     950    2401    3933    5324   18820
```

Now, I want to get more statistics on the price of the diamonds. I answer questions like:

How many diamond cost less than $500? To answer this question, first I get a summary of the price for those under $500. After executing the below command, the summary looks logical and gives true and false statements. True for those numbers below $500 and False for those above $500.

```r
summary(diamonds$price < 500)
```

```
##    Mode   FALSE    TRUE    NA's
## logical   52211    1729       0
```

To make the above logical statement look nicer, I create a new variable called 'less_than_500' and assign it to 'NA'. Then I create a conditional statement: If the price is under $500, return 1; else return 0.

```r
less_than_500 <- NA
diamonds$less_than_500 <- ifelse(diamonds$price < 500, 1, 0)
diamonds$less_than_500 <- factor(diamonds$less_than_500)
summary(diamonds$less_than_500)
```

```
##     0     1
## 52211  1729
```

Next question, how many diamonds cost less than $250? The same approach for the previous question will be applied here.

```r
less_than_250 <- NA
diamonds$less_than_250 <- ifelse(diamonds$price < 250, 1, 0)
diamonds$less_than_250 <- factor(diamonds$less_than_250)
summary(diamonds$less_than_250)
```

```
##     0
## 53940
```

The above calculations show that the conditional statement returns only those that do NOT cost less $250 dollars. This means there are no diamonds that cost less than $250.

Nest question, how many diamonds cost $15000 or more? Let's apply the same approach in the 2 previous questions to this question as well.

```r
more_than_15000 <- NA
diamonds$more_than_15000 <- ifelse(diamonds$price >= 15000, 1, 0)
diamonds$more_than_15000 <- factor(diamonds$more_than_15000)
summary(diamonds$more_than_15000)
```

```
##     0     1
## 52284  1656
```

There are 1656 diamonds that cost $15000 or more.

Cheaper Diamonds

Now, I want to explore the largest peak on the histogram I created earlier. I will use different binwidth, and limit the x-axis to observe the largest peak better.

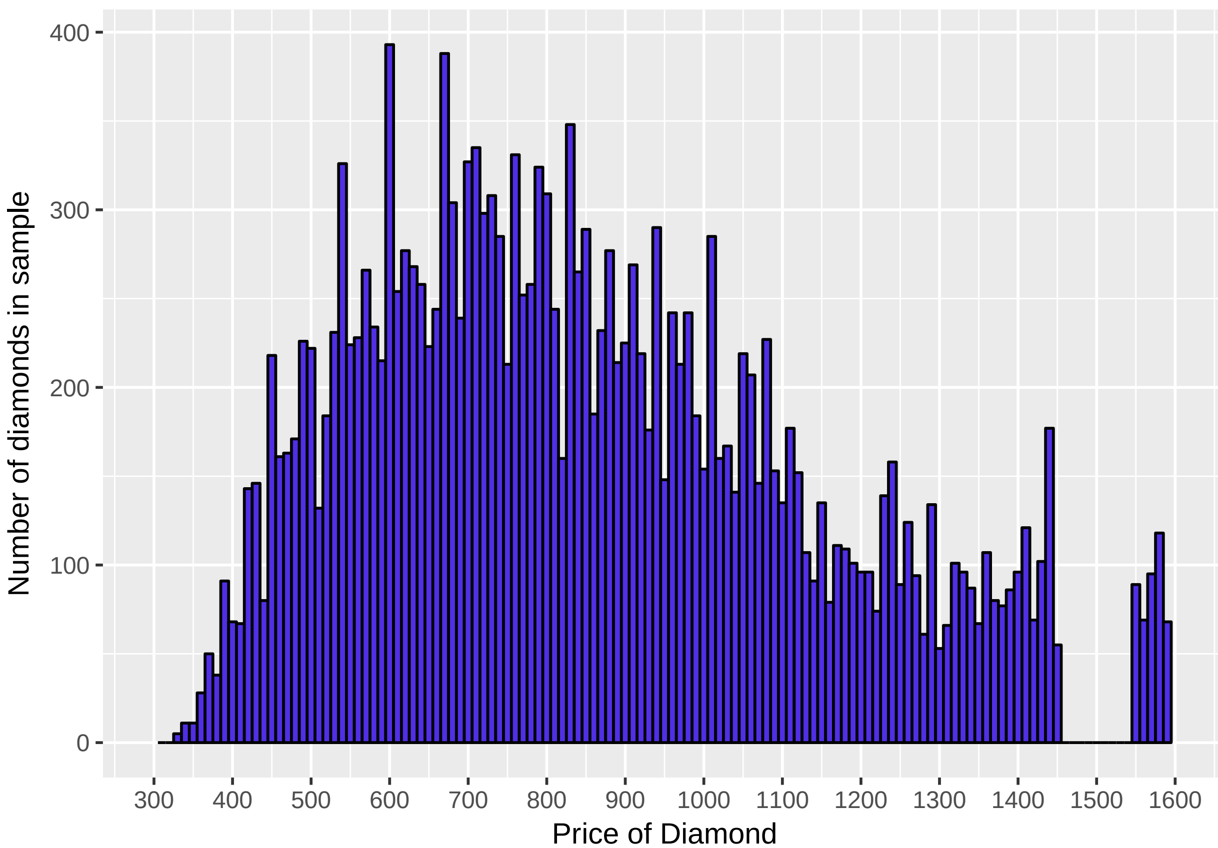
According to my previous histogram, it's safe to say that the peak occured somewhere between 0-2000. I decided to set limits and breaks to get a closer look on the peak. I first started to set the limit(0,2000), and breaks=seq(0,2000,100), which gave me a much clear view that I can easily ignore anything under $300.

Also, I could see there were no diamonds that cost $15000.

I decided to set my final limit between $300-1600.

```
qplot(data = diamonds, x = price,
      color = I('black'), fill = I('#502FE4'),
      binwidth = 10) +
  xlab('Price of Diamond') +
  ylab('Number of diamonds in sample') +
  scale_x_log10() +
  scale_x_continuous(limits = c(300,1600), breaks = seq(0, 1600, 100)) +
  ggsave('priceHistogram.png')
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.

## Saving 6.5 x 4.5 in image

## Warning: Removed 33450 rows containing non-finite values (stat_bin).

## Warning: Removed 33450 rows containing non-finite values (stat_bin).
```



From this histogram, we can see the main peaks are between $600-900.

Price by Cut Histograms

I want to break the histogram of diamonds prices by their cut. Let's have a look at the cut summary.

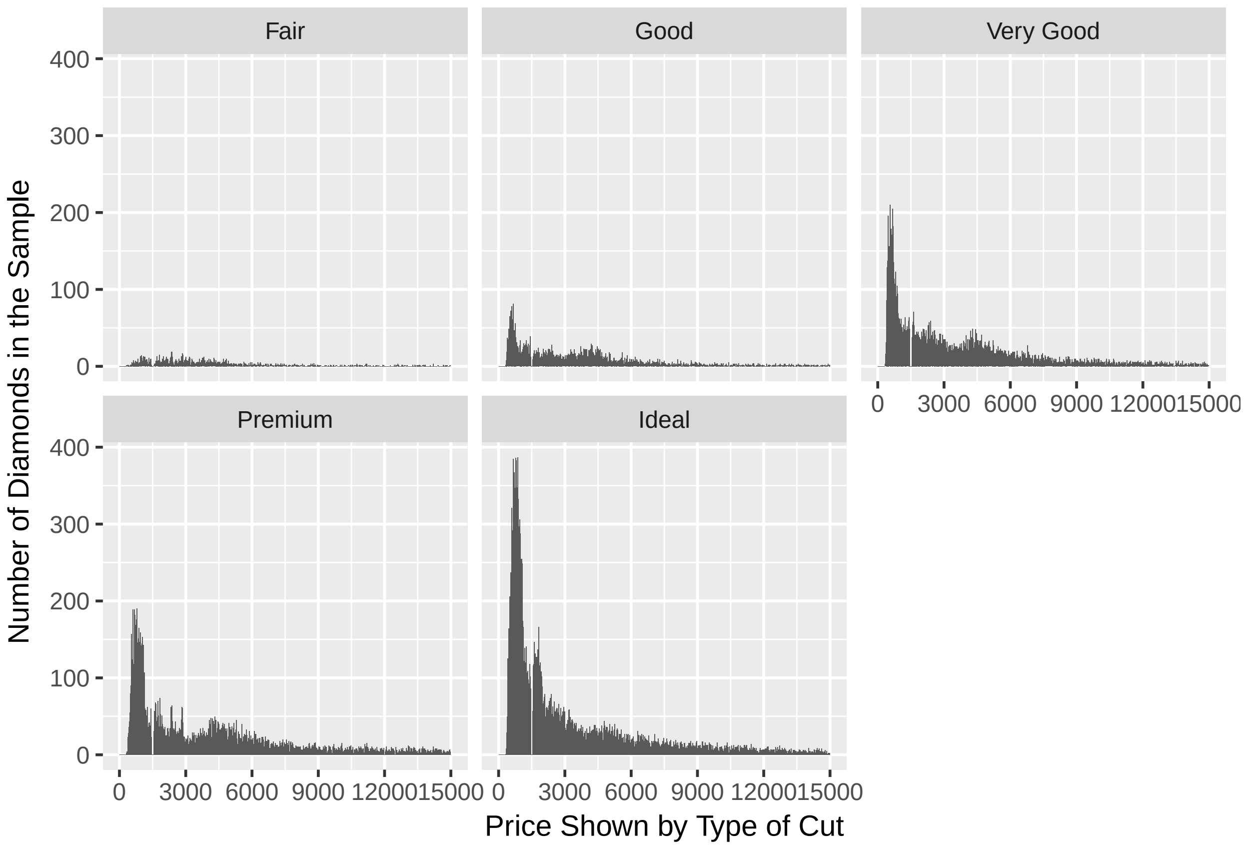
```
summary(diamonds$cut)
```

```
##      Fair      Good Very Good   Premium     Ideal
##      1610      4906     12082     13791     21551
```

There are 5 categories of cut: Fair, Good, Very Good, Premium, and Ideal.

```
qplot(data = diamonds, x = price, binwidth = 25) +
  xlab('Price Shown by Type of Cut') +
  ylab('Number of Diamonds in the Sample') +
  scale_x_continuous(limits = c(0, 15000), breaks = seq(0, 15000, 3000)) +
  facet_wrap(~cut)
```

```
## Warning: Removed 1655 rows containing non-finite values (stat_bin).
```



Diving the prices based on the diamond cut, shows that the distribution looks similar in Ideal, Very Good and Premium categories; all long-tailed with a price peak. In the Good category, it is also long-tailed but the peak is not as much as in the first 3. In the Fair category the distribution looks uniform on the left with long tails on the right.

Price by Cut

Now, let's answer some questions about the price of the diamonds based on their cuts:

Which cut has the highest priced diamond?

To answer this question, I use 'by' to get the summary of each cut. The information below shows that Very Good and Premium cuts have the maximum value of $18820.

```
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
```

Which cut has the lowest priced diamond?

Cuts Ideal and Premium have the lowest priced diamond $326.

Which cut has the lower median price?

Cut Ideal has the lowest Median price of $1810.

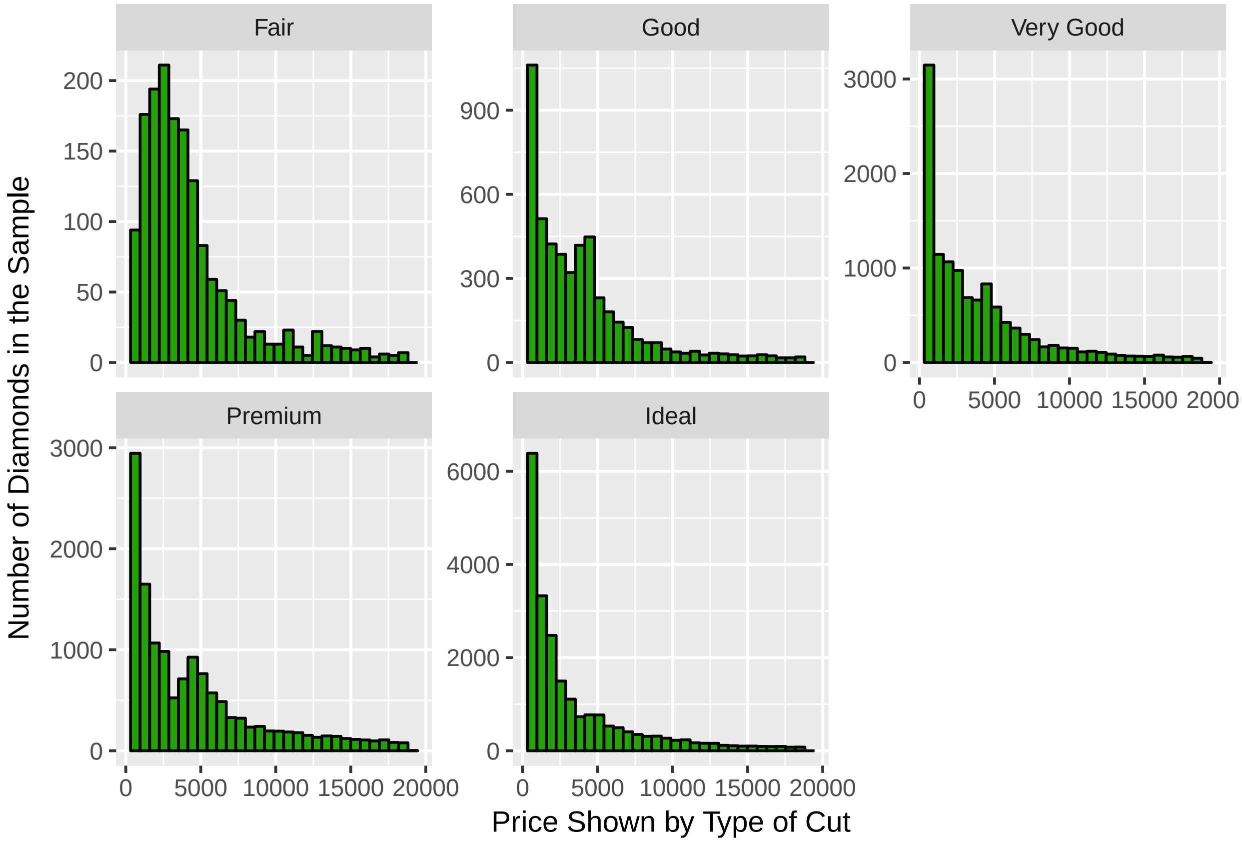
Scales and Multiple Histograms

Getting a summary of the prices based on cuts shows that the distribution of the prices should be somewhat the same. However, in the histograms I created for prices based on cuts, we saw that only the distributions of Premium, Very Good and Ideal look the same. The distribution of Fair and Good look somehow uniform.

Let's look into this more. I now create histograms of prices based on cuts, where the y-axis is not fixed. I do it by scaling the y-axis.

```
qplot(x = price, data = diamonds,
      color = I('black'), fill = I('#25A00A')) +
  xlab('Price Shown by Type of Cut') +
  ylab('Number of Diamonds in the Sample') +
  facet_wrap(~cut, scales="free_y")
```

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



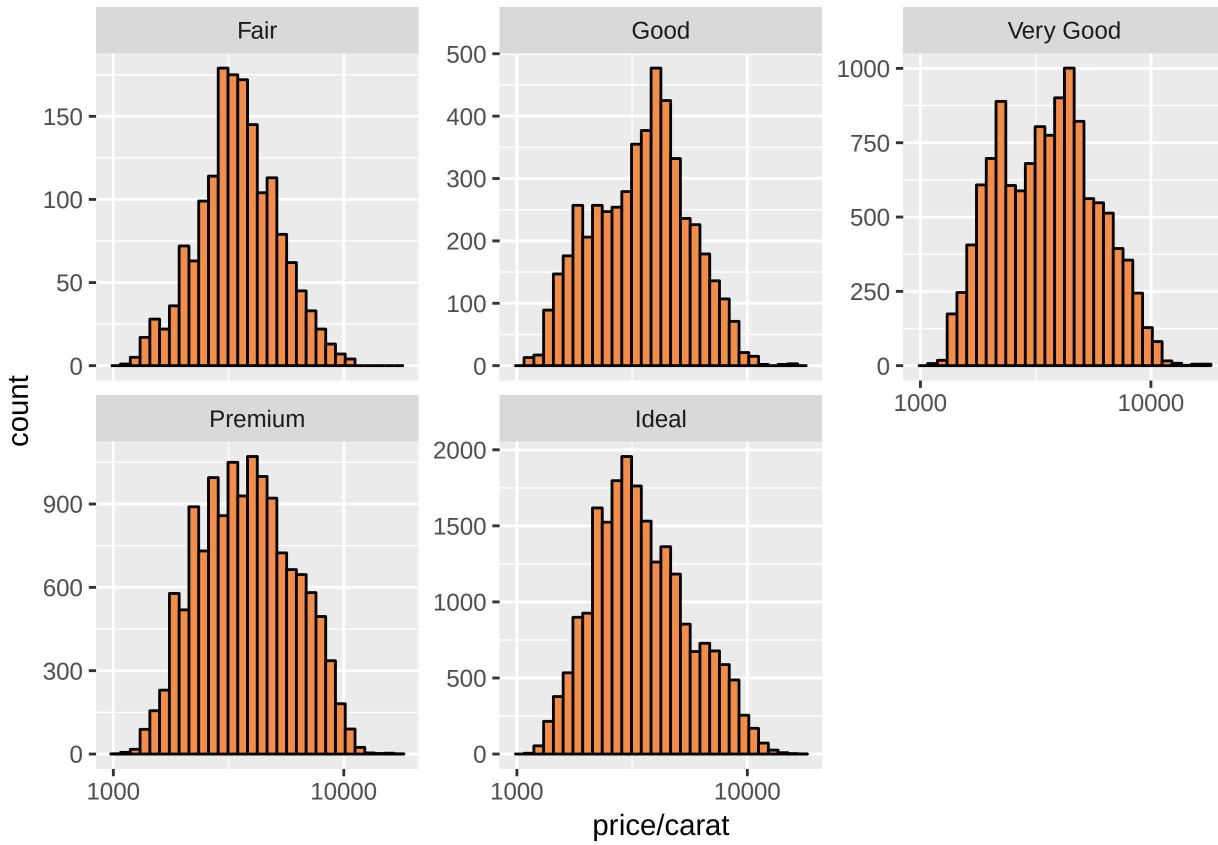
Scaling the y-axis makes it different on some of the panels in the histogram. For example, for Fair cuts, the y-axis goes from 0 to over 200, whereas, in Premium it goes from 0 to about 3000.

Price per Carat by Cut

I want to create a histogram of price per carat and facet it by cut. Since histograms only takes a single series and splits it into bins, for the 'x' variable, I use 'price/carat' to get the distribution for price per carat. I wrap it by cut, and add scales for the y-axis to get different values on the y-axis. I complete my histogram by scaling it using log10. After adding log10, I can see how the distribution has changed to a somehow normal one.

```
qplot(data = diamonds, x = price/carat,
      color = I('black'), fill = I('#F28D47')) +
  scale_x_log10() +
  facet_wrap(~cut, scales = "free_y")
```

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



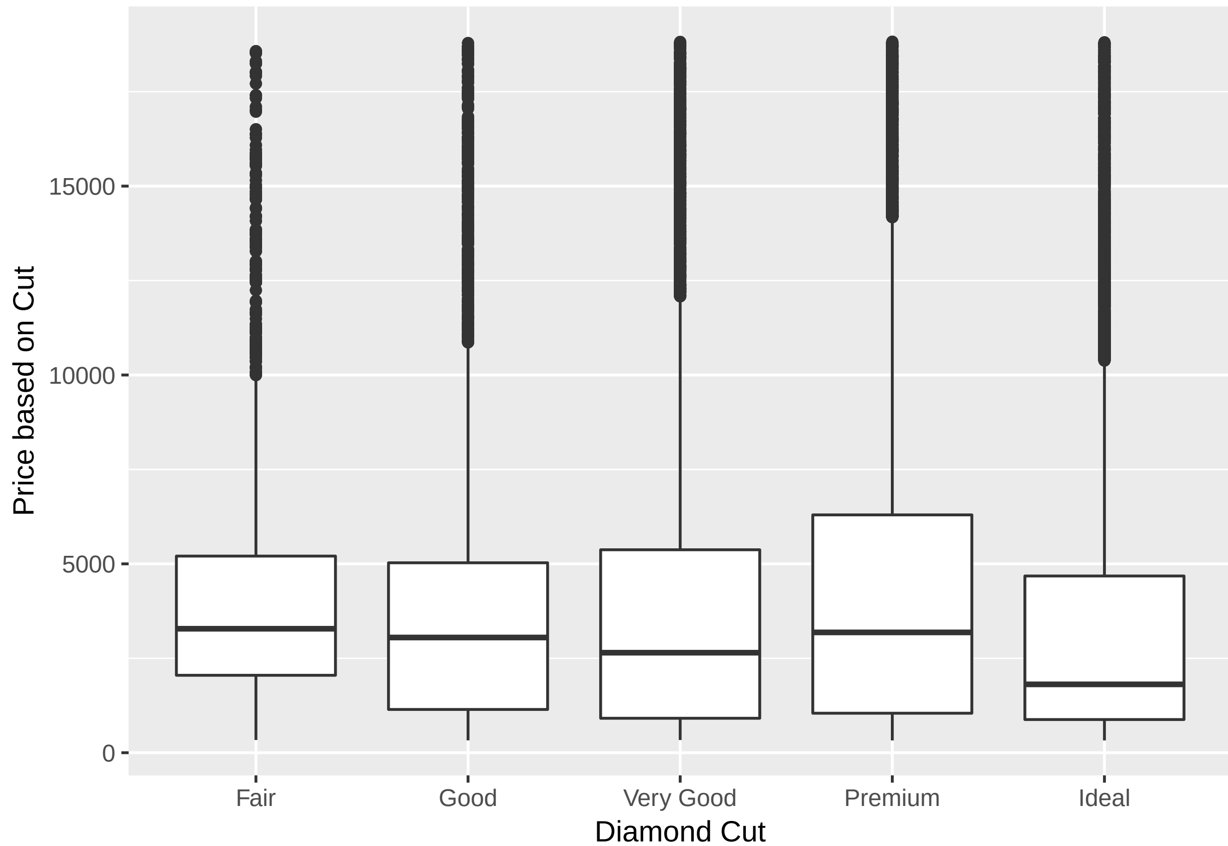
Price Box Plots

I want to investigate the price of diamonds using box plots, numerical summaries, and the following categorical variables: cut, clarity, or color.

Box Plot by Cut

To get the boxplot based on cut, I set the x-axis to show different cuts, and the y-axis to show the price of those cuts.

```
qplot(data = diamonds, geom = 'boxplot',
      x = cut, y = price) +
  xlab('Diamond Cut') +
  ylab('Price based on Cut')
```

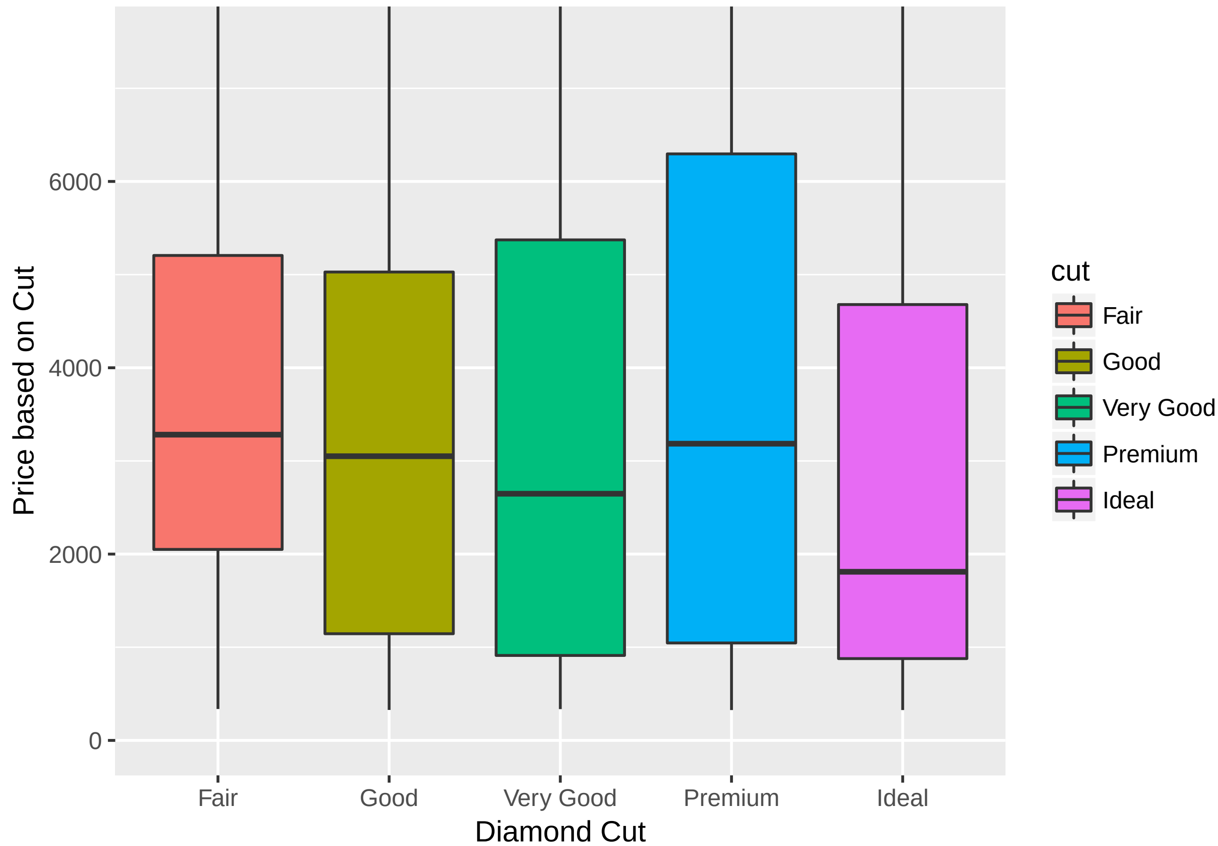


These boxplots show there are a lot of outliers which fall over $10000 price. I need to get a closer look at the boxes without having to deal with the outliers. To do that, I use the 'coord_cartesian' which can magnify the boxes for me.

The Cartesian coordinate system is the most familiar, and common, type of coordinate system. Setting limits on the coordinate system will zoom the plot (like you're looking at it with a magnifying glass), and will not change the underlying data like setting limits on a scale will. Read more on: http://docs.ggplot2.org/0.9.3.1/coord_cartesian.html

Since the boxplots lie between 0 and $7000, I adjust my y-axis to limit the prices up to 7500. I also add a bit of color to the boxplots for a nicer look.

```
qplot(data = diamonds, geom = 'boxplot',
      x = cut, y = price,
      fill=cut) +
  xlab('Diamond Cut') +
  ylab('Price based on Cut') +
  coord_cartesian(ylim = c(0, 7500))
```



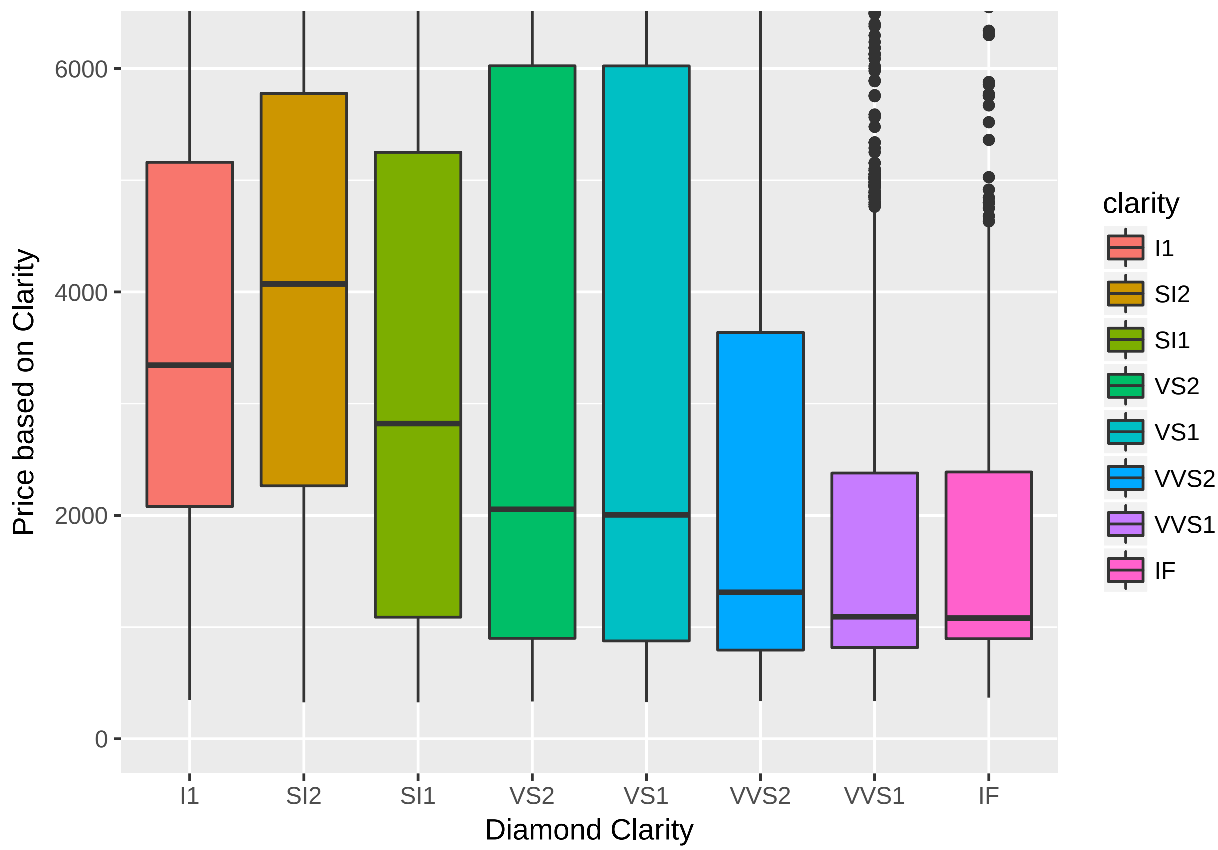
The barplots show that the Premium cut has the largest range and highest median (i.e. the black horizontal line in the box), where the Ideal cut has the lower price.

Box Plot by Clarity

To get the boxplot based on clarity, I set the x-axis to show different clarities, and the y-axis to show the price of them.

I first generate my boxplot without using cartesian coordinates. This will give me the chance to see the ranges and where the outliers are. After investigating these things, I adjust my coordinates.

```r
qplot(data = diamonds, geom = 'boxplot',
      x = clarity, y = price,
      fill=clarity) +
  xlab('Diamond Clarity') +
  ylab('Price based on Clarity') +
  coord_cartesian(ylim = c(0, 6200))
```



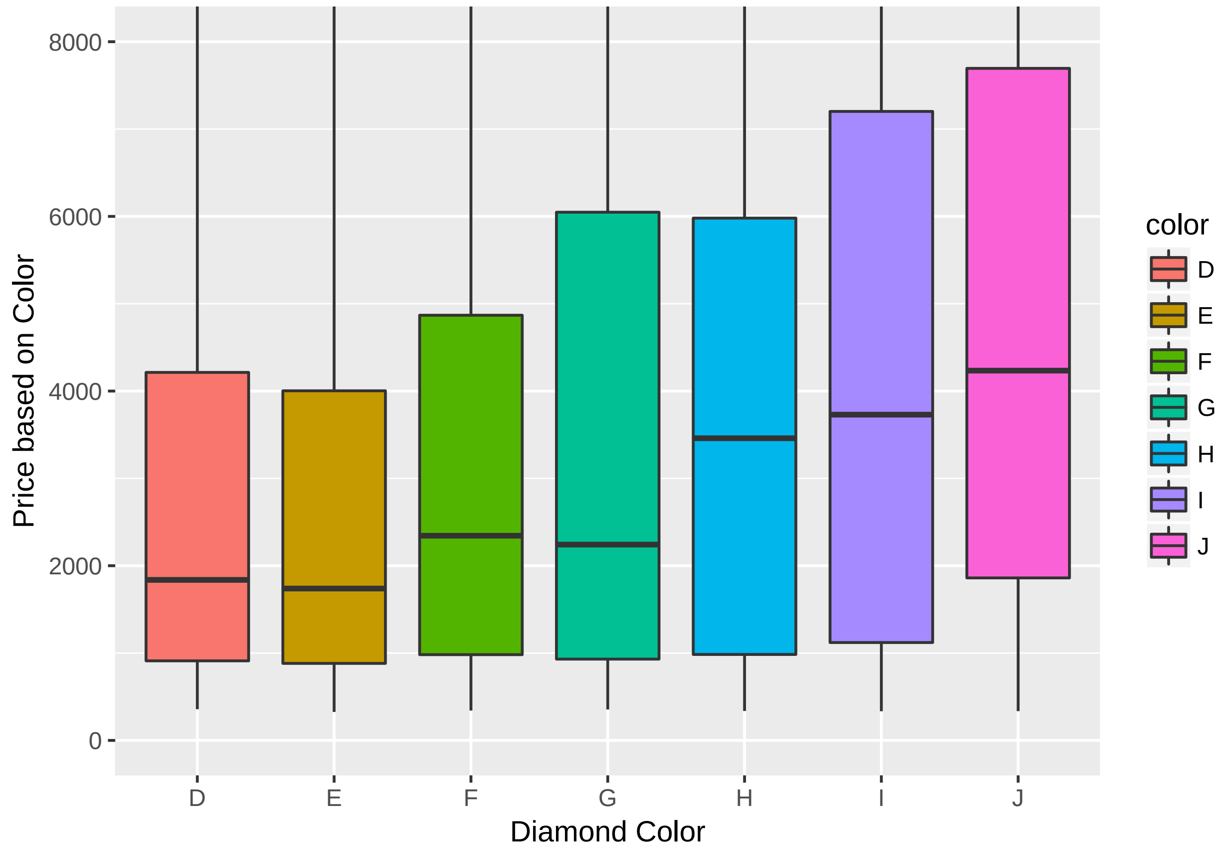
The barplot show VS1 and VS2 have somehow the same range, with the Median of VS2 being slightly larger than VS1. The largest Median in price goes to SI2 clarity categorty.

Box Plot by Color

To get the boxplot based on clarity, I set the x-axis to show different colors, and the y-axis to show the price of them.

I first generate my boxplot without using cartesian coordinates. This will give me the chance to see the ranges and where the outliers are. After investigating these things, I adjust my coordinates.

```
qplot(data = diamonds, geom = 'boxplot',
      x = color, y = price,
      fill=color) +
  xlab('Diamond Color') +
  ylab('Price based on Color') +
    coord_cartesian(ylim = c(0, 8000))
```



The barplots show that color E seems to have the lowest price, and the largest Median belongs to color J.

Interquartile Range - IQR

Now I will work with the IQR() function and try to get some info regarding price and color of diamonds. IQR() computes the interquartile range of the x values.

Let's first investigate a bit on the price range based on color. I use the 'by' command to get a summary of prices, categorized by color.

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

```
## diamonds$color: D
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     357     911    1838    3170    4214   18690
## --------------------------------------------------------
## diamonds$color: E
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     882    1739    3077    4003   18730
## --------------------------------------------------------
## diamonds$color: F
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     342     982    2344    3725    4868   18790
## --------------------------------------------------------
## diamonds$color: G
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     354     931    2242    3999    6048   18820
```

```
## ----------------------------------------------------------
## diamonds$color: H
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     337     984    3460    4487    5980   18800
## ----------------------------------------------------------
## diamonds$color: I
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     334    1120    3730    5092    7202   18820
## ----------------------------------------------------------
## diamonds$color: J
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     335    1860    4234    5324    7695   18710
```

Based on this summary, I can for example say that the first quartile(25%) of diamonds with color D has the price 911 dollors, and the third quartile(75%) of them are 4214 dollors.

What is the IQR for the diamonds with the worst color?

Based on the documentation for the diamonds dataset, in diamonds colors, 'J' is the worst and 'D' is the best.

```
IQR(subset(diamonds, color=="J")$price)
```

```
## [1] 5834.5
```

What is the IQR for the diamonds with the best color?

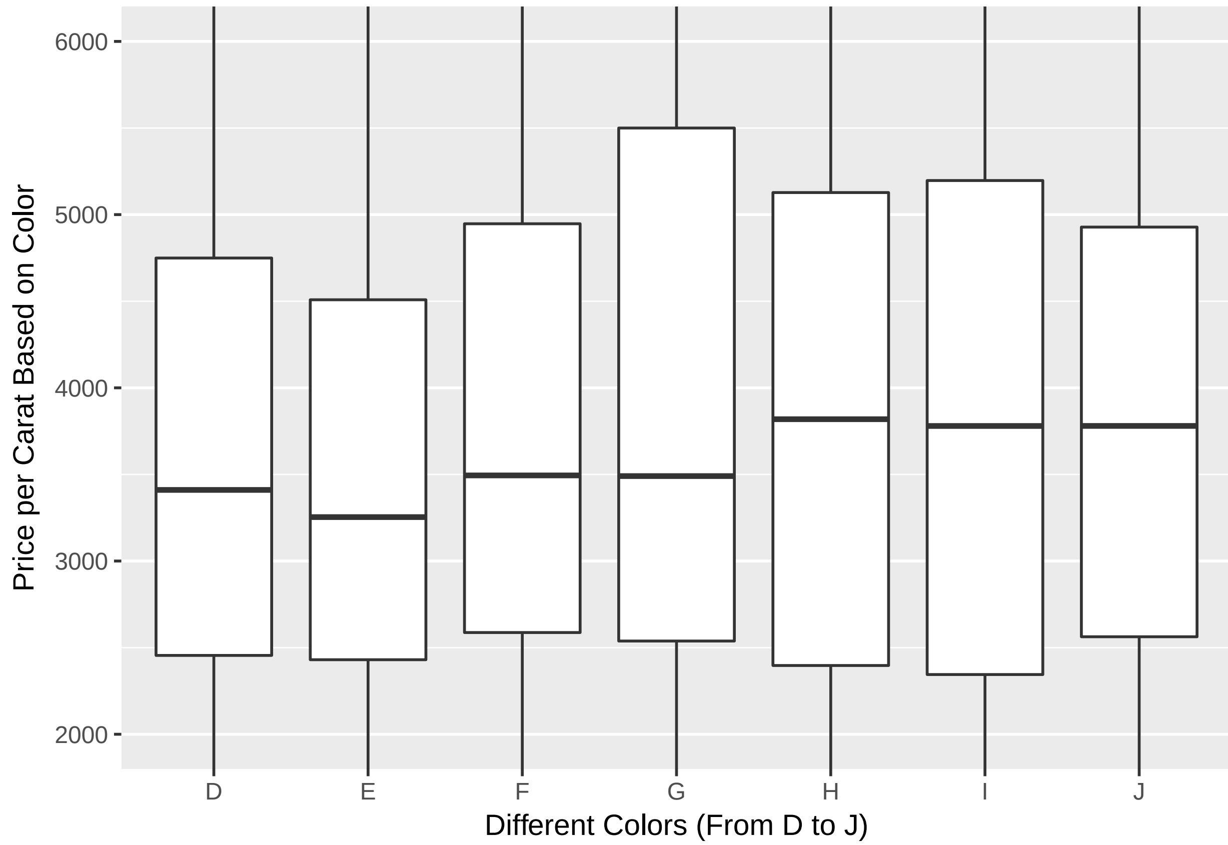
```
IQR(subset(diamonds, color=="D")$price)
```

```
## [1] 3302.5
```

Price per Carat Box Plots by Color

I want to investigate the price per carat of diamonds across the different colors of diamonds using boxplots.

```
qplot(data = diamonds, geom = 'boxplot',
      x = color, y = price/carat) +
  xlab('Different Colors (From D to J)') +
  ylab('Price per Carat Based on Color') +
  coord_cartesian(ylim = c(2000, 6000))
```

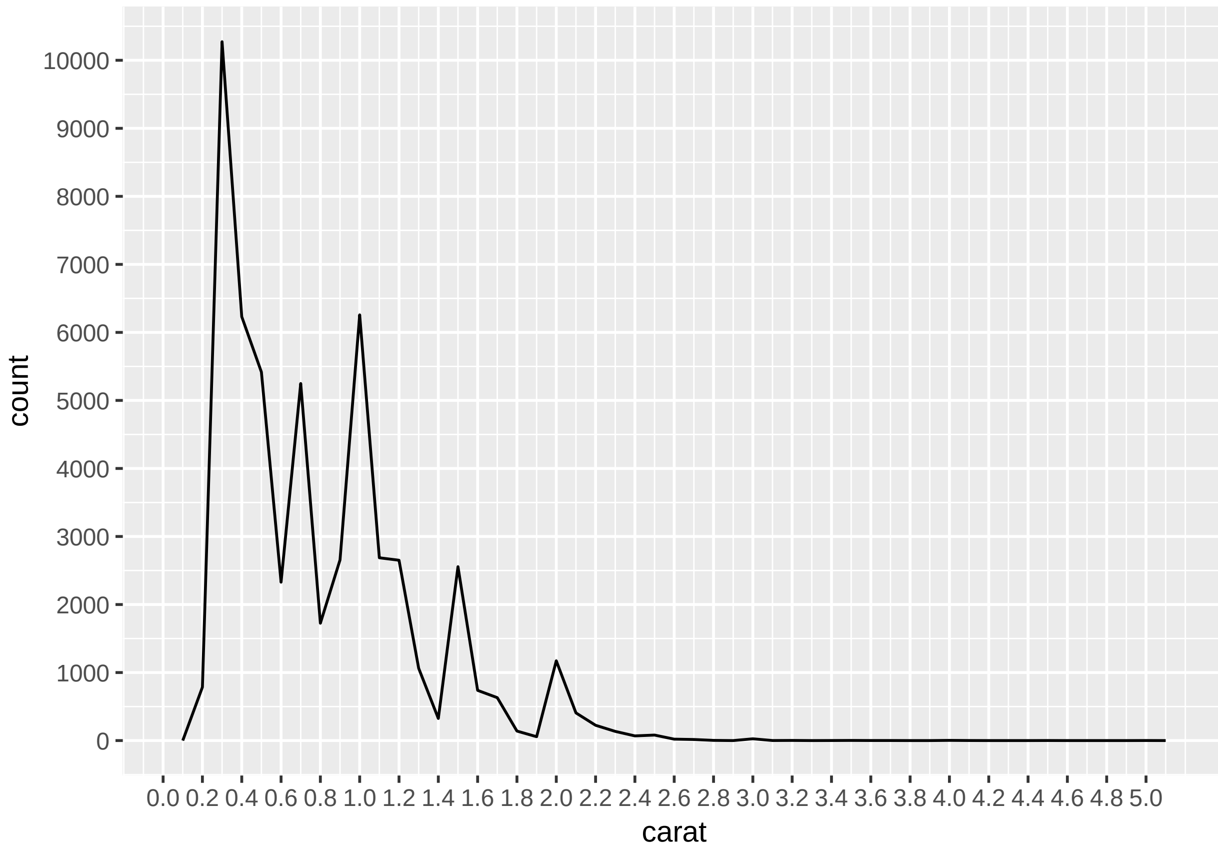


Among different colors, color 'G' has the most range, and color 'I' has the lowest price. The highest Median goes to color 'H'.

Carat Frequency Polygon

Using a frequency polygon, I want to find out what carat size has a count greater than 2000.

```
qplot(data = diamonds, x = carat, binwidth = 0.1,
  geom = 'freqpoly') +
  scale_y_continuous(breaks = seq(0, 15000, 1000)) +
  scale_x_continuous(breaks = seq(0, 5, 0.2))
```



From this plot, I can see that for example 0.3 and 1.01 sizes (carats) have more than 2000 counts, while anything after 2.0 has fewer than 2000 counts.

Gapminder Data

The Gapminder website contains over 500 data sets with information about the world's population. My task will be to download a data set of my choice, and create 2-5 plots based on its data.

I decide to go with the data about female students who are out of primary school. I downloaded the main file in xlsx, and converted it to CSV to read it in R.

```
getwd()
```

```
## [1] "/Users/nazaninmirarab/Desktop/Data Science/P4"
```

```
female_out_of_school <- read.csv('data.csv', header = T, check.names = F)
```

After executing the command above, I have 'female_out_of_school' dataset created in my environment.

There are many values in the dataset marked as '..' and I want to change them to NA, so that I can later filter them out of the actual values when I am creating plots.

I also name the first column in the dataset 'Country' since it is representing the countries.

```
#Replacing the '..' value with NA
female_out_of_school[female_out_of_school == '..'] = NA

#Changing the name of the first column to 'country'
colnames(female_out_of_school)[1] <- 'Country'
```

I do a little bit more cleanup on the original dataset. In my dataset, the values are between years 1999-2006. So I want to have a table with only these years present.

Also, I want to change the original table to a table where there are three columns: country, year, and count. Country and year are self-explanatory. Count will be the number of female students out of school for the specific year and country.

To do this tidying up, I use the 'tidyr' package.

```
#install.packages("tidyr")
library(tidyr)

#Creating a new dataset 'female_out_gather' with 3 columns: country, year and count
female_out_gather <- gather(female_out_of_school, year, count, 2:ncol(female_out_of_school))
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```
#Extracting years 1999-2006 from the dataset as only these years have meaningful values in them
female_out_gather_1999_2006 <- subset(female_out_gather, year >= 1999, year <= 2006)
```

For getting plots, I use my 'female_out_gather_1999_2006' dataset that I have cleaned up already. I want to get a plot from the data, excluding the NA values.

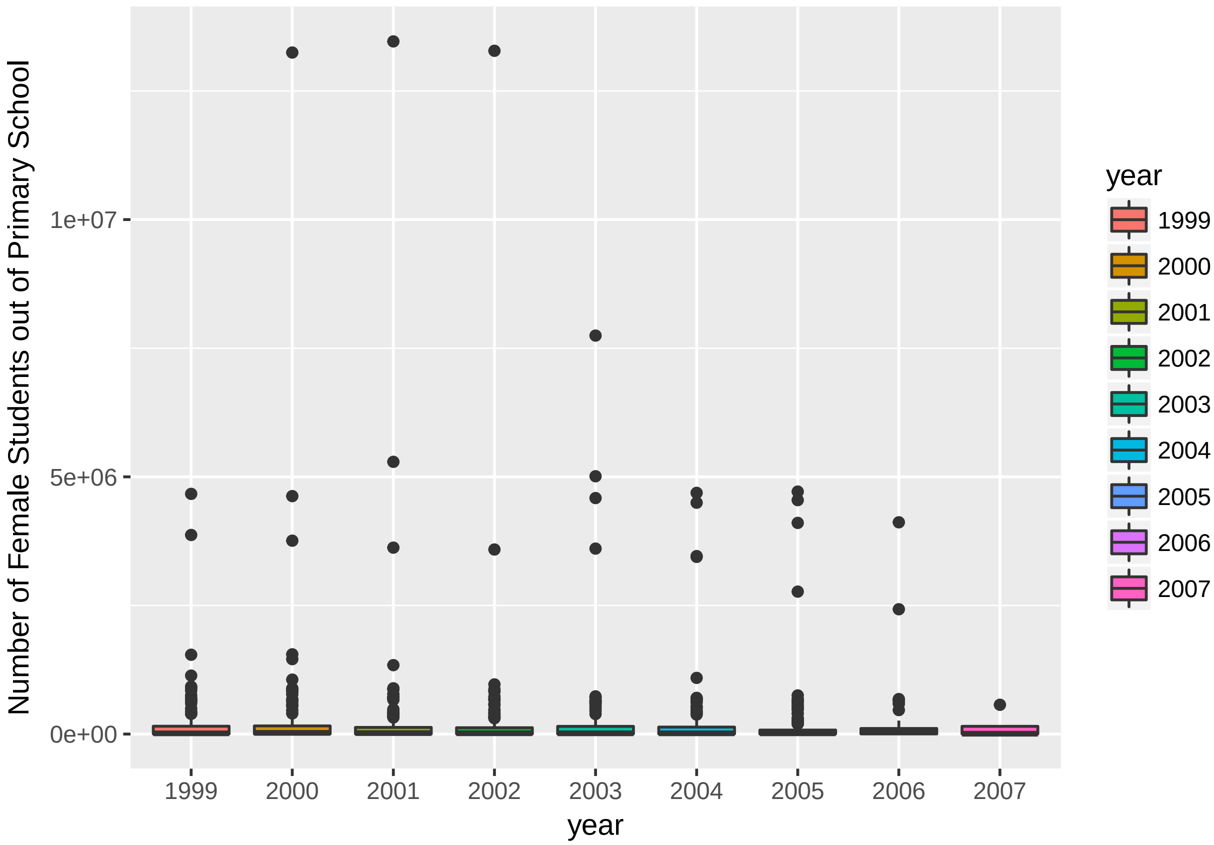
For the y-axis, I use 'as.numeric' to make sure the values returned are numbers.

```
qplot(data = subset(female_out_gather_1999_2006, !is.na(count)),
      x = year, y = as.numeric(count),
      geom = 'boxplot', fill=year) +
  ylab('Number of Female Students out of Primary School') +
  ggsave('boxplot-1.png')
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: Removed 9 rows containing non-finite values (stat_boxplot).
```

```
## Warning: Removed 9 rows containing non-finite values (stat_boxplot).
```



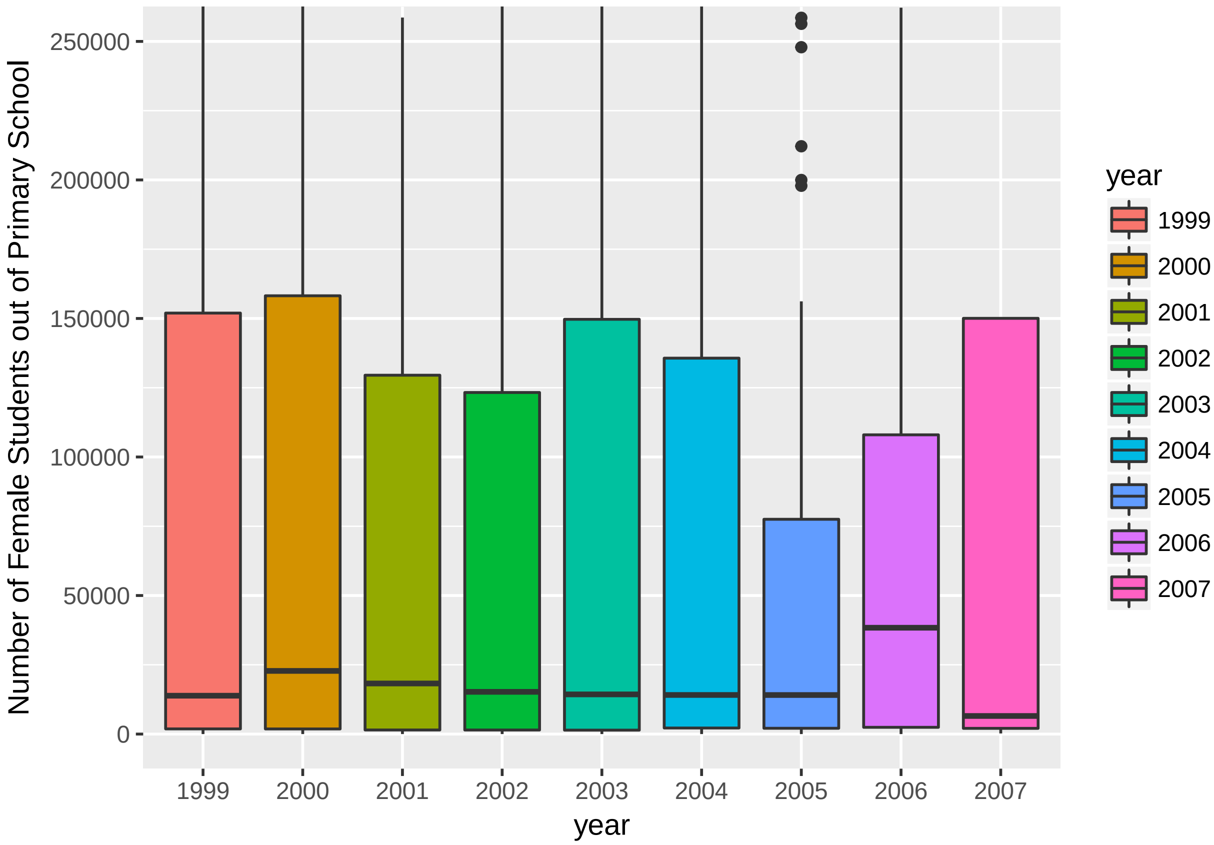
After getting the initial look on the plot, I see there are many outliers which have made the boxplots look very small. I use cartesian coordinates to magnify the boxes.

```
qplot(data = subset(female_out_gather_1999_2006, !is.na(count)),
      x = year, y = as.numeric(count),
      geom = 'boxplot', fill=year) +
    coord_cartesian(ylim = c(0, 250000)) +
  ylab('Number of Female Students out of Primary School') +
  ggsave('boxplot-2.png')
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: Removed 9 rows containing non-finite values (stat_boxplot).
```

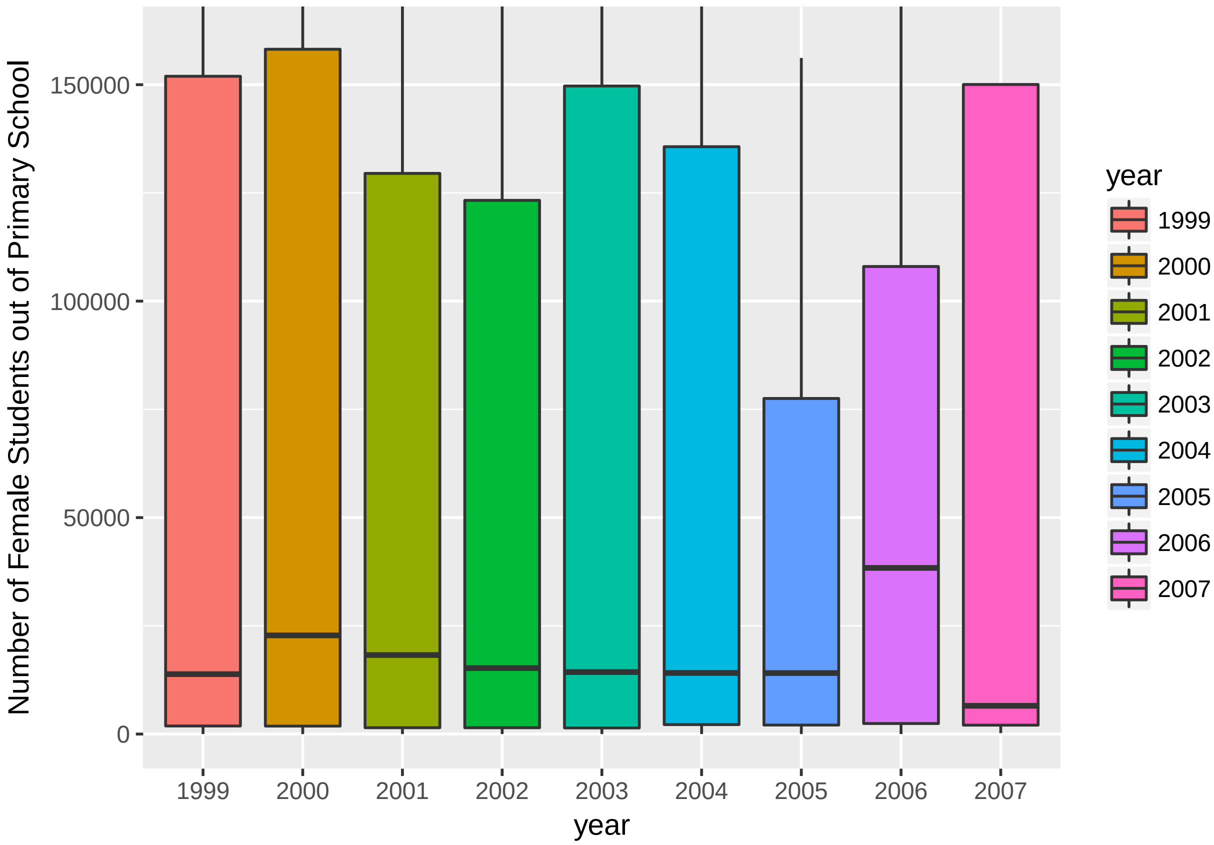
```
## Warning: Removed 9 rows containing non-finite values (stat_boxplot).
```



Using 'coord_cartesian(ylim = c(0, 250000))' as coordinates is good, but I want a closer look, to be able to distinguish the Medians in each year better.

```
qplot(data = subset(female_out_gather_1999_2006, !is.na(count)),
      x = year, y = as.numeric(count),
      geom = 'boxplot', fill=year) +
    coord_cartesian(ylim = c(0, 160000)) +
  ylab('Number of Female Students out of Primary School') +
  ggsave('boxplot-3.png')
```

```
## Saving 6.5 x 4.5 in image

## Warning: Removed 9 rows containing non-finite values (stat_boxplot).

## Warning: Removed 9 rows containing non-finite values (stat_boxplot).
```

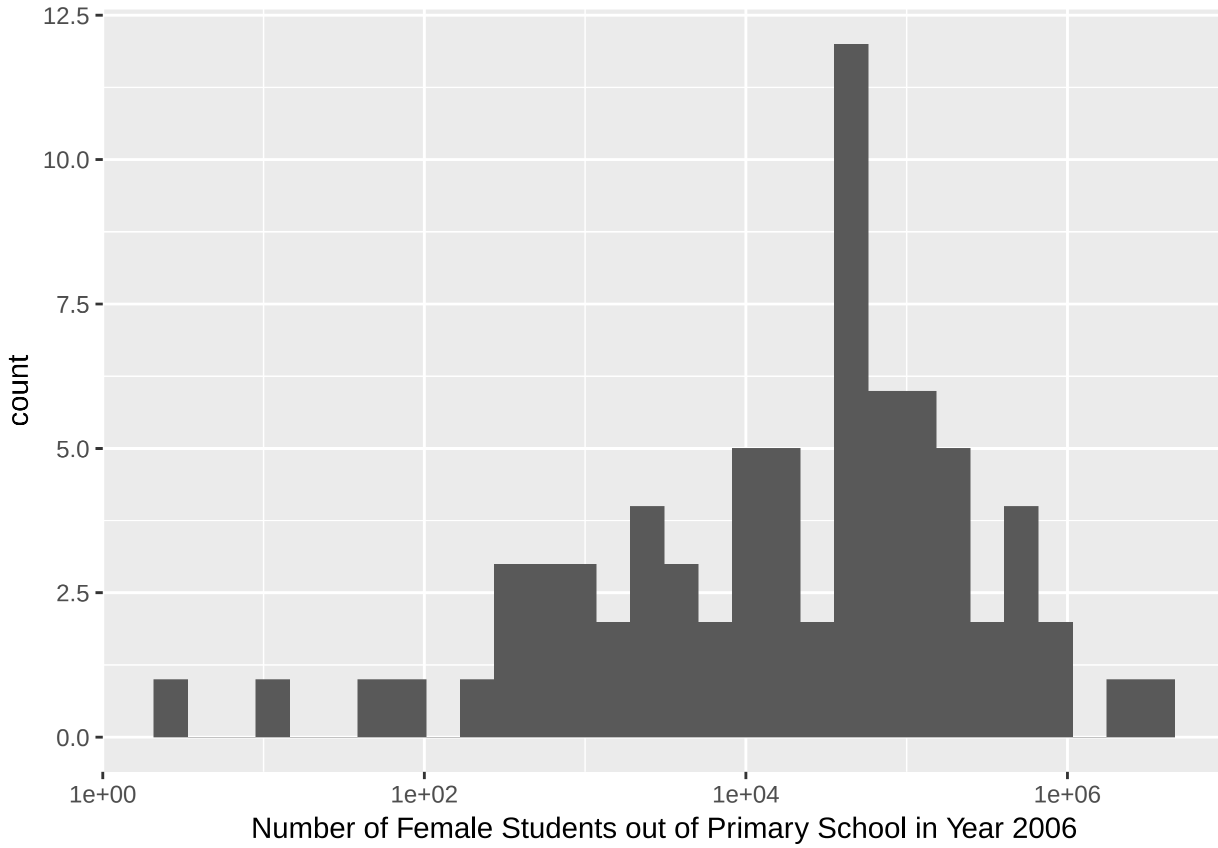


Magnifying the boxplots by setting the limite between 0-160000 suffices for me to see that year 2006 has the highest Median in female students out of primary school. Also, year 2000 seems to have a wider range in comparison to other years.

The plots show that the Median was decreasing from year 2000 until it reaches year 2006 with a sudden increase. So let's focus on year 2006 a bit more.

```
out_of_school_2006 <- subset(female_out_gather_1999_2006, year == 2006)
qplot(data = out_of_school_2006, x = as.numeric(count)) +
  xlab('Number of Female Students out of Primary School in Year 2006') +
  scale_x_log10() +
  ggsave('plot-4.png')
```

```
## Saving 6.5 x 4.5 in image

## Warning: Transformation introduced infinite values in continuous x-axis

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

## Warning: Removed 134 rows containing non-finite values (stat_bin).

## Warning: Transformation introduced infinite values in continuous x-axis

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

## Warning: Removed 134 rows containing non-finite values (stat_bin).
```



Using log10, I could achieve a normal-ish distribution of the data.