Diamond Exercises

Kathryn Gadberry

April 16, 2016

Exploratory Data Analysis

First, I need to set-up my workspace.

```
suppressMessages(library(dplyr))
suppressMessages(library(tidyr))
suppressMessages(library(ggplot2))
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
suppressMessages(library("datasets"))
diamonds <- data.frame(diamonds)
```

This set contains prices and attributes of almost 54,000 diamonds. Most of the ten variables listed below are straight forward. Clarification is needed for variable x (length in mm), variable y (width in mm), and variable z (depth in mm).

```
names(diamonds)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

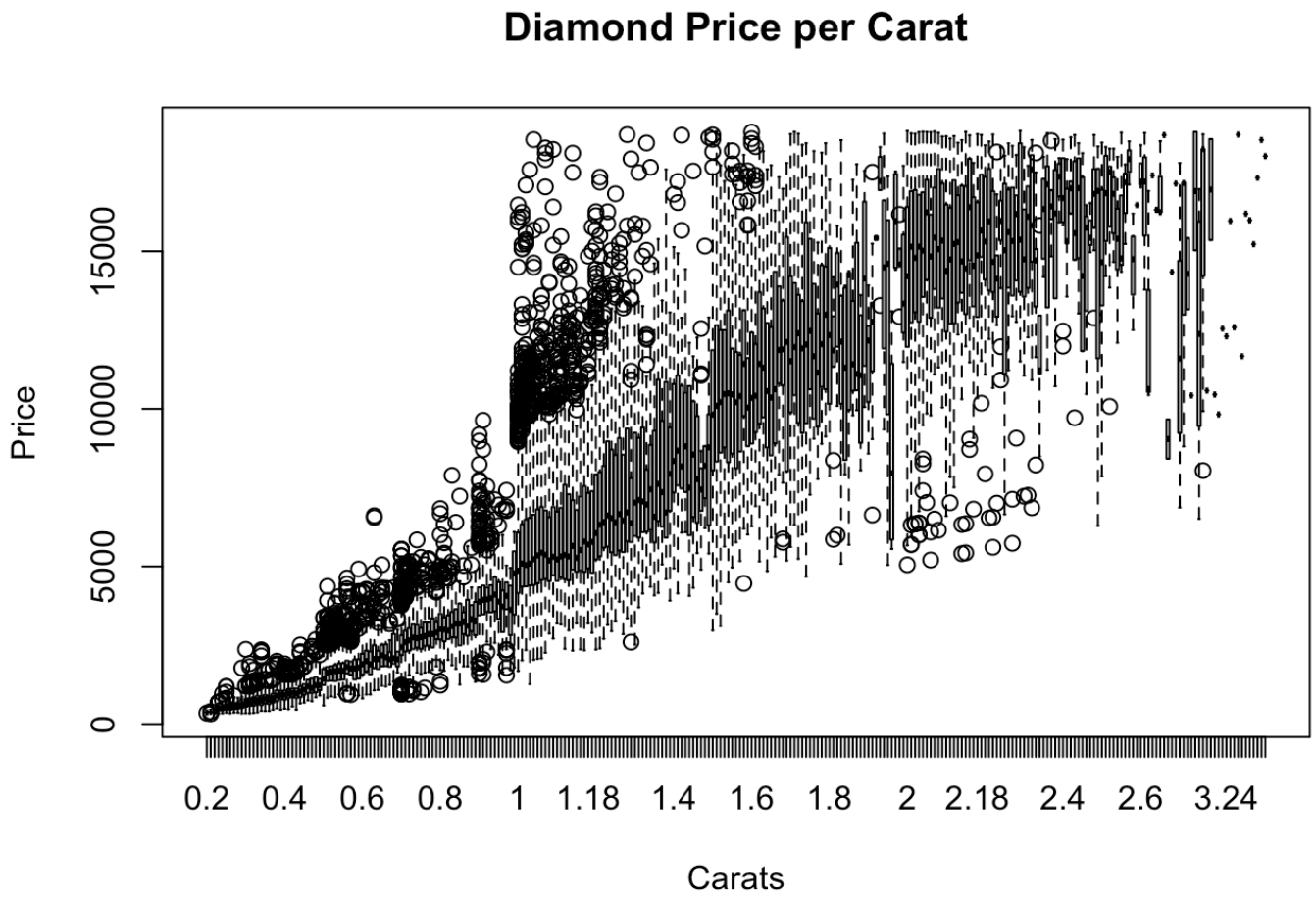
```
glimpse(diamonds)
```

```
## Observations: 53,940
## Variables: 10
## $ carat   (dbl) 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, ...
## $ cut     (fctr) Ideal, Premium, Good, Premium, Good, Very Good, Very ...
## $ color   (fctr) E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J,...
## $ clarity (fctr) SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, S...
## $ depth   (dbl) 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, ...
## $ table   (dbl) 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54...
## $ price   (int) 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339,...
## $ x       (dbl) 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, ...
## $ y       (dbl) 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, ...
## $ z       (dbl) 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, ...
```

Part I: Boxplots

I wanted to investigate the price per carat of diamonds across the different colors of diamonds using boxplots. An intial look at this data, using the boxplot function, might look something like this:

```
boxplot(price~carat, data = diamonds, main = 'Diamond Price per Carat', xlab = 'Ca
rats', ylab = 'Price')
```



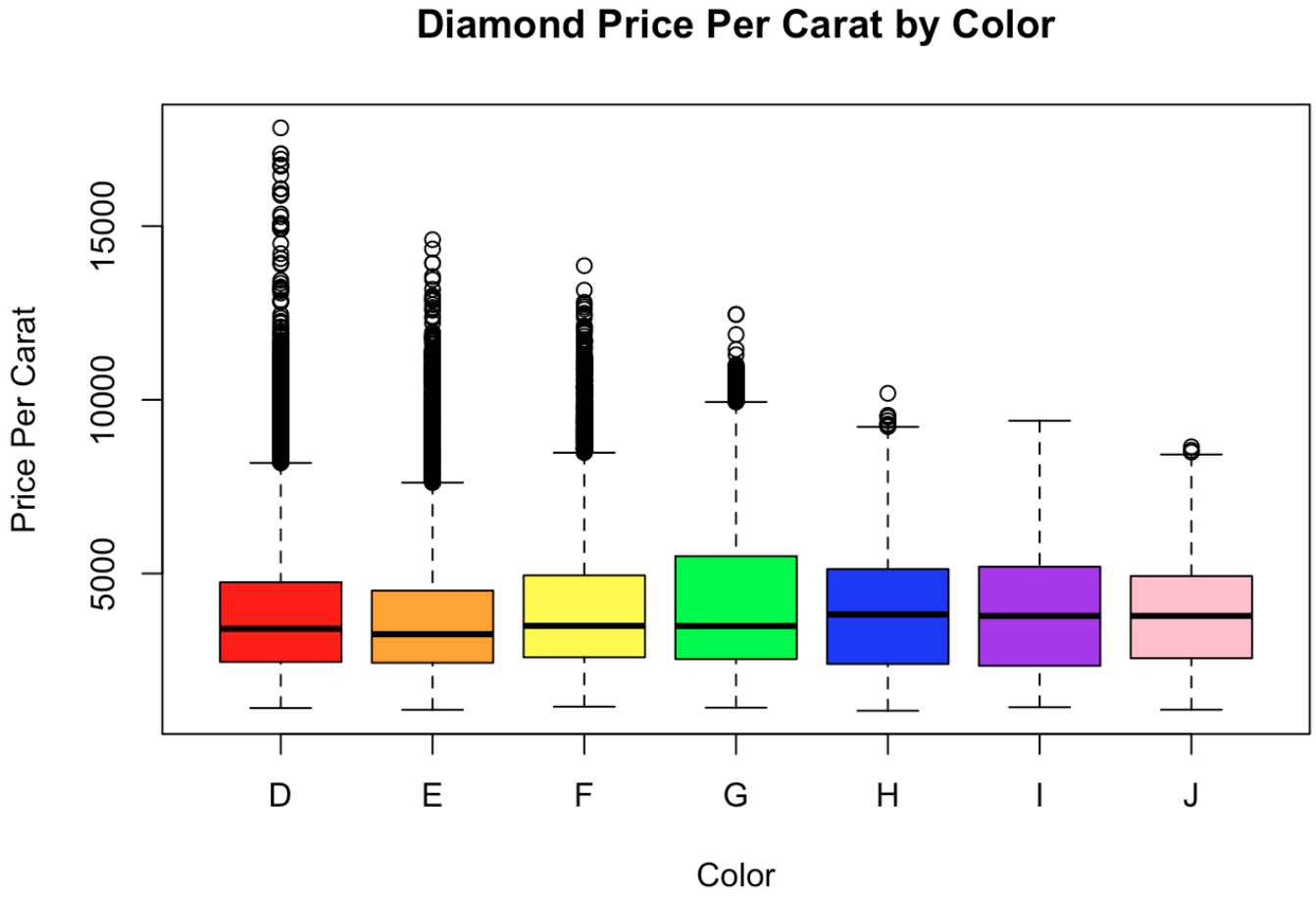
To clean up this data and add the third variable (color), I need to create a variable dividing price by carat.

```
diamonds$price_per_carat <- diamonds$price/diamonds$carat
head(diamonds$price_per_carat)
```

```
## [1] 1417.391 1552.381 1421.739 1151.724 1080.645 1400.000
```

Now, I can plot the Price Per Carat in comparison to the color of each diamond. I customized the color scale to define each of the seven diamond color levels: D, E, F, G, H, I, J. These individual boxplots show us the range, average(median), as well as the first and third quartile average price for each color diamond.

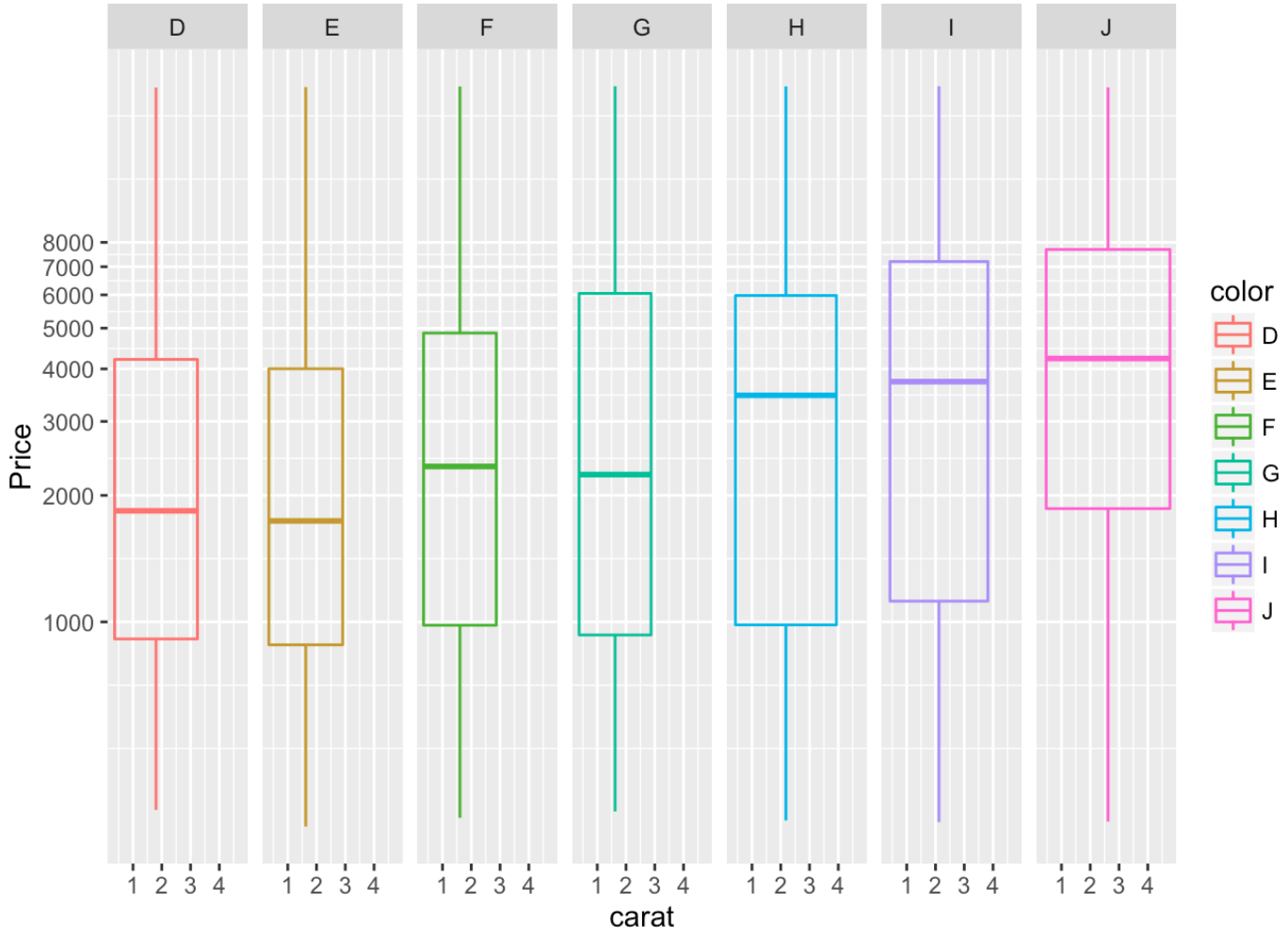
```
boxplot(diamonds$price_per_carat ~ diamonds$color, main = "Diamond Price Per Carat
by Color", xlab = "Color", ylab = "Price Per Carat", col = c('red', 'orange', 'yel
low', 'green', 'blue', 'purple', 'pink'))
```



Investigating using ggplot syntax

When adding the ggplot function, I assinged the line of code to the object gp. Then, I'm able to add more commands and explore different ways to manipulate and visualize the data using the same base. This chart gives us more insight into the differences in the average price per carat over the seven different colors. On average, the J colored diamonds are the most expensive, but have a higher carat size compared to the E colored diamonds, which are the smallest size and least expensive out of the group.

```
gp <- ggplot(diamonds, aes(carat, price))
gp + geom_boxplot(aes(color = color), alpha = 1/20)+
  facet_wrap(~color, ncol = 7)+
  scale_y_log10(name = 'Price', breaks = seq(0, 8000, 1000))
```



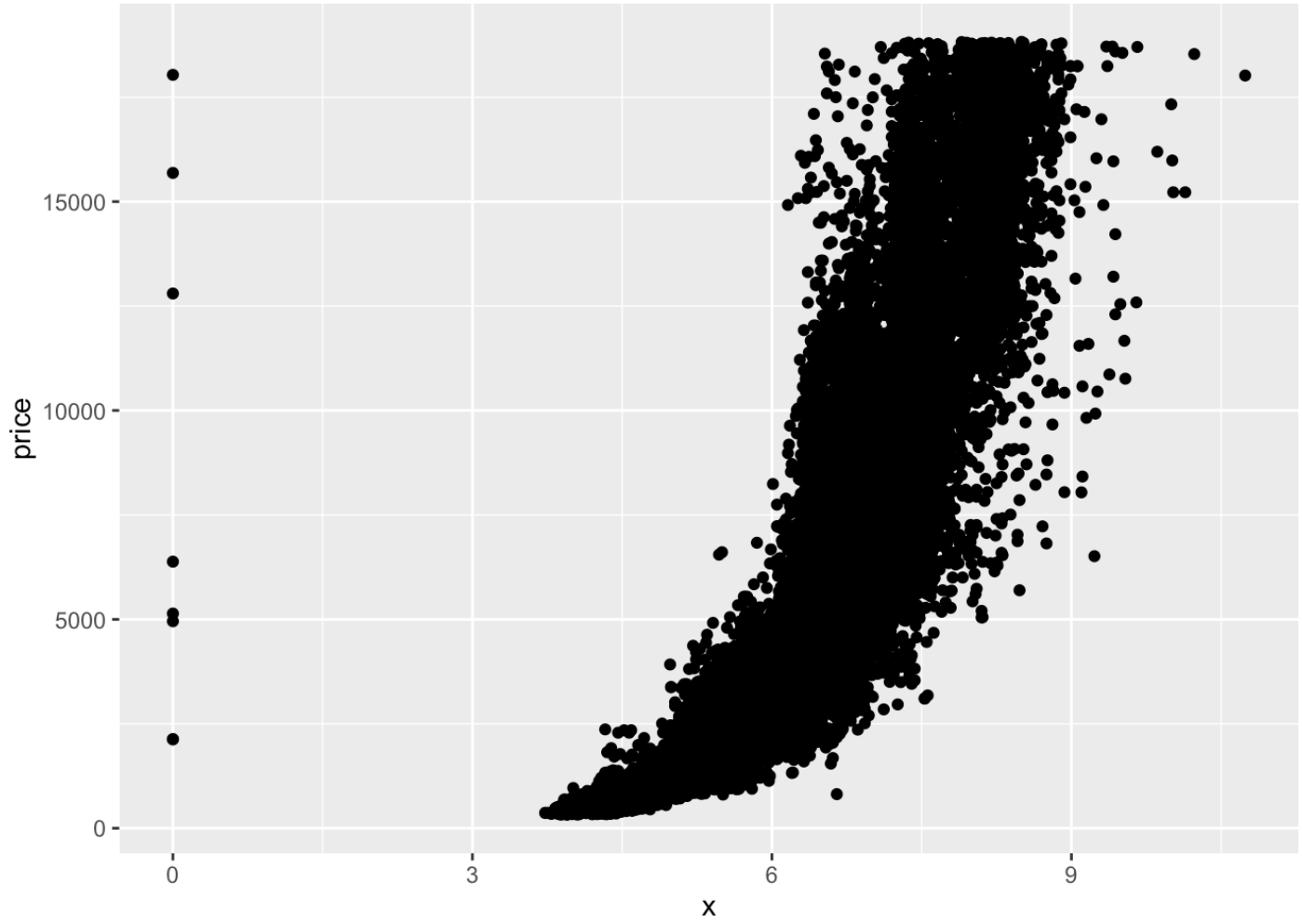
Part II: Scatterplots

For the next section, I want to create a scatterplot of price vs. x (width) using the ggplot syntax. First, I need to learn a little more about variable x.

```
summary(diamonds$x)
```

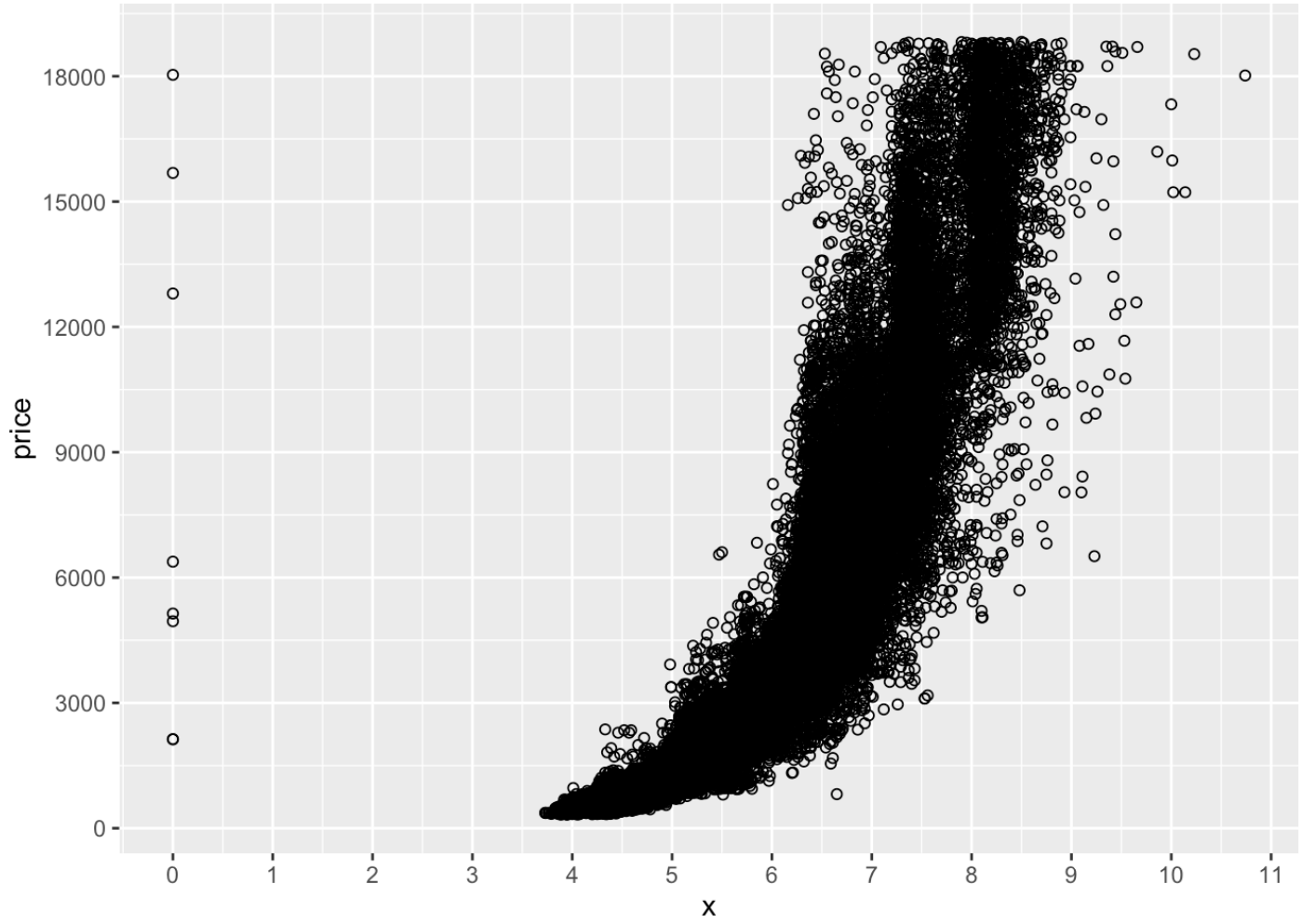
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.710   5.700   5.731   6.540  10.740
```

```
qplot(x, price, data = diamonds)
```



Initially, I get a graph that looks something like the data above. The density of these data points doesn't allow us to determine much, except that there does seem to be a positive correlation between price and width. Using ggplot, I can set the scatterplot with the function 'geom_point', as well as, scale my x and y axis.

```
ggplot(diamonds, aes(x = x, y = price))+
geom_point(shape = 1)+
scale_x_continuous(breaks = seq(0, 12, 1))+
scale_y_continuous(breaks = seq(0, 18000, 3000))
```



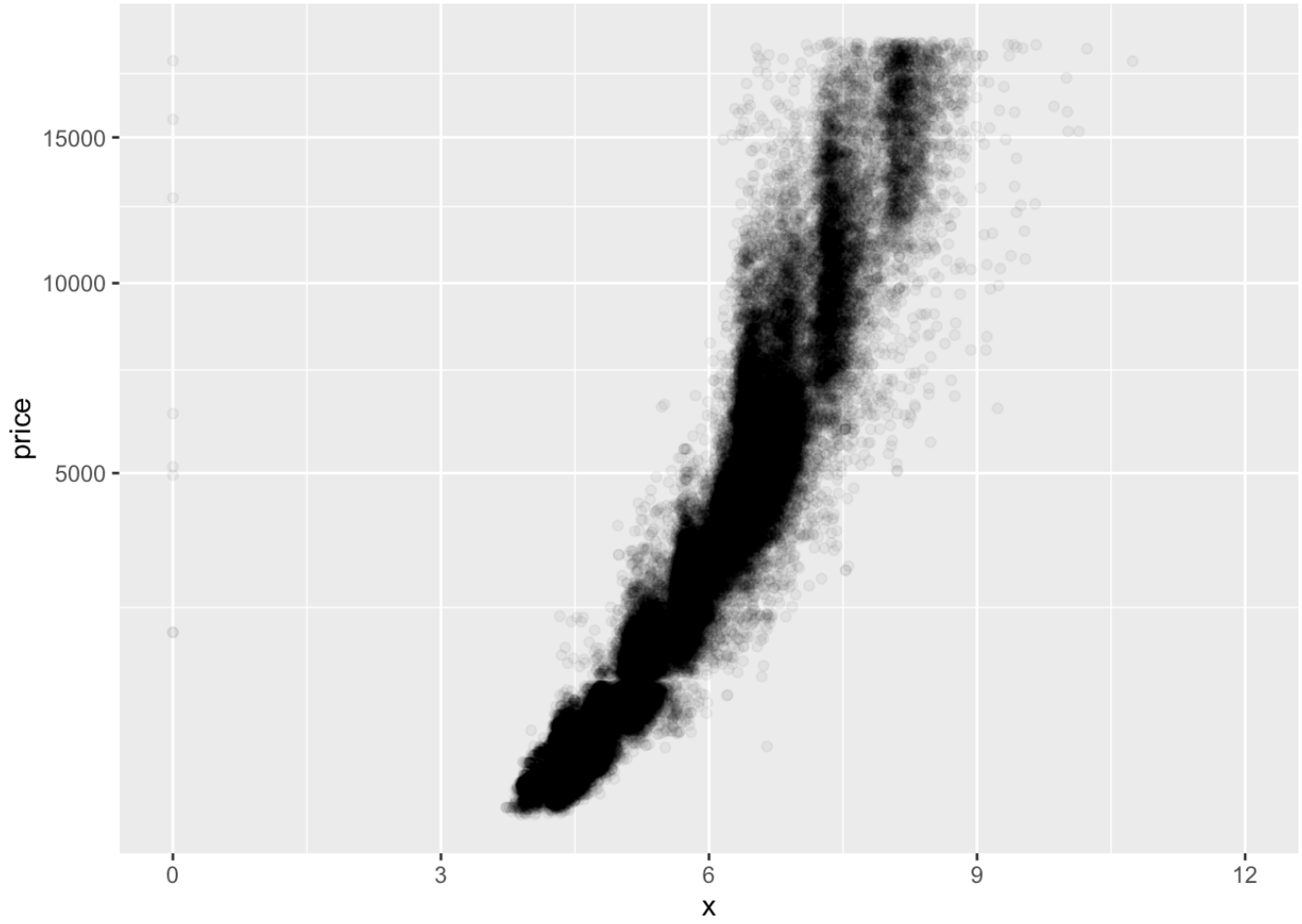
There are many differen't ways to analyze the data. See two differen't plots of the same price vs. x comparison below.

```
gp1 <- ggplot(aes(x = x, y = price), data = diamonds)+
  geom_point(alpha = 1/20)+
  xlim(0, 12)+
  coord_trans(y = 'sqrt')

gp2 <- ggplot(aes(x = x, y = price), data = diamonds)+
  geom_jitter(alpha = 1/20)+
  xlim(0,12)
  ylim(300, 15000)
```

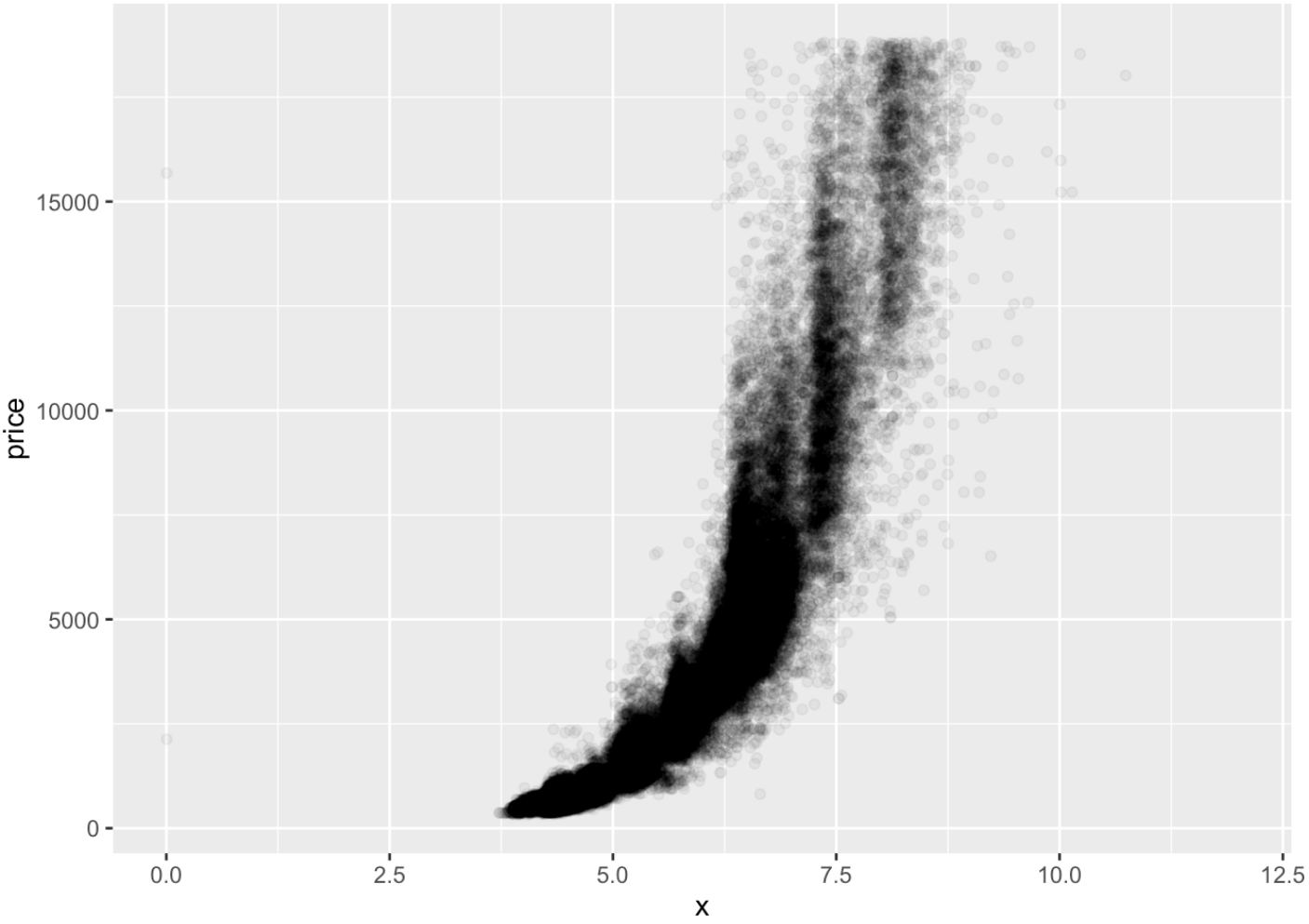
```
## <ScaleContinuousPosition>
##  Range:
##  Limits:  300 -- 1.5e+04
```

```
gp1
```




```
gp2
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



Correlation

We can also look further into what the actual correlation is between price and width of the diamonds. the cor.test function shows us the correlation confidence interval is .88. Depending on the threshold, this could be evidence there there is a significant correlation between these two variables.

```
cor.test(diamonds$x, diamonds$price, method = "pearson")
```

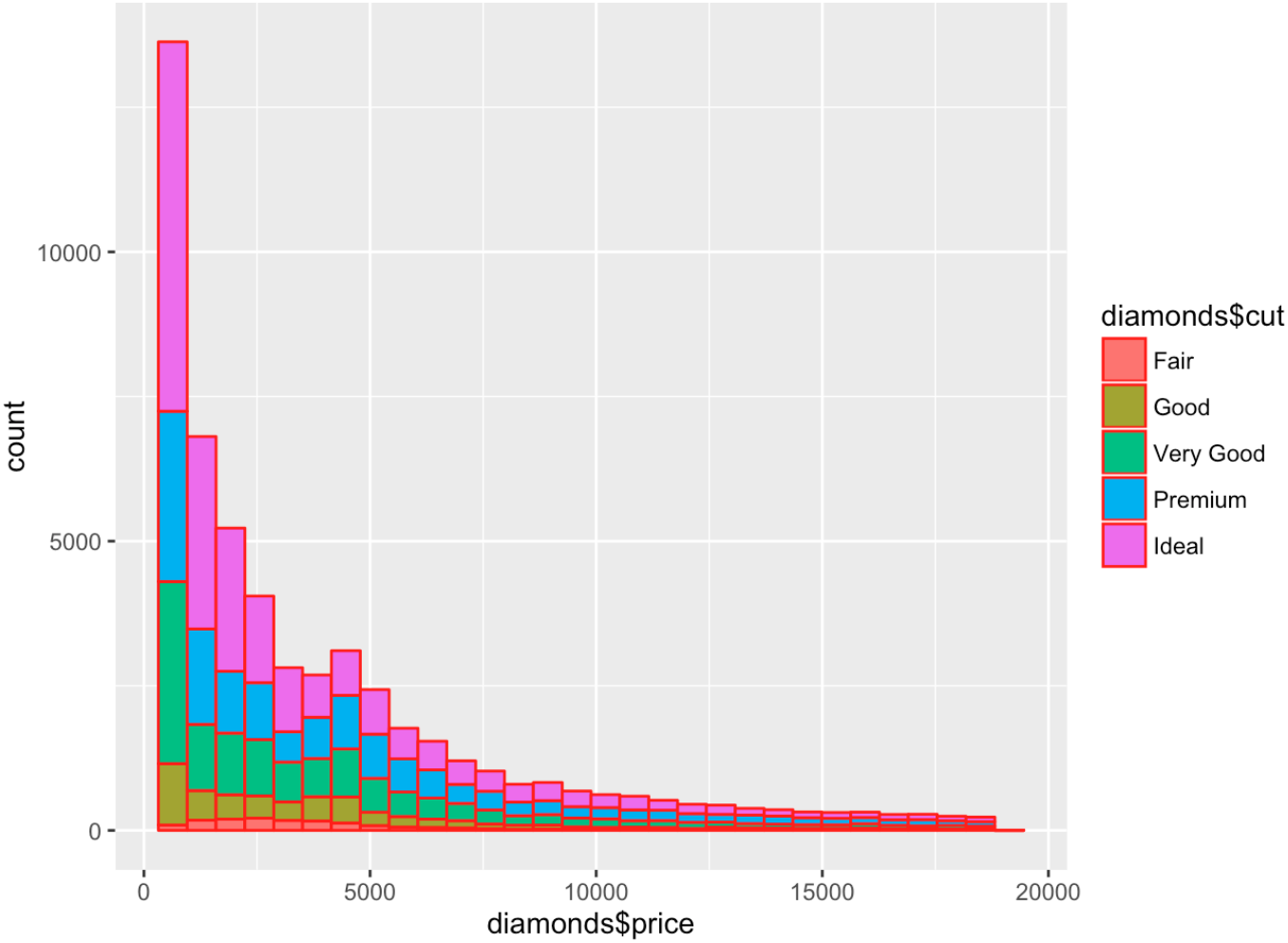
```
##
##  Pearson's product-moment correlation
##
## data:  diamonds$x and diamonds$price
## t = 440.16, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8825835 0.8862594
## sample estimates:
##       cor
## 0.8844352
```

Part III: Histogram

In this final section, I wanted to create a histogram of diamond prices and facet the histogram by diamond color. I used the diamond cut variable to color the histogram.

```
 ggh <-  ggplot(data = diamonds, aes(diamonds$price)) +
    geom_histogram(col="red",
                   aes(fill = diamonds$cut))
ggh
```

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




```
 ggh + facet_wrap(~color, ncol = 2)+
xlim(300,15000)+
ylim(0, 5000)
```

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

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

```
## Warning: Removed 35 rows containing missing values (geom_bar).
```