

EDA-group6

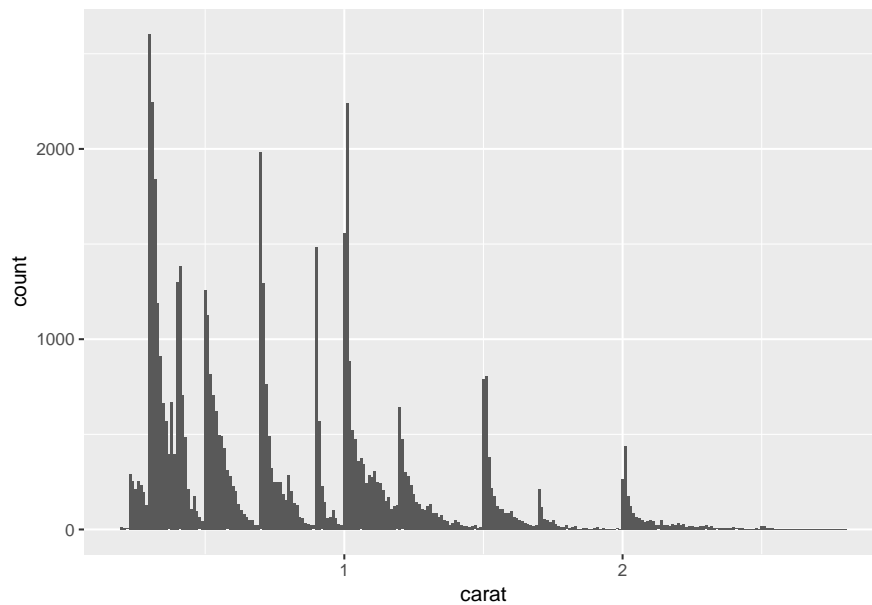
*group 6: Finn Womack, Lorne Curran, Martin Leandro Uranga Priore, Chenyang Duan,
Emerson Webb, Jiarui Xu*

5/16/2019

Visualizing distributions with `geom_freqpoly()` (7.3.1) - Lorne

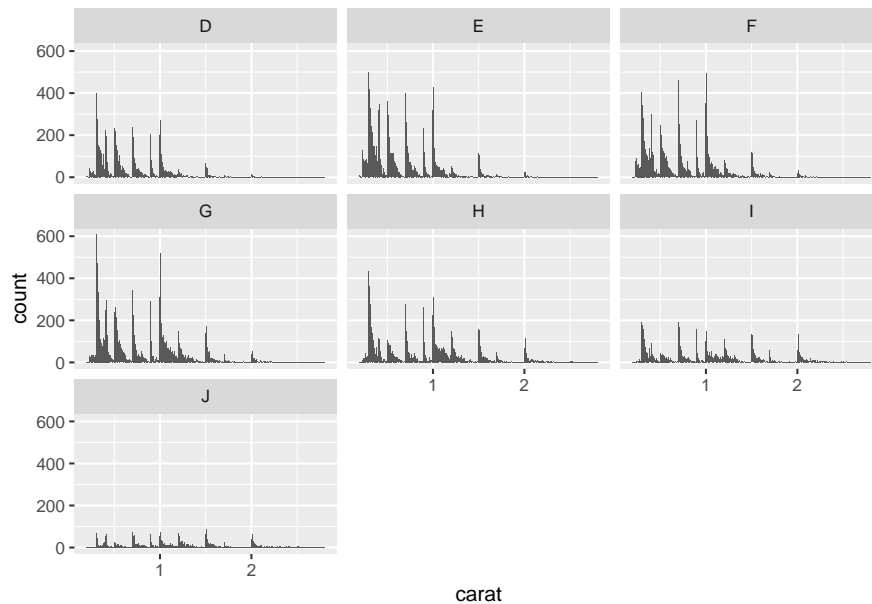
Histograms display the distribution of a continuous variable, preferably with a user-selected bin width:

```
smaller <- diamonds %>%  
  filter(carat < 3)  
smaller %>% ggplot(mapping = aes(x = carat)) +  
  geom_histogram(binwidth = 0.01)
```



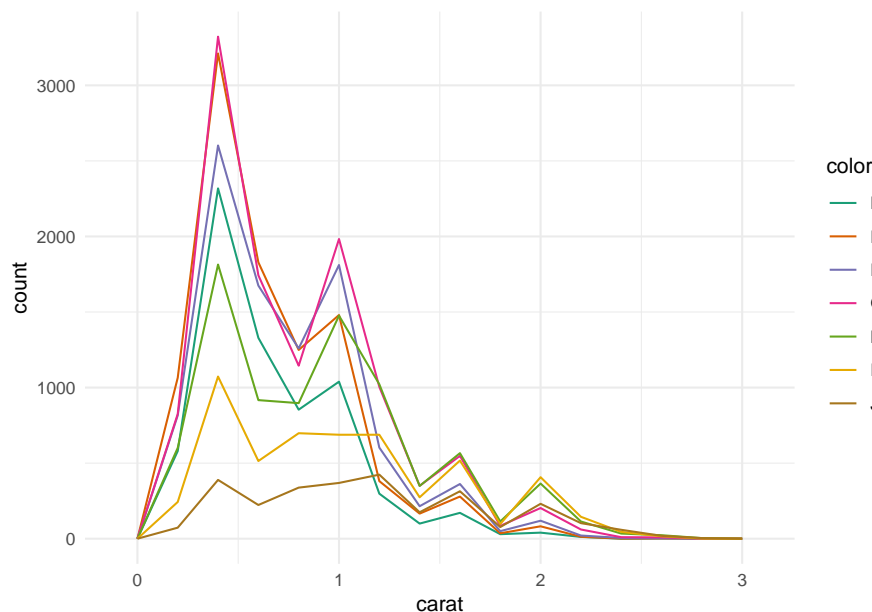
To look at `carat`'s distribution within groups of a categorical co-variable, we might use `facet_wrap`:

```
smaller %>% ggplot(mapping = aes(x = carat)) +  
  geom_histogram(binwidth = 0.01) +  
  facet_wrap(~ color)
```



Or we can employ `geom_freqpoly` to display the distribution of `carat` for each level of `color` as lines on the same plot. This is great for direct comparisons.

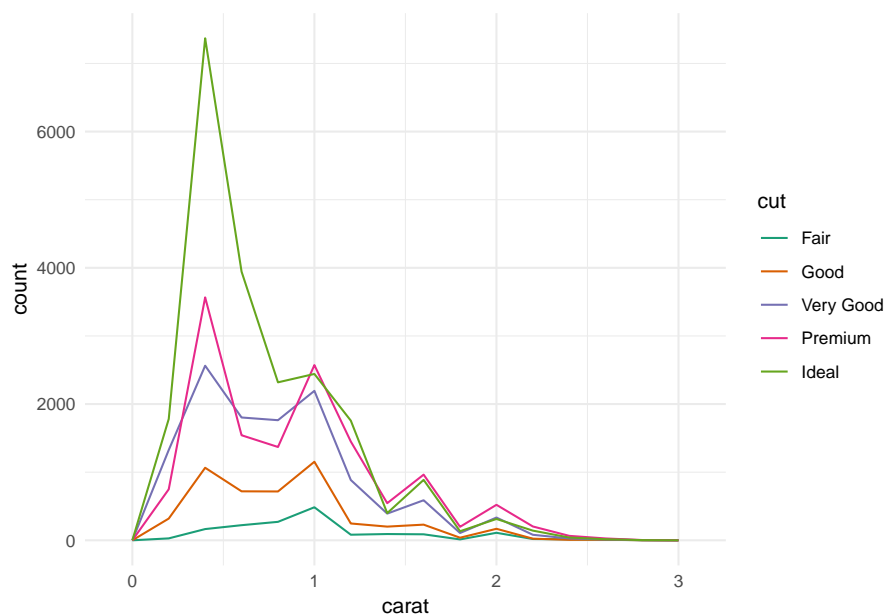
```
smaller %>% ggplot(mapping = aes(x = carat, color = color)) +  
  geom_freqpoly(binwidth = 0.2) + # upping binwidth  
  scale_color_brewer(palette = "Dark2") +  
  theme_minimal()
```



There's 2 problems with this particular example. The number of levels displayed is at the limit of what we may want to present in a single graph. And, despite the increase in bin width, it's hard to distinguish the distribution of a given group when its values are much smaller than others. To avoid the first situation, we

could go back to the `carat` and `cut` example (below) for demonstration purposes. To solve the 2nd problem when group sizes are so disparate, we'd go on to R4DS 7.5.1 **A categorical and continuous variable**.

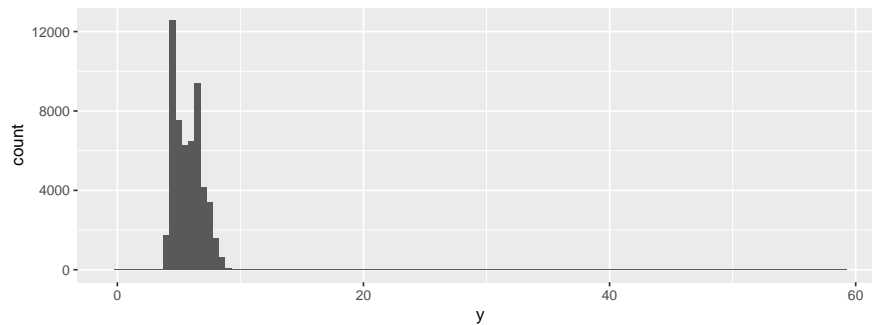
```
smaller %>% ggplot(mapping = aes(x = carat, color = cut)) +  
  geom_freqpoly(binwidth = 0.2) +  
  scale_color_brewer(palette = "Dark2") +  
  theme_minimal()
```



To deal with unusual values (7.3.3) (7.4) - Sam

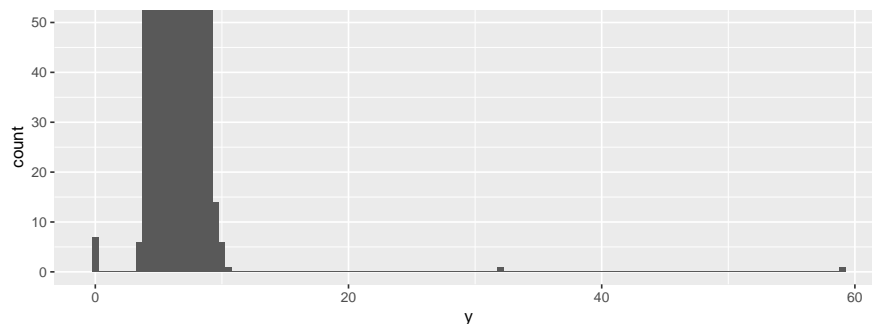
Usually when we visualizing the distribution of a variable using histogram, we may see an unpleasant scale of the axis of variable like:

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



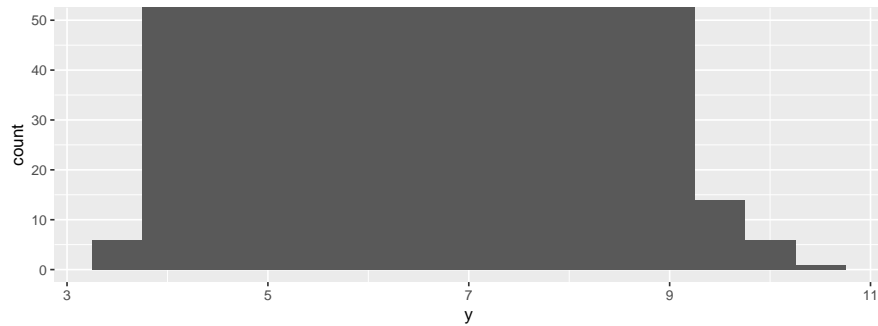
In this plot we can see the scale of x varies from 0 to 60 with most of the area blank, which is unpleasant. One guess is there might be unusual values like outliers in the data. To check them out we can reduce limits of y-axis to look into small parts of the plot:

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



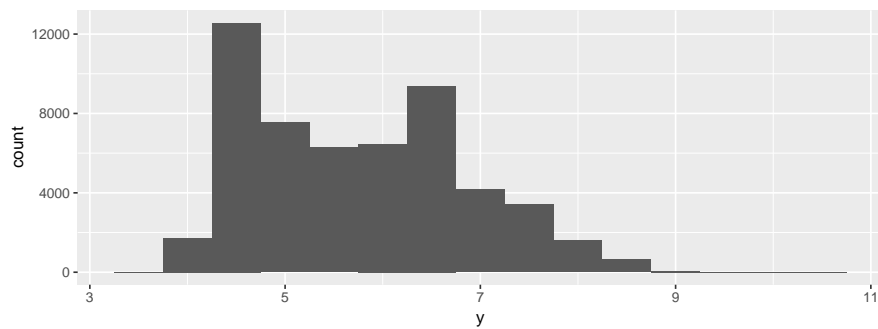
Here we can see a value of 0, a value around 32 and a value around 58 might be 2 unusual values in the dataset. The next thing we want to do is to remove them. Intuitively we can do this by pick out the part of data where y is between 3 to 20, using `between()`:

```
diamonds2 <- diamonds %>%  
  filter(between(y, 3, 20))  
ggplot(diamonds2) +  
  geom_histogram(mapping = aes(x = y), binwidth = 0.5) +  
  coord_cartesian(ylim = c(0, 50))
```



Now we see unusual values are no longer there. But this might cause problem if there do have some important data outside of this range. So instead we can generate another variable where we set unusual values to 'NA' using `ifelse()`. When using a new variable with 'NA's' generates annoying message that "Removed xx rows.". To remove this message a simple just use the statement `'na.rm = TRUE'`:

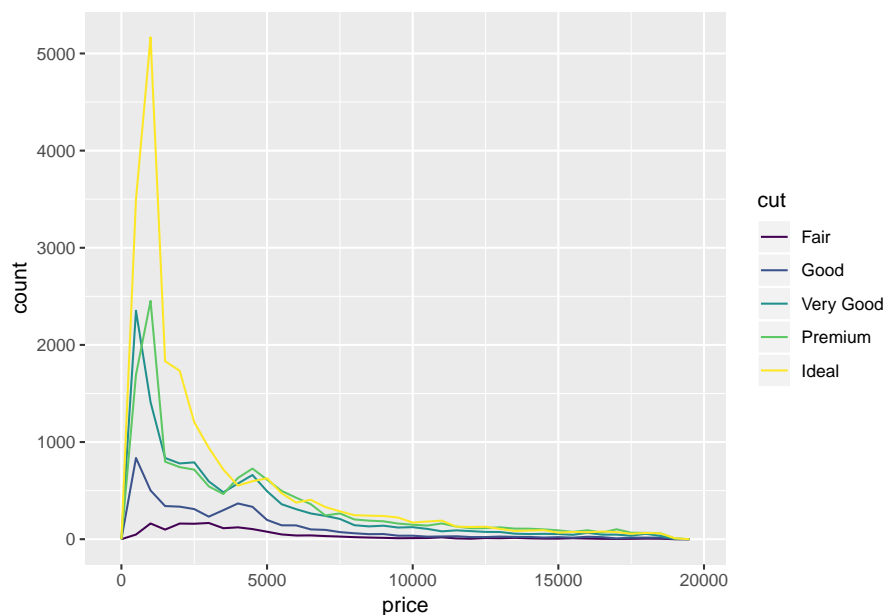
```
diamonds2 <- diamonds %>%
  mutate(y = ifelse(y < 3 | y > 20, NA, y))
ggplot(diamonds2) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5, na.rm = TRUE)
```



7.5.1 A categorical and continuous variable - Emerson

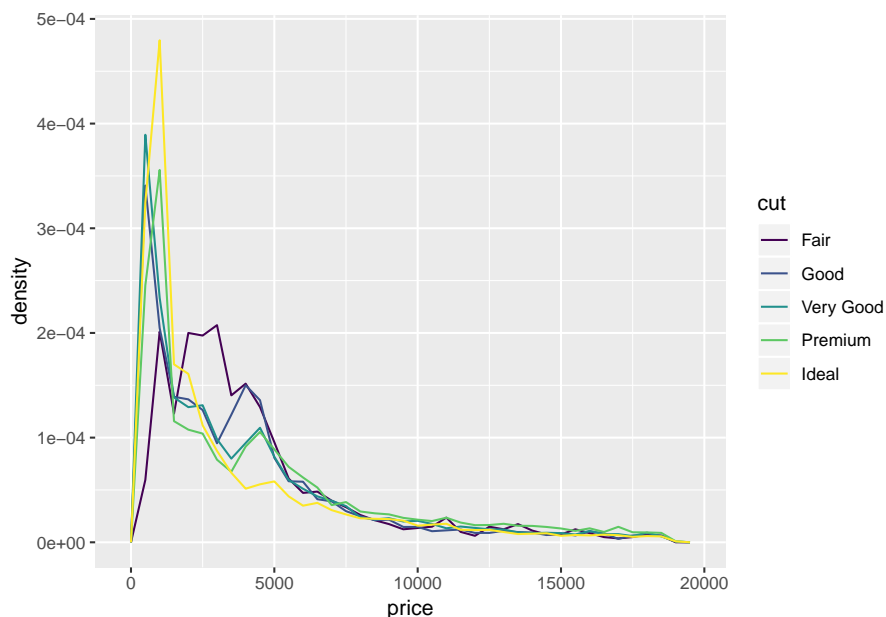
If we are interested to see how a continuous variable varies by the levels of a categorical variable, we have a few options. The first option is to use a frequency polygon:

```
ggplot(data = diamonds, mapping = aes(x = price)) +  
  geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```



One issue with this display is that group sizes vary between diamond cuts, so a plot based on counts is not very useful. Instead, inside of the mapping argument above, we can specify `y=..density..` which standardizes each distribution:

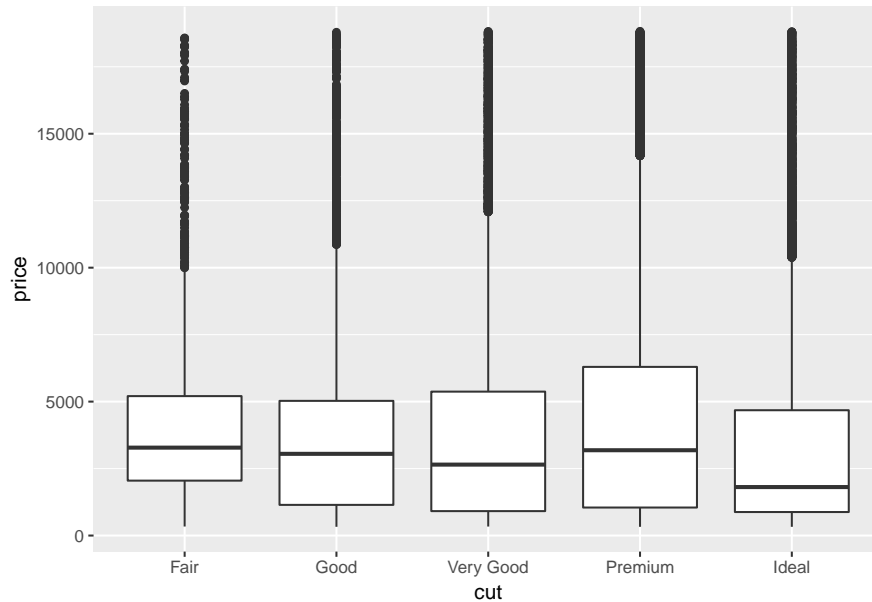
```
ggplot(data = diamonds, mapping = aes(x = price, y = ..density..)) +  
  geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```



Even with this it might be difficult to compare the variability between groups. For example, it seems that

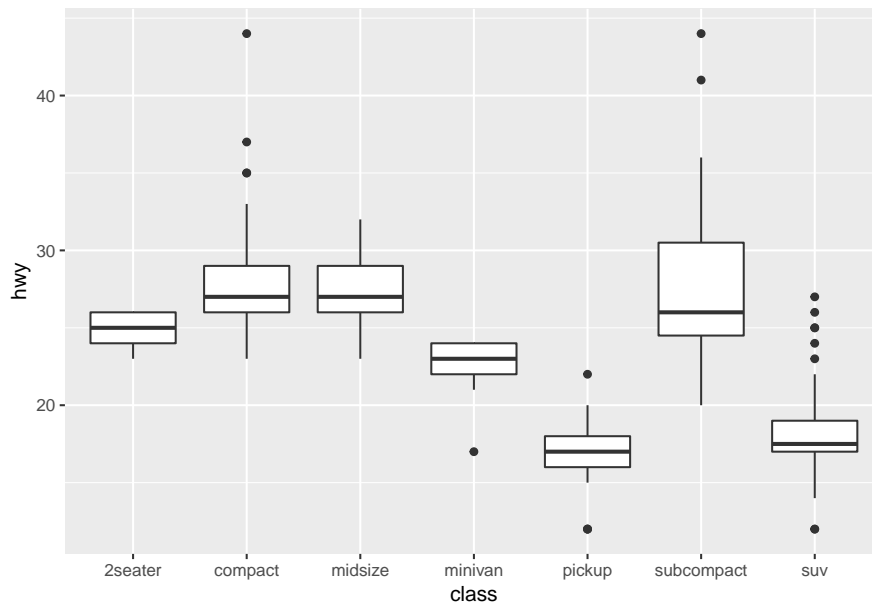
fair diamonds have the highest average price even though they are the lowest quality cut. However, since the plot is so cluttered the comparisons are not easy to make. An alternative plot to explore the variation between groups is a boxplot. With a boxplot some of the geometric information about the distribution is lost, but boxplots are much more compact and allow for easy comparisons. With this boxplot, it is easier to see that fair diamonds on average have a higher price than higher quality diamonds.

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



Finally we can make use of the **reorder** function to change the order of the categorical variable in a boxplot. As an example, we can look at the **class** variable and the **hwy** variable in the **mpg** dataset. Notice how there is no intrinsic ordering of the **class** variable.

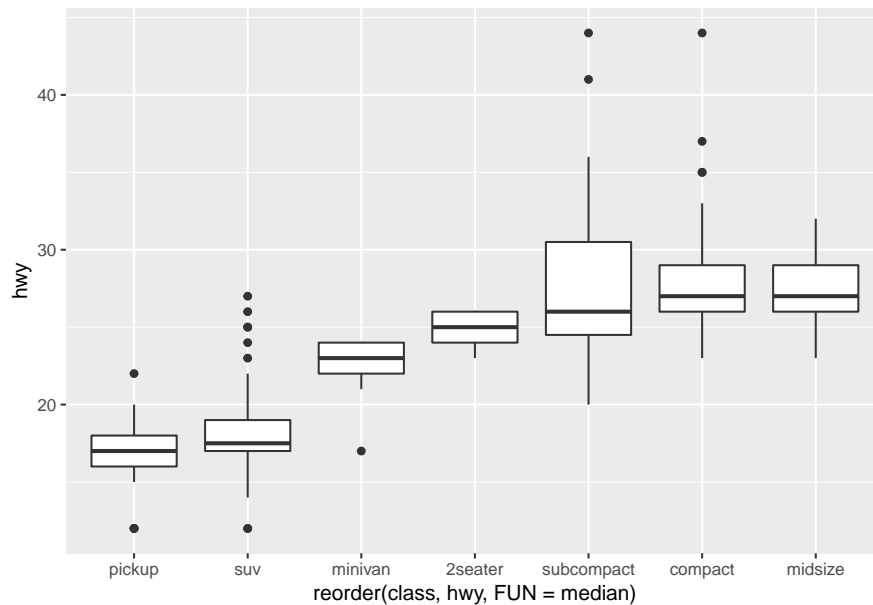
```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +  
  geom_boxplot()
```



We could use the **reorder** function to order **class** by the median value of **hwy** which makes comparisons

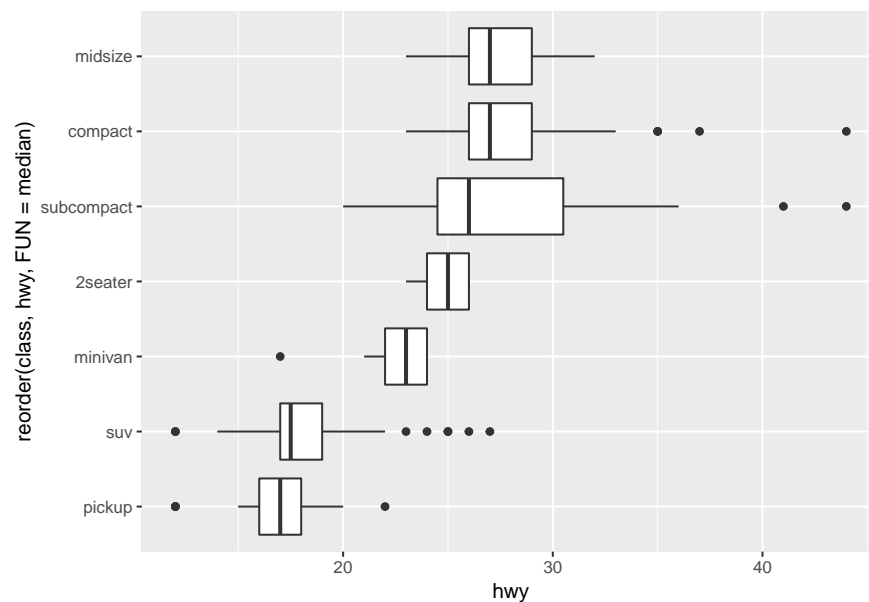
easier to make.

```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy))
```



Finally if there are long variable names, then using `coord_flip` can make the plot easier to read.

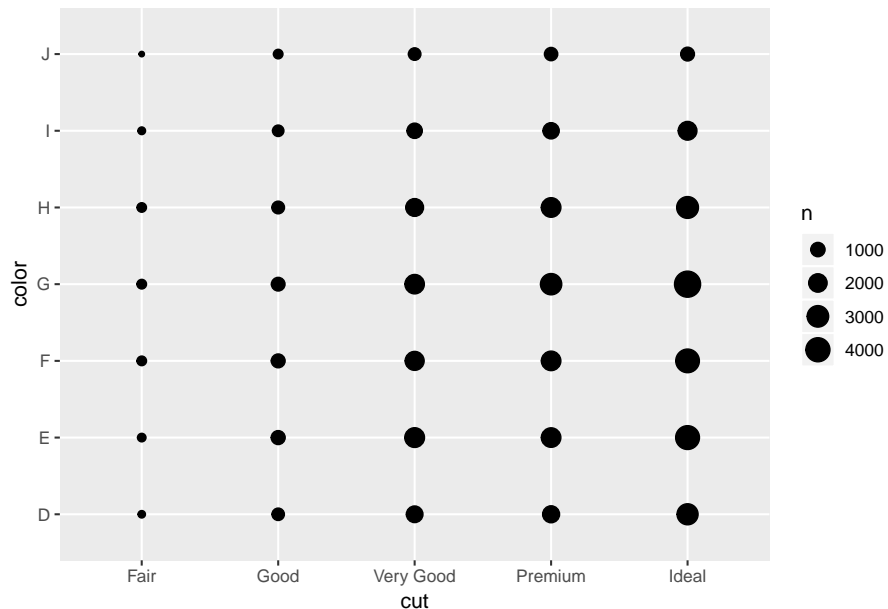
```
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy)) +  
  coord_flip()
```



Covariation between two categorical variables (7.5.2) - Martin

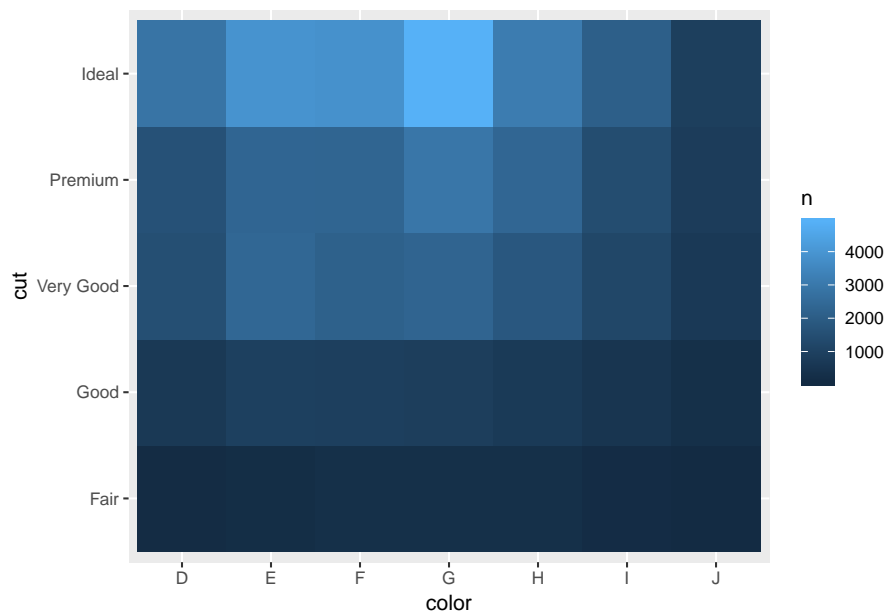
To see how two categorical variables are related we can count the number of observations for each level combination of both variables. We can do this using *geom_count* like in the following example:

```
ggplot(data = diamonds) +  
  geom_count(mapping = aes(x = cut, y = color))
```



This graph has the disadvantage that we have to interpret the size of the dots which unless the difference between to neighbors is huge it can be difficult to assess the magnitude. One option, it to use *count* of *ddply* and then use the function *geom_tile* which make the visualization easier to interpret since use colors to display the covariation between both variables.

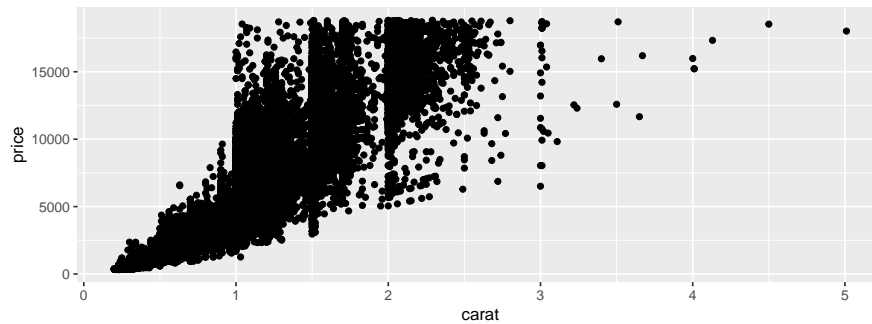
```
diamonds %>%  
  count(color, cut) %>%  
  ggplot(mapping = aes(x = color, y = cut)) +  
    geom_tile(mapping = aes(fill = n))
```



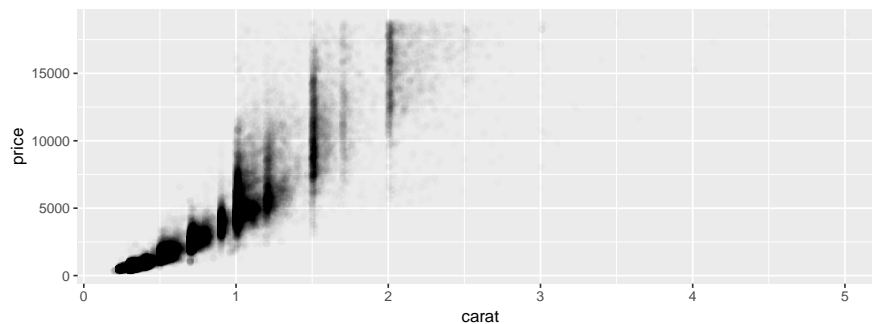
Covariation between two continuous variables (7.5.2) (7.6) - Finn

We've already used scatterplots to examine the covariance between two continuous variables which tend to get cluttered with a lot of data. In the past we've added transparency using `alpha` to combat this when examining scatterplots with a lot of data:

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

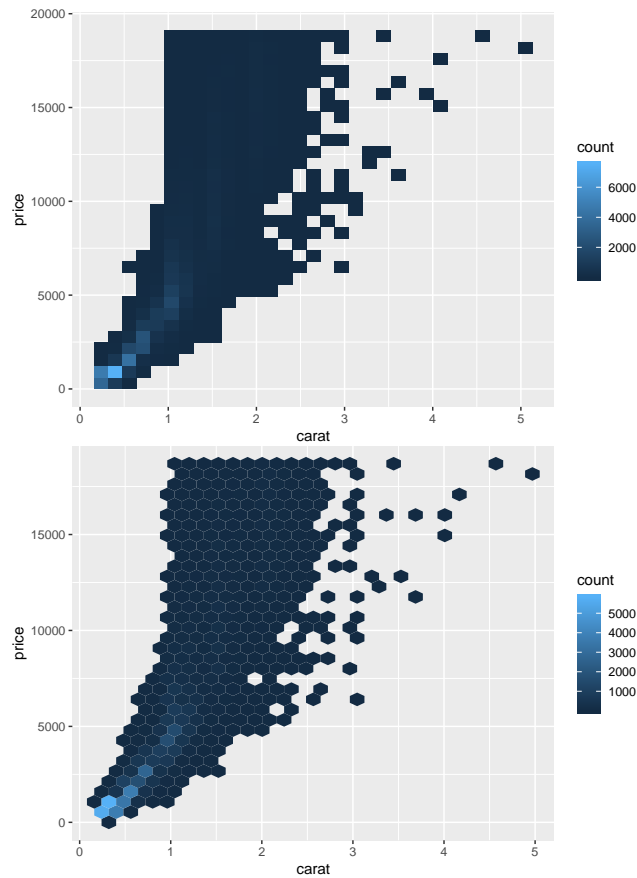


```
ggplot(data = diamonds) +  
  geom_point(mapping = aes(x = carat, y = price), alpha = 1 / 100)
```



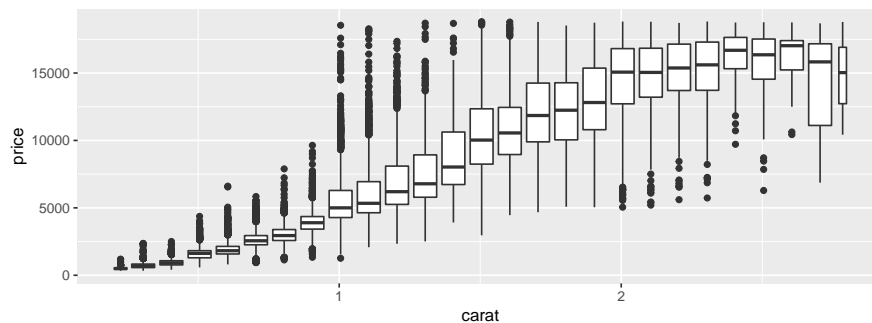
However, this technique of adjusting `alpha` can sometimes be difficult to use. If the data is too large and transparency proved challenging then another possible way of exploring covariance between two continuous variables, which might be useful in this case, is to put them into bins and adjusting the color instead of transparency by using the `geom_bin2d()` and `geom_hex()` functions.

```
ggplot(data = diamonds) +  
  geom_bin2d(mapping = aes(x = carat, y = price))  
ggplot(data = diamonds) +  
  geom_hex(mapping = aes(x = carat, y = price))
```



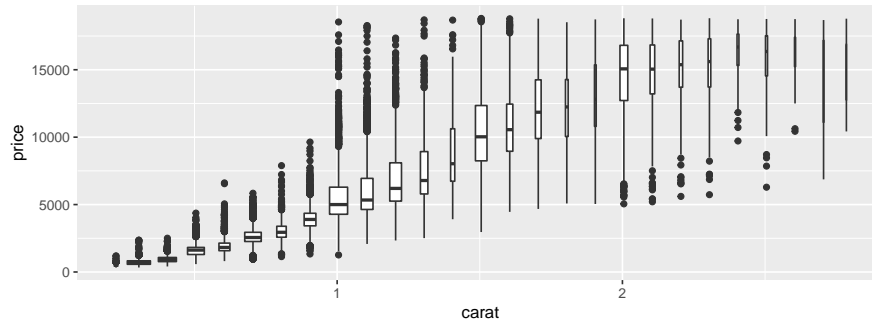
The binning strategy can also be selectively applied to just one of the variables then you can plot them as if one were continuous and the other categorical.

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



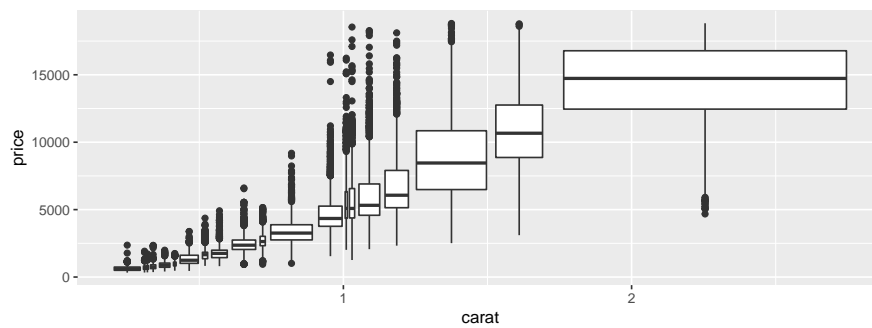
If we want to also have an indicator as to how many points each plot is summarizing we can use `varwidth = TRUE`.

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



This way we can compare the boxes size to see which ones are summarizing smaller or larger sets of datapoints. Since in this case some of the boxes can be hard to see another way of dealing with this problem is to instead divide the variable into equally sized partions for each bin with the size of the bins instead dictating the range over the axis (since they now all have roughly the same number of data points) using `cut_number()`.

```
ggplot(data = smaller, mapping = aes(x = carat, y = price)) +  
  geom_boxplot(mapping = aes(group = cut_number(carat, 20)))
```

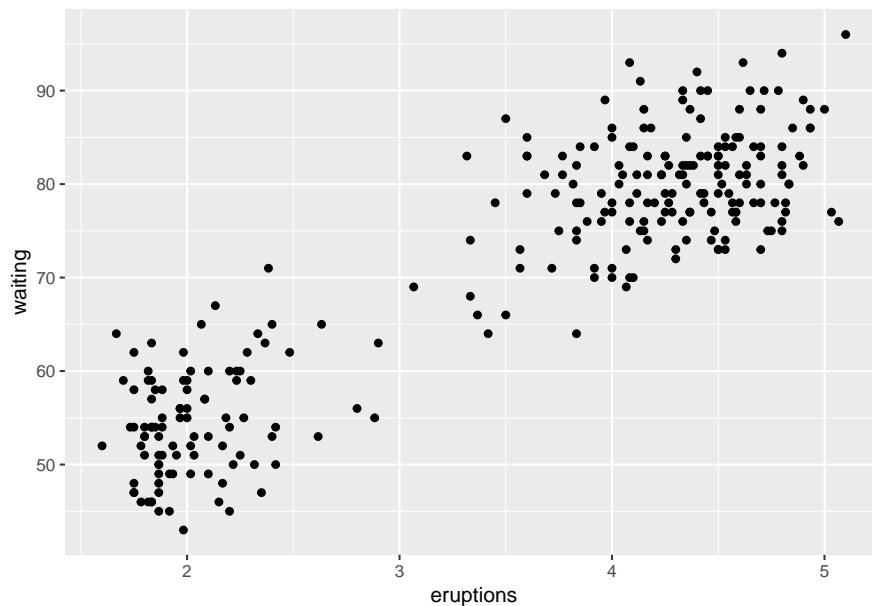


Patterns and models (7.6) -Chenyang

Patterns in your data provide clues about relationships. If a systematic relationship exists between two variables it will appear as a pattern in the data.

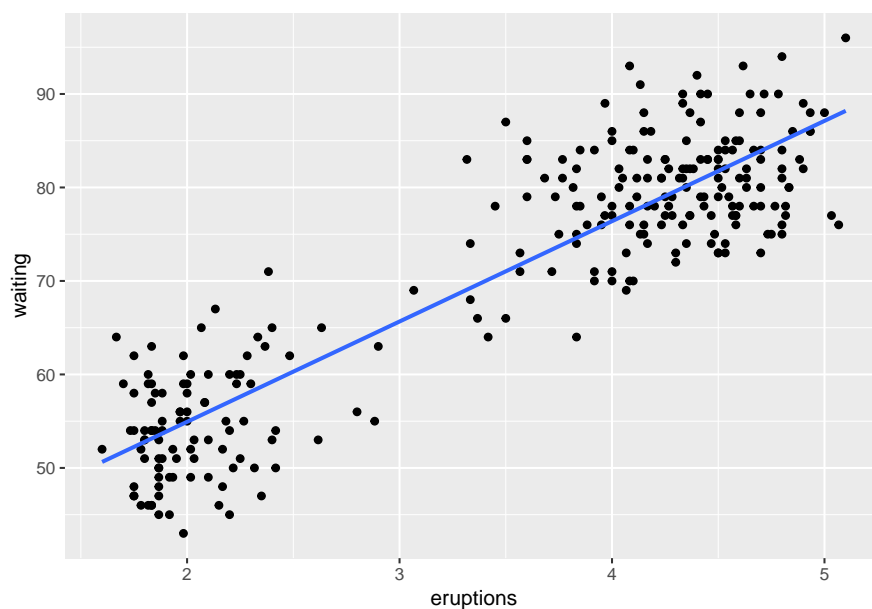
For example, if we draw a scatterplot of Old Faithful eruption lengths versus the wait time between eruptions. Obviously, we can find two patterns in the following plot.

```
ggplot(data = faithful) +  
  geom_point(mapping = aes(x = eruptions, y = waiting))
```



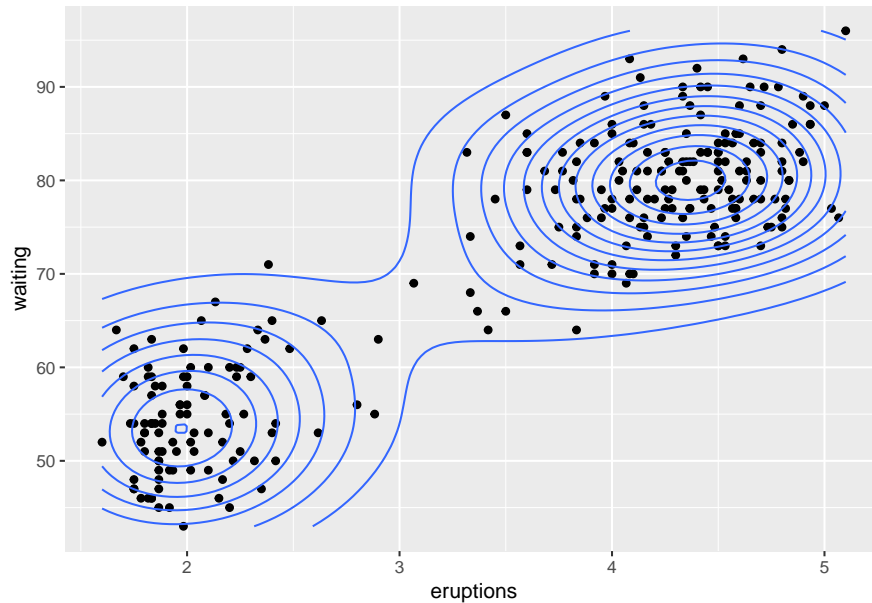
First of all, there is positive correlation between the Old Faithful eruption lengths and the waiting time.

```
ggplot(data = faithful) +  
  geom_point(mapping = aes(x = eruptions, y = waiting)) +  
  stat_smooth(mapping = aes(x = eruptions, y = waiting), method = "lm", se=FALSE)
```



Secondly, the scatterplot also displays the two clusters.

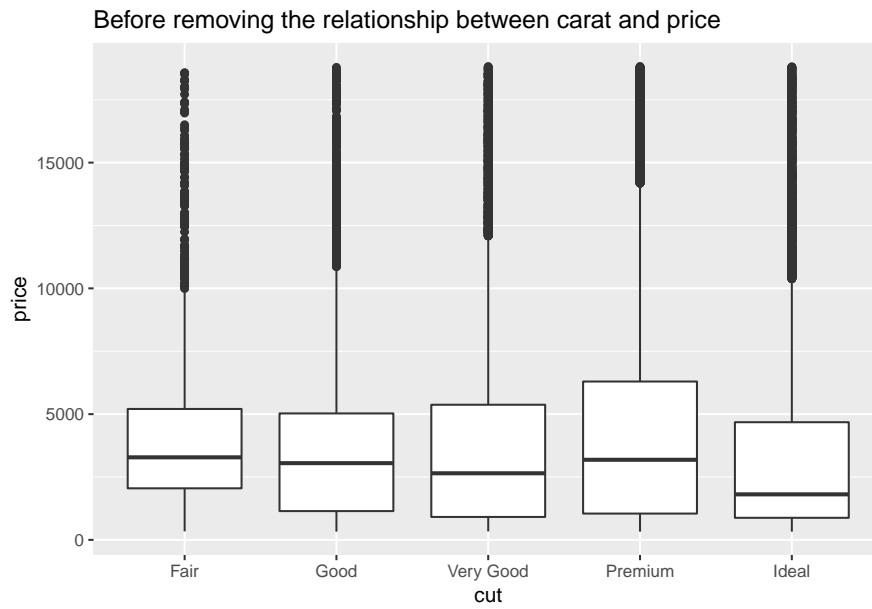
```
ggplot(data = faithful, aes(x=eruptions,y=waiting)) +  
  geom_point() +geom_density_2d()
```



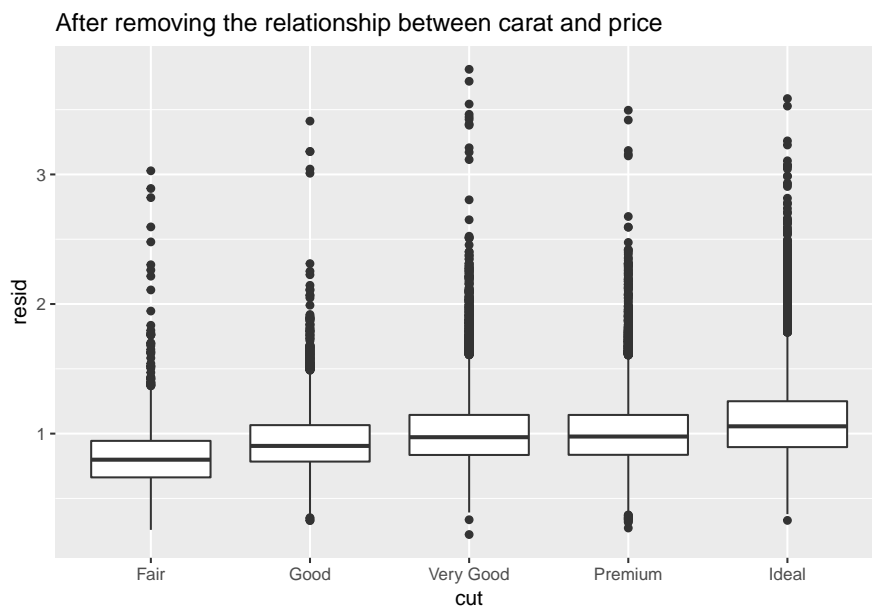
Patterns provide one of the most useful tools for data scientists because they reveal covariation.

Models are a tool for extracting patterns out of data. For example, consider the diamonds data. It's hard to understand the relationship between cut and price, because cut and carat, and carat and price are tightly related. It's possible to use a model to remove the very strong relationship between price and carat so we can explore the subtleties that remain.

```
library(modelr)  
  
mod <- lm(log(price) ~ log(carat), data = diamonds)  
  
diamonds2 <- diamonds %>%  
  add_residuals(mod) %>%  
  mutate(resid = exp(resid))  
  
ggplot(data = diamonds, mapping = aes(x = cut, y = price)) +  
  geom_boxplot() + labs(title = "Before removing the relationship between carat and price")
```



```
ggplot(data = diamonds2) +  
  geom_boxplot(mapping = aes(x = cut, y = resid)) + labs(title = "After removing the relationship between
```



And we can see that once we remove the relationship between carat and price. We can clearly see that better quality diamonds are more expensive.