EDA Techniques Summary

Group 6

5/11/2020

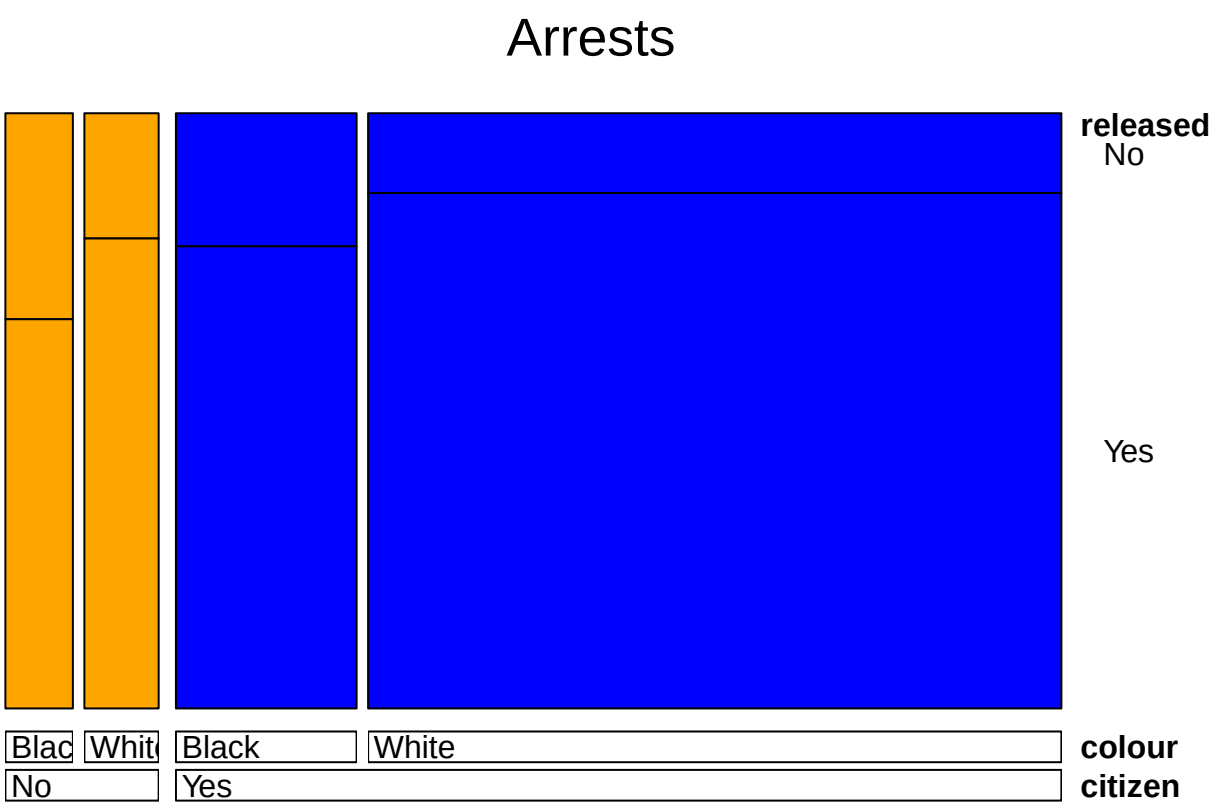
7.3.1 Visualizing Distributions

doubledecker()

doubledecker() is a mosaic plot that is used for categorical data to visualize contingency tables. The plots offer a graphical way to display dependencies and associations among variables in a data set. The api is in the vcd package. For example, the Arrests data from Toronto, If you want to look at the influence of "citizen" and "colour" on being "released", you can use a doubledecker plot to do this.

doubledecker() varies from a standard mosaic in that you specify one dependent variable of interest which permutates the table and is the last dimension specified in the plot. Here's an example of the generated plot.

```
dat <- Arrests
doubledecker(released ~ citizen +colour, data=Arrests, gp =( gpar(fill = c("orange", "blue"))), main=TRU
```

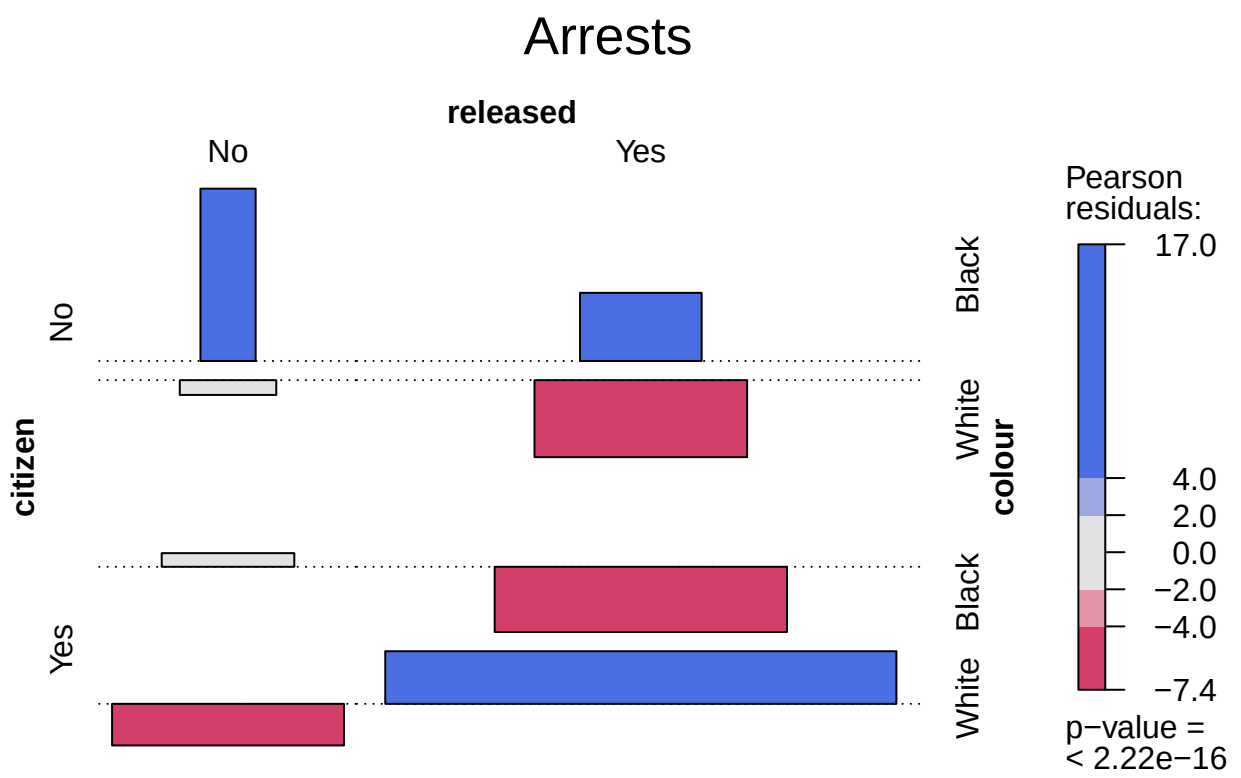


Using gp in this context applies the supplied vars to only the dependent variable.

assoc ()

Another useful mosaic plot for visualizing associations is **asssoc()**. The area of the box is proportional to the difference in observed and expected frequencies. The placement of the box is relative to the baseline for independence (0), so if it's greater than the baseline, it's above the line, otherwise its below the line.

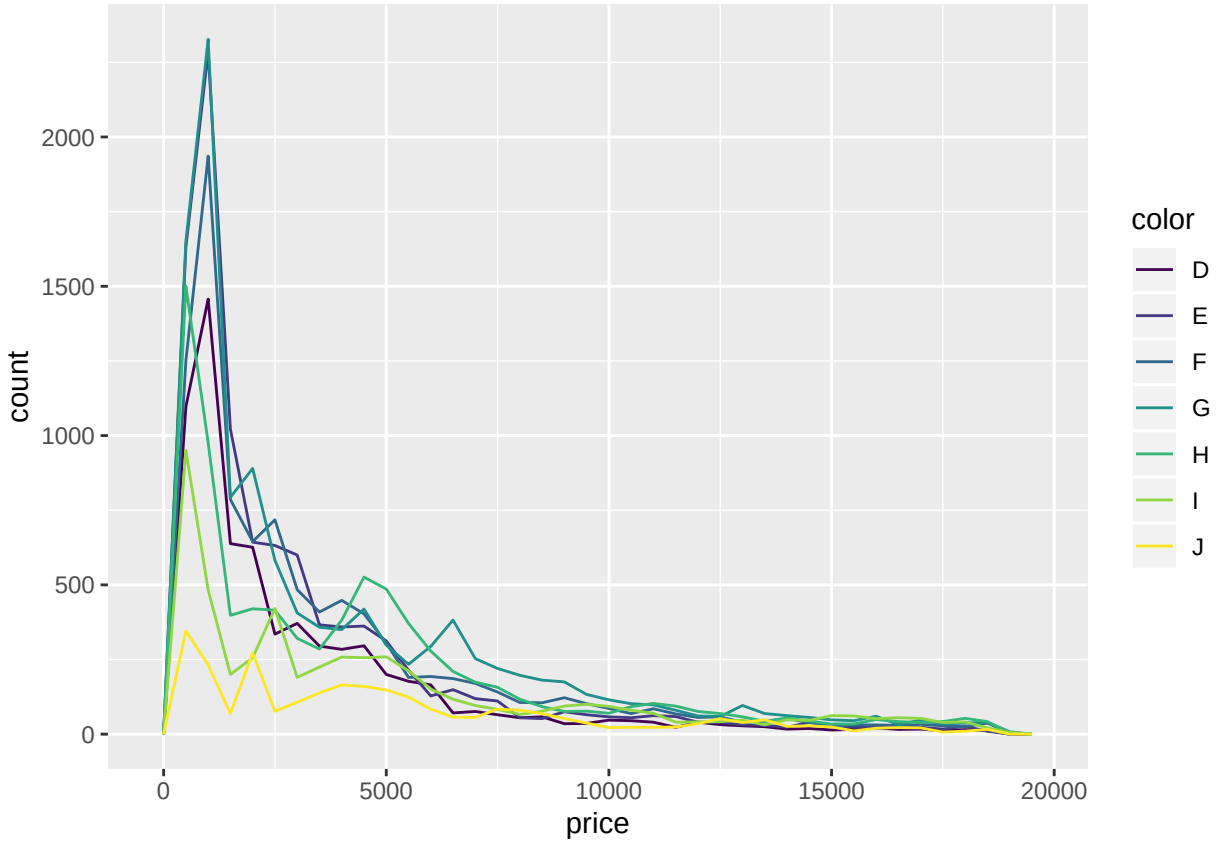
```
assoc(released ~ citizen + colour, data=Arrests, shade=TRUE, main=TRUE, comnpress = FALSE)
```



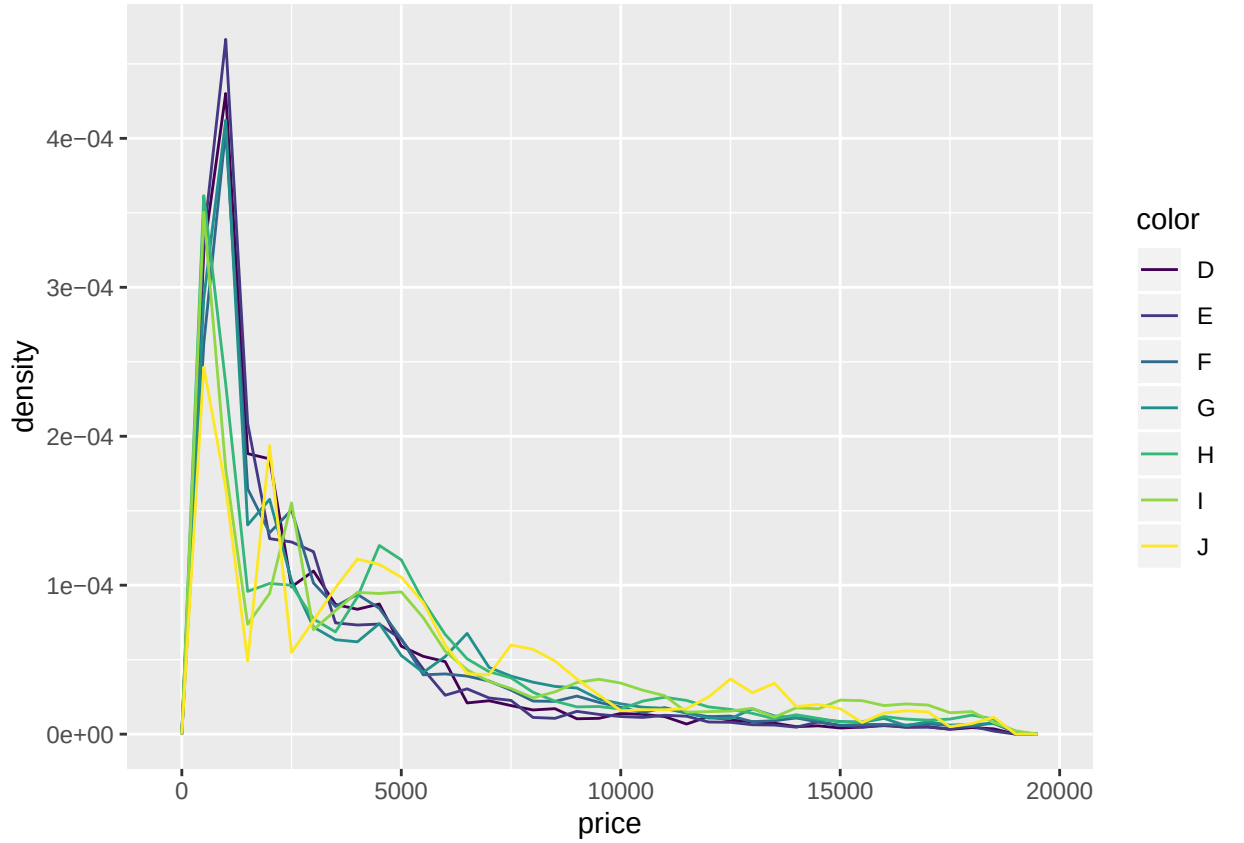
Covariation of a Categorical and a Continuous Variable:

The book gave two examples of how to deal with a categorical and continuous variable that have covariation. One technique that was given was comparing the **density** of variables rather than the **count**. This can lead to more interpretable visualization when the counts across categorical variables are not very even. The below plots compare the same data of the diamonds data set, but rather than looking at count the second plot looks at density. You can see that colors H and I and J appear to more regulary have diamonds in a higher price point.

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

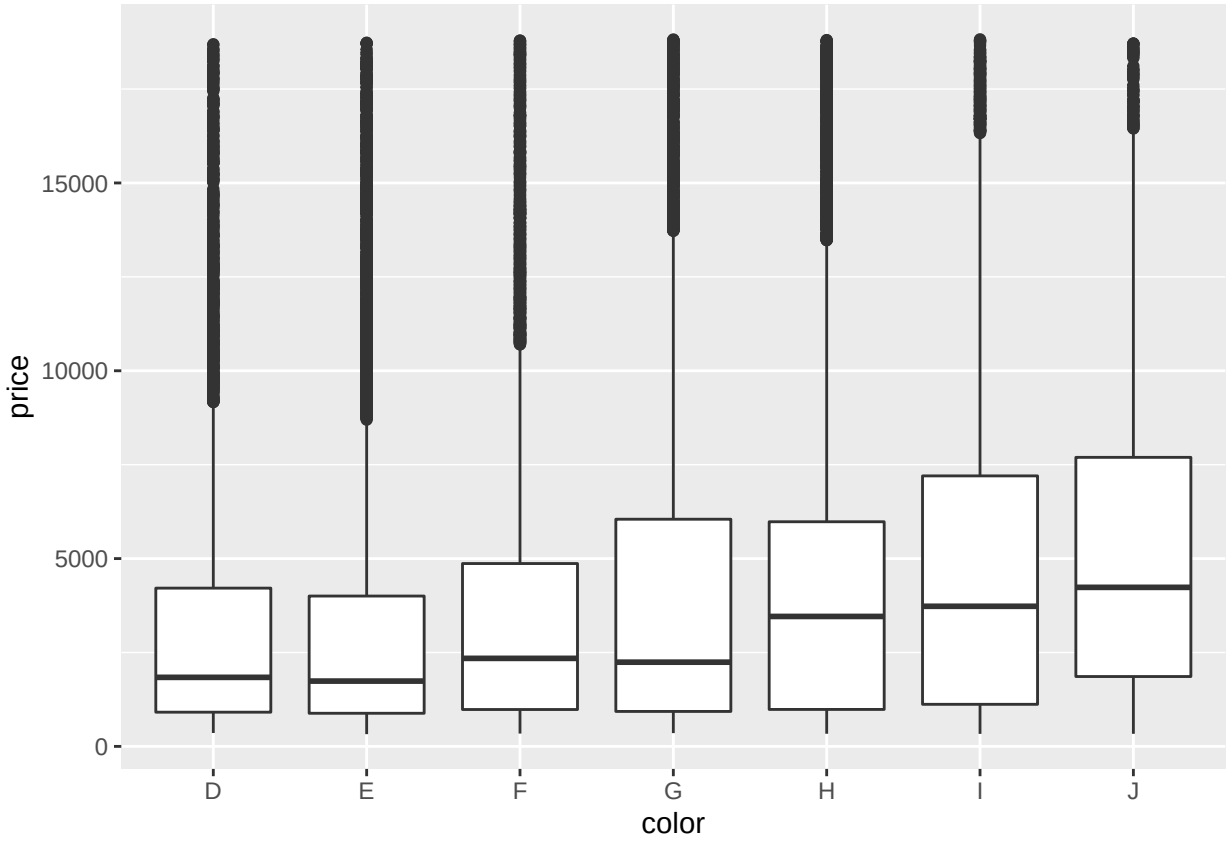



```
ggplot(data = diamonds, mapping = aes(x = price, y = ..density..)) +
  geom_freqpoly(mapping = aes(colour = color), binwidth = 500)
```



The freq_poly plots produced 7 categories, making it difficult to read. Another example would be a box plot. The box shows the 25th, 50th and 75th percentile, with outliers as individual points and points outside of those percentiles as whiskers. Below it is much more clear that H I and J have higher average prices, but they also have wider spreads.

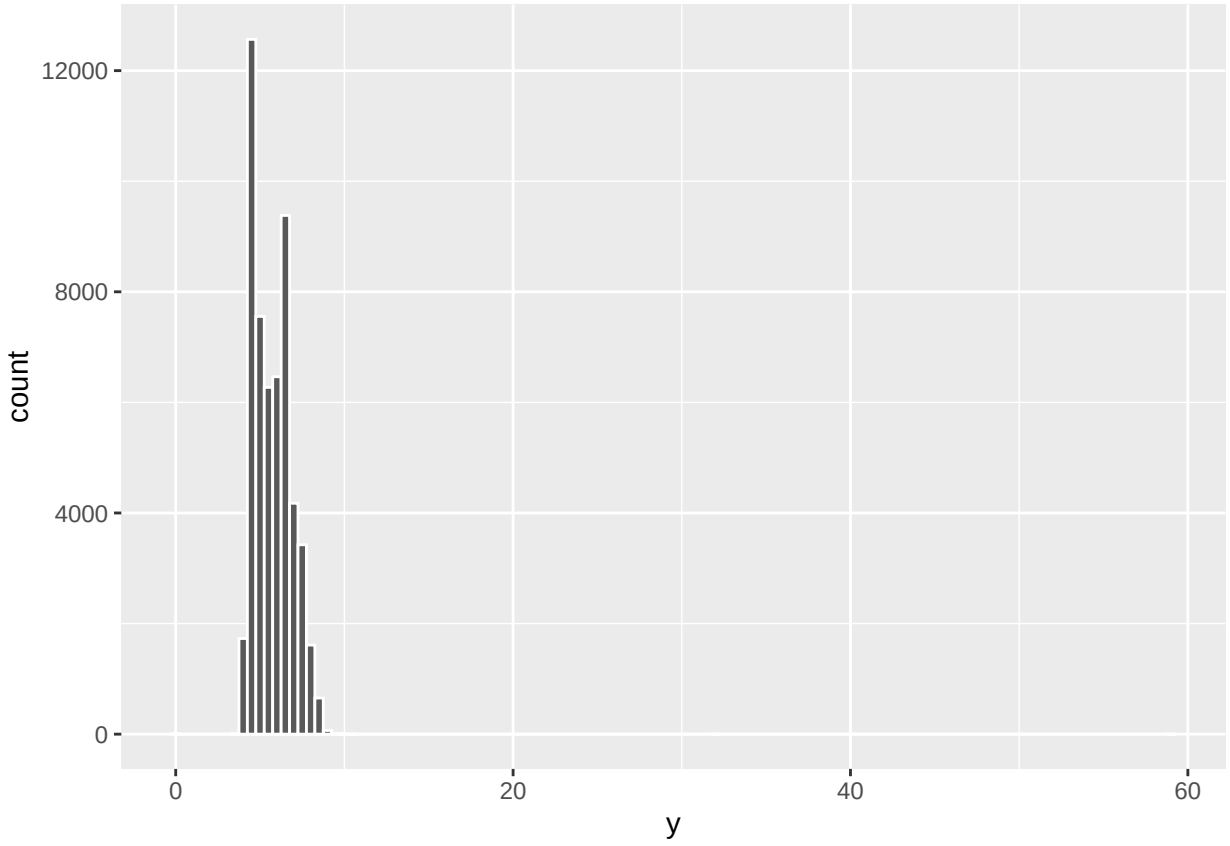
```
ggplot(data = diamonds, mapping = aes(x = color, y = price)) +
  geom_boxplot()
```



Unusual Observations:

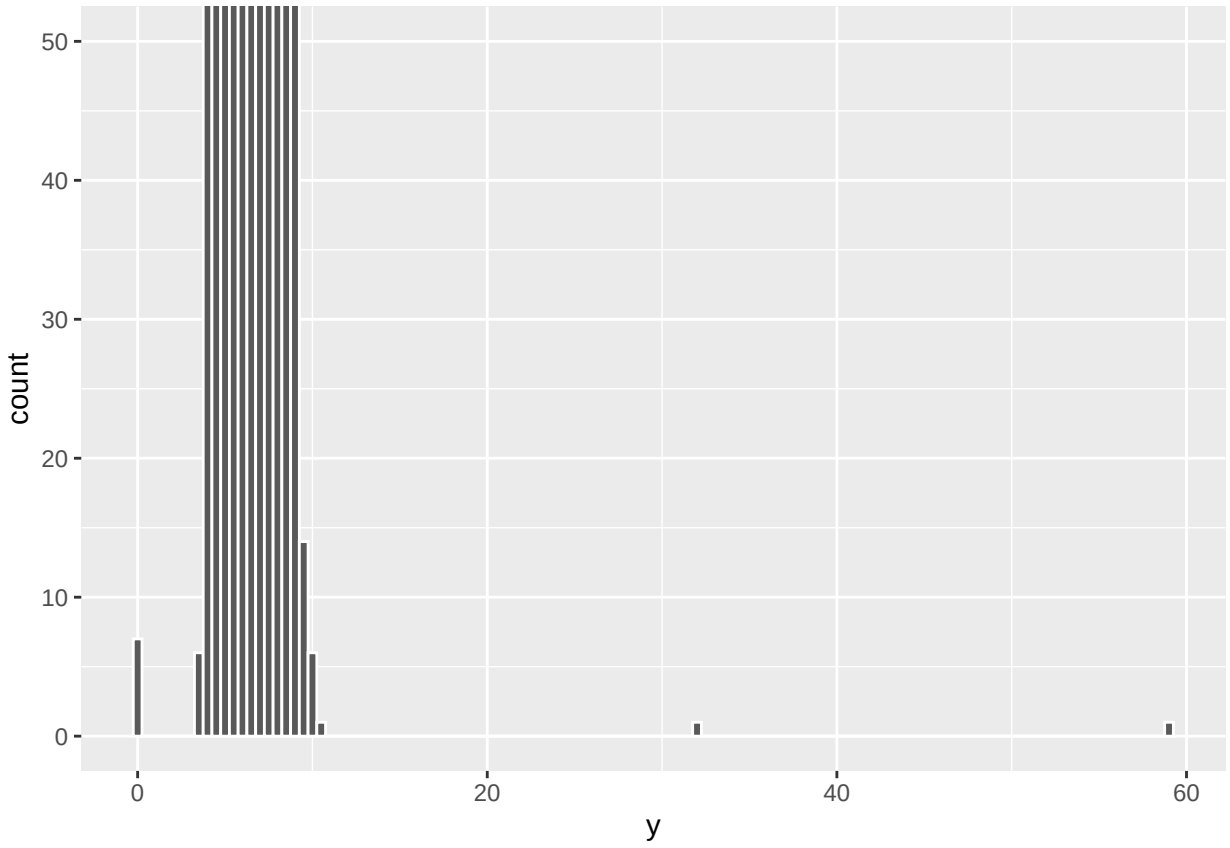
During exploratory data analysis it's always a good idea to check for outliers or unusual observations. Sometimes outliers are errors in a dataset, and sometimes they are genuine observations that are just outside the normal bounds for a particular variable. Often times it is difficult to tell, so further exploration is generally required. Let's examine the distribution of diamond sizes by plotting a histogram of the y-dimension measured in mm.

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



The unusually wide x-axis gives us reason to suspect that there may be a few outliers with counts so low that they don't even show up on a histogram. Change the scale of the y-axis to get a better look.

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



We can see two outlying observations around 30 and 60, and a handful at 0. Let's examine these cases further. We'll look at the `x` and `z` variables as well to get a better idea of the actual sizes and we will include `price` to see if there is anything unusual there.

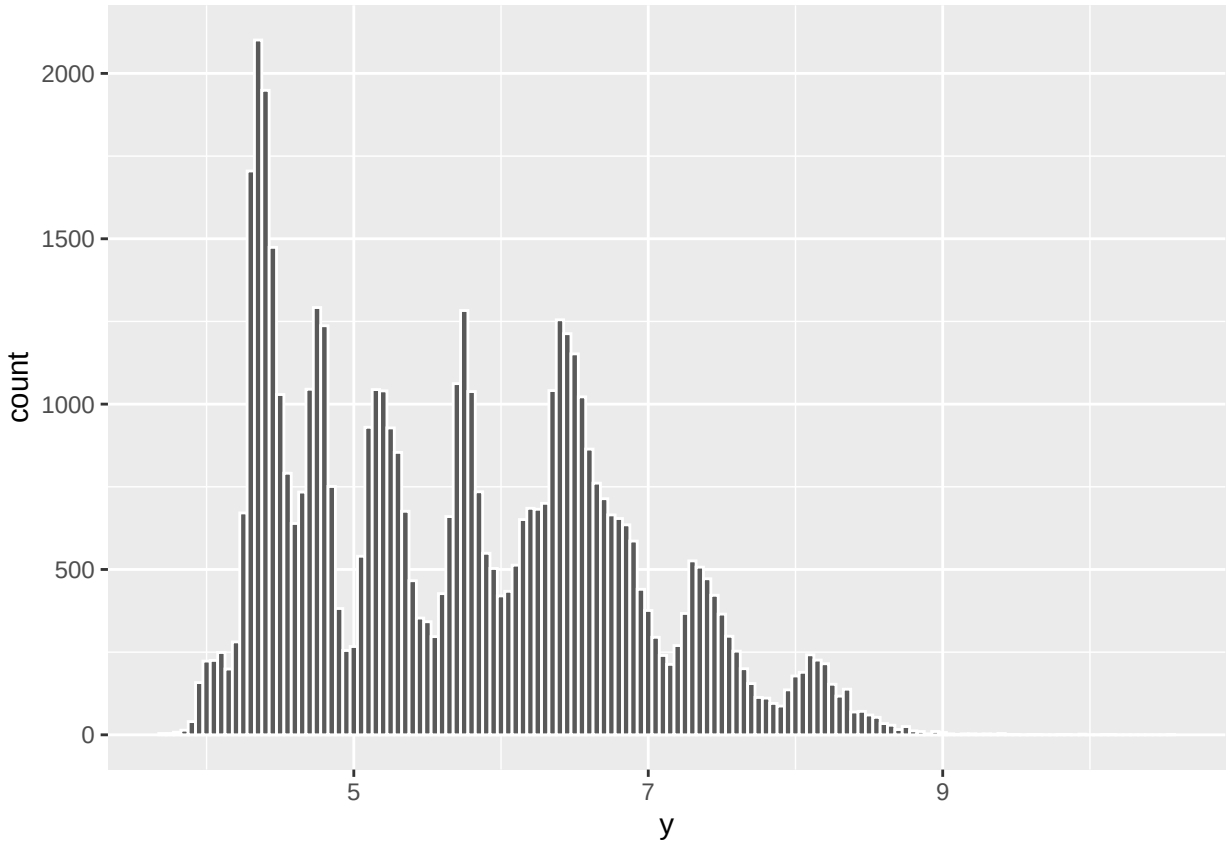
```
unusual <- diamonds %>%
  filter(y < 3 | y > 20) %>%
  select(price, x, y, z) %>%
  arrange(y)

unusual
```

```
## # 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
```

It is impossible for a diamond to have dimensions of 0, so we can assume these are measurement errors or perhaps a coded value which means something else. The other two observations have more reasonable `x` and

z values, so it seems reasonable to assume that these are measurement errors. Let's remove all the unusual observations by subsetting the data inside the ggplot() function, and see what the distribution looks like.

```
ggplot(diamonds[(diamonds$y < 20) & (diamonds$y > 0),]) +
  geom_histogram(aes(y), binwidth = 0.05, color = "white")
```



By removing unusual observations we can get a much more detailed look at how the sizes of diamonds are distributed. I would guess the multiple peaks are due to jewelers rounding up to meet a particular size with a prespecified price point, but we never would have seen this interesting pattern without removing the outliers.

Section 7.5.3. Two Continuous Variables

Overplotting becomes an issue when there are many points in the same area, and using transparency, while simple, doesn't always help. Often there are SO many points in a close area that it still doesn't do the volume of points justice. There are a few other options to help with this issue - one of them is using hexagonal bins and color scale to represent the density of points in an area. This is demonstrated below:

First, load the package "hexbin":

```
#install.packages("hexbin") (Commented out for smoother run)
library(hexbin)
```

```
## Warning: package 'hexbin' was built under R version 3.6.3
```

Then, load in and format the data to be used:

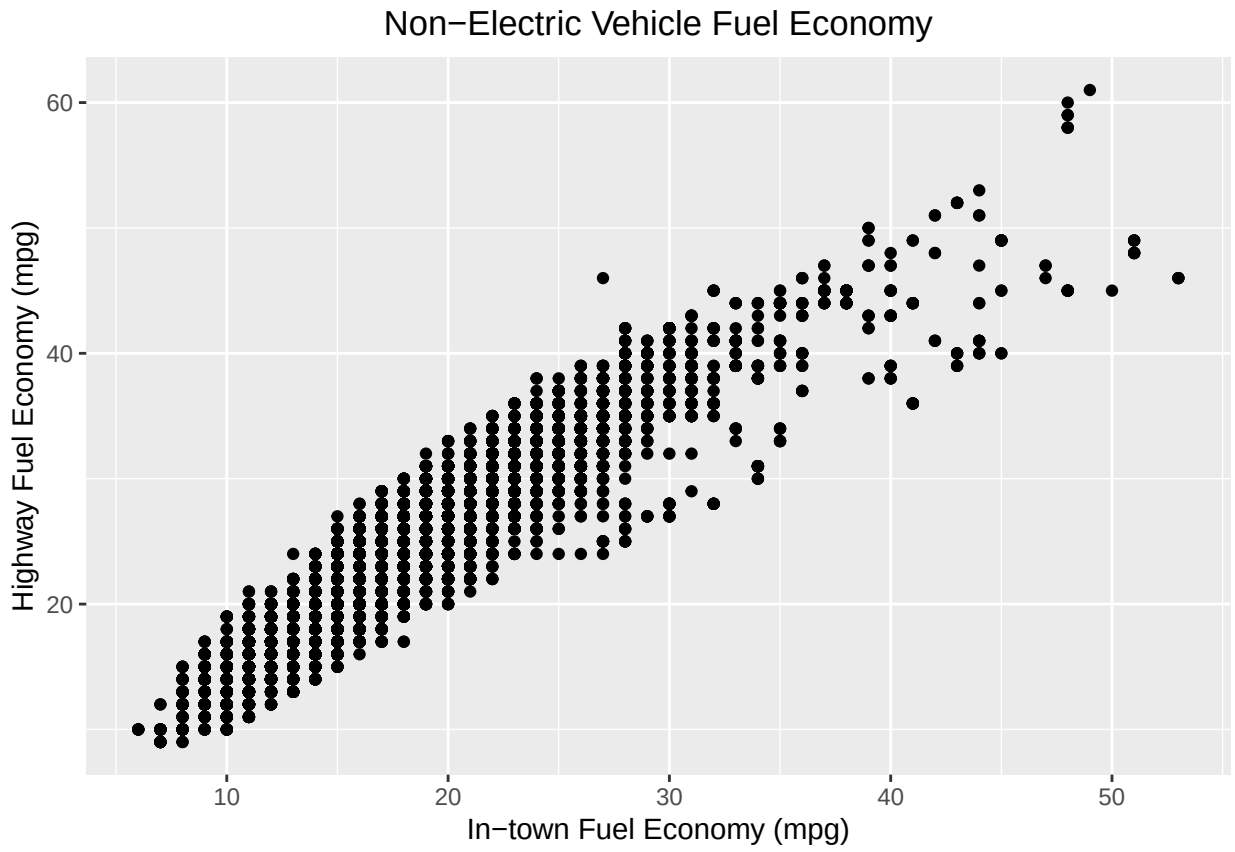
```
library(fueleconomy)
```

```
## Warning: package 'fueleconomy' was built under R version 3.6.3
```

```
vehicles <- vehicles
nonelectricvehicles <- subset(vehicles,vehicles$fuel!="Electricity")
```

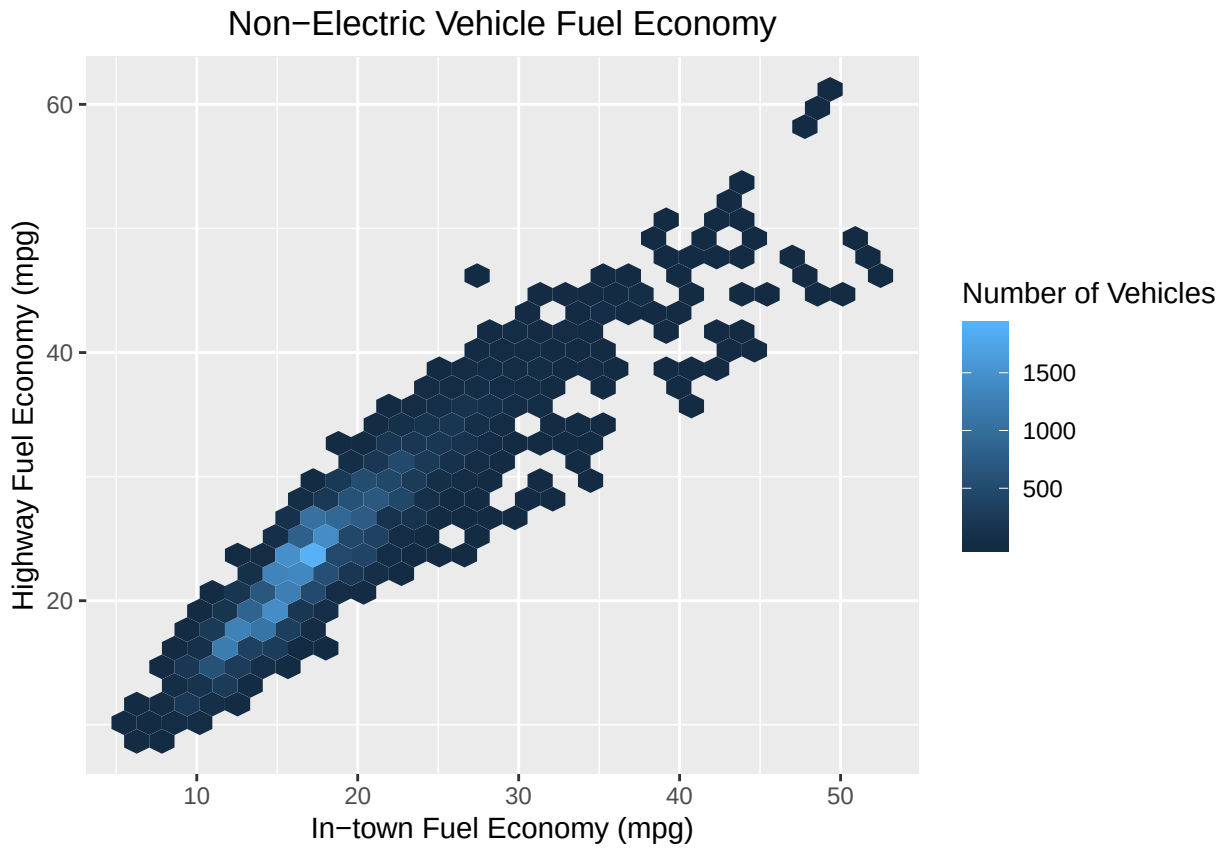
This dataset contains a massive number of vehicles with non-electric fuel sources. So when we attempt to plot in-town and highway fuel economies (both continuous variables) for all non-electric cars in the dataset, we get an overplotting issue:

```
library(ggplot2)
ggplot() +
  geom_point(mapping=aes(cty,hwy),data=nonelectricvehicles) +
  ggtitle("Non-Electric Vehicle Fuel Economy") +
  xlab("In-town Fuel Economy (mpg)") +
  ylab("Highway Fuel Economy (mpg)") +
  theme(plot.title = element_text(hjust = 0.5))
```



We can tell that these points are overplotted because there are over 33,000 vehicles plotted in the figure, and there appear to be FAR fewer points than that (meaning a lot of them are overlapping exactly). So we can implement the hexagonal bins method of dealing with overplotting:

```
ggplot(data=nonelectricvehicles) +
  geom_hex(mapping = aes(cty,hwy)) +
  ggtitle("Non-Electric Vehicle Fuel Economy") +
  xlab("In-town Fuel Economy (mpg)") +
  ylab("Highway Fuel Economy (mpg)") +
  scale_fill_continuous(name="Number of Vehicles") +
  theme(plot.title = element_text(hjust = 0.5))
```



This plot does a much better job of showing the distribution of fuel economies. The color scale allows the viewer to see that a much larger number of vehicles fall into a small middle range of highway and in-town fuel economies. This method will be most useful in situations similar to this one - where there are many many observations with similar values, which still need to be explored carefully, and with patterns which the creator of the graphic does not want to be obscured by the sheer volume of points.