Data Management and Visualization, Part II

Andie Creel

1 Picking up from Data Management and Visualization, Part I

We are going to pick up with the project we set up in Part I, and use the clean data to make six charts then polish one.

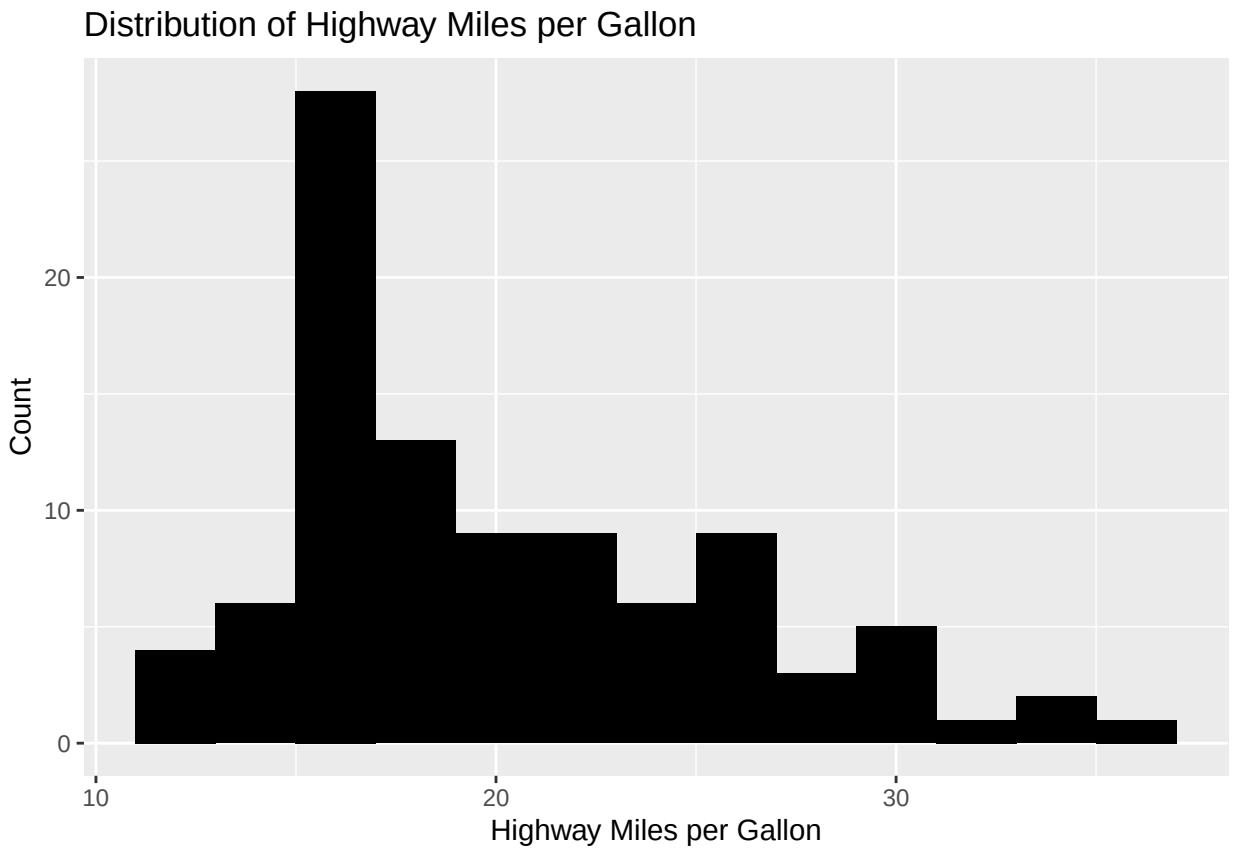
2 Data Visualization

- We'll do 6 charts together
  - Good data visualization requires choosing the right chart
  - Get a feel for it
- Then we'll polish one
  - see the types of things we can change
  - get some best practices for nice figures

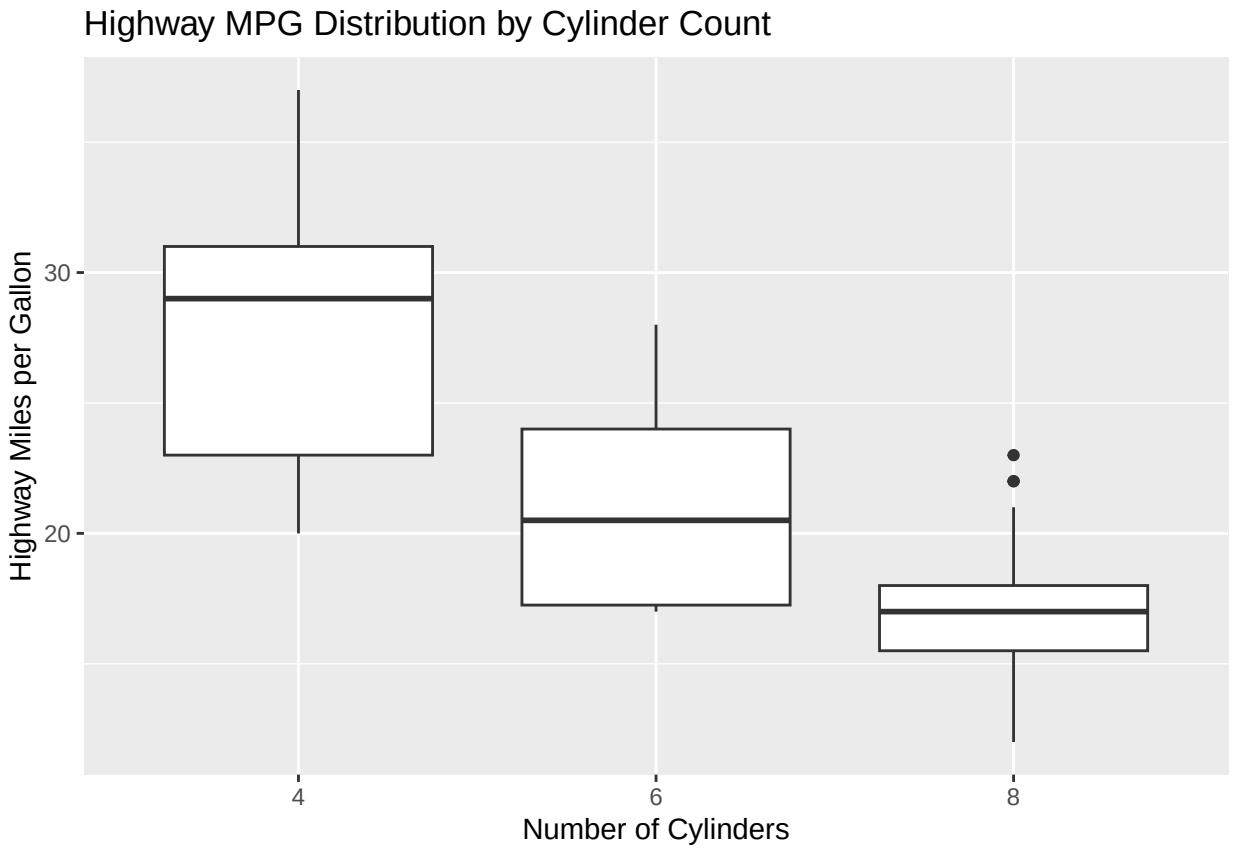
Create a `2_figures.R` script in the `scripts` folder

```
# Andie Creel / Goal: create figures / Started: Nov, 2024

# ------------------------------------------------------------------------
# load libraries
# ------------------------------------------------------------------------
rm(list = ls()) # clear work space at beginning of script
library(dplyr)
library(readr)
library(ggplot2)

# ------------------------------------------------------------------------
# read in clean data
# ------------------------------------------------------------------------
myData <- read_csv("data/clean_data/my_clean_data.csv")

# ------------------------------------------------------------------------
# Histogram -- the distribution of a single numerical variable: geom_histogram()
# ------------------------------------------------------------------------
#AC: run these one line at a time
#   - aes: aesthetics
#   - tells you the axis you'll be plotting
#       - x axis: hwy mpg

# dataframe is an input to the plot function
```

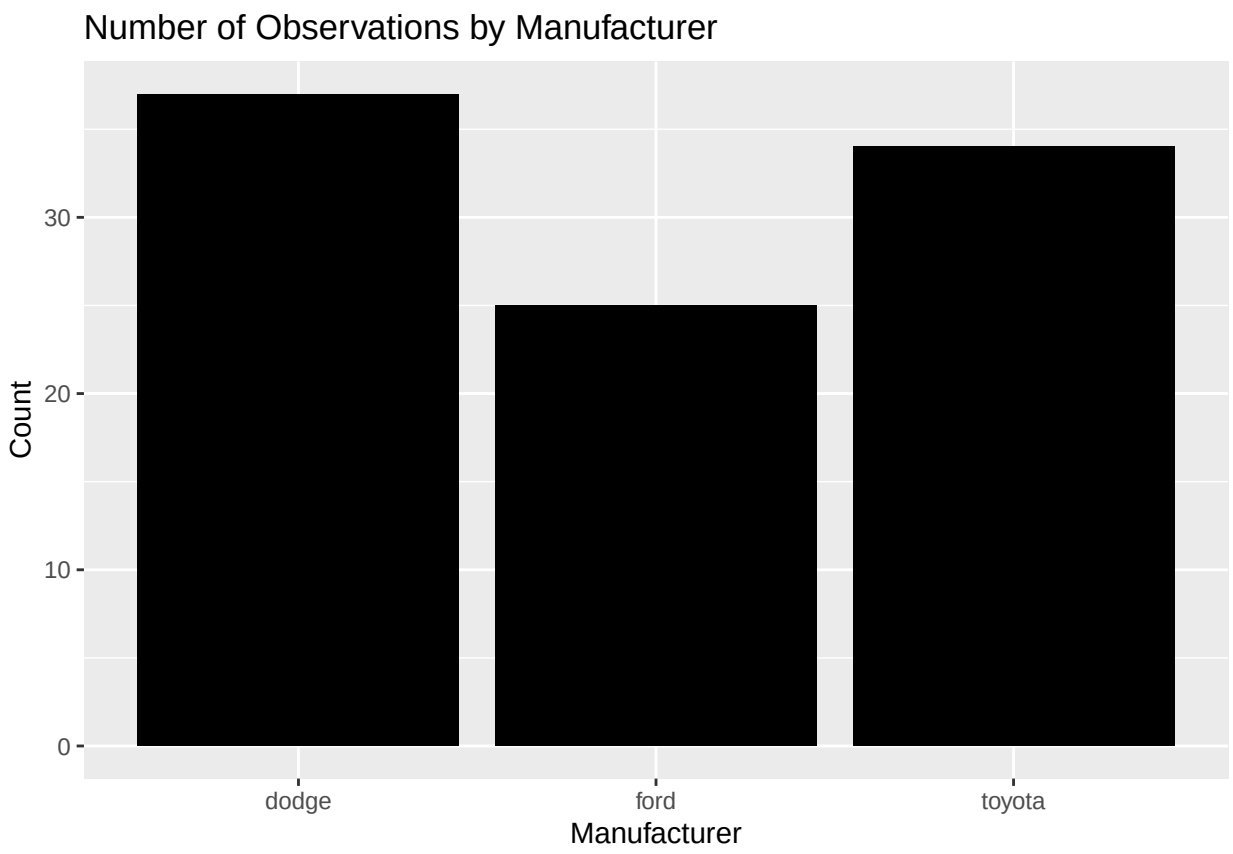
```r
ggplot(myData, aes(x = hwy)) +
  # geom_histogram() +
  geom_histogram(binwidth = 2, fill = "black") + # inputs change look
  labs(title = "Distribution of Highway Miles per Gallon",
       x = "Highway Miles per Gallon",
       y = "Count")
```



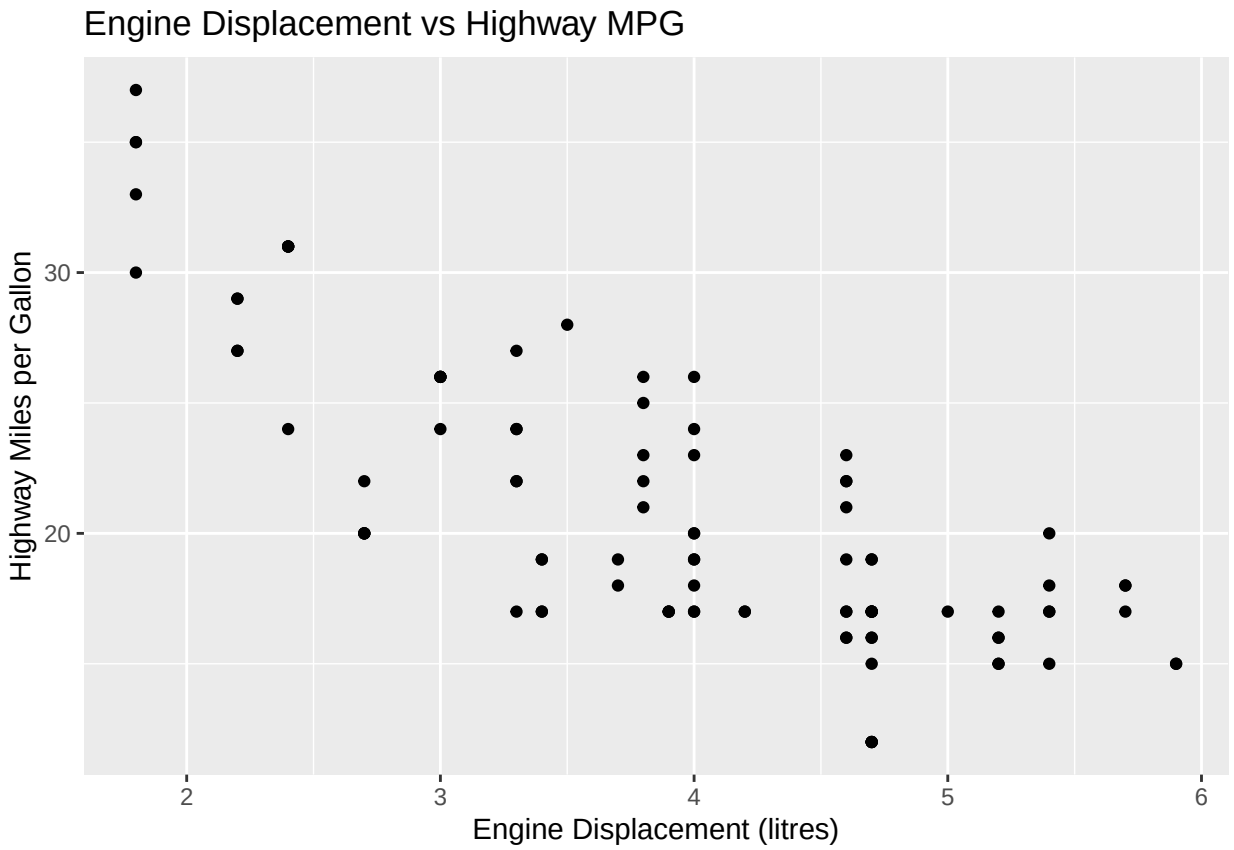

```r
# ------------------------------------------------------------------------
# Box plot -- continuous variable for different categories: geom_boxplot()
#   cyl: number of cylinders
# ------------------------------------------------------------------------
myData %>% # Pipe data
  mutate(cyl = as.factor(cyl)) %>%  # didn't read in as a factor
  ggplot(aes(x = cyl, y = hwy)) +  # make the plot
    geom_boxplot() +
    labs(title = "Highway MPG Distribution by Cylinder Count",
         x = "Number of Cylinders",
         y = "Highway Miles per Gallon")
```



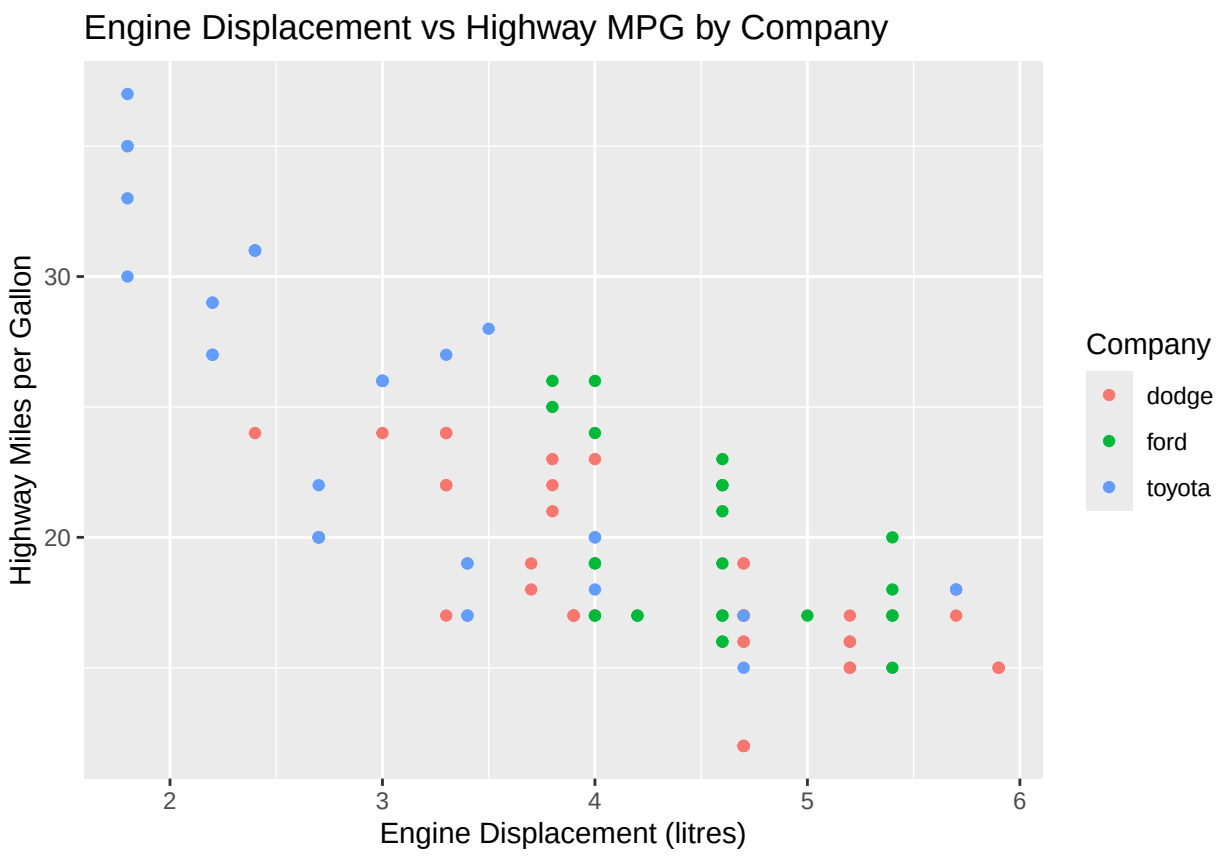

```
# ------------------------------------------------------------------------
# Bar chart -- count of observations in different categories: geom_bar()
# ------------------------------------------------------------------------
ggplot(myData, aes(x = manufacturer)) +
  geom_bar(fill = "black") +
  labs(title = "Number of Observations by Manufacturer",
       x = "Manufacturer",
       y = "Count")
```



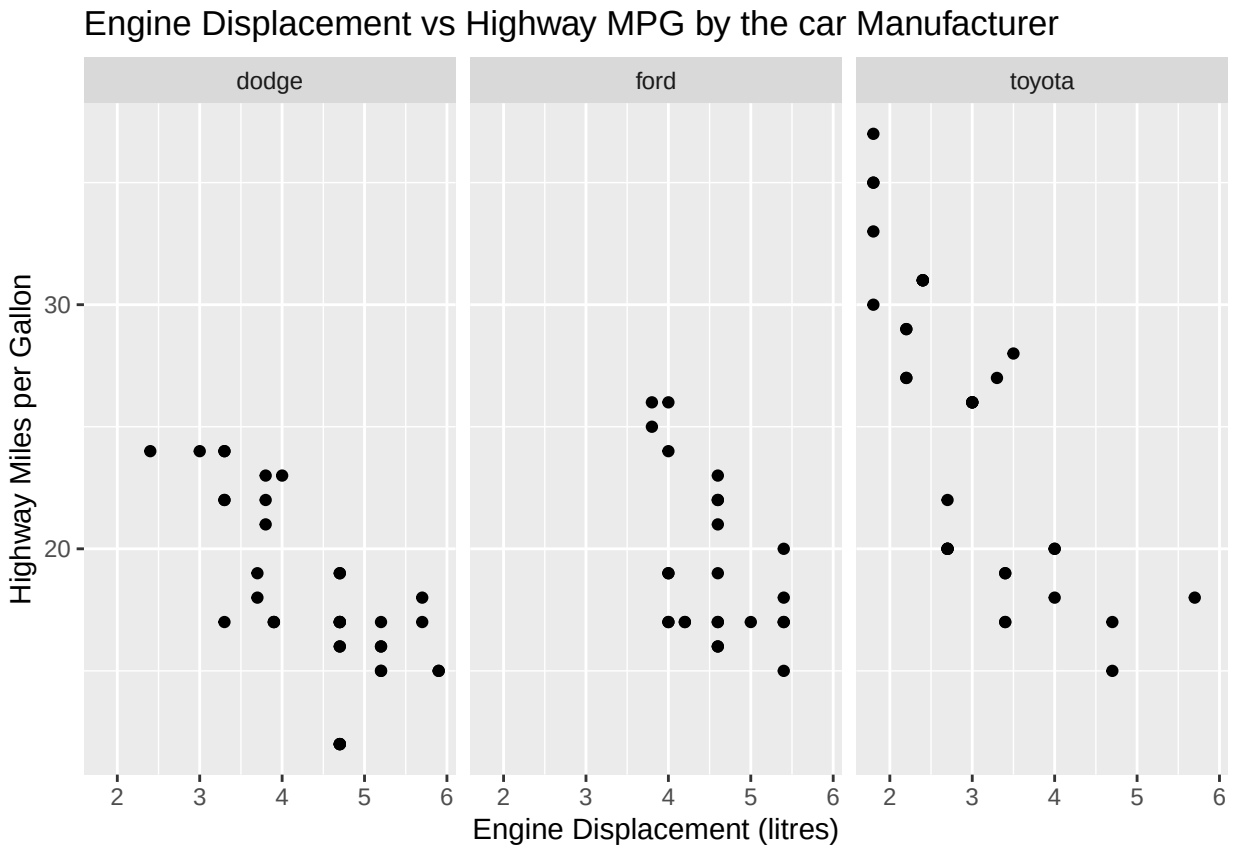

```
# ------------------------------------------------------------------------
# Scatter plot -- two continuous variables: geom_point()
#   hwy: highway miles per gallon
#   displ: engine displacement which is approx. engine size
# ------------------------------------------------------------------------


ggplot(myData, aes(x = displ, y = hwy)) +
  geom_point() +
  labs(title = "Engine Displacement vs Highway MPG",
       x = "Engine Displacement (litres)",
       y = "Highway Miles per Gallon")
```



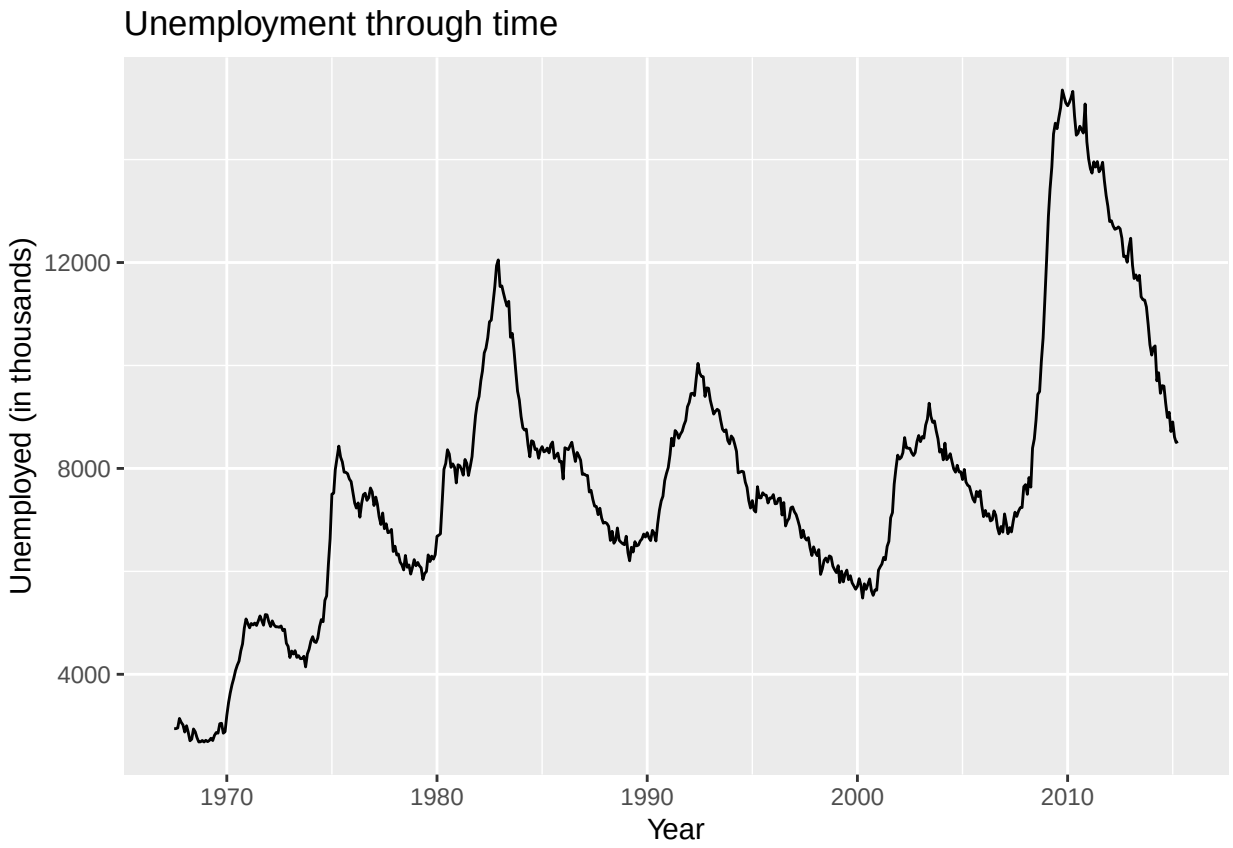

```
# Third color axis: groups that you want shown in different colors.
ggplot(myData, aes(x = displ, y = hwy, color = manufacturer)) +
  geom_point() +
  labs(title = "Engine Displacement vs Highway MPG by Company",
       x = "Engine Displacement (litres)",
       y = "Highway Miles per Gallon",
       color = "Company")
```




```
# ------------------------------------------------------------------------
# Multiple plots -- same graph for different categories: facet_wrap()
#     - same information last chart with the color
# ------------------------------------------------------------------------

# can store a graph as an object
myFacet <-  ggplot(myData, aes(x = displ, y = hwy)) +
  geom_point() + # type of graph you wanna see multiple times
  facet_wrap(~manufacturer) + # seperated by what?
  labs(title = "Engine Displacement vs Highway MPG by the car Manufacturer",
       x = "Engine Displacement (litres)",
       y = "Highway Miles per Gallon")

# display the graph
myFacet
```




```
# ------------------------------------------------------------------------
# Line chart -- How a numeric variable changed through time: geom_line()
# ------------------------------------------------------------------------

# Use the economics data set (ggplot2 package)
data(economics)

myLine <- economics %>%
  ggplot(aes(x = date, y = unemploy)) +
  geom_line() +
    labs(title = "Unemployment through time",
       x = "Year",
       y = "Unemployed (in thousands)")

# Print chart
myLine
```




```
# ------------------------------------------------------------------------
# Save a chart
# ------------------------------------------------------------------------

# Line graph
ggsave("results/line_chart.png", # file path and name
       plot = myLine, # plot you want to save
       width = 10, # width of plot (inches)
       height = 8, # height of plot (inches)
       dpi = 300) # dots per square inch

# Facet
ggsave("results/facet.png",
       plot = myFacet,
       width = 10,
       height = 4, # specialize the height for the figure
       dpi = 300)
```

Open up one of the plots in our results folder.

These are okay, but not presentation ready.

3 A few of the best practices for data visualization

Why is data visualization so important?

- You put hundreds of hours into data collection and analysis
- Most people are just going to look at your figures
- If bad, they won't read your paper
- If good, they'll know your key results from a glance
- Figures make or break whether people know what you found

Let's take one of our charts and make it polished following a few basic best practices. Data visualization is an art all on it's own, and it's worth taking advantage of other resources in data visualization. However, these best practices will get you pretty far.

Best Practices

1) Choose the right graph: let your research question guide this
2) Write clear labels and titles
3) Simple as possible while not becoming reductive
4) All parts of graph are legible
5) Colors are not horrible and also work for people who are color blind
6) Use ggplot/code for as much as possible (saves so much time in long run if you get new data, change your mind about something, etc)

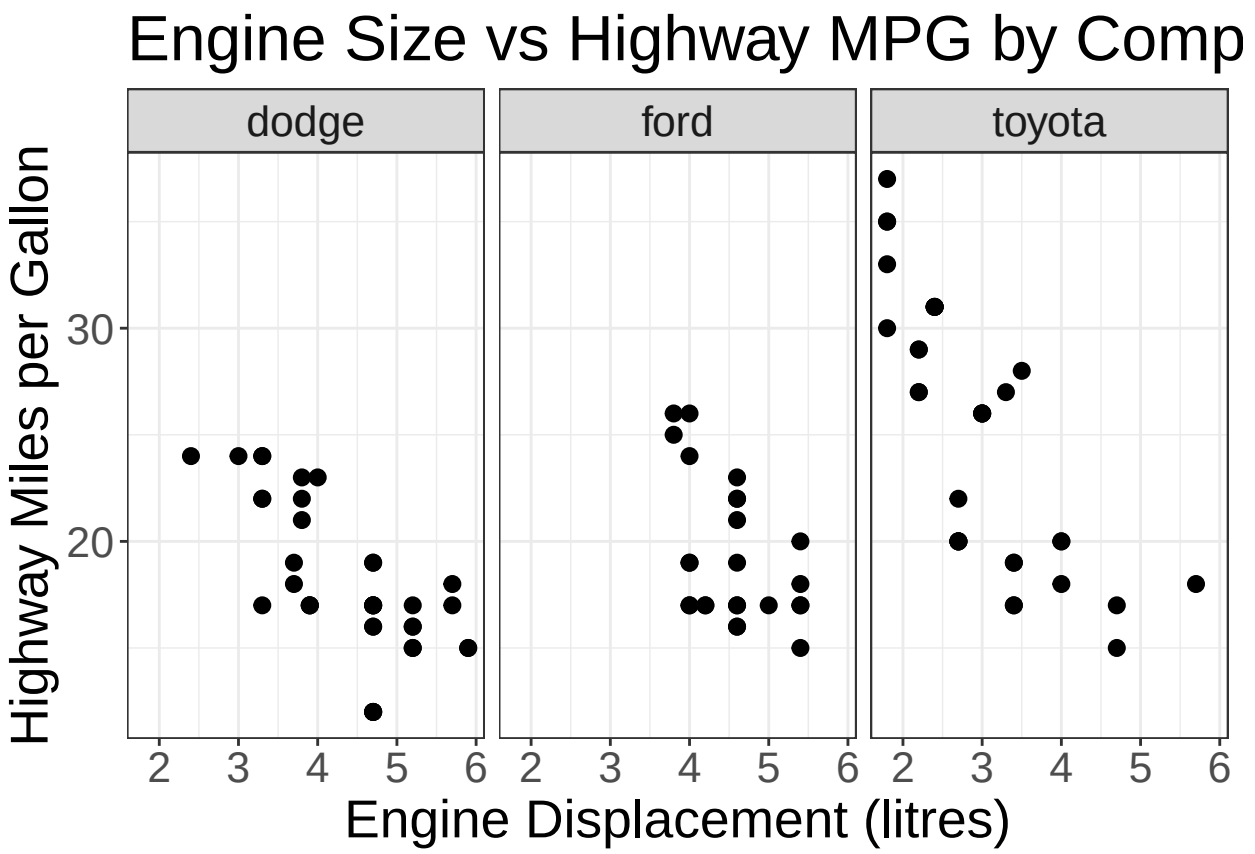
These six rules will get your pretty far.

Let's work with a plot we already made in the `2_results.R` file.

4 Facet chart

Consider if our research questions gas mileage relative to engine size, compared across companies.

```
# ------------------------------------------------------------------------
# Work with out facet plot
# ------------------------------------------------------------------------

myFacet <- ggplot(myData, aes(x = displ, y = hwy)) +
  geom_point(size = 2.5) + # make points legible
  facet_wrap(~manufacturer) +
  labs(title = "Engine Size vs Highway MPG by Company", # clean up our title
       x = "Engine Displacement (litres)",
       y = "Highway Miles per Gallon") +
  # theme_minimal() +
  theme_bw() +                                          # choose simple theme
  theme(text = element_text(size=20))                   # make it legible

# print chart
myFacet
```




```
ggsave("results/facet.png",
       plot = myFacet,
       width = 10,
       height = 4,
       dpi = 300)
```

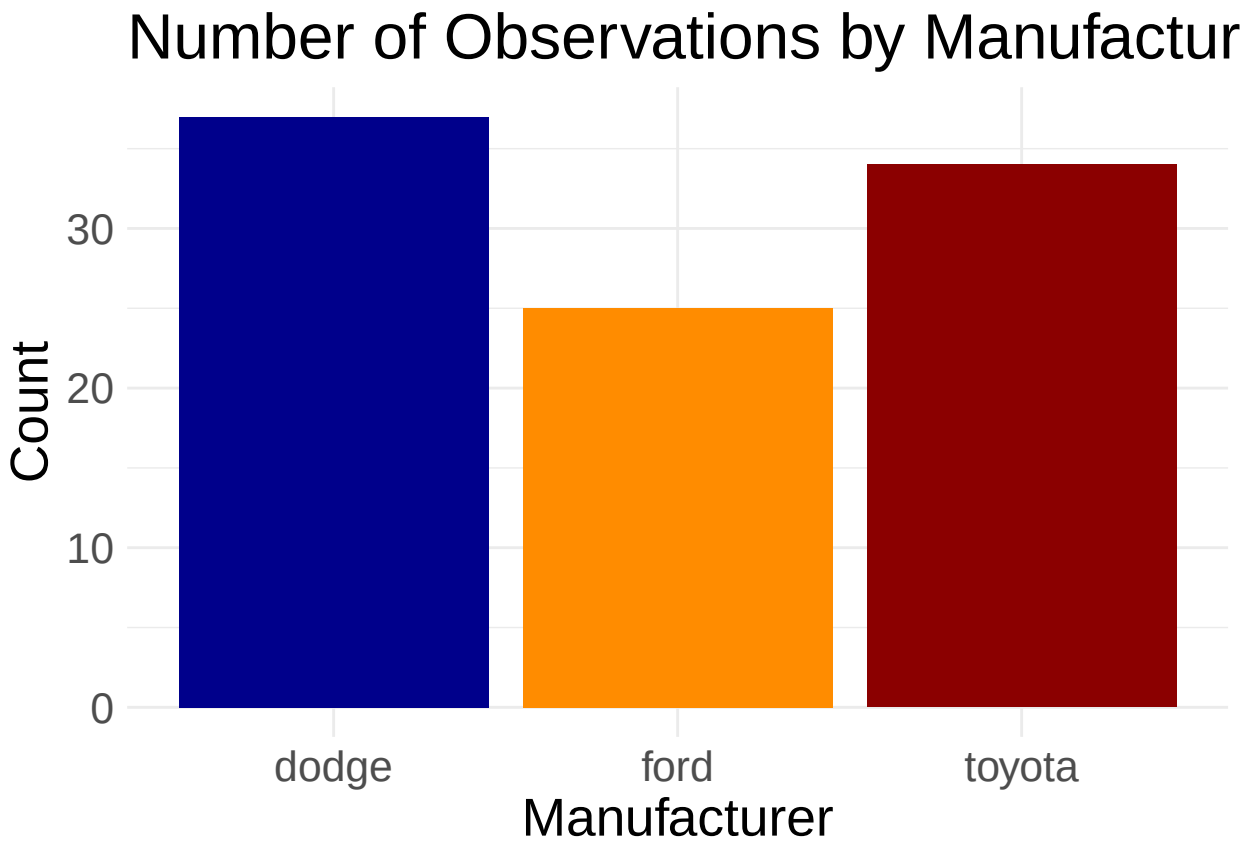
5 Bar chart (colors)

My main advise is to keep your colors as simple as possible.

Consider our bar chart.

```
# ------------------------------------------------------------------------
# Working with our old bar chart
# ------------------------------------------------------------------------

# colors I use: darkred, darkblue, darkgreen, darkorange a lot.

myBar <-  ggplot(myData, aes(x = manufacturer)) +
  geom_bar( fill = c("darkblue", "darkorange", "darkred")) + # change colors
  labs(title = "Number of Observations by Manufacturer",
       x = "Manufacturer",
       y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  # theme_bw() +
```

```
  theme_minimal() +                                    # make it simple
  theme(text = element_text(size=20))                  # make it legible

# print chart
myBar
```




```
# show the difference between png and jpeg
ggsave("results/bar.jpeg",
       plot = myBar,
       width = 10,
       height = 8,
       dpi = 300)

ggsave("results/bar.png",
       plot = myBar,
       width = 10,
       height = 8,
       dpi = 300)
```

6 Wrap up

There are million ways to tweak charts, and you'll probably spend loads of time tweaking your figures in R. However, once you get over the learning curve, making them in R will be so much faster than if you're editing them in Power Point or Excel.

When it's all code, you can regenerate your figures extremely quick. This is so valuable when you get more data, or in the worst case where you realize you made a mistake data cleaning and need to fix that then regenerate all your data.