Descriptive Visualizations

Irfan Kanat

July 3, 2017

In this learning activity we will learn about creating descriptive visualizations of data sets to see how variables are distributed, or how they relate to each other.

R base comes with adequate plotting functionality and there are lots of small specialized packages to create different plots. In this activity we will focus on ggplot2 package. The reason for this selection is because ggplot2 is a very versatile package that enables a broad range of plots to be created. I will focus mostly on qplot() function, and discuss ggplot structure only briefly.

I will first demonstrate most common types of plots (histograms, scatter plots, and line plots) with qplot and move on to how the same results can be achieved in ggplot.

Here is a little exercise for you (you should know this from Packages and Package Management learning activity from previous module). Install and activate ggplot2 package (solution is at the bottom of this document if you can not figure out how).

Introducing the Datasets

We will use several datasets in this learning activity. You should remember the bagsOfOranges, bagsOfApples, and mtcars data sets from the previous learning activity Descriptive Statistics.

I will create an additional dataset to to use with line plots, don't worry about the code below at this point.

```
set.seed(2017) # Set seed for random number generator for replicability
profits<-data.frame(company=rep(LETTERS[1:3], each=5), year=rep(2013:2017, 3),
                    profit=c((seq(from=102, to=110, by=2)+rnorm(5, mean = 0, sd=2)),
                             (rep(50, 5)+rnorm(5, mean = 0, sd=1)),
                             (seq(from=70, to=90, by=5)+rnorm(5, mean = 0, sd=2))))
# Export data into a csv file.
write.csv(profits, file="data/profits.csv")
```

Let us view the new dataset.

```
View(profits)
```

```
## Warning in View(profits): X cannot set locale modifiers
```

This (profits) simulated dataset shows the profits for three companies (A,B,C) over a 5 year period between 2013 to 2017.

Plotting with qplot()

Now we can get to the fun part. qplot simplifies the ggplot functionality by automating most common tasks. We will use qplot for most common plots.

```
# Load the ggplot package
library(ggplot2)
# Review function syntax
?qplot
```

qplot is a simplified interface for ggplot functionality. It tries to figure out what you may need by the data and parameters you provide. It allows you to skip a lot of the details, by filling in the blanks you leave with reasonable defaults. If you ever want to change the details though, you will inevitably switch back to ggplot.

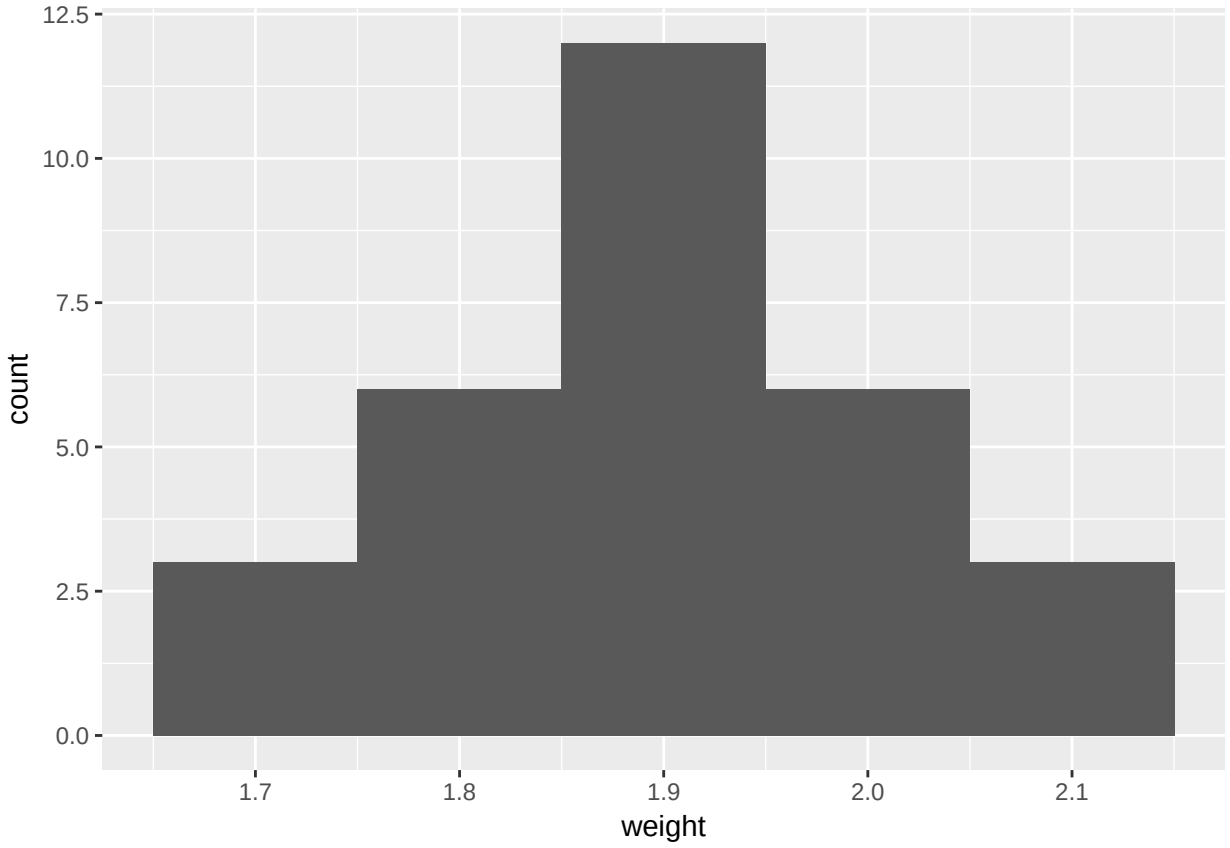
Histogram

A histogram is useful if you are interested in how a continuous variable is distributed. For continuous variables, histogram will show how many observations (frequency) fall into a range (bin). An example would be number of orange bags that weigh between 2 and 2.1 pounds.

Let's say, we are interested in how the weight of bags of oranges vary. To answer this question we can use a histogram.

Let us plot this example.

```
qplot(x = weight, data = bagsOfOranges, geom = "histogram", binwidth = .1)
```



Let us go over the function call.

```
qplot(x = weight, data = bagsOfOranges, geom = "histogram", binwidth = .1)
```

Here I call the qplot() function with various parameters:

1. x = *weight*: I name the variable I want to be plotted on x axis.
2. data = bagsOfOranges: I point to the data source to look for the variable.
3. geom = "histogram": I specify the type of plot I want (more on geom later).
4. binwidth: I specify how wide of a range I want for each bin.

Barcharts

If the variable is categorical, a bar chart can be used to show frequencies of a categorical variable's levels. Such as number of cars with automatic or manual transmission. A bar chart is useful beyond frequencies as we will discuss later.

Let us say we want to know, the number of cars in each gear category. How many cars with 3 gears, how many with 4 and so on. . .

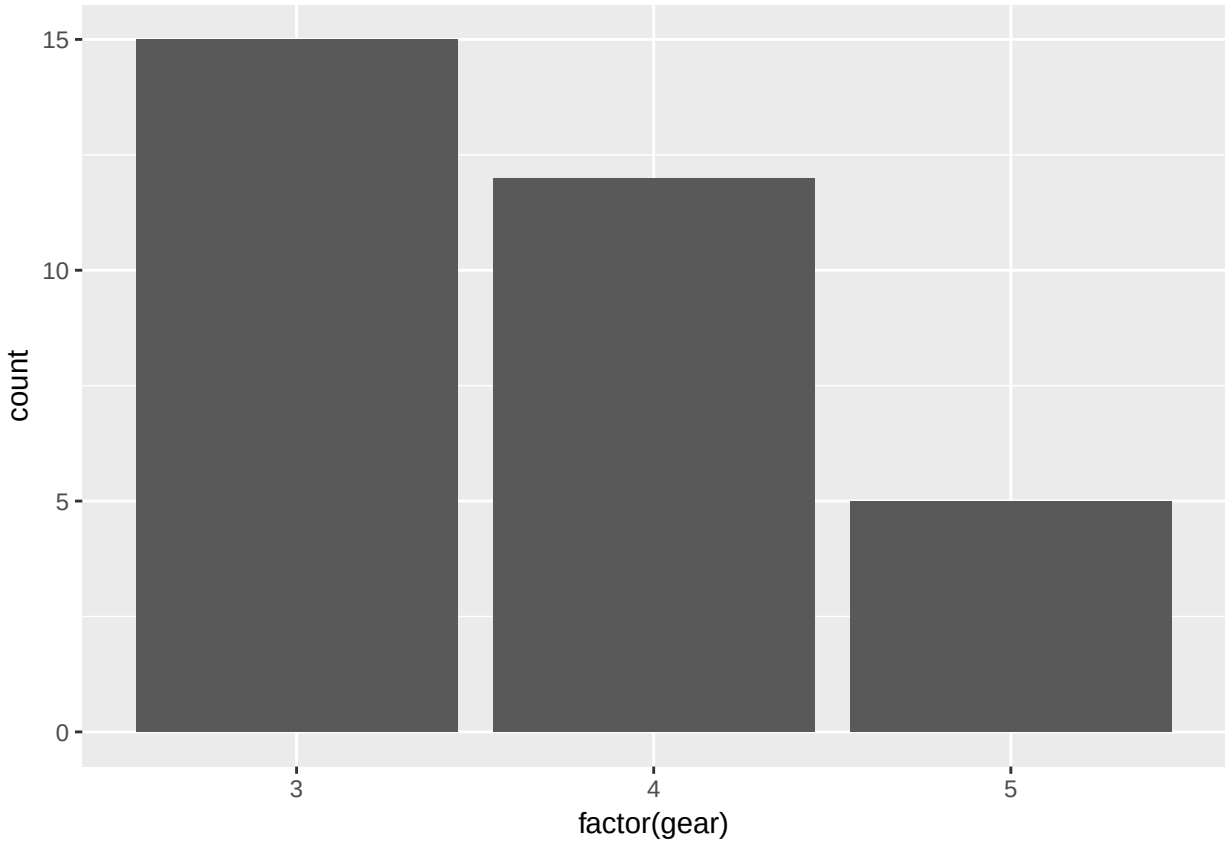
Let remember the previous learning activity. Remember, we tabulated the number of gears. So we already know the quantitative answer.

```
table(mtcars$gear)
```

```
##
##  3  4  5
## 15 12  5
```

We can visualize this information with a bar chart.

```
# Let us report the number of cars with differing number of front gears
qplot(x = factor(gear), data = mtcars, geom = "bar") # factor() for categorical
```



The bar chart shows we have 15 cars with 3 gears, 12 with 4 gears, and 5 cars with 5 gears.

Let us breakdown the command used to create the plot.

`qplot(factor(gear), data = mtcars, geom = "bar")`

I call the qplot() function with qplot()

1. x = factor(gear): I specify the variable I want plotted (*gear*) but I make R know this variable is categorical with factor() function.
2. data = mtcars: I tell qplot where to find *gear*, by declaring data source as mtcars
3. geom = "bar": I tell qplot what type of plot I desire *geom="bar"* basically geometry is bar chart

qplot assumes I want frequencies by default.

Next step after looking at a variable by itself is to look at a variable in relation to other variables. You can use the bar chart to demonstrate the frequency of one categorical variable, broken down by another categorical variable. Thus establishing relationship between two categorical variables.

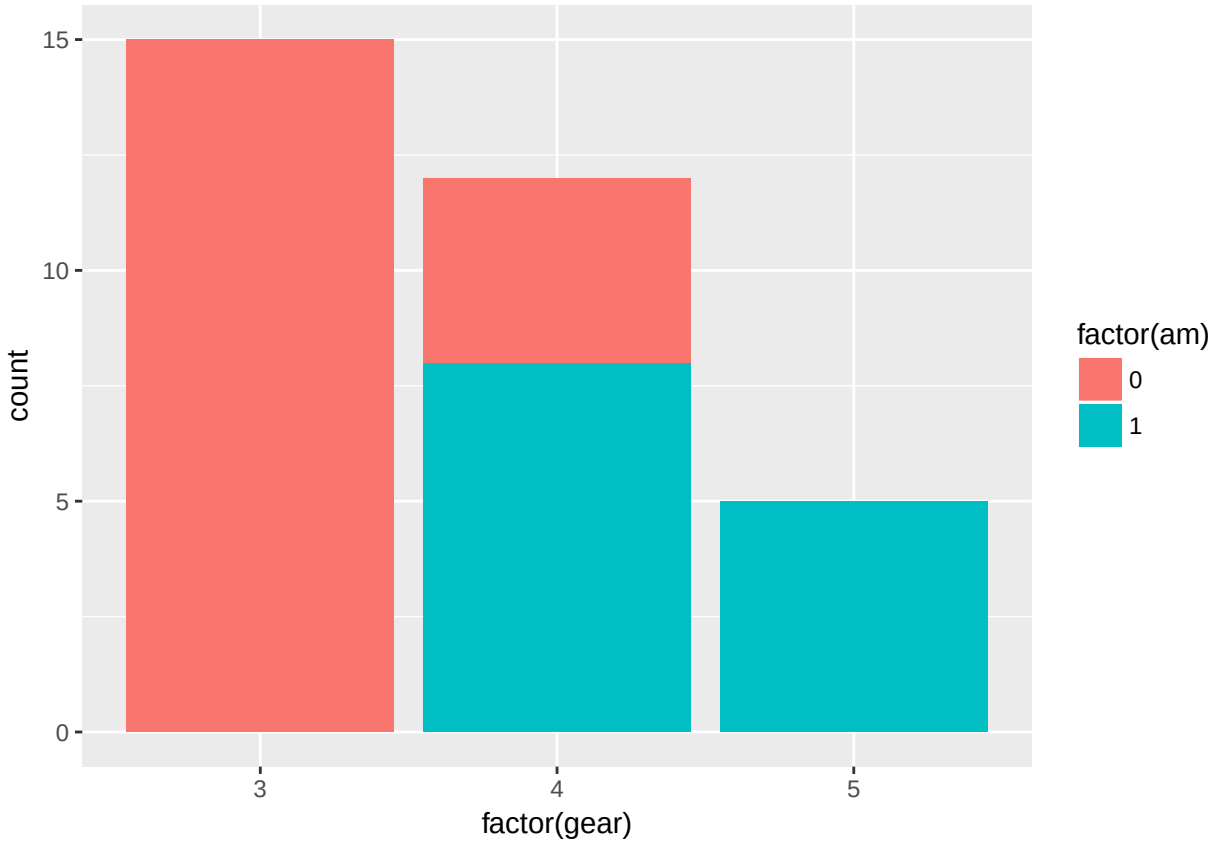
Let us start with a tabulation to see what raw data says about the relationship between transmission type (Automatic = 0) and number of gears.

```
table(mtcars[, c("am", "gear")])
```

```
##    gear
## am   3  4  5
##   0 15  4  0
##   1  0  8  5
```

Same data can be visualized with a bar chart as follows:

```
qplot(x = factor(gear), data = mtcars,  fill = factor(am), geom = "bar") # used factor to declare categ
```



This visualization demonstrates that certain number of gears are only observed in automatic or manual transmissions (remember the data is from 1974). Giving us an insight, basically gear would be a good predictor for transmission type.

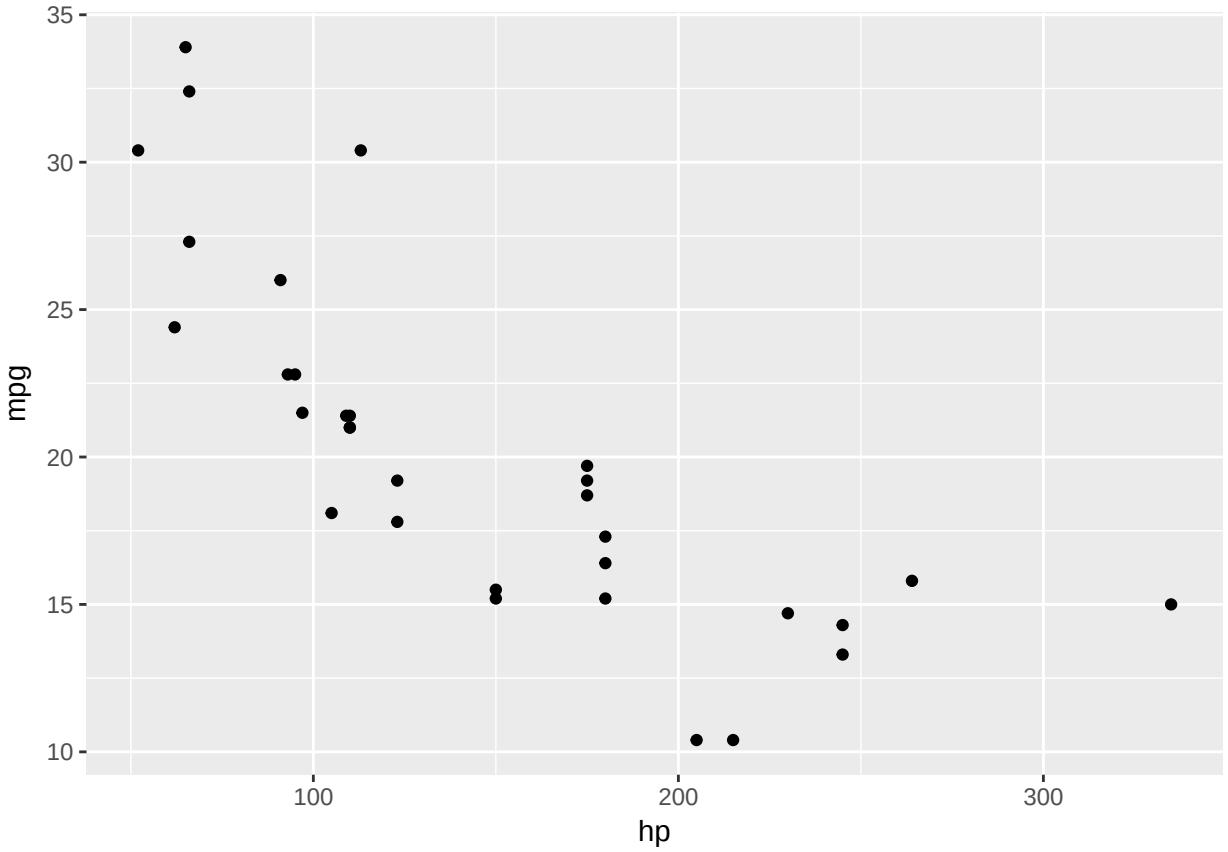
Here I added an additional parameter, specifying which variable to use in filling the bar. fill=factor(am) makes qplot use am (transmission type) to change the color of fillings of the bars.

Scatter Plots

If you are interested in the relationship between two continuous variables, you can use scatter plots. A scatter plot is like a correlation, it shows you how the two variables act together.

Let us say you are interested how the strength of the engine (hp) is related to fuel consumption. You can use a scatter plot for this.

```
qplot(hp, mpg, data = mtcars)
```



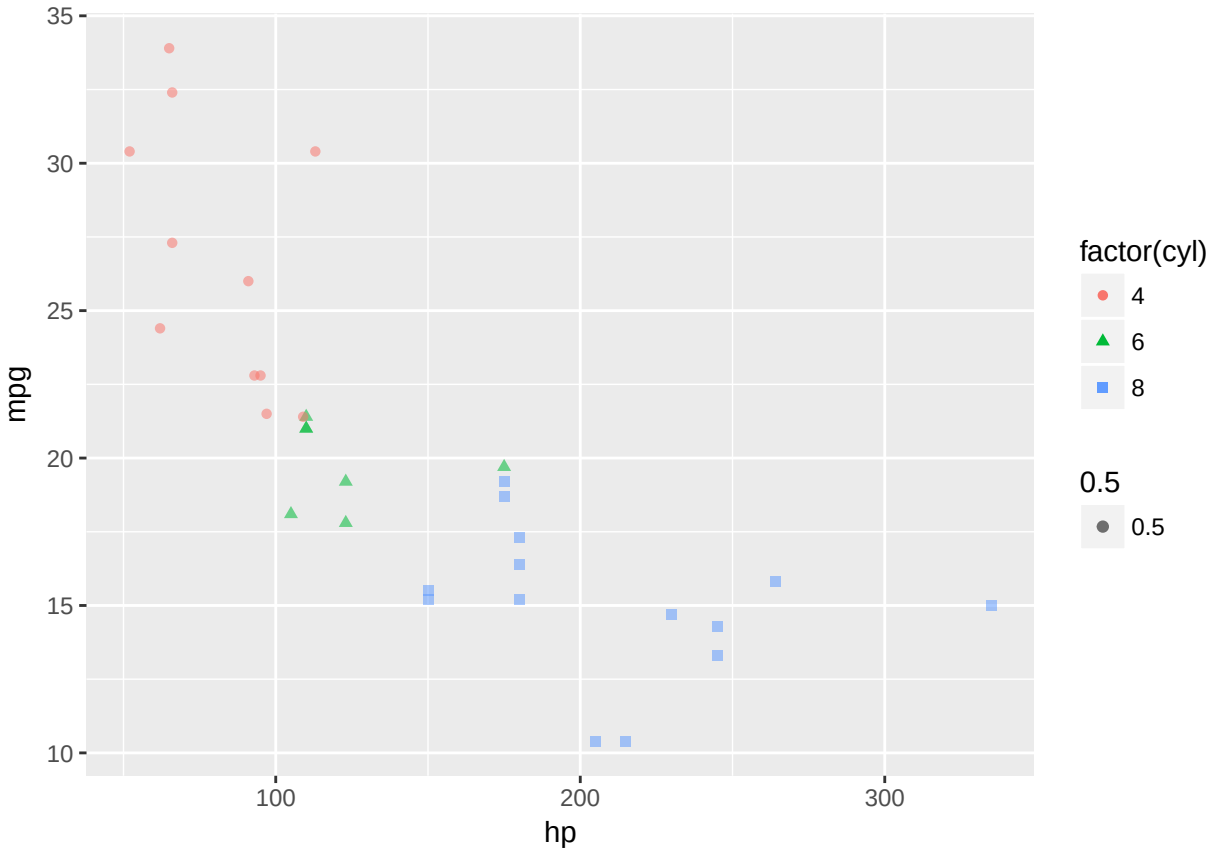
I did not specify a geom this time, qplot assumes "point" by default.

The plot shows that there is an inverse relationship between horse power and fuel efficiency. Basically stronger cars guzzle up more gas.

Again, we can up the ante by introducing an additional variable. Since this is a two dimensional plot, we need to use something other than x or y axes to map to the additional variable.

Let us say you are also interested in the effect of number of cylinders on these two (hp and mpg) variables. You can use colors or symbols to overlay additional variables. So the variable does not show just two, but three variables. Let us color the dots by the number of cylinders.

```
qplot(hp, mpg, data = mtcars, color = factor(cyl), shape = factor(cyl), alpha = .5)
```



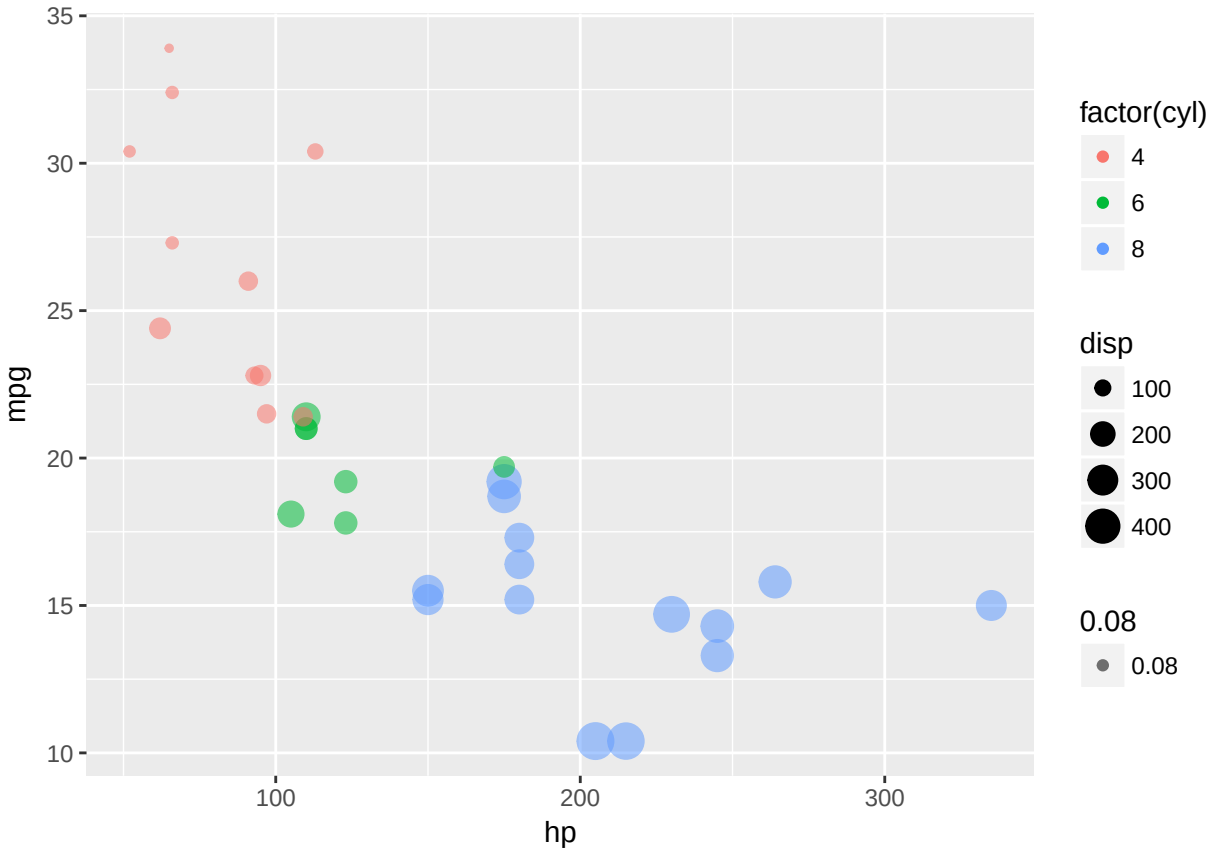
The additional parameter of color = factor(cyl) should be self explanatory. I instruct qplot to color the dots by number of cylinders.

Similarly, shape = factor(cyl) means use different shapes for different number of cylinders (this is useful if your report will be viewed in black and white).

alpha = .5 is the level of transparency of dots. If the dots are overlaid, it is hard to distinguish many opaque dots, so transparency will allow you to get darker color where there are many dots converging.

This is quickly getting out of hand, but let us say you want to visualize FOUR variables. We can make the size of dots dependent on a continuous variable (displacement - volume of engine).

```
qplot(hp, mpg, data = mtcars, color = factor(cyl), size = disp, alpha = .08)
```



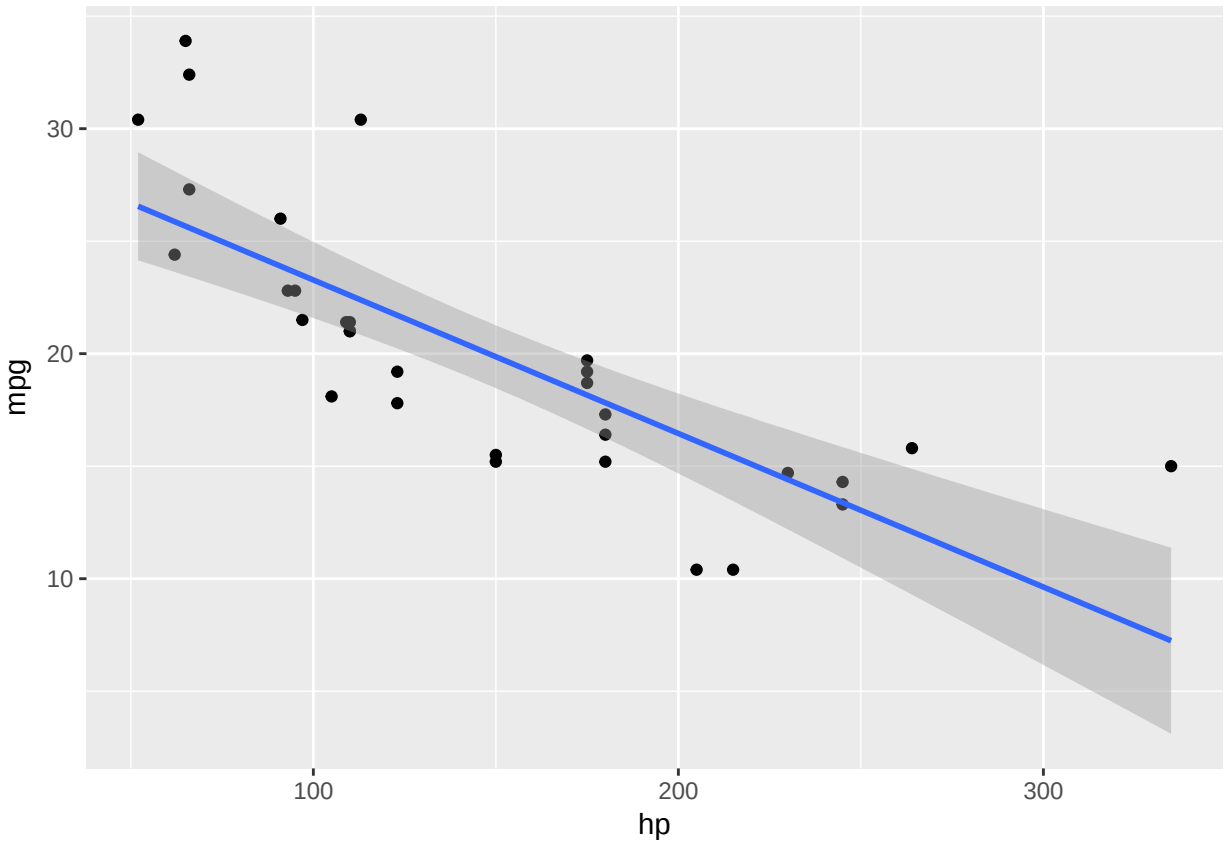
Let us review what we have learned so far. We can map variables to the axes of the plot, we can map them to the shape, color, or size of the dots. Using these different dimensions you can create informative visualizations that communicate what you are trying to explain. A word of caution: Don't over do it. Human mind can only process so much data at once. More variables do not necessarily mean a better visualization. Keep it simple. Use a visualization to communicate a relation you want to emphasize, not to explain everything about the dataset in one graph. Most of the time it is better to have a few simple visualizations than to have a very complex one.

Another way to express the relation between two variables is to fit a regression line. Basically the slope of the line shows you how much one variable changes when you change the other variable by one unit.

This is where things start to get a bit ggplotty. You can overlay ggplot layers on top of qplot outputs (you will understand what this means better in a second).

```
qplot(hp, mpg, data = mtcars) +
  geom_smooth(method = lm, sd = F)
```

```
## Warning: Ignoring unknown parameters: sd
```



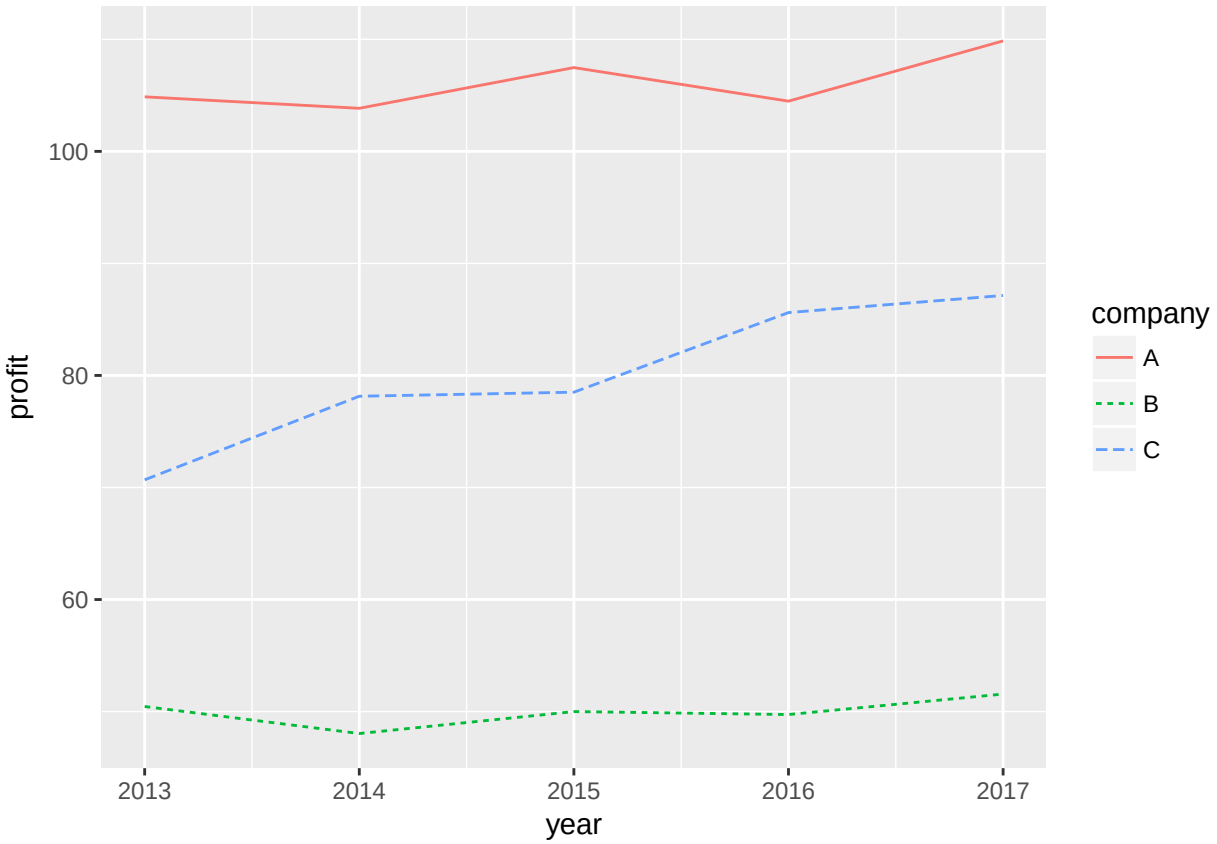
Line Plots

If you have a variable that changes over time, a line plot is a good way to visualize the trajectory over time.

We will use the profits dataset. profits shows profits for three companies (A, B, C) over a five year period between 2013 and 2017.

Let us say, we want to see how profits of these companies change over the years.

```
qplot(data = profits, x = year, y = profit, color = company, linetype = company, geom = "line")
```



We can see that Company B had a pretty flat trajectory, while the other companies have been increasing their profits. It seems like the fastest growth has been in company C.

After the previous examples, the syntax should be familiar by now. Let us review:

```
qplot(data = profits, x = year, y = profit, color = company, linetype = company, geom = "line")
```

I call qplot with 6 parameters.

1. data=profits: Where to find the variables.
2. x=year: year variable should be mapped to x axis.
3. y=profit: profit variable should be mapped to y axis.
4. color=company: the color of the lines should be determined by the company variable.
5. linetype=company: the type of lines (dashed, dotted, solid) should be determined by the company variable. This is useful if your report will be viewed in black and white.
6. geom="line": This is where I specify I want a line plot.

Plotting with ggplot

qplot is a wrapper for ggplot. It eases use of ggplot for most common use cases. ggplot can do everything qplot can do. Unfortunately, the opposite is not always true. qplot is a simplified interface and some use cases dictate use of ggplot.

I will get you started with ggplot and teach you just enough to be dangerous, but after that you are on your own. You need to figure the rest out by yourself (perhaps using the sources from Resources Learning Activity in Module 1 and the ones at the end of this learning activity).

Some say ggplot is a separate language in itself. I would say it is a dialect of R, with its own specific jargon. Let us learn some basic words in this new dialect.

1. aes: Aesthetic mapping, how variables will be plotted. The first aesthetic mapping in ggplot() will be used as default for additional layers.
   - x: What will be on x axis.
   - y: What will be on y axis.
   - col: Which variable will determine colors.
   - size: Which variable will determine size.
   - shape: which variable will determine shape.
   - lty or linetype: which variable will determine the line type.
2. geom: Geometric object, specifies the type of plot. These often form additional layers. Various options exist:
   - point: for scatterplot (geom_point)
   - smooth for fitted curves (geom_smooth)
   - boxplot box and whisker plots
   - path and line for line plots
   - histogram and freqpoly frequency plots for continuous variables
   - bar for bar charts of categorical variables
3. Others such as titles, labels, legends, transformations...

ggplot works with layers. You can initialize a plot and keep adding layers to it using "+" sign.

This may all be very confusing when you read about them. It is much easier to understand the layers when seen in action.

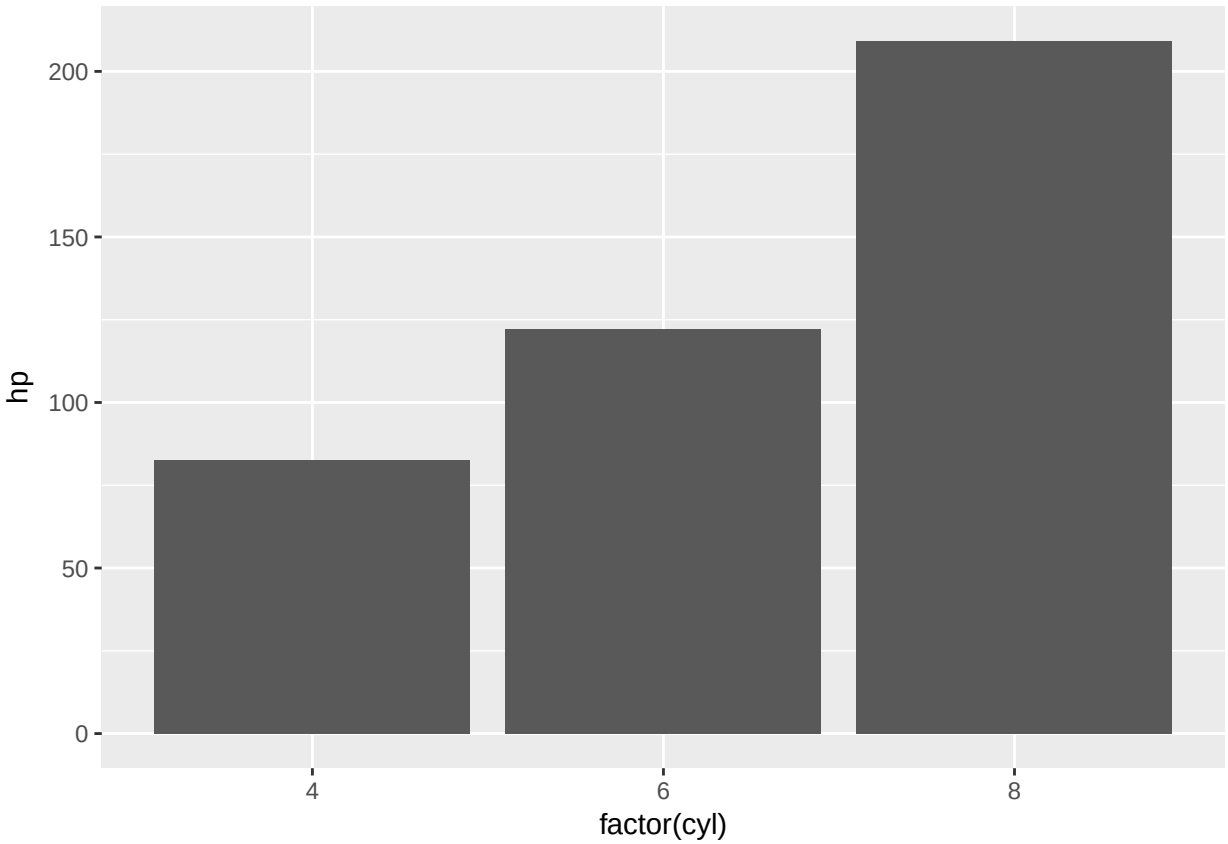
More Bar Charts

Now it is time to go over an example that is a bit more advanced. I will introduce two new concepts: (1) other uses of bar charts, and (2) use of ggplot.

A histogram is specific to continuous variables and frequencies, a bar chart is more versatile. It can be used for many more purposes, such as reporting average horse power (continuous) for different number of cylinders.

Let us plot that example of horse power over number of cylinders.

```
ggplot(data = mtcars, aes(x = factor(cyl), y = hp)) +
  stat_summary(fun.y = "mean", geom = "bar")
```



First, let us verify what we see is what we wanted. Don't worry about the code below right now, I am interspersing these to familiarize you with numeric representation of these plots. You will learn about aggregate function in the next learning activity.

I will calculate average horsepower for different number of cylinders. If the plot is accurate the information below should match the plot.

```
aggregate(hp ~ cyl, data = mtcars, FUN = mean)
```

```
##   cyl        hp
## 1   4  82.63636
## 2   6 122.28571
## 3   8 209.21429
```

The plot seems to be spot on (SURPRISE! SURPRISE!).

Now let us go over the syntax for ggplot function call.

```
ggplot(data = mtcars, aes(x = factor(cyl), y = hp)) + stat_summary(fun.y = "mean", geom = "bar")
```

1. What I do here is call ggplot to initialize a plot, I use two parameters.

a. data = mtcars: I specify where the variables are.
b. aes(): pass the aes function as the second parameter, aes function specifies how variables are mapped to plot objects. Additional layers will use the initial aesthetic mapping in gplot call.
   - x = factor(cyl): x axis will show categorical variable cylinders.
   - y = hp: y axis will show continuous variable horse power.

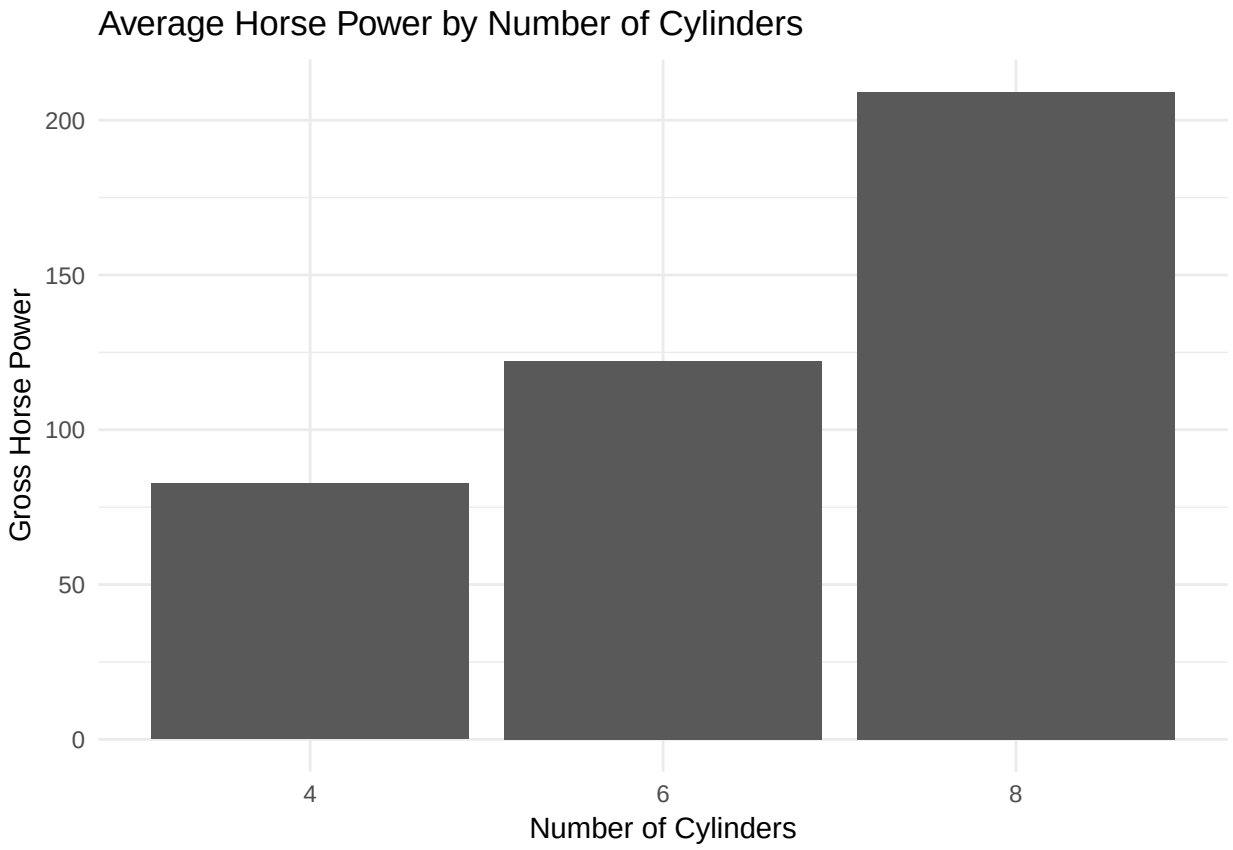
2. Then I overlay a layer with the + sign. This layer will instruct ggplot on how to treat the variables specified in aes.

a. fun.y = "mean": plot the average of variable on y axis

b. geom = "bar": plot a bar chart

I don't expect you to become ggplot experts, but I expect you to be able to follow an example if you see one. Below is an example that builds upon the plot above. Should be easy to see how layering works.

```
# Initialize the plot, map variables
ggplot(data = mtcars, aes(x = factor(cyl), y = hp)) +
  # Specify transformations to variables
  stat_summary(fun.y = "mean", geom = "bar") +
  # Title of the plot
  ggtitle("Average Horse Power by Number of Cylinders") +
  # X label
  xlab("Number of Cylinders") +
  # Y label
  ylab("Gross Horse Power") +
  # themes are available to get prettier results
  theme_minimal()
```

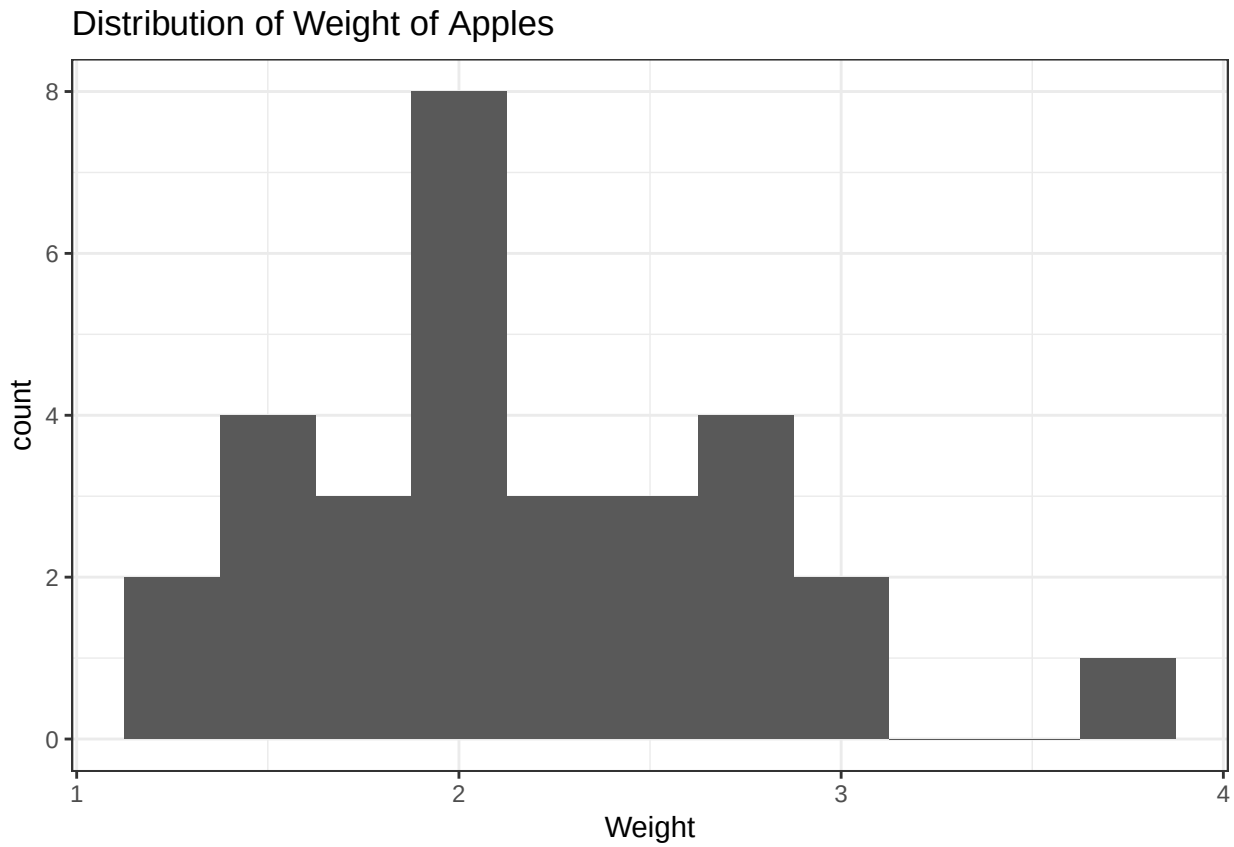


Histogram

Remember histogram is for continous variables and shows frequency.

```
# Initialize the plot with variables of interest
ggplot(bagsOfApples, aes(x=weight)) +
  # Instruct ggplot to plot bin width (range) of .25
  geom_histogram(binwidth=0.25) +
  ggtitle("Distribution of Weight of Apples") +
```

```
  xlab("Weight") +
  theme_bw() # make it pretty with a theme
```




```
ggplot(bagsOfApples, aes(x = weight)) +
  geom_histogram(binwidth = 0.25) +
  ggtitle("Distribution of Weight of Apples") +
  xlab("Weight") +
  theme_bw()
```

1. I initialize ggplot with two parameters:

a. bagsOfApples: datasource to find variables specified in aes
b. aes: aesthetic mapping, which variable goes where.
   - x = weight: I specify a weight as the variable mapped to x axis.

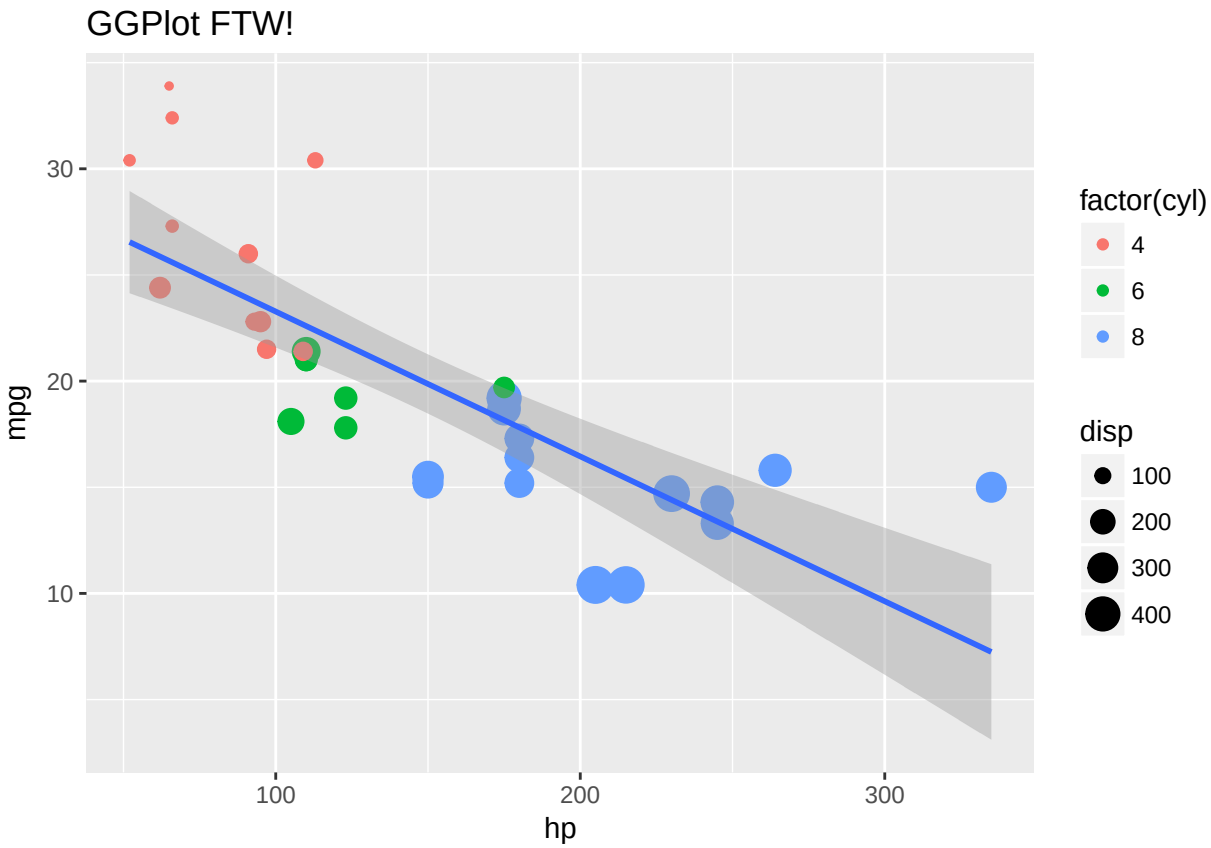
2. I layer a histogram with geom_histogram() with a single parameter:
   - binwidth = 0.25: This specifies the width of each bin (range)
3. I layer a title with ggtitle().
4. I layer x label to rename what is shown by ggplot with xlab().
5. I choose a theme to make it all pretty with theme_bw().

Here is a little exercise for you, experiment with different binwidths to understand what bin means.

Scatter Plot

This example should be familiar to you from the part on qplot.

```r
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(color = factor(cyl), size = disp)) + # For scatter plot
    geom_smooth(method = lm) + # Add a regression line
  ggtitle("GGPlot FTW!") # Add a title
```



Let us go over what is going on here.

```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(color = factor(cyl), size = disp)) + # For scatter plot
    geom_smooth(method = lm) + # Add a regression line
  ggtitle("GGPlot FTW!") # Add a title
```

1. I initialize ggplot with two parameters:
   a. mtcars: data source, this is where ggplot will find our variables.
   b. aes: Aesthetic mapping, which variable goes where. Here I map horse power to x axis and mpg to y axis.
2. I layer a scatter plot with geom_point() it a single parameter:
   a. aes: Aesthetic mapping, on top of what was specified in ggplot's aes parameter, I put two further constraints on points.
   - color = factor(cyl): Use number of cylinders to color the points.
   - size = disp: Use displacement for size of points.
3. I layer a fitted regression line with geom_smooth(). It takes the variables specified in ggplot's aes parameter and fits a line.
   a. method = lm instructs geom_smooth to use a simple linear regression.
4. I layer a title with ggtitle()

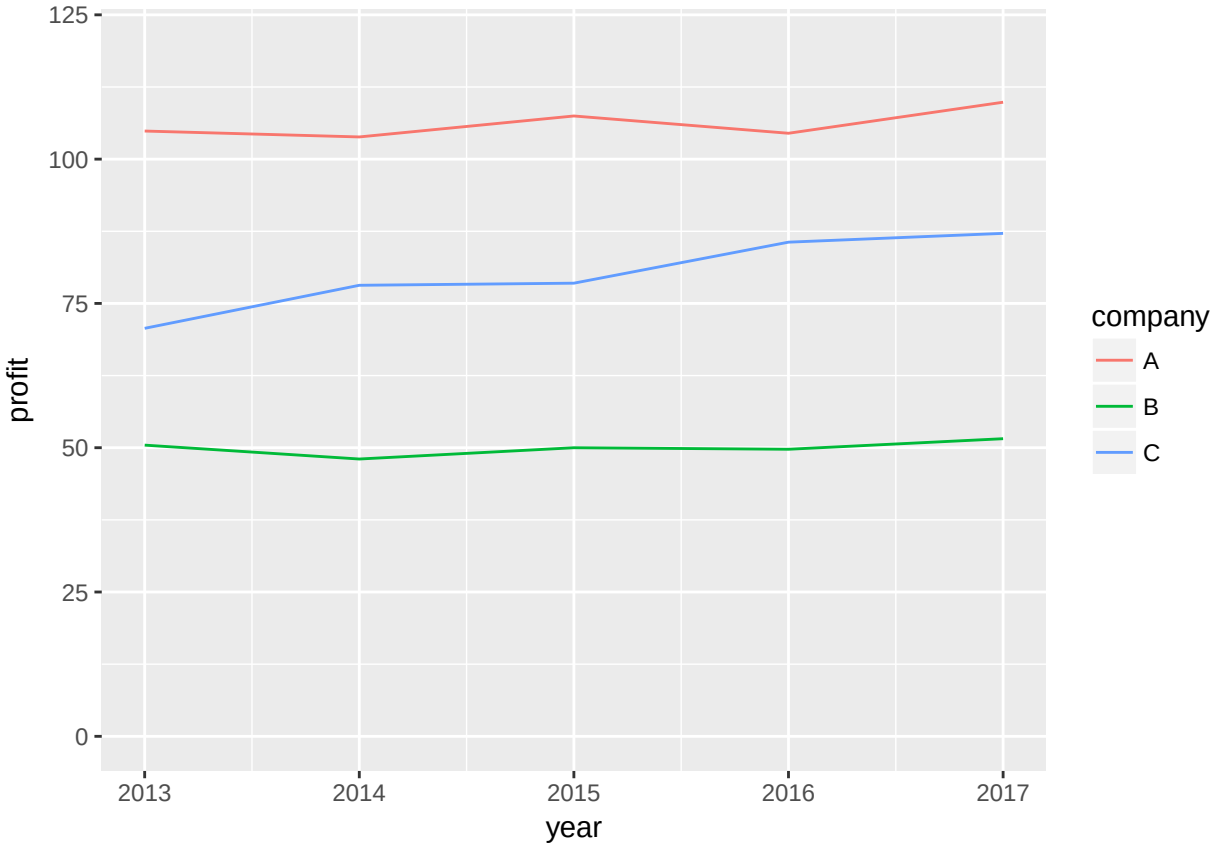
Here is an exercise for you. Experiment with layers by removing any one of the layers and checking the

results out.

Line Plot

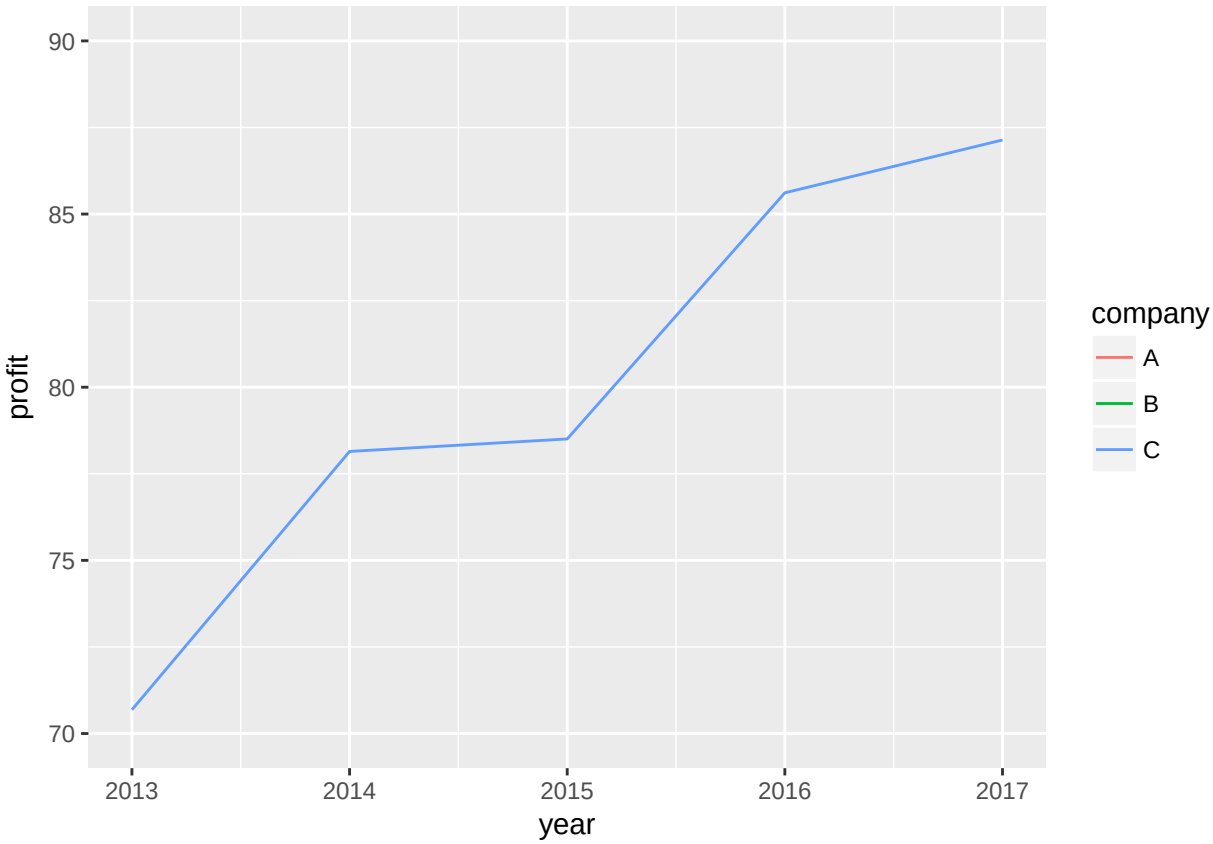
We will go over the earlier example in qplot with ggplot to see how it looks in ggplot.

```
ggplot(data = profits, aes(x = year, y = profit, color = company)) +
  geom_line() +
  ylim(0, 120) # Change the range of Y variable to put a more objective slant on things
```



The syntax should be familiar to you by now. The only new addition is the ylim, basically I specify the range of Y variable to plot so that the profits can be seen in a more objective light. Skewing the range of Y variable is how advertisers sometimes use visualizations to mislead people. If you hold the range small, a tiny change may appear big. See below:

```
ggplot(data = profits, aes(x = year, y = profit, color = company)) +
  geom_line() +
  # Change the y range to mislead people and be a terrible human being.
  ylim(70, 90)
```



caret and Visualizations

We will cover caret a little in machine learning samples. This package provides convenient shortcuts to ggplot functionality. Simplifying most common plotting tasks in machine learning. Please refer to the project page for further reference.

Further Reading:

If you want to learn about ggplot2, I recommend heading to ggplot2.org. Reading the manuals will get you out of a tight spot every once in a while but to understand the logic of the language you may need to read some overarching source such as ggplot2: Elegant Graphics for Data Analysis.

Solutions to Exercises

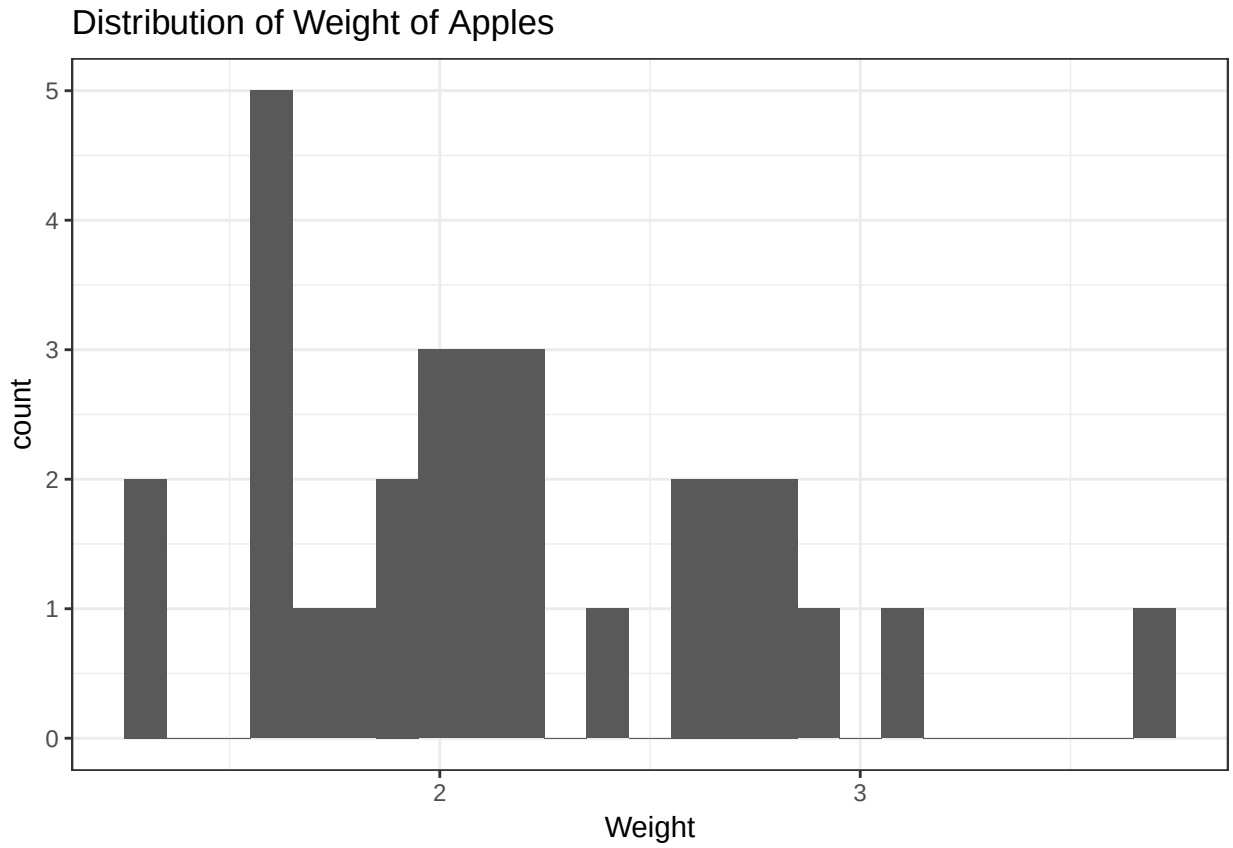
1. Install ggplot2 package.

```
# UNCOMMENT THE BELOW CODE TO RUN
# install.packages("ggplot2")
```

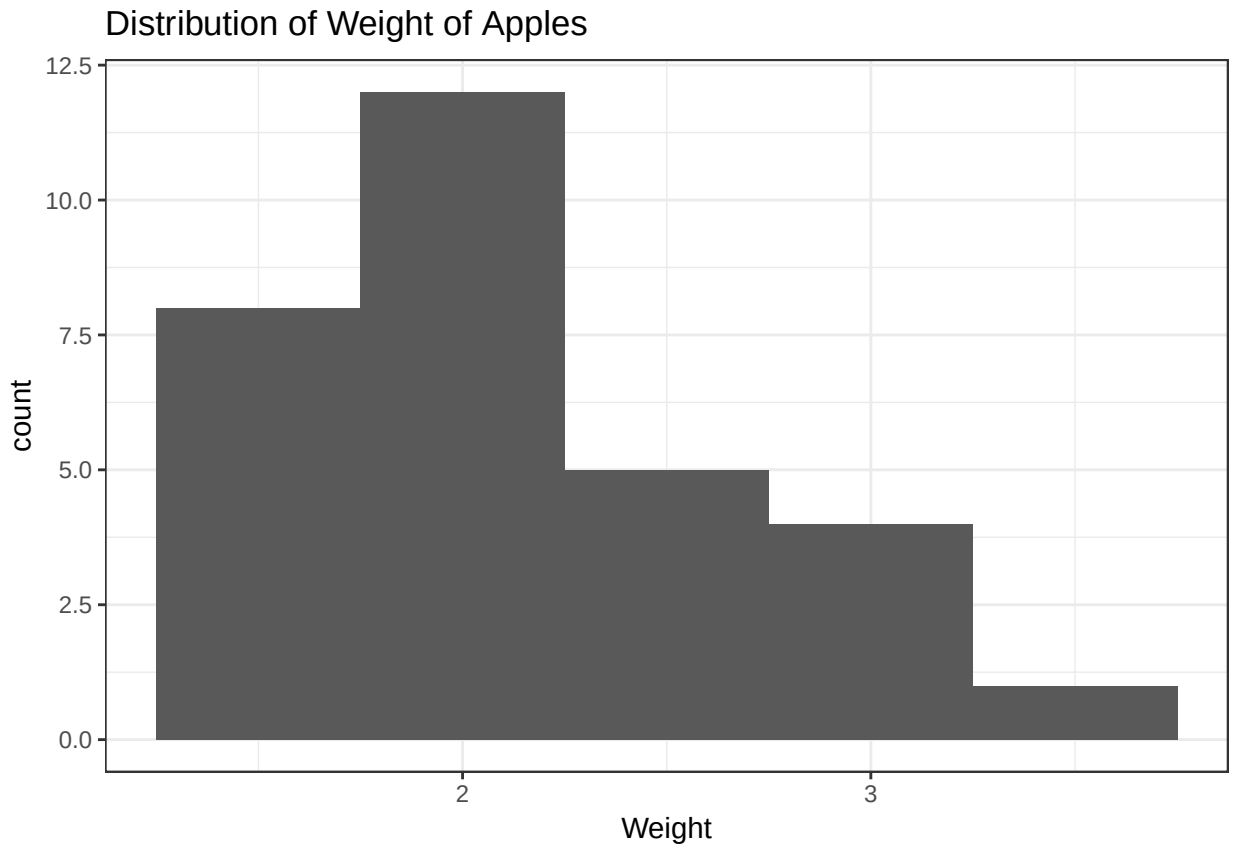
2. Experiment with different binwidths to understand what bin means.

```
# Initialize the plot with variables of interest
ggplot(bagsOfApples, aes(x = weight)) +
  # Instruct ggplot to plot bin width (range) of .3
  geom_histogram(binwidth = 0.1) +
```

```
  ggtitle("Distribution of Weight of Apples") +
  xlab("Weight") +
  theme_bw()
```




```
# Initialize the plot with variables of interest
ggplot(bagsOfApples, aes(x = weight)) +
# Instruct ggplot to plot bin width (range) of .3
  geom_histogram(binwidth = 0.5) +
  ggtitle("Distribution of Weight of Apples") +
  xlab("Weight") +
  theme_bw()
```



3. Experiment with layers by removing any one of the layers and checking the results out.

```
# Just comment out any line beyond the first.
ggplot(mtcars, aes(x = hp, y = mpg)) +
  # For scatter plot
  geom_point(aes(color = factor(cyl), size = disp)) +
    # geom_smooth(method=lm) + # Add a regression line OR NOT
  ggtitle('GGPlot FTW!') # Add a title
```