Getting Comfortable with your Data in R - Visualization and Descriptive Statistics

A Michelle Edwards, Ph.D., MLIS

October 29, 2019

Table of Contents

Quick Review

Last workshop we talked about collecting our data, cleaning it to a point that would allow us to enter it into Excel and eventually into SAS or R.

Items we discussed included:

- Best practices for naming our variables or column headings
- Knowing your data - what were the expected ranges of values? typos?
- How to get the data into R

Research Question

Remember how all of our research is driven by a research question. We do not go out and collect data just for the fun of it. We collect data with a purpose and that purpose will be linked back to your research question.

Now that we have our data collected and loaded into our statistical package, let's start to get more comfortable with it, by visualizing it and then running some descriptive statistics.

Data Visualization - what is it?

Data visualization can mean many different things to different people. To me, it all comes down to the purpose. WHY? and for WHOM? Think of data visualization as a tool to help you tell your story. I've always seen data analysis or statistical analysis as a way to help me to tell the story of my research. There are so many "tools" that you can use, so let's think of data visualization and statistical analyses as 2 tools that you will use to tell your research story.

Remember that visualizing your data is "critical to the understanding of the contents of your data." We use data visualization to help us examine, scrutinize and validate our data.

There are a great many resources out there for us to use as a guide to the world of Data Visualization. I will use 2 in particular - but please use the ones that speak to you. The resources I like to use are:

- "Effective Data Visualization: The Right Chart for the Right Data" (2017) by Stephanie D.H. Evergreen
- "Making Sense of Data II: A Practical Guide to Data Visualization, Advanced Data Mining Methods, and Applications" (2009) by Glenn J. Myatt and Wayne P. Johnson

Five General Principles behind Data Visualization:

1. Show the data
2. Simplify - you want to keep the message simple and remove all the flowery bits of your visualizations
3. Reduce the clutter - do you REALLY need all those grids or ticks on your graph??
4. Revise your visualizations - creating a graph or a table should be viewed as part of your writing. You don't write something and leave it. You write, you revise, you may write again and revise again. Treat you visualizations in the same manner.
5. Be HONEST! This may sound funny - but let's face it, there are times where the visualizations we create may have an element of exaggeration added in. Those y-axes - where do they start? at 0 or somewhere else? Are we exaggerated the differences between those lines?

Graphic Design Principles

Yes there are some graphic design principles that you may want to consider as you begin to think about creating your data visualizations. This is a link to a powerpoint presentation that I use when talking about these principles. Please take a few minutes when you have the time to review them. Each has a visual to explain the principle.

Data Types

The data that we colect from our research projects may not always be the same type. What do I mean by a Data Type? Let’s review the basic types:

Quantitative

- Measures a quantity
  - Continuous
    - a measure of something
  - Categorical
    - Nominal: you are in one group or another, there is NO order to the groups
    - Ordinal: you are in one group or another, there IS an order to the groups
- Examples of:
  - Continuous: height, weight, age,
  - Nominal: Yes/No, Program of Study, Gender
  - Ordinal: shirt size, LIkert scale, Year of study

Qualitative

- Measures a quality
- Descriptive, words,

Digging into Visualizing Examples

Let’s begin by loading data into our program - R. The data we will be using is the Fisher’s Iris dataset whic is included with the Base R program. To view this dataset please use the following R statements.

```
data(iris)

# View the top observations in Fisher's Iris data
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa

# Let's take a look at the variable names in Iris data
names(iris)

## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

Note that there are 5 variables:

- Species of the Iris measured
- Sepal length measured
- Sepal width measured
- Petal length measured
- Petal width measured

Remember that R is case-sensitive, take careful note of the variable names.

Exercise 1

1. What type of data are each of the 5 variables available in the Iris dataset?

Visualizing ONE number (univariate)

Let's start with the Species variable. We have 1 variable - so univariate - and we want to see what's happening with it. **What kind of visualizations could you do?**

- _
- _
- _

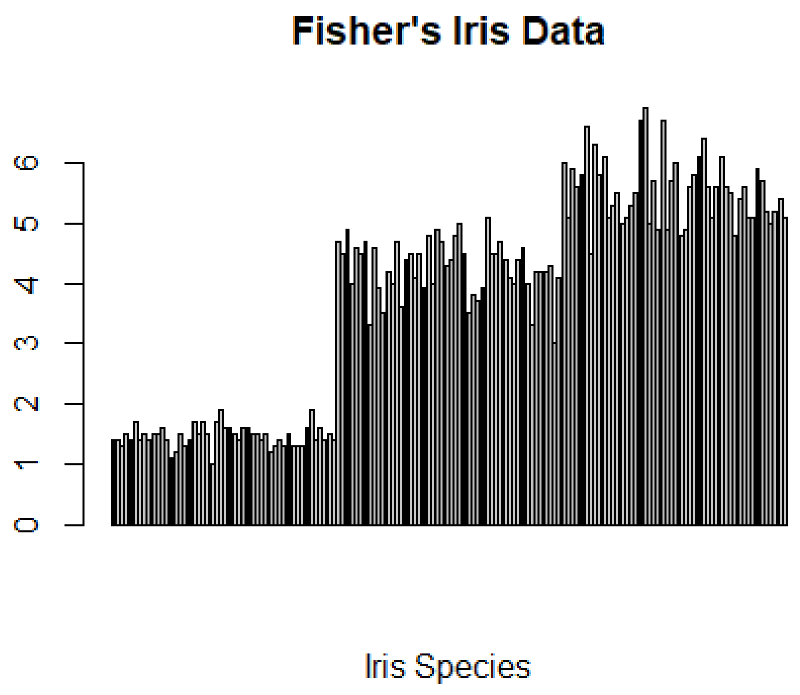
Bar Chart

Bar chart can show us the frequency or the count of observations in Species, and we could also break it up for each species. In R, let's start with the basic Barplot function. To use this function, we need to create a value that we want to plot. In this example we are creating an object called counts by using the table function, which will sum how many observations are in each Species.

main= is the Main title for the chart.

xlab= is the label for the X-axis

```r
# Simple Bar Plot

barplot(iris$Petal.Length, main="Fisher's Iris Data",
        xlab="Iris Species")
```



Is there anything unusual about this barplot? What were you expecting to see?

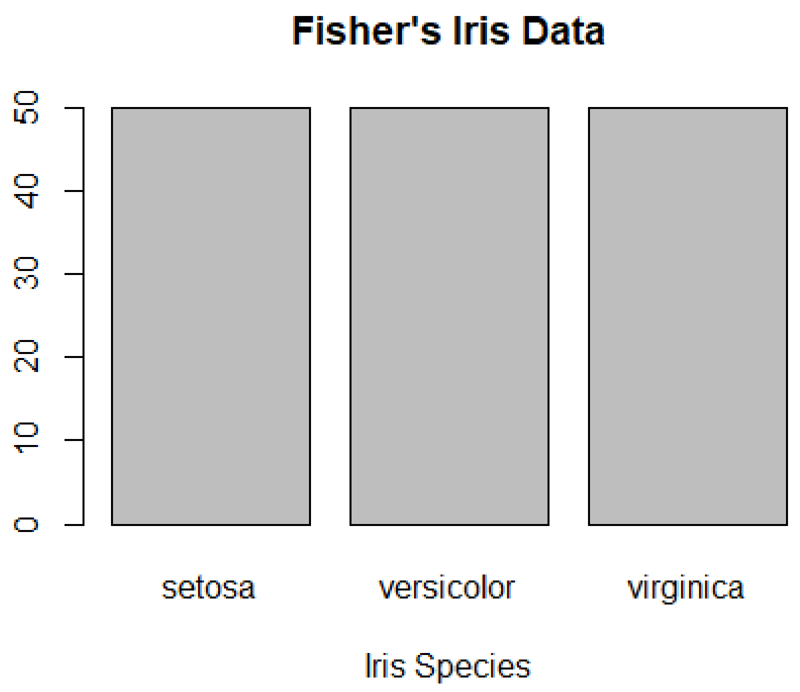
Try running the plot again- but changing Petal.Length to Species. What happens??

When we use the Base R barplot function, we need to specify the exact information that we want to see plotted. I was expecting to see a barplot with 3 bars, each representing the number of observations in each Species. To create this using the barplot() function we first need to create a new object that contains the counts for each species. We will create an object called **counts** that contains the numb er of observations of each Species.

```r
counts <- table(iris$Species)
```

Now we will use this new object to create the barplot.

```r
barplot(counts, main="Fisher's Iris Data",
   xlab="Iris Species")
```



Phew! the barplot I was expecting with a few extra steps. We will try this again in a few minutes with the ggplot2 package.

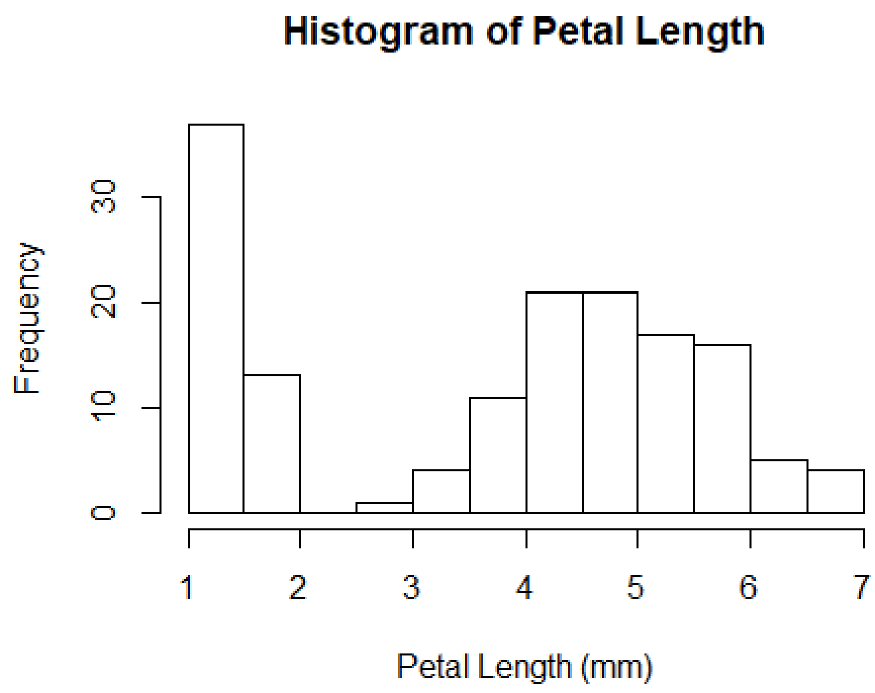
Exercise 2

1. What story does this graph tell?
2. When would you use a barchart with data that are continuous? Such as petal width?

Histograms

What is a histogram? Show frequency or counts of your data in a group or an interval. Generally used for continous data. Let's try it on petal length using the Base R function of hist()

```
hist(iris$Petal.Length,
     main="Histogram of Petal Length",
     xlab="Petal Length (mm)")
```



We know that we have 3 species of Irises in this dataset, so let's try to pull them apart, by creating a histogram for each species. To create this barplot in R, we can use the base R function barplot(), but we would need to convert Species from textual to numeric format. This would take a few extra steps of programming, so let's turn our attention to the ggplot2 package in R. This package is probably the most commonly used package to create graphics in R.

Our first step will be to install the package and then load the package onto our own computers. Remember from our first R workshop, that there are different ways to install an R package. To review please visit the Introduction to R workshops notes.

```r
# Install the ggplot2 package
#install.packages("ggplot2")

# Load the package
library(ggplot2)
```

Before we dig into ggplot - let's take a step back and talk about the ggplot2 package.

ggplot2

ggplot2 was created by Hadley Wickham in 2005. It is an implementation of Leland Wilkinson's Grammar of Graphics-a general scheme for data visualization which breaks up graphs into semantic components such as scales and layers. Since 2005, ggplot2 has grown in use to become one of the most popular R packages. source

Advantages of ggplot2:

- consistent underlying grammar of graphics (Wilkinson, 2005)
- very flexible
- themes that allow you to create a polished plot appearance
- mature and complete graphics system
- many users with an active mailing list - support!

You cannot create the following with ggplot2:

- 3-dimensional graphics - see the "rgl"" package
- graph-theory type graphs - see the "igraph" package
- interactive graphs - see the "ggvis" package

What is The Grammar Of Graphics?

The basic idea is that you can create the plot building blocks independently and later combine them to create any kind of graphical display you want. The Building Blocks of a graph include:

- data
- geometric object (geom) - objects drawn that represents the data: bars, points, etc…
- asethetic mapping - visual properties of the geoms: line colour, point shapes, etc…
- statistical transformations
- scales
- coordinate system
- position adjustments
- faceting

Grammar Of Graphics - Geometric Objects and Aesthetics

Geometric Objects (geom)

geoms are the actual points that are on plot. Think of it as the datapoint on the plot. We can have different types of geoms:

- points (geom_point, used in scatter plots, dot plots, etc…)
- lines (geom_line, used for time series, trend lines, etc…)
- boxplot (geom_boxplot)

A plot must have at least on geom - if you think about it we need to plot something and we need to tell R what we want to plot. You can have more than one geom on a plot at the same time. To do this we simply add a + operator.

Curious about the types of geoms available to you? Try this bit of code to see all the options.

```
# help.search("geom", package="ggplot2")
```

Aesthetic Mapping

"aesthetic" means "something you can see". These are attributes that you can add to the geom of your plot.

- position on the axes
- colour
- fill - colour
- shape of points
- type of line
- size

Each type of geom accepts only a subset of all the aesthetics. Please refer to the help of each geom to see what is available. The aesthetic mappings are set with the aes() function.

General code form for ggplot2

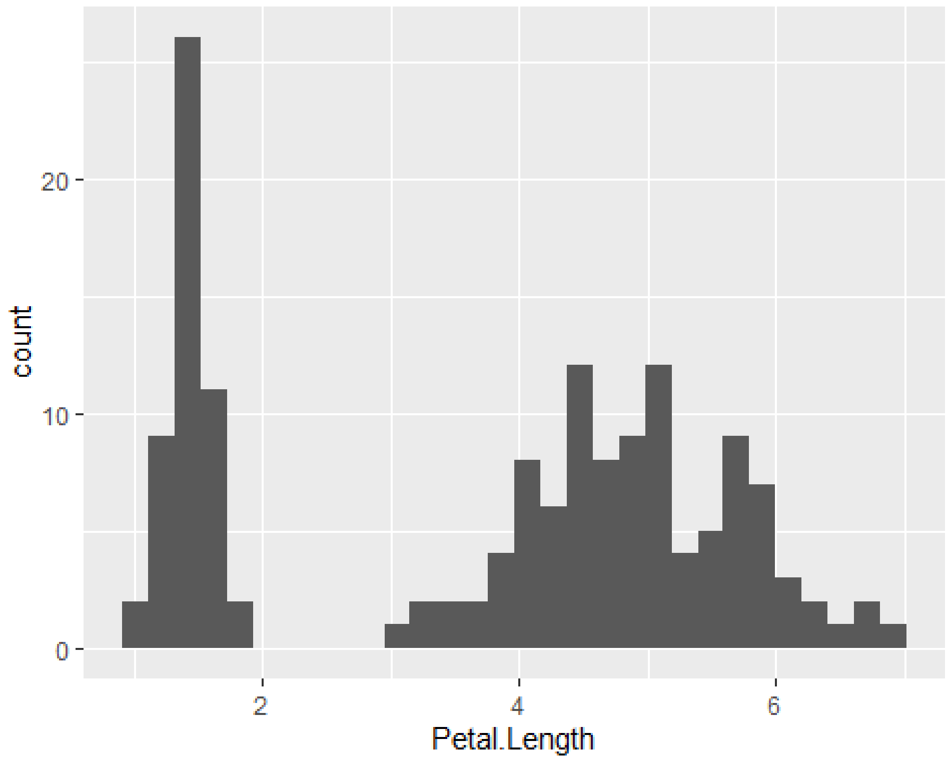
Being aware of the background of ggplot2 and what it is based on, will help you to create code for the different types of plots you are interested in.

Just remember that aes() contains the aesthetic characteristics of your plots, whereas geom_…() specifies the type of plot you want to create.

So - let's go back and work through the histogram example again using ggplot2.

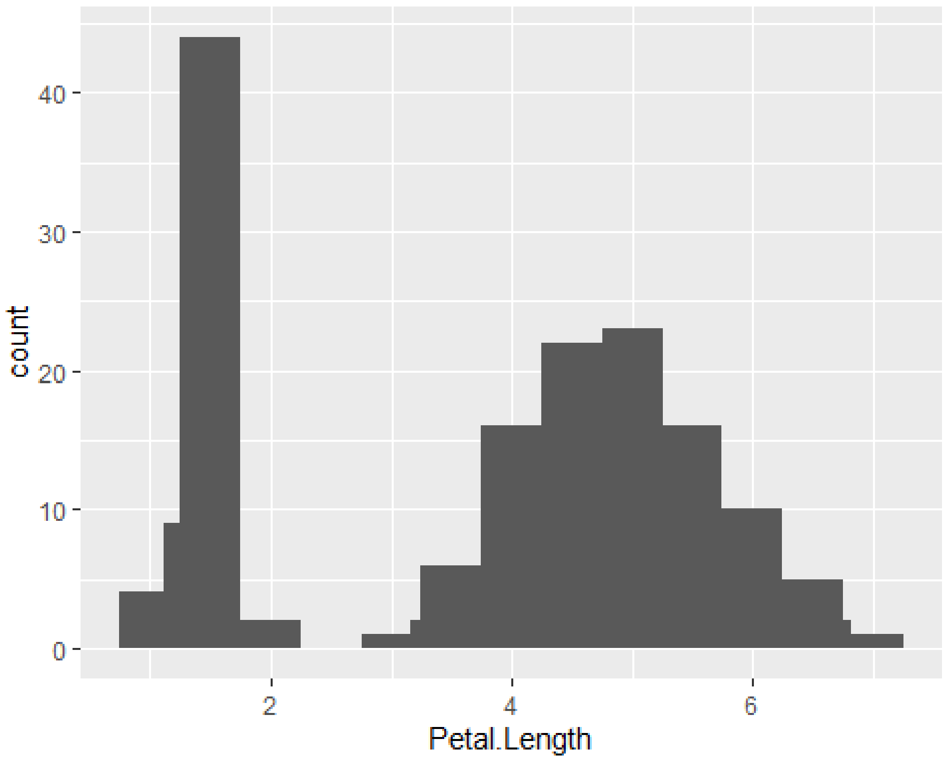
We first specify the dataset name, followed by the aes() characteristics - these will change depending on the type of plot you are creating. In this case - we only need to specify which variable is our x-axis - Petal.Length. Then we tell R what type of plot we want - in this case it is the geom_histogram. Let's try it out to see what happens.

```
## Use different breaks and different y-axis limits for each graph
ggplot(iris, aes(x=Petal.Length)) +
  geom_histogram()

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



See the note that comes along with this plot. To change the bin_width, use the following code. Try running this a few times, while changing the value of the binwidth.

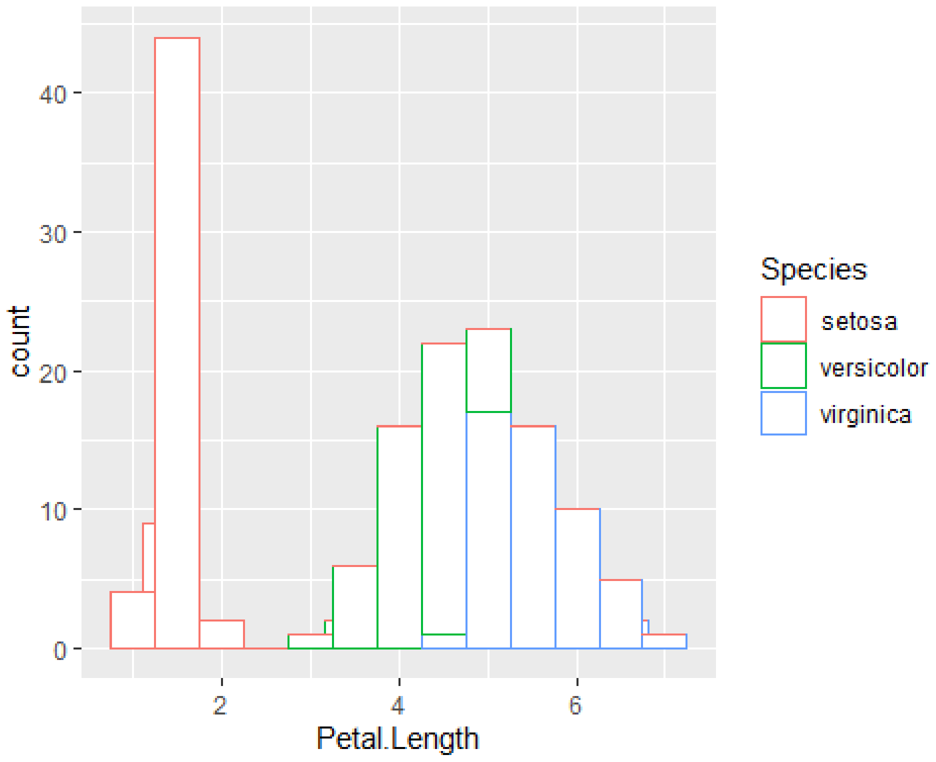
```
ggplot(iris, aes(x=Petal.Length)) +
  geom_histogram()+
  stat_bin(binwidth=0.5)

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



We also know that we have 3 species in this data, let's change the aesthetics of the graph by doing 2 things:

1. Change the fill of the bars to white
2. Change the colour of the outline of the bars to match the Species

```
ggplot(iris, aes(x=Petal.Length, colour=Species)) +
  geom_histogram(fill="White") +
  stat_bin(binwidth=0.5, fill="white")

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



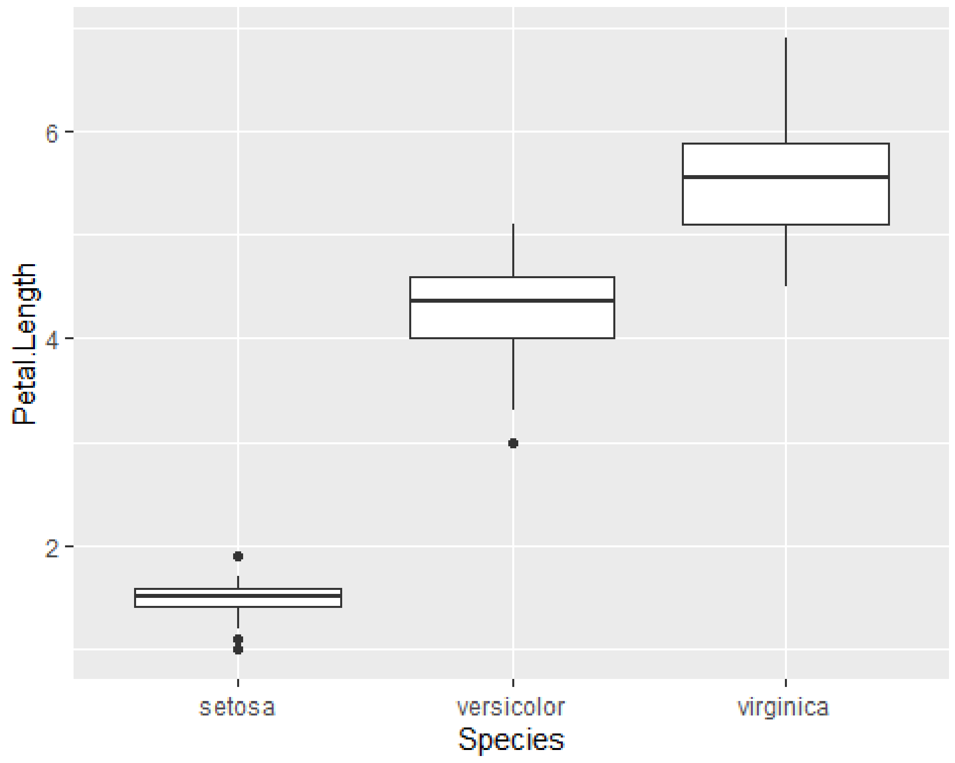
Exercise 3

1. Working with a partner, review the last R code and take turns explaining what each line means.
2. Try resetting the bins to create a new histogram.
3. Create another set of histograms for Sepal Length.
4. Try to explain why it appears that we see 2 sets of bars?

Boxplots

What is a boxplot? A standardized, visual representation of the distribution of your data. Oh now that's a mouthful! It's a way of looking at your data to see whether the values you've collected are evenly distributed across the range you collected. Oish! How bout this - bell curve - but in a box format? A classic visual of a box plot will show you the 4 quartiles of your data - so where is 25% of your data, 50% (median), 75%. It will also show you where the mean of your data resides. It can also show you if you have any outliers - any data points that are beyond the expected normal distribution. I know you've all seen these, and you will use them when or rather if you conduct any GLMM analysis. These are a very handy plot to help examine your residuals.

```
ggplot(iris, aes(x=Species, y=Petal.Length)) +
  geom_boxplot()
```



Exercise 4

1. With a partner, review these last plot and describe what you are seeing? What story is the data trying to tell?
2. Describe what each part of the R code means.
3. Starting to think about descriptive statistics, what numbers/values/statistics might you want to add here to help fill in the story?

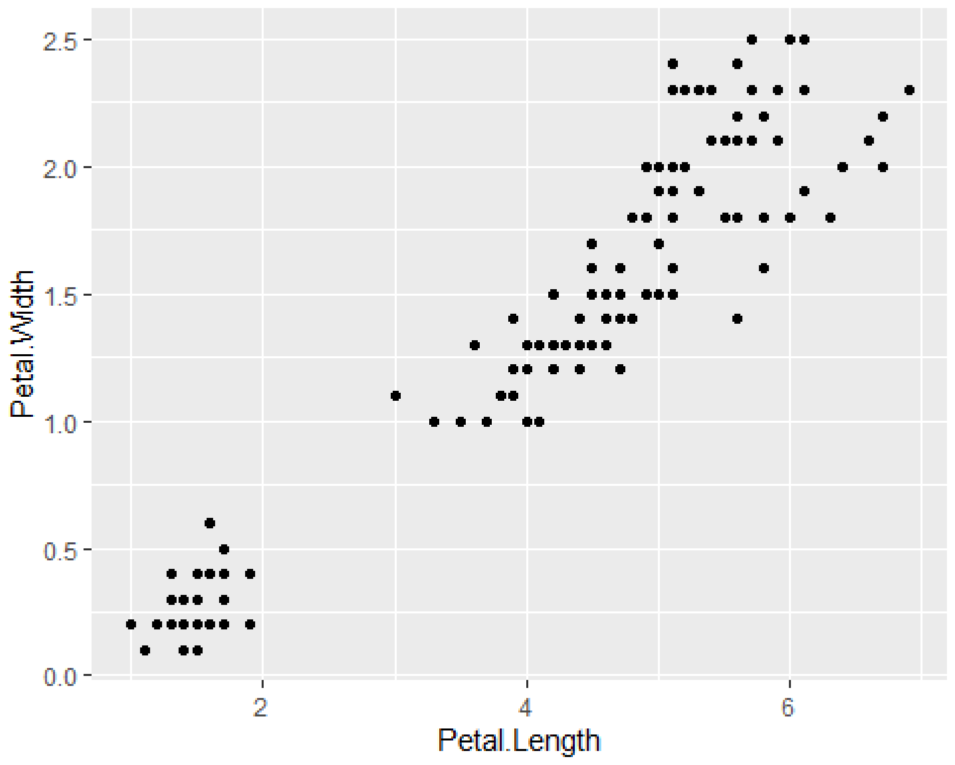
Visualizing MORE than ONE number (bivariate or multivariate)

When we have more than one variable that we are interested in plotting at the same time, then we have more options. Maybe a scatterplot to see if there is a relationship between the variables, or a side-by-side barchart to see how things may be different? Let's take a quick peak at a scatterplot and a barchart.

Scatterplot

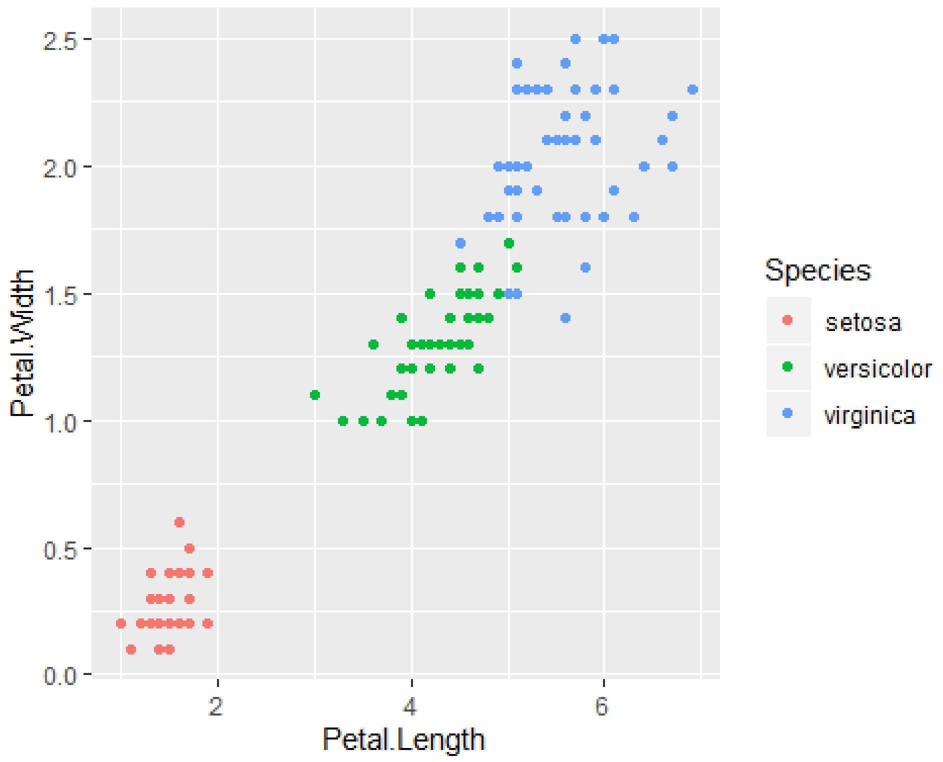
As the name suggests we are creating a plot with one variable as the Y-axis and the second variable as the X-axis. Let's try a scatterplot with petal width as our Y and petal length as our X.

```
ggplot(iris, aes(y=Petal.Width, x=Petal.Length)) +
  geom_point()
```



Let’s change the colour of the dots to reflect the 3 different species represented in this dataset. To do this we are going to add some aesthetics to the points:

```r
ggplot(iris, aes(y=Petal.Width, x=Petal.Length)) +
  geom_point(aes(color=Species))
```



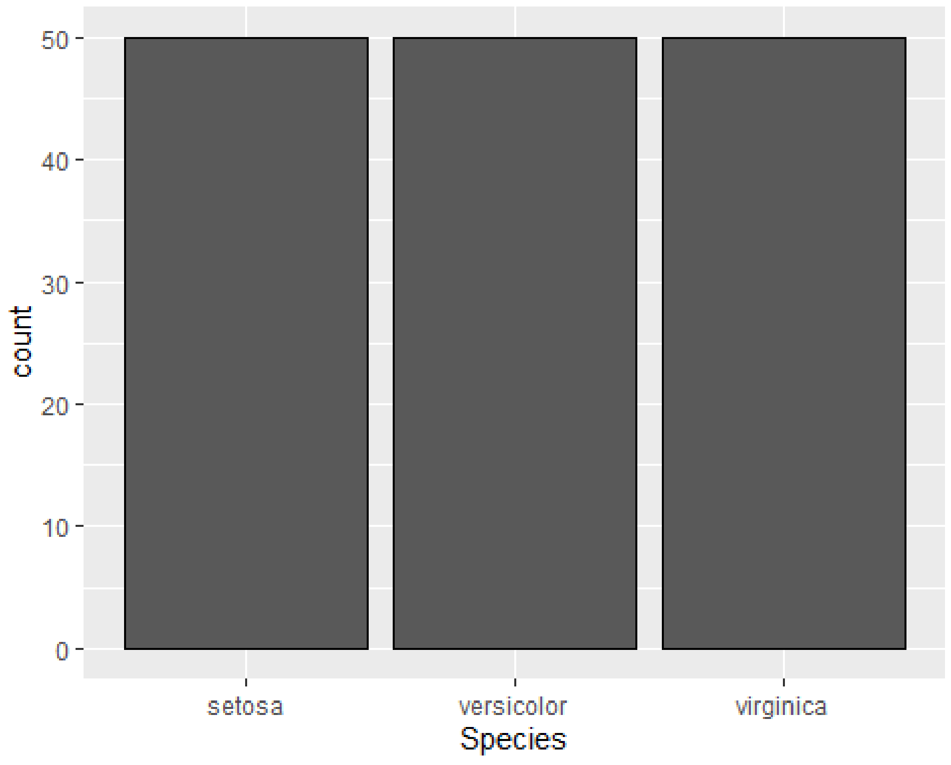
Exercise 5

1. With a partner work out the story that this data visualization is telling?
2. Thinking ahead to the statistics - what might you want to run to help fill in this story?

Bar chart with means

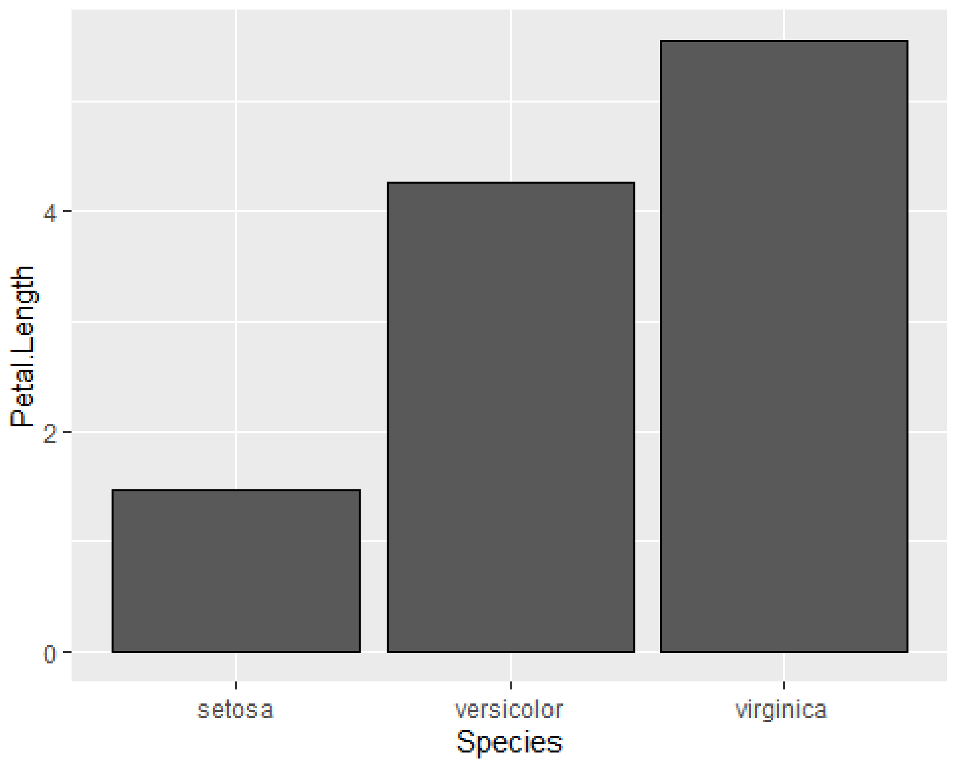
We often like to see our data represented by our groups in a bar chart. So let's use the ggplot2 package and see what happens. Much easier than using the Base R barplot() function.

```r
ggplot(iris, aes(x=Species)) +
  geom_bar(stat="count", color="black")
```



Now let’s see about changing the bars to represent the mean of Petal.Length for each species. First, we need to add the value of the y-axis. Then we need to change the stat to “summary”, and finally let R know that we are interested in the mean.

```
ggplot(iris, aes(x=Species, y = Petal.Length)) +
  geom_bar(stat="summary", color="black", fun.y = "mean")
```



Now we would like to add one more piece of information - a measure of variability for the bars. To accomplish this, we need to calculate the measure of interest first, standard error or confidence limits. So… let’s put this aside for now, take a look at the descriptive stastistics and then come back to this.

Descriptive Statistics

The next step on your data visualization and analysis path, are descriptive statistics. As much as we would all love to jump ahead and get to the "meat" of our research and run our models, we should all specnd time getting comfortable with your data. We've just spent time visualizing the data - getting a sense as to what the data looks like, but now let's get a better sense of the numbers behind some of the visualizations.

Descriptive statistics is exactly that - statistics that will help you describe your data. These statistics may include:

- mean, median, mode
- frequencies
- distributions or ranges
- measures of variation

We reviewed the different types of data above. If you need to review these again, please take a couple of moments to reread the different data types. Why you may ask? The type of data you are working with till depict, in most cases, what type of statistic you will calculate.

Summary()

R has a versatile function called Summary() that provides a summary of your analysis results. A function that we will use on a regular basis as we move to working more with our data and its analysis. To begin though, let's see what Summary() provides us with our iris dataset.

```
summary(iris)

##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##                 
##                 
## 
```

Take note that it provides some basic descriptive statistics for each variable in the dataset. Let's work through these together.

We ran the Summary() on the entire dataset, but what if we are only intersted in one particular variable? We would modify the code to specify the variable name within the dataset as this:

```
summary(iris$Petal.Length)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   1.600   4.350   3.758   5.100   6.900
```

Now, the results we are seeing correspond to the entire dataset, so all 3 species. After reviewing the plots we created earlier, we know that there are differences between the 3 iris species. So let's review the descriptive statistics for the species separately. To do this in R, we need to use another function summarize() and learn how to use pipes.

We can view the mean of Petal.Length for each Species in our Iris dataset using a couple of different ways. First let's look at the **tapply()** function and then we'll take a look at uing the pipe feature in R.

tapply()

We can use the tapply() function to see the means, standard deviation, variation, and N for any variable in our data - let's continue to use Petal.Length. To use this function, we first specify the variable of interest Petal.Length, followed by the variable we would like it broken down by Species, followed by the statistic of interest.

Let's try looking at the mean, sd, variance, and N.

```
# Calculationg the MNean of Petal.Length for each species
tapply(iris$Petal.Length, iris$Species, mean)

##     setosa versicolor  virginica 
##      1.462      4.260      5.552

# Calculationg the Standard Deviation of Petal.Length for each species
tapply(iris$Petal.Length, iris$Species, sd)

##     setosa versicolor  virginica 
##  0.1736640  0.4699110  0.5518947

# Calculationg the Variance of Petal.Length for each species
tapply(iris$Petal.Length, iris$Species, var)

##     setosa versicolor  virginica 
## 0.03015918 0.22081633 0.30458776

# Calculationg the Number of Observations of Petal.Length for each species
tapply(iris$Petal.Length, iris$Species, length)

##     setosa versicolor  virginica 
##         50         50         50
```

dplry

Now let’s do a little bit of R programming to learn more about PIPES in R. Our goal is to obtain the mean of each of our Species in the iris dataset. To accomplish this in a different way, we first need to let R know that we want to group the results by Species, or in other words, we want to see the results separately for each Species listed in our dataset. The second step is to request the means of a particular variable - so in our example we will use Petal.Length. We could try to run each of these steps separately and save the results, but let’s run it all together by taking advantage of pipes in R.

To learn more about using pipes in R, we need to install another package dplyr. Please install this package to your computer and then run the library().

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

means.species <- iris %>%  # specifying the dataset we are using
  group_by(Species) %>%    # specifying the group that we want to see
  summarize(PL = mean(Petal.Length), PW = mean(Petal.Width))  # creating the
output means

means.species

## # A tibble: 3 x 3
##   Species       PL    PW
##   <fct>      <dbl> <dbl>
## 1 setosa      1.46 0.246
## 2 versicolor  4.26 1.33
## 3 virginica   5.55 2.03
```

Exercise 6

1. What do these results mean? What story are they telling about our data?

Standard Deviation and Standard Error in R

Notice that when we requested means - we only received the mean, nothing else. If you use SAS or SPSS, as examples, when you ask for means, you usually get some measure of variability with them. In R, we need to do a bit more coding to obtain them.

Standard Deviation

To obtain the stddev of a variable in your data, it is quite straightforward, use the sd() function. Let's try it out on the Iris dataset we've been working with:

```
# Standard Deviation of Petal.Length measures in the Iris dataset
sd_pl <- sd(iris$Petal.Length)

sd_pl

## [1] 1.765298
```

Standard Error

Unfortunately, I have not been able to find a similar function for the standard error. So, we need to calculate the standard error based on our standard deviation. Here is the code:

```
#Standard Error calculation for Petal.Length measures in the Iris dataset
se_pl <- sd(iris$Petal.Length)/sqrt(length(iris$Petal.Length))

se_pl

## [1] 0.144136
```

Adding the standard error bars to the barplots is a bit more involved process. There are many examples available for you to follow. Essentially it involves creating a dataframe that contains only the means and the standard errors that you would like to plot. I'm not going to review that process in this workshop. But should you need help with this for your own research, please feel free to reach out to me.

Frequencies or tables

One last piece of information we may want to obtain from our data, are frequencies. We've already seen one way to create these when we created the counts for our barplot(). Let's review that here again. The basic function to calculate frequencies is the table()

```
table(iris$Species)

## 
##     setosa versicolor  virginica 
##         50         50         50
```

Normality test - Shapiro-Wilk

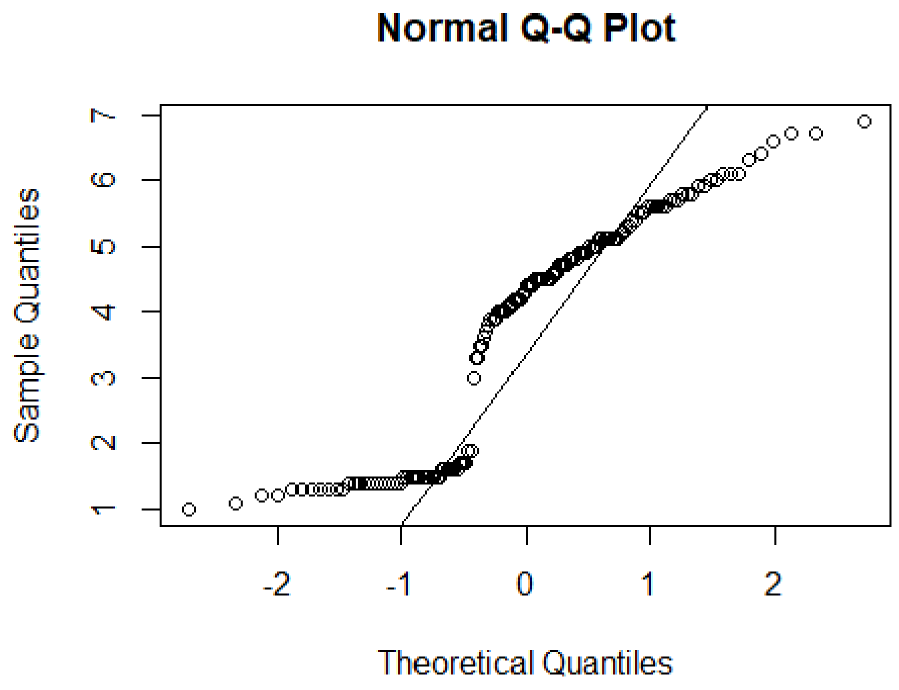
One last descriptive statistic we may be interested in viewing is the Shapiro-Wilk statistic for testing whether our data are normally distributed. This is something we tend to use to test our assumptions for an ANOVA, but let's see how to use it on our Petal.Length variable in the Iris dataset.

```
shapiro.test(iris$Petal.Length)

##
##  Shapiro-Wilk normality test
##
## data:  iris$Petal.Length
## W = 0.87627, p-value = 7.412e-10
```

How do we interpret these results? Take note that the Null hypothesis for this test is that the data come from a Normal Distribution. So a p-value < 0.05 (as a guide) will help us to say that we reject the Null hypothesis and conclude that the Petal.Length data is not from a normal distribution. However, there is more to interpreting this result. Please note the value of the W statistic - in this case it is 0.87627. The goal of this statistic is that it approaches the value of 1.0. If the value of the W statistic is close to 1 (as a rough guide - > 0.80), take a look at a histogram or a normal probability plot before agreeing with the p-value.

Let's look at the normal probability plot for the Petal.Length variable

```
qqnorm(iris$Petal.Length)
qqline(iris$Petal.Length)
```



Hmmm… not quite what we would like to see. Remember we would like to see our data along the qqline. So based on this plot and out Shapiro-Wilk statistic, I would conclude that our Petal.Length variable does not meet a normal distribution.

But… We already knew this - didn't we?

Exercise 7

1. Run the normality test on another variable in our data. Sepal.Width as an example
2. With a partner discuss the results.
3. Why did I say we already knew these would not be normal??

Workshop Review

Data Visualization

- 5 General Principles Behind Data Visualization
- Graphic Design Principles
- Different Data Types
- Different examples
  - Bar Chart
  - Histograms
  - Boxplots
  - Scatterplot
  - many more….

Descriptive Statistics

- Summary()
- Standard Deviation and Standard Errors
- Table()
- Normality Statistics and plots