Introduction to Data Visualization with R

Scott McCain

Today we will go into some basic introductions to R, but mostly to a data visualization package ggplot2. This is also a first-run at the R notebook, the format that you see here. It allows a mix of plots, code, and text. This is in constrast with an R script by itself, which is just code (hopefully also with comments!)

Data visualization in R can be done in two main avenues: ggplot2, and "base" R graphics. I will demonstrate the use of ggplot2, but base R graphics can be very useful as well. I would argue that ggplot2 is significantly better at exploratory plots, and base R graphics are slightly better for publication level plots (but this is just my humble opinion!). Base R gives you ultimate control over every aspect of your plots.

We will use a dataset that is automatically uploaded to R known as iris. So let's take a look at this dataset!

```
#First we need to upload all the necessary packages into the working space.
#You'll notice that these need to be downloaded first, eg:
#install.packages("ggplot2")
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(magrittr)
```

These are some useful functions for when you've got your data uploaded into R. Uploading data into R can be a challenge by itself. I tend to use mostly ".csv" files, and the `read.csv()` function. We can cover that on another time!

```
#What does top of the dataset look like?
head(iris)#Check out the function tail() also!
```

```
##   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
```

```
#What is the structure? (str function)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
#What are the dimensions?
dim(iris)
```

```
## [1] 150   5
```

I don't like writing capital letters, so lets change these names...

```
names(iris) <- tolower(names(iris))
```

There are many ways to do the same thing in R. Do you find the above code statement hard to read? I do! That's because we sort of have to read it inside out - that can get very confusing when you have a lot of functions in one line! Here is a neater alternative:

```
names(iris) <- iris %>% names() %>% tolower()
```

We are using the pipe command, `%>%`. You can read this as "then". So we take the dataframe "iris" *then* get the names, *then* converting to lower case.

"Plot the data, plot the data, plot the data" - R.A. Myers

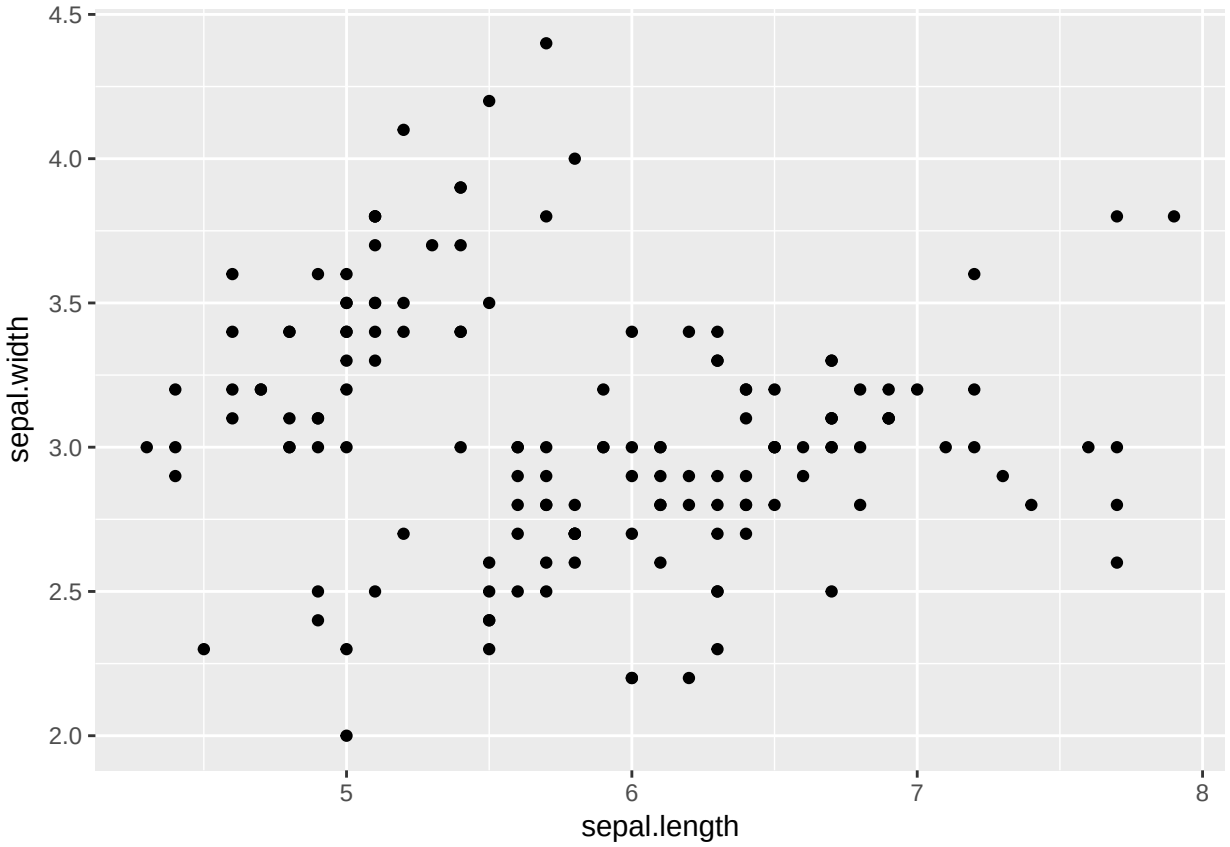
What does it **look** like!

Start off with the base function `ggplot()`. What are the key arguments? You tell me! (`?ggplot`)

```
knitr::opts_chunk$set(fig.width = 4.5, fig.height = 3.5)
names(iris)
```

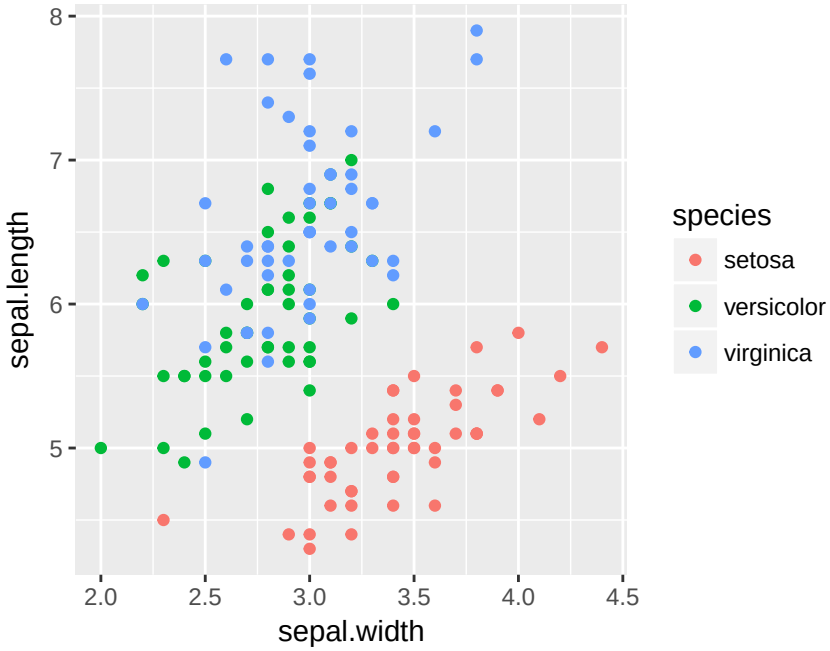
```
## [1] "sepal.length" "sepal.width"  "petal.length" "petal.width"
## [5] "species"
```

```
ggplot(data = iris, aes(x = sepal.length, y = sepal.width)) +
  geom_point()
```



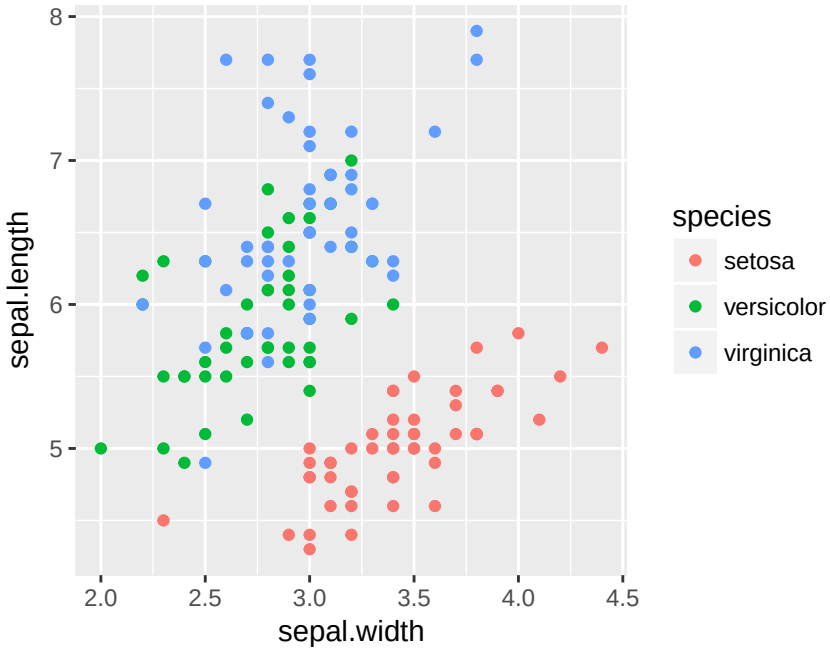
This is a pretty great plot already! But I want to colour code by species.

```
ggplot(data = iris, aes(x = sepal.width, y = sepal.length)) +
  geom_point(aes(colour = species))
```

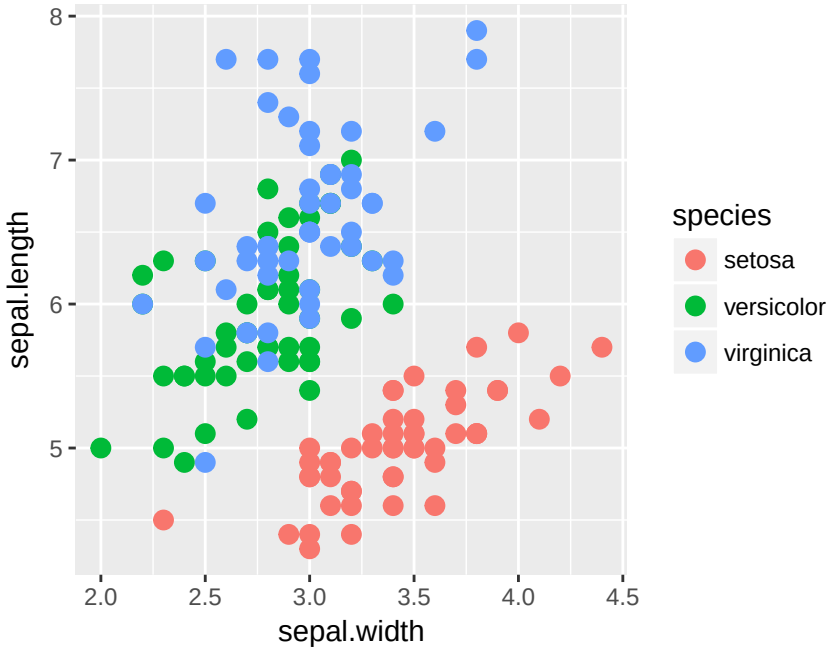


Great! But I don't want to rewrite that code chunk every time. Let's assign this plot to an object:

```
plot_1 <- ggplot(data = iris, aes(x = sepal.width, y = sepal.length)) +
              geom_point(aes(colour = species))
print(plot_1)
```



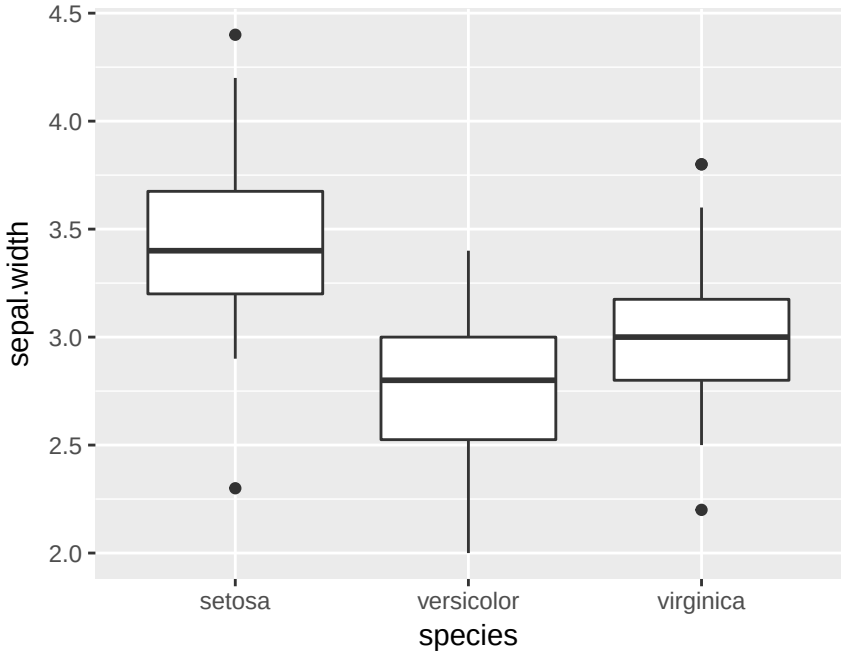

```
plot_1 + geom_point(aes(colour = species), size = 3)
```



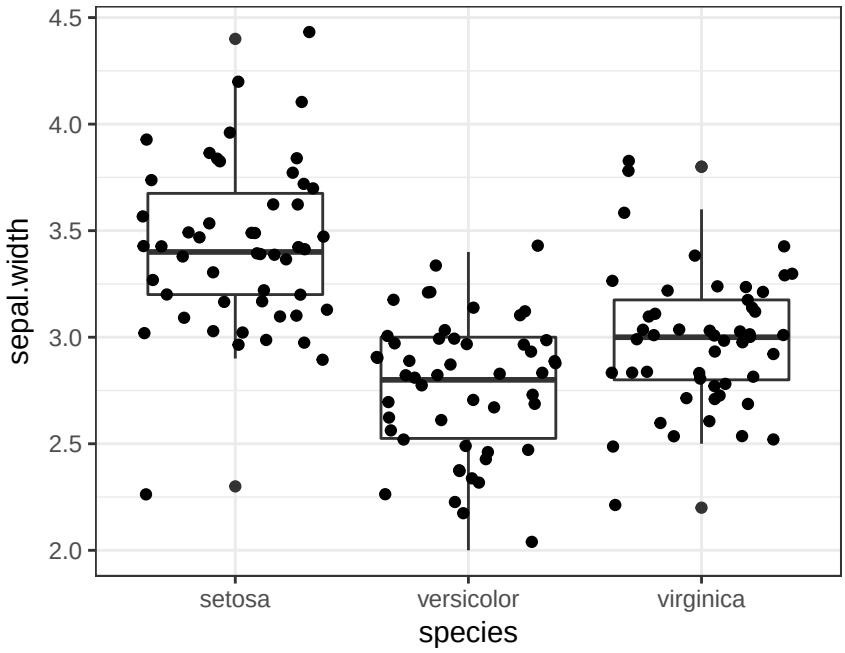
Note what happens when I put the `size` argument outside of `aes()`. It applies to all of the observations, it's not mapped to a variable.

What if we want to look at the distribution of these by species?

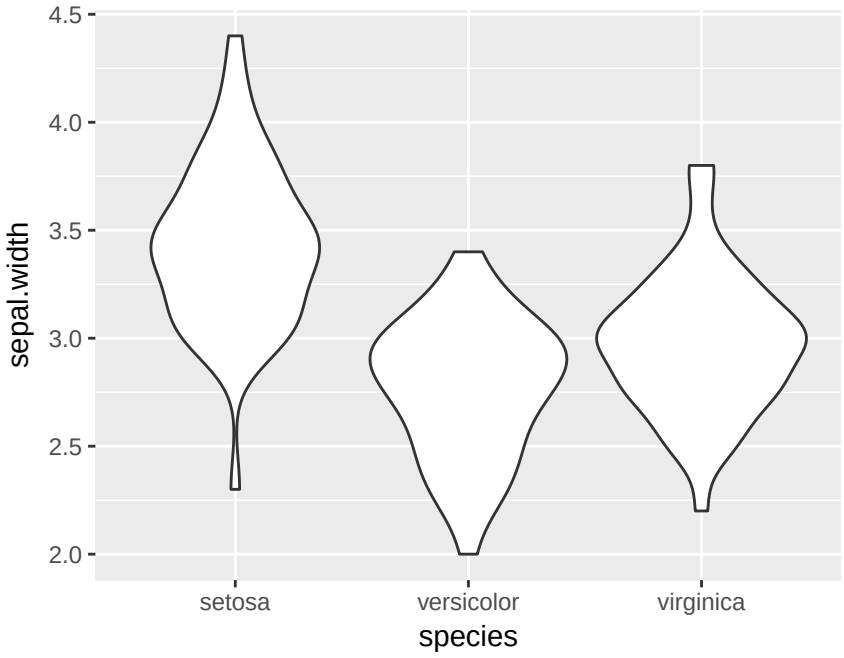
```
plot_2 <- ggplot(iris, aes(species, sepal.width)) + geom_boxplot()
print(plot_2)
```



```
plot_2 + theme_bw() + geom_jitter()
```

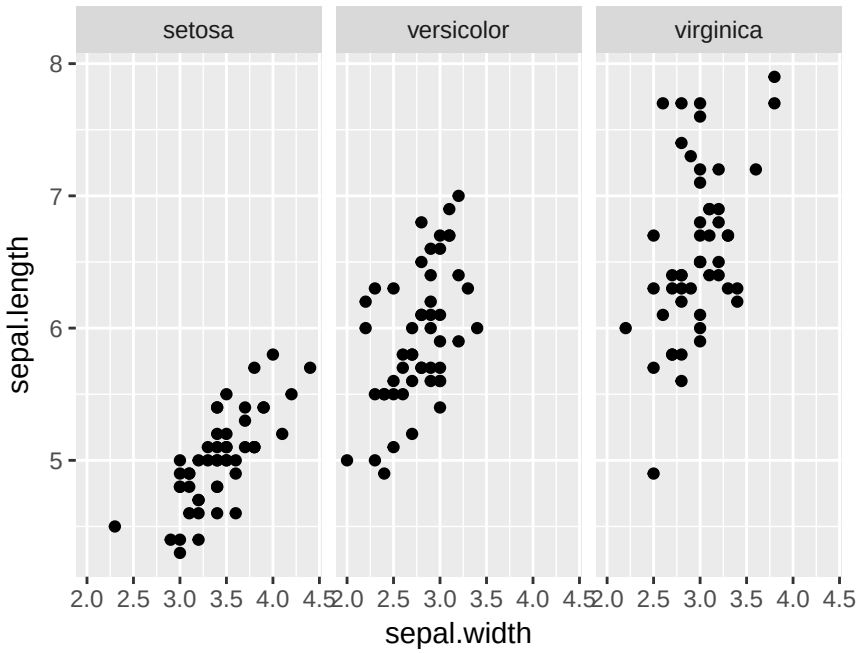



```
ggplot(iris, aes(species, sepal.width)) + geom_violin()
```



What if we want to look at the relationship between two variables, but separate it by each species?

```
ggplot(data = iris, mapping = aes(sepal.width, sepal.length)) +
  geom_point() +
  facet_grid(~species)
```



This is just scraping the surface of ggplot2, you go from idea to visualization with just a few lines of code. It's extremely effective for exploratory data visualization. To get to a publication level plot, it's debatable which is better: ggplot2 or base graphics. In the end of the day, whichever your prefer!

There many options for visualizing your data with ggplot2. This website is amazing for exploring the different types!