R: An Introduction to Basic Statistics, ggplot2, and the tidyverse

Matt C. McFee

Institute of Biomaterials and Biomedical Engineering

April 8, 2020

What is R and what will you learn today?

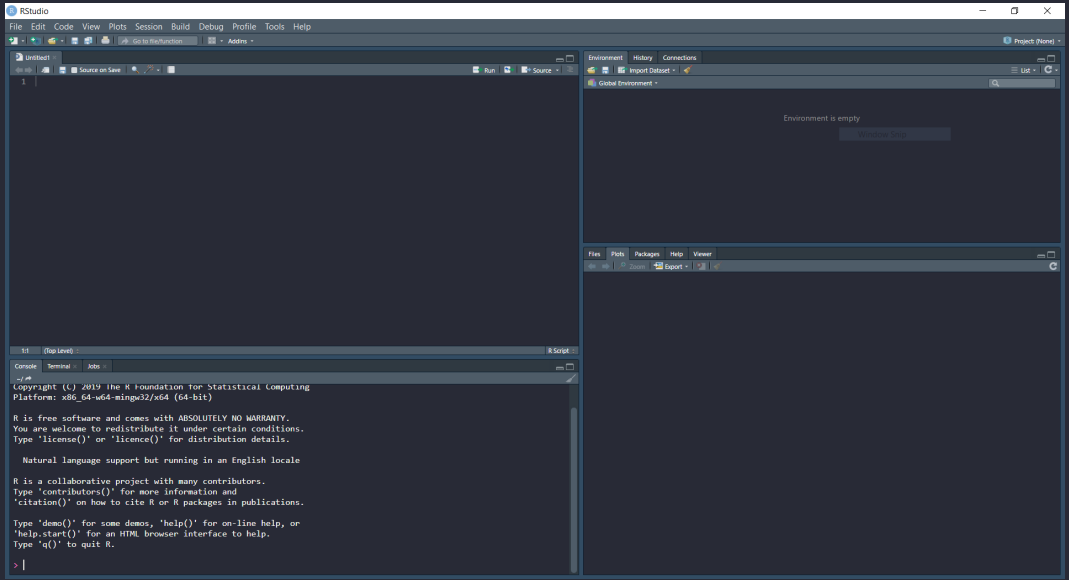
- A statistical programming language that’s great for statistics and data science.
- You will learn how to install pre-written packages in R to solve specialized tasks.
- You will learn how to do t-tests, ANOVA, and how to generate some nice figures.
- You will learn how to do basic data manipulations.

Why use R?

- It is free, and widely used by statisticians and data scientists.
- It is incredibly extensible (someone has already made a package for the analysis you want to do).
- It is open source and deployable on pretty much any computer system out there (no more "mysterious" copies of GraphPad Prism on your USB drives).
- Programming is fun, and an essential skill nowadays.

Note: You are also all incredibly smart, so I'm 100% sure you can all pick this up in a few weeks.

The RStudio user interface



Installing packages in R

Once you've found a package that will allow you to do what you need to do, it can be installed by issuing commands of the form below.

```
install.packages('tidyverse')
install.packages('ggplot2')
install.packages('ggpubr')
install.packages('your_package_name')
```

This might fail due to issues with software version differences. You can install the latest copy of the package off of Github and this should resolve the issue.

```
install_github('kassambara/ggpubr')
```

How do we load our newly installed tools?

This is very simple, but every time you restart your R session, or when you are writing a script, you will have to load any packages you need before you start.

```
library(tidyverse)
library(ggplot2)
library(ggpubr)
```

Note: Some pre-existing commands may be replaced with a package command with the same name. R will notify you of this when you load the package.

How do I get my data into R?

R can handle a wide variety of data files such as Excel, csv, Stata files etc.
I'll show you two commands that will allow you to read Excel and csv files.

```
data <- read_csv('/home/matt/example.csv')
data <- read_excel('/home/matt/example.xlsx')
data <- read_csv(file.choose())
```

Whatever file type you have, I guarantee R can handle it.
Issuing the follow command in R should show you a print out of your file in the R console

```
data
```

What about more complicated scenarios?

- Try googling "statistical test name here" + "R."
- Use the built in tool tips in RStudio.
- Use the R help() command.

```
help('read_csv')
```

The t-test in R

Suppose you have two columns of data in your newly imported data. Column "y" is the dependent variable, and "x" is the independent variable. The t-test can be run as follows.

```
t_test_result <- t.test(data$x, data$y, var.equal = TRUE)
t_test_unequal_variance <- t.test(data$x, data$y)
t_test_paired <- t.test(data$x, data$y, paired = TRUE)
t_test_one_samp <- t.test(data$y, mu = 0)

summary(t_test_result)
```

Note: The "$" symbol can be interpreted as the user telling R to access the data you imported (remember we named this variable data) and select the specified column.

Analysis of Variance in R

ANOVA is just as easy but the syntax will be a little different. Again, assuming we have a variable containing our imported data named "data" and two columns "x" and "y" we can perform ANOVA as follows.

```
aov.res <- aov(y ~ x, data = data)

summary(aov.res)
```

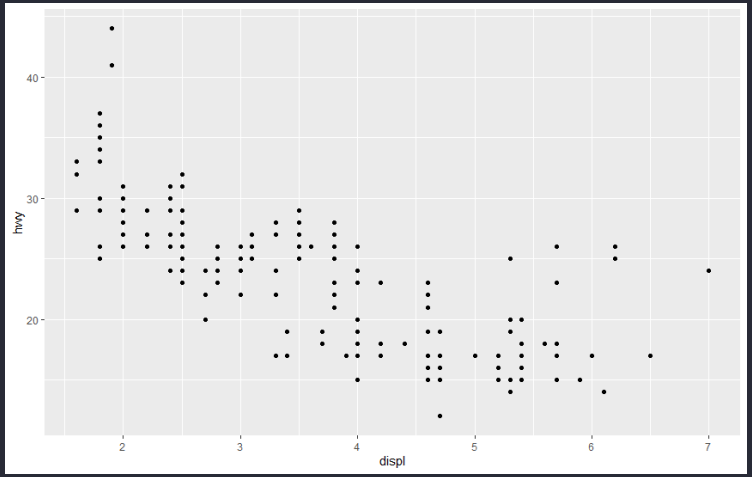
What is ggplot2?

It is a graphical library primarily written by our friend Hadley Wickham, that provides a logical, systematic way to generate beautiful graphics.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point()
```

Tell ggplot where the data is stored, and then map variables to aesthetics. This should become more clear in a second.

Our first graph



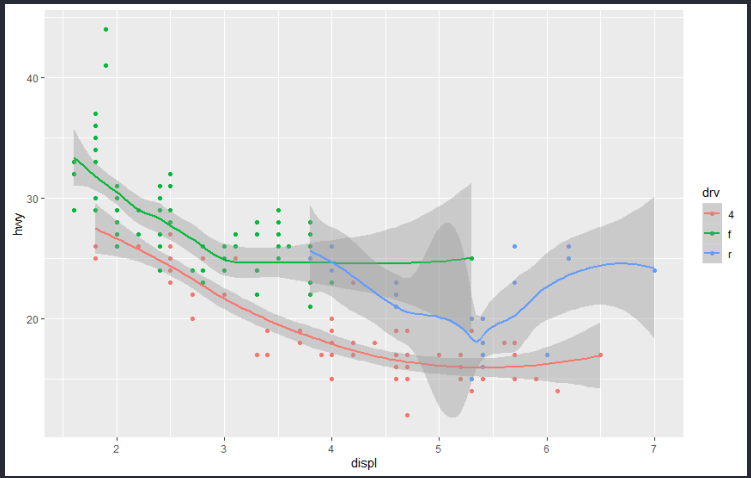
Try this: Remove geom_point() from your code. What happens?

More graphing with ggplot2

Let’s make a few small changes to our previous code.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, colour = drv)) +
geom_point() + geom_smooth()
```

A more complicated example



Try this: Try moving colour = drv outside of aes(). What happens?

Homework: What other objects can you add to a ggplot? How do you think you add labels? Annotations?

The language of graphics!

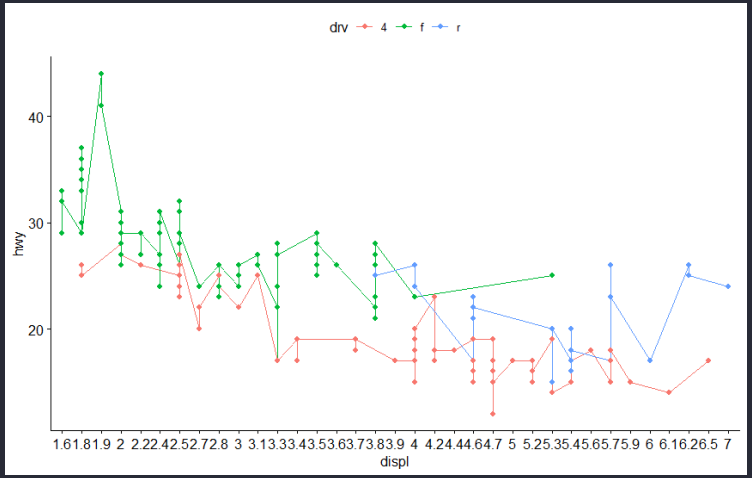
So that’s really the nice language of graphics. First you map your data to aesthetics. For example, map variable1 value to x-coordinate and variable2 value to y-coordinate, or map the colour of the objects to variable3.
You can make incredibly complex graphics this way. Really, the sky is the limit.

A bioinformatician’s package allows you to cheat!

A bioinformatician ended up making a ggplot2 wrapper called "ggpubr" that gives you sane "default" ggplot settings with a single command. Consider the following.

```
graph <- ggline(mpg, x = ’displ’, y = ’hwy’, fill = ’displ’,
xlab = ’displ’, ylab = ’hwy’, color = ’drv’)
```

Our first graph using ggpubr for ggplot2



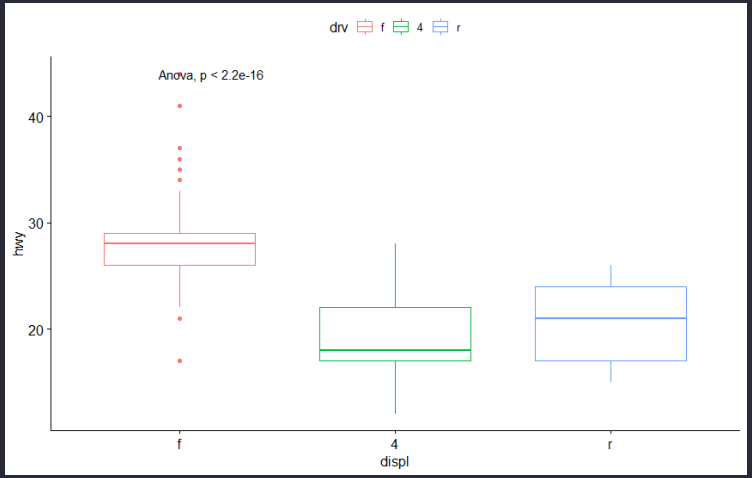
Homework: What other graph types are including in ggpubr?

You also gain some stats shortcuts using ggpubr

You can also perform hypothesis tests and report the results on the graph as below.

```
graph <- ggboxplot(mpg, x = ’drv’, y = ’hwy’, fill = ’displ’,
xlab = ’displ’, ylab = ’hwy’, color = "drv")
+ stat_compare_means(method = ’anova’)
```

Another ggpubr graph



Homework: How do you display comparisons and p-values?

What is this tidyverse you keep mentioning?

The tidyverse is a family of R packages developed by Hadley Wickham (and others) to provide useful tools for manipulating data, doing stats, and visualizing data. For example, ggplot2 is a member of the tidyverse. I suggest you take some time to look into the tidyverse yourself. I'm going to introduce you to two useful commands that you will use often.

The concept of tidy data

This is a way to store your data, that will not only make your life and analysis easier, but it will also make statisticians and data scientists happy. Believe it or not, Statistics majors in Canada take a course dedicated to teaching them how to beg scientists to consult them about Statistics and data.

Data Point #	Var1	Var2	Var3	VarN

The tidyverse’s filter and mutate commands

Filter allows you to select and store a subset of your data based on logical operators.

```
filtered_dat <- filter(mpg, displ > 2, hwy > 20)
```

Mutate allows you to modify existing columns or append new columns to your data that are calculated using existing data columns.

```
mutated_dat <- mutate(filtered_dat, new_val = displ + hwy)
```

Homework: Check out the arrange and add_column functions provided by the tidyverse. They will also come in handy. Finally, look up ways you can handle NA values in your dataframes.

The end of our little journey. What is a pipe?

This section will briefly introduce you to the idea of pipes in R. Essentially, you can take the output of one function and pipe it to another function to act as the new functions input data. A pipe in R is written as %>%.

```
final_dat <- mpg %>% filter(displ > 2, hwy > 20) %>%
mutate(new_val = displ + hwy)
```

As you can see this is much cleaner, and makes your manipulations a logical, ordered process.

Additional resources for your R programming education

Hands-on R Programming: https://rstudio-education.github.io/hopr/
Start here. This is an R Project approved introduction to the language that is totally free.
R for Data Science: https://r4ds.had.co.nz/
This was written by a huge contributor to the R programming language. Again, this book is 100% free of charge.