A (Very) Short Introduction to R for Wet Lab Scientists

Kieran O'Neill

What is R, and Why Should I Use It?

What is R?

R is a versatile, open source programming language that was specifically designed for data analysis. As such R is extremely useful both for statistics and data science. Inspired by the programming language S.

- Open source software under GPL.
- Superior (if not just comparable) to commercial alternatives. R has over 5,000 user contributed packages at this time. It's widely used both in academia and industry.
- Available on all platforms.
- Large and growing community of peers.
- Bioconductor: largest (and arguably the best) free collection of software for biological data analysis anywhere.

Why Not Just Use Excel, FlowJo, GraphPad, etc?

1. Reproducibility
   - Its really important that you know what you did.
   - More journals/grants/etc. are also requiring this.
   - The best way to know what you did is to provide all the code.
   - GUI software makes this difficult
   - If you keep a lab notebook, why not do the same thing with your analysis?
2. Flexibility, capabilities and pretty pictures
   - R can handle much larger data sets, much faster, and much more easily than Excel.
   - Huge range of statistical tests, biological data types, etc.
   - Plotting in R is far more sophisticated than any available GUI.

Proof of What I Mean By Pretty Pictures:

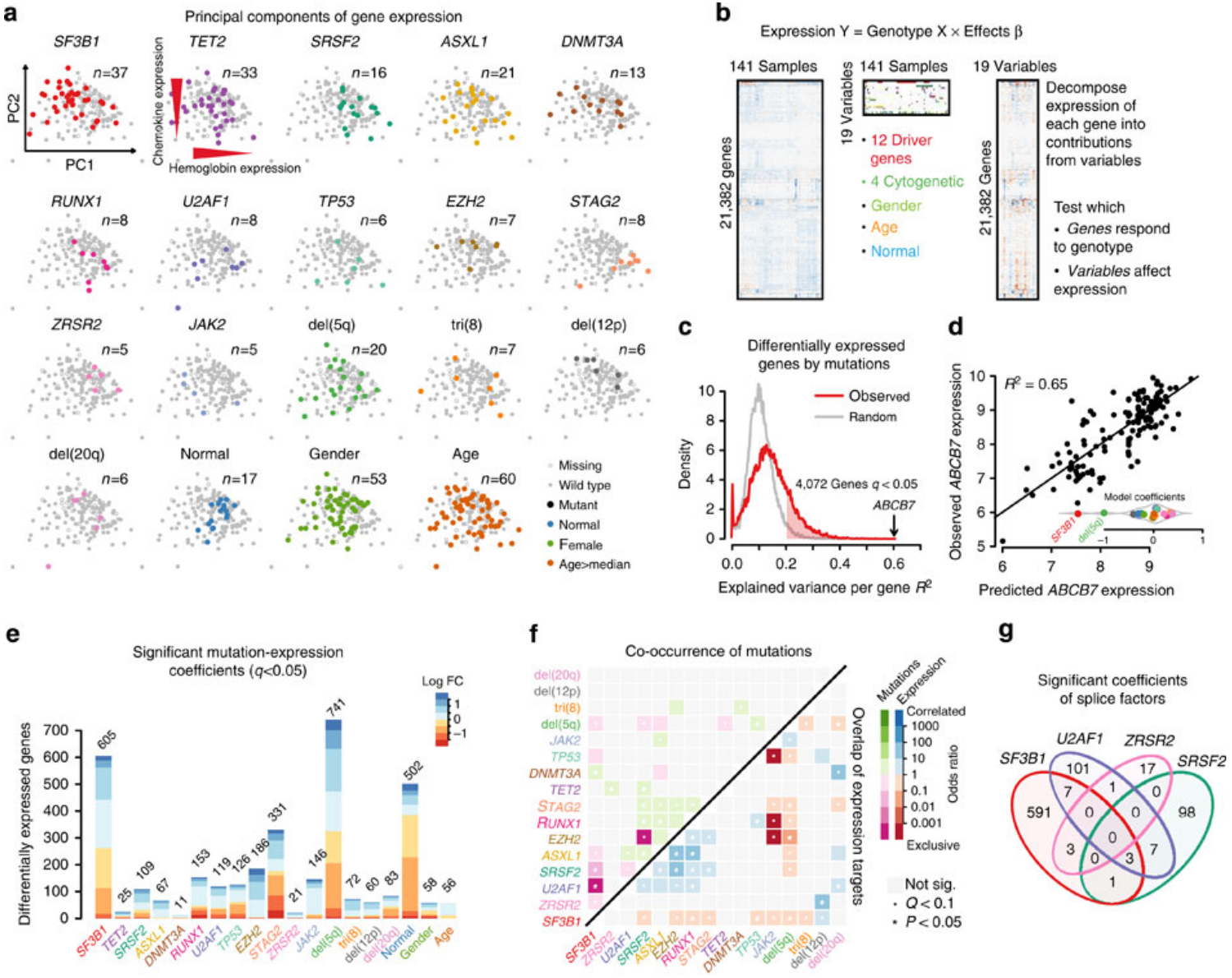


Figure 1: Gerstung et al (2015) Nature Communications (CC-BY)

RStudio

Set up a new project

- Click ‘file’, then ‘New project’
- Click ‘New directory’m then ’Empy Project’, then pick a directory
- With the project set up, click ‘file’, then ‘new’ (or ctrl+shift+n)
- Click ‘File’, ‘Save’ (ctrl+s)
- Save the file as something meaningful, like `lecture1_examples.R`

Note: for Mac users, where I say ‘ctrl’, use your weird Mac control key instead.

Quick overview of RStudio

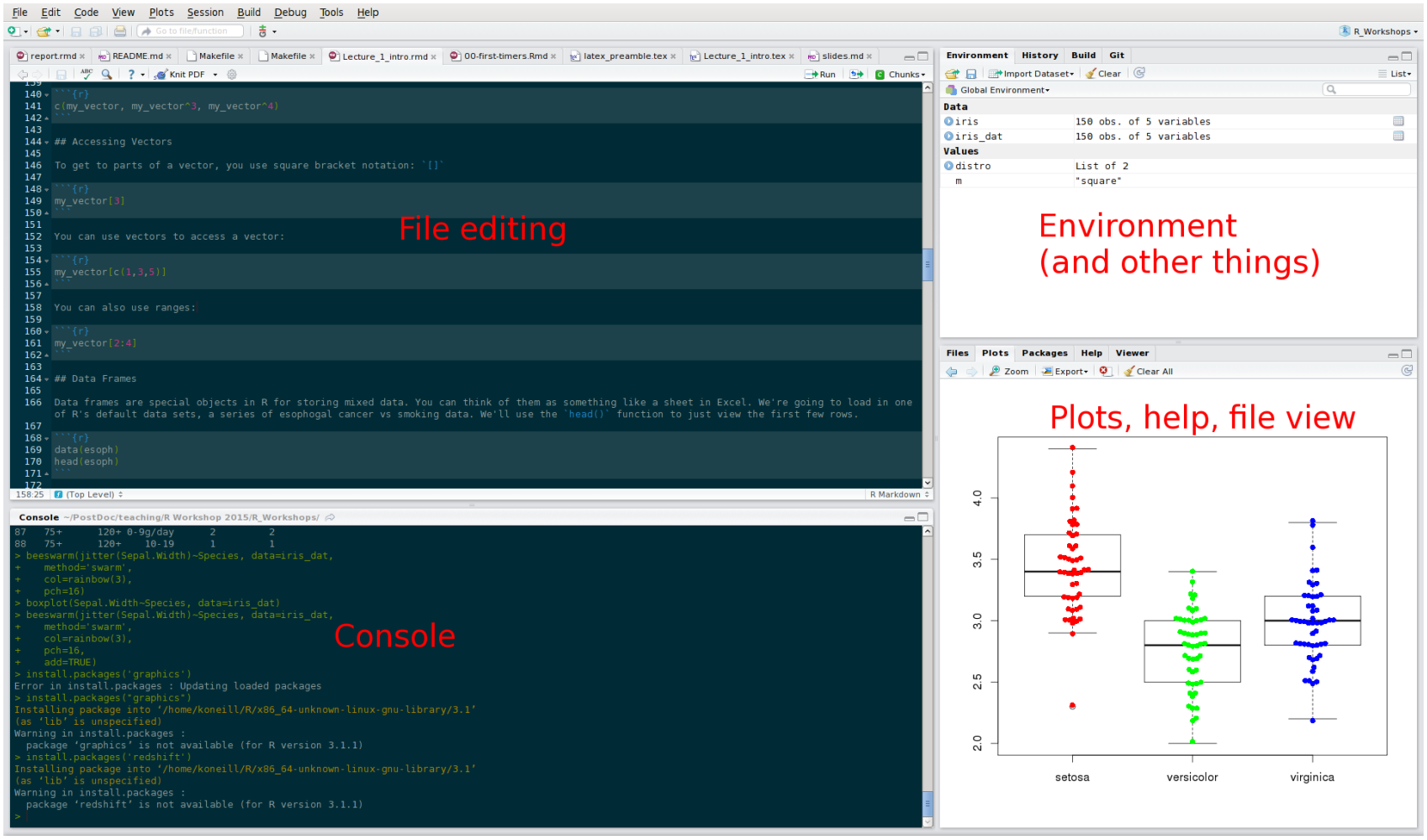


Figure 2: RStudio Interface

Working between the script and console

*Type the following into the console, and press enter:

```
print("Hello")
```

```
## [1] "Hello"
```

- Now type it into the file window, and with the cursor on that line, press ctrl+enter
- Messing around in the console is fun.
- But it's better to keep your work in a file which you save often.

R Commands and Objects

Objects

You can assign values to objects:

```
some_number <- 5
some_number + 3
```

```
## [1] 8
```

```
some_other_number <- some_number ^ 3
some_other_number
```

```
## [1] 125
```

Take a look in your environment window in RStudio. You can also see what objects are defined using the `ls()` command:

```
ls()
```

```
## [1] "some_number"       "some_other_number"
```

Basic Data Types

You can find out the type of an object using `typeof()`:

```
typeof(some_number)
```

```
## [1] "double"
```

```
some_text <- "5"
typeof(some_text)
```

```
## [1] "character"
```

Numeric vs Character

```
some_number + 5
```

```
## [1] 10
```

```
some_text + 5   # This would give an error -- try it.
```

```
as.numeric(some_text) + 5
```

```
## [1] 10
```

Vectors

Vectors are one-dimensional objects. You can create them with the `c()` function:

```
my_vector <- c(1,3,5,6,7,8)
```

You can apply operations to a whole vector.

```
my_vector^3
```

```
## [1]   1  27 125 216 343 512
```

You can join vectors with `c()`:

```
c(my_vector, my_vector^3, my_vector^4)
```

```
##  [1]    1    3    5    6    7    8    1   27  125  216  343  512    1   81
## [15]  625 1296 2401 4096
```

Accessing Vectors

To get to parts of a vector, you use square bracket notation: []

```
my_vector[3]
```

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

You can use vectors to access a vector:

```
my_vector[c(1,3,5)]
```

```
## [1] 1 5 7
```

You can also use ranges:

```
my_vector[2:4]
```

```
## [1] 3 5 6
```

Data Frames

Data frames are special objects in R for storing mixed data. You can think of them as something like a sheet in Excel. We're going to load in one of R's default data sets, a series of esophogal cancer vs smoking data. We'll use the `head()` function to just view the first few rows.

```
data(esoph)
head(esoph)
```

```
##   agegp     alcgp    tobgp ncases ncontrols
## 1 25-34 0-39g/day 0-9g/day      0        40
## 2 25-34 0-39g/day    10-19      0        10
## 3 25-34 0-39g/day    20-29      0         6
## 4 25-34 0-39g/day      30+      0         5
## 5 25-34     40-79 0-9g/day      0        27
## 6 25-34     40-79    10-19      0         7
```

Also try clicking on `esoph` in the Environment window in RStudio.

Working With Data Frames:

You can access columns in a data frame using $, or rows, columns, or individual values using []

```
head(esoph$agegp)      # column
```

```
## [1] 25-34 25-34 25-34 25-34 25-34 25-34
## Levels: 25-34 < 35-44 < 45-54 < 55-64 < 65-74 < 75+
```

```
head(esoph[,'agegp']) # column using []
```

```
## [1] 25-34 25-34 25-34 25-34 25-34 25-34
## Levels: 25-34 < 35-44 < 45-54 < 55-64 < 65-74 < 75+
```

Working With Data Frames (ctd):

```
esoph[2,]              # row
```

```
##   agegp     alcgp tobgp ncases ncontrols
## 2 25-34 0-39g/day 10-19      0        10
```

```
esoph[2,'agegp']       # Single value
```

```
## [1] 25-34
## Levels: 25-34 < 35-44 < 45-54 < 55-64 < 65-74 < 75+
```

```
esoph[2,1]             # Single value using numbers
```

```
## [1] 25-34
## Levels: 25-34 < 35-44 < 45-54 < 55-64 < 65-74 < 75+
```

Aother Useful Function: Summary()

```
summary(esoph[,3:5])
```

```
##        tobgp        ncases          ncontrols
##  0-9g/day:24   Min.   : 0.000   Min.   : 1.00
##  10-19   :24   1st Qu.: 0.000   1st Qu.: 3.00
##  20-29   :20   Median : 1.000   Median : 6.00
##  30+     :20   Mean   : 2.273   Mean   :11.08
##                3rd Qu.: 4.000   3rd Qu.:14.00
##                Max.   :17.000   Max.   :60.00
```

Installing Add-on Packages

CRAN

- Most of R's power comes from free third-party add-ons
- CRAN is the Comprehensive R Archive Network
- It is the main repository for R packages
- You can install packages like so:

```
install.packages('beeswarm')
```

When you start a new session, you can then load a package using `library`:

```
library('beeswarm')
```

Bioconductor

- Bioconductor is a big part of what makes R awesome for biologists.
- Bioconductor is a repository specifically for (molecular) biology R packages.
- It has very stringent rules for those packages regarding documentation, examples and code quality.
- There are packages to handle a vast range of data, from BAM files to microarrays to flow cytometry and many more.

Check it out at www.bioconductor.org

To install Bioconductor packages (note: don't run this now, it can take ten minutes or more the first time):

```
source("http://bioconductor.org/biocLite.R")
biocLite('flowCore') #Or whatever the package is called.
```

Loading in Files

The Iris Data Set

Loading in CSV

Loading in from Excel ?

Basic Statistical Tests

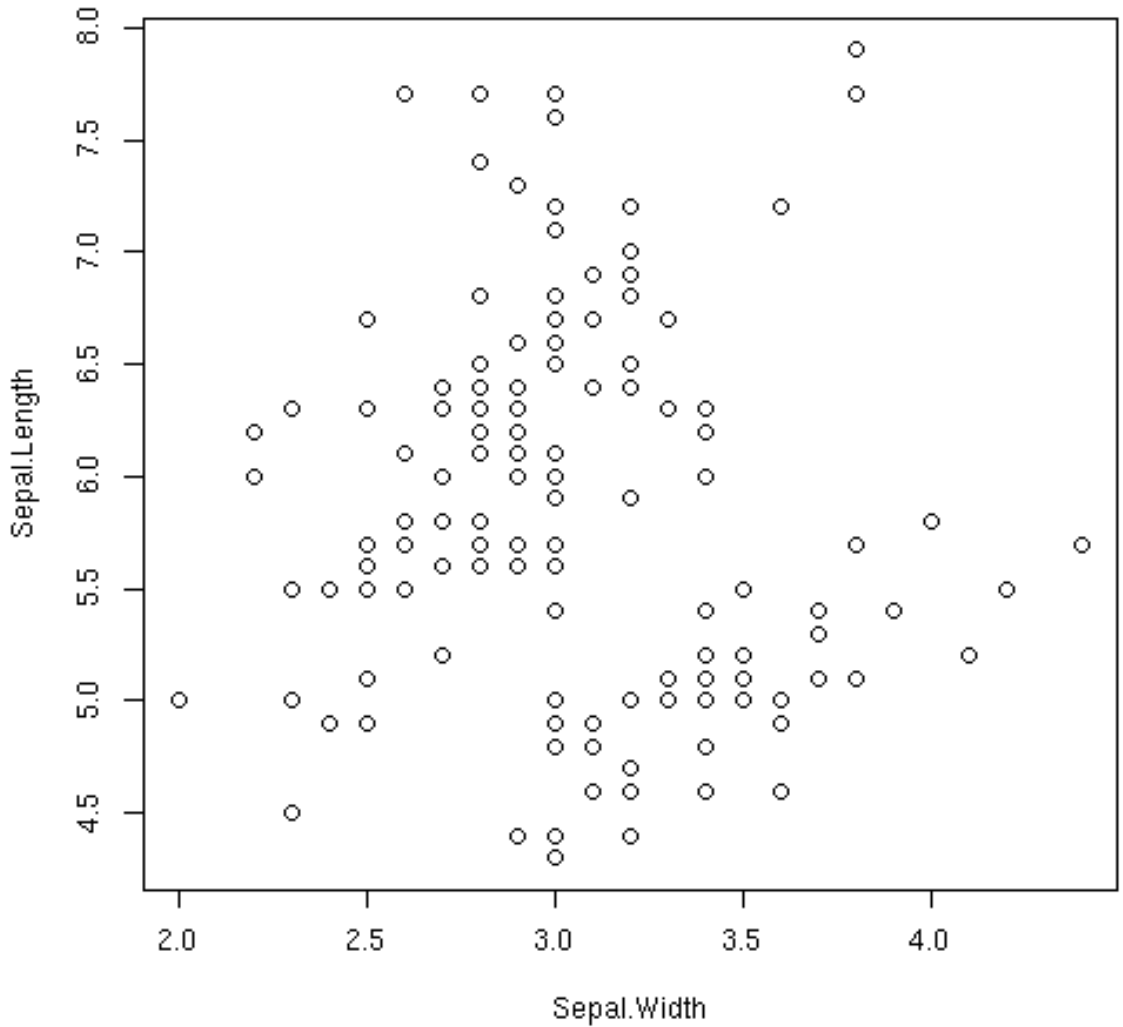
Student's T Test

Examining the Results

Basic Plots

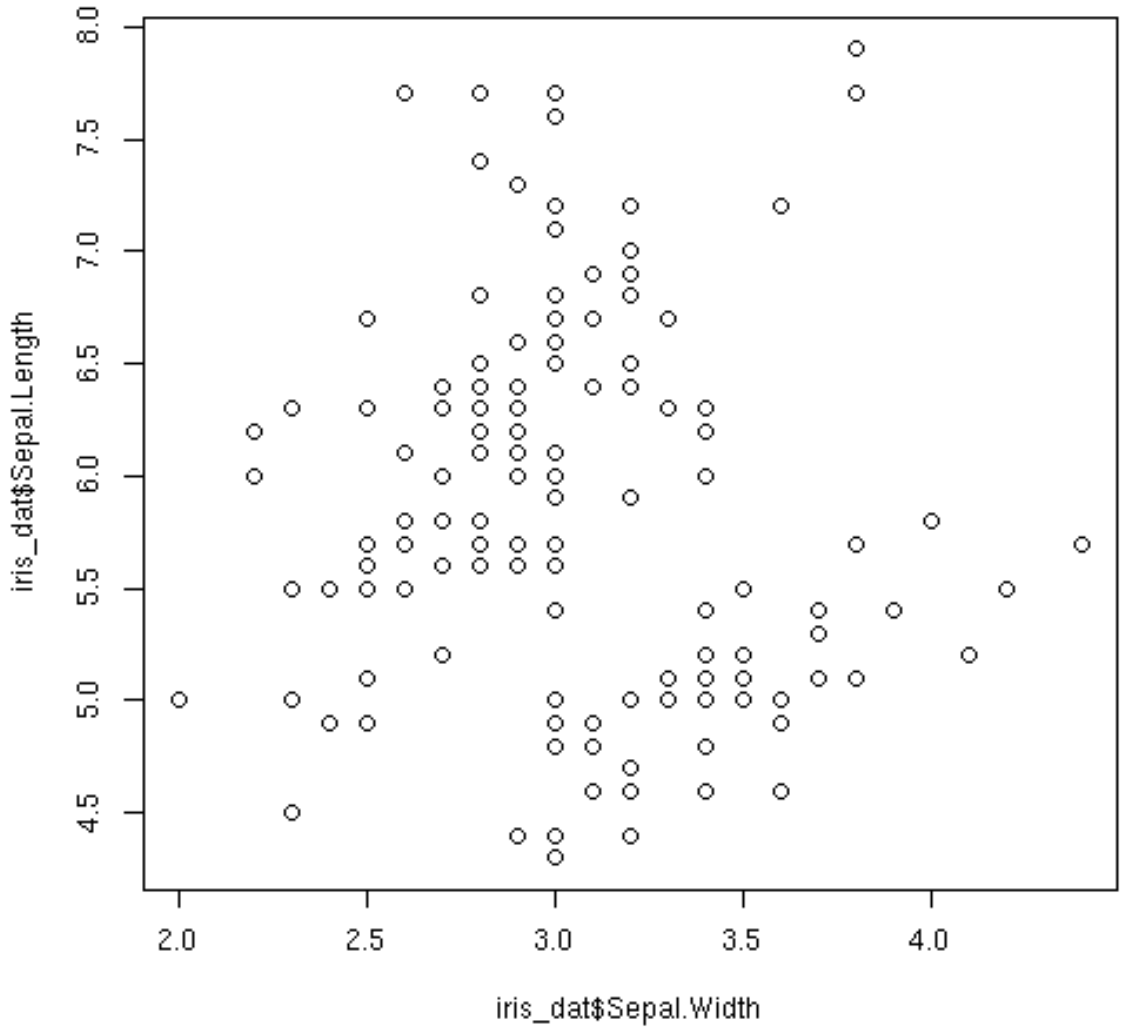
Scatter Plot

```
data(iris)
iris_dat <- iris
plot(Sepal.Length~Sepal.Width, data=iris_dat)
```



Scatter Plot, Alternate Way of Calling

```
plot(iris_dat$Sepal.Width, iris_dat$Sepal.Length)
```



Scatter Plot, With Some Options

```
plot(iris_dat$Sepal.Width, iris_dat$Sepal.Length,
    pch=16,
    col=iris$Species,
    main='Sepal Length vs Sepal Width',
    xlab='Length',
    ylab='Width')
```

Scatter Plot, With Some Options

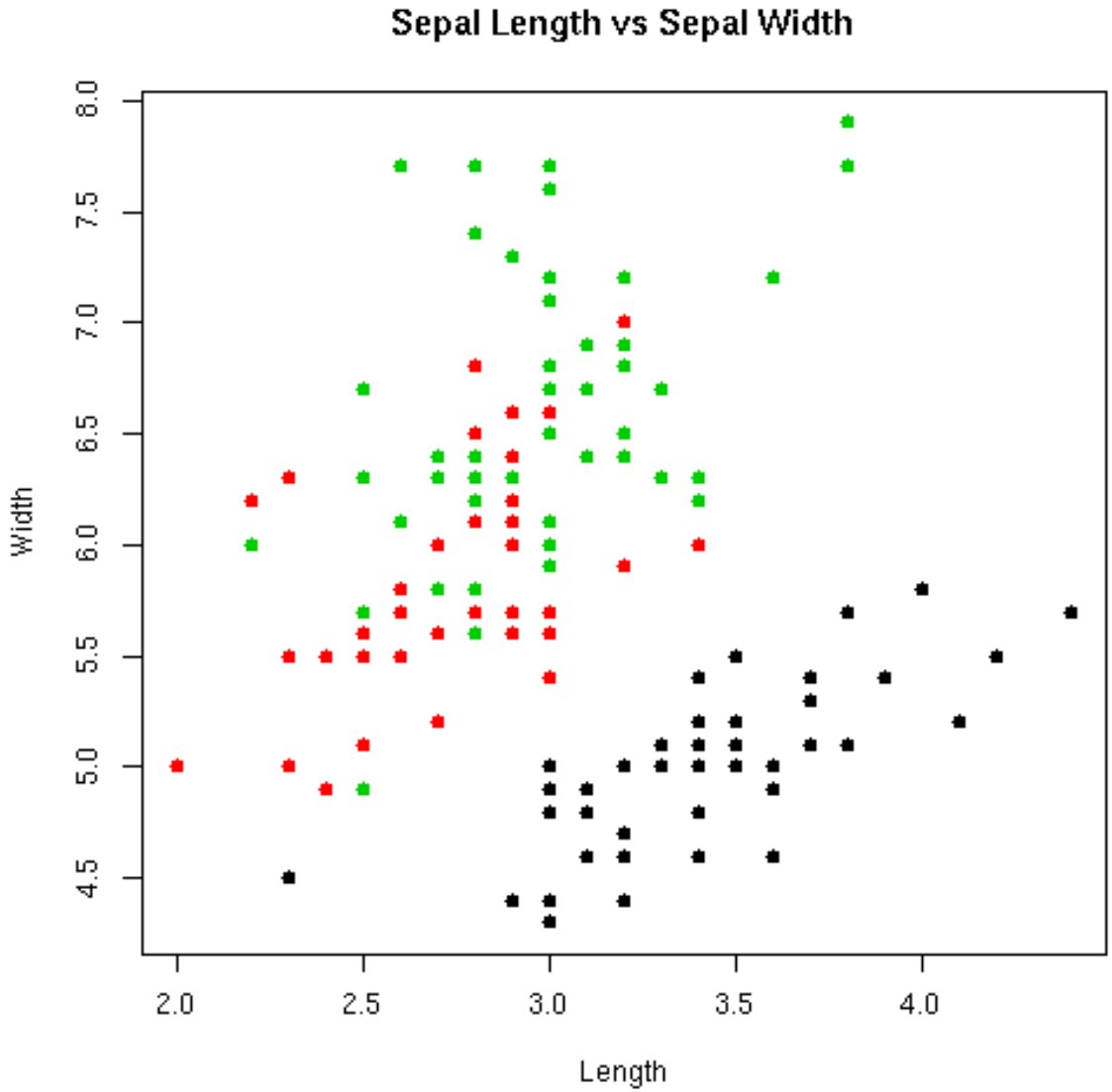


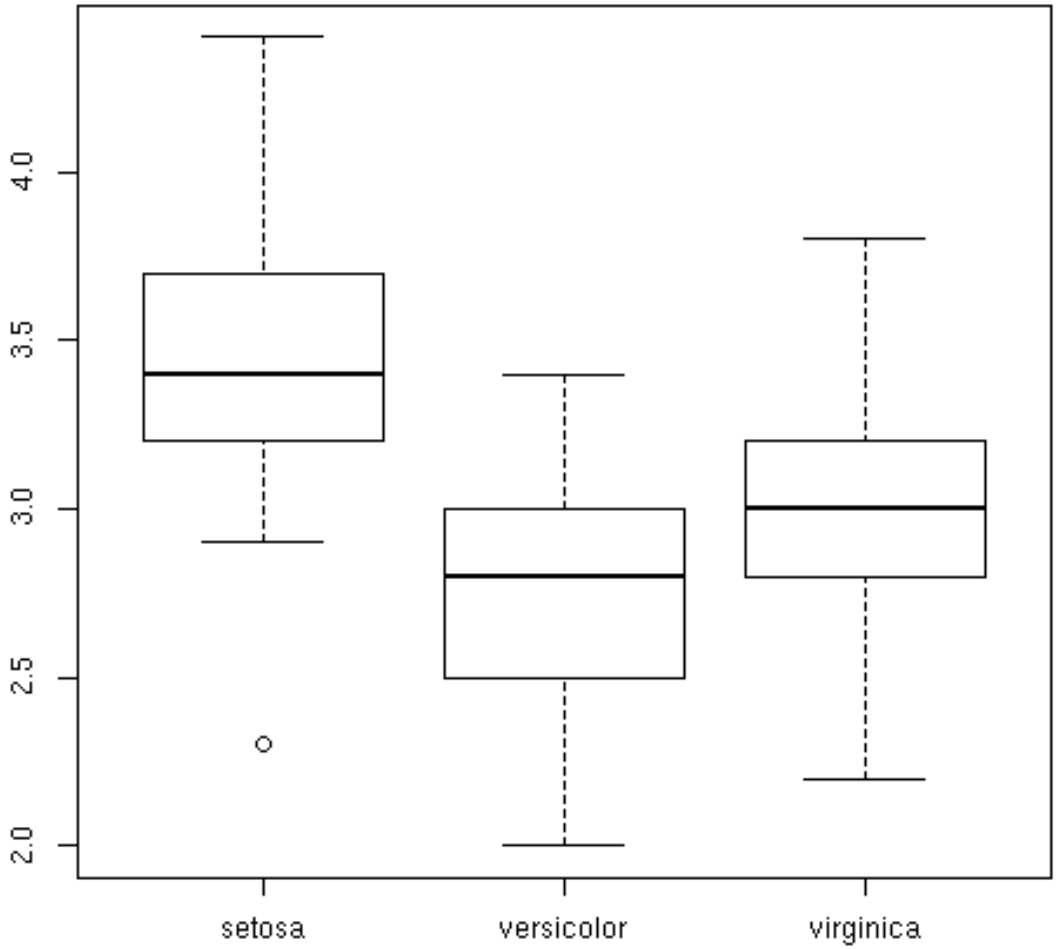
Figure 5: plot of chunk unnamed-chunk-25

Dynamite Plots

- A lot of papers use bar plots with error bars to show data with multiple measurements per treatment.
- These have a lot of shortcomings: data being hidden, assumptions about the confidence intervals used, and wasted ink.
- Unsurprisingly, R does not have an way to to these.
- Instead, R does allow box plots, which are much better.
- There is also a package for beeswarm plots.

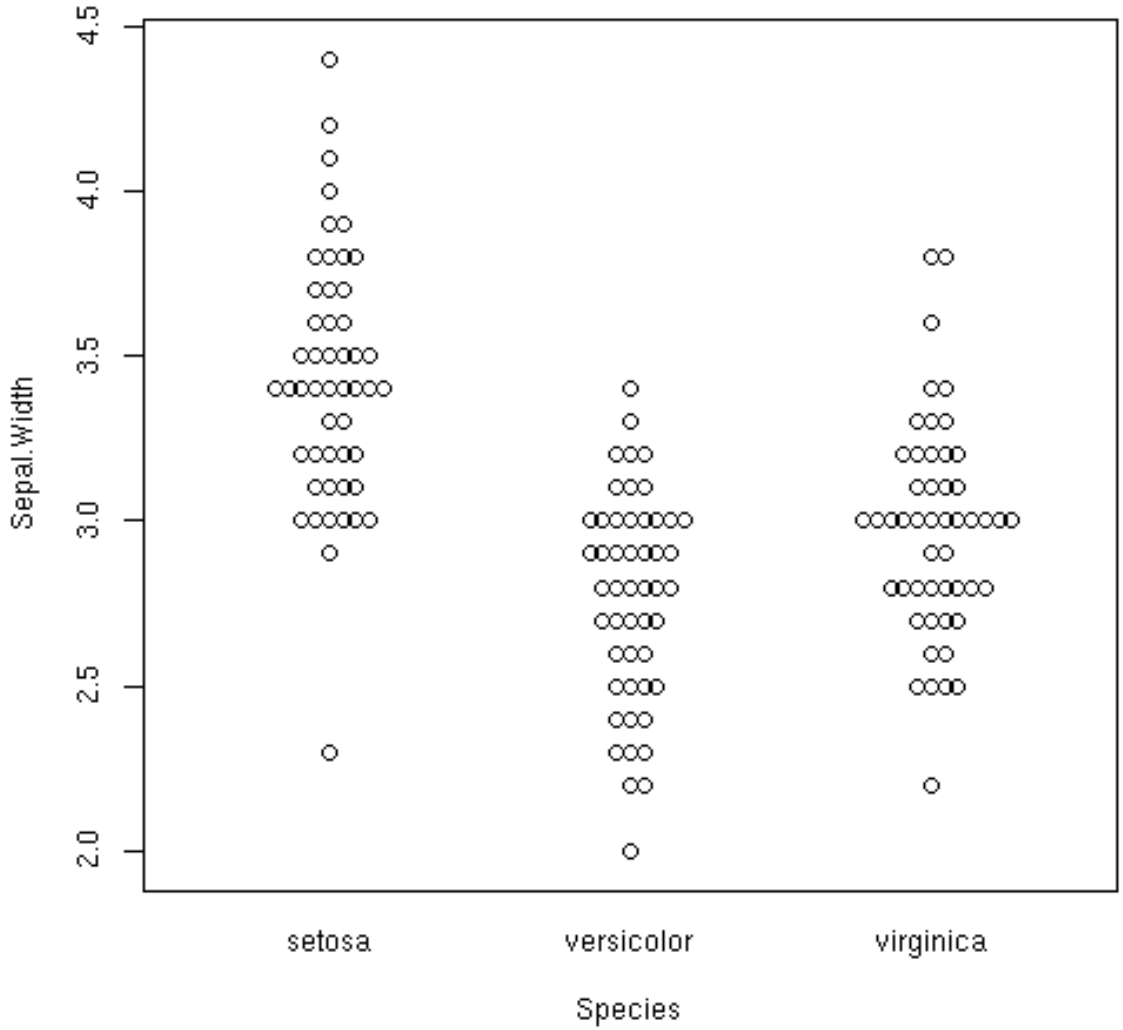
Box Plots

```
boxplot(Sepal.Width~Species, data=iris_dat)
```



Beeswarm Plots

```
library(beeswarm)
beeswarm(Sepal.Width~Species, data=iris_dat)
```



Beeswarm Plots With More Options

```
beeswarm(jitter(Sepal.Width)~Species, data=iris_dat,
         method='swarm',
         col=rainbow(3),
         pch=16)
```

Beeswarm Plots With More Options

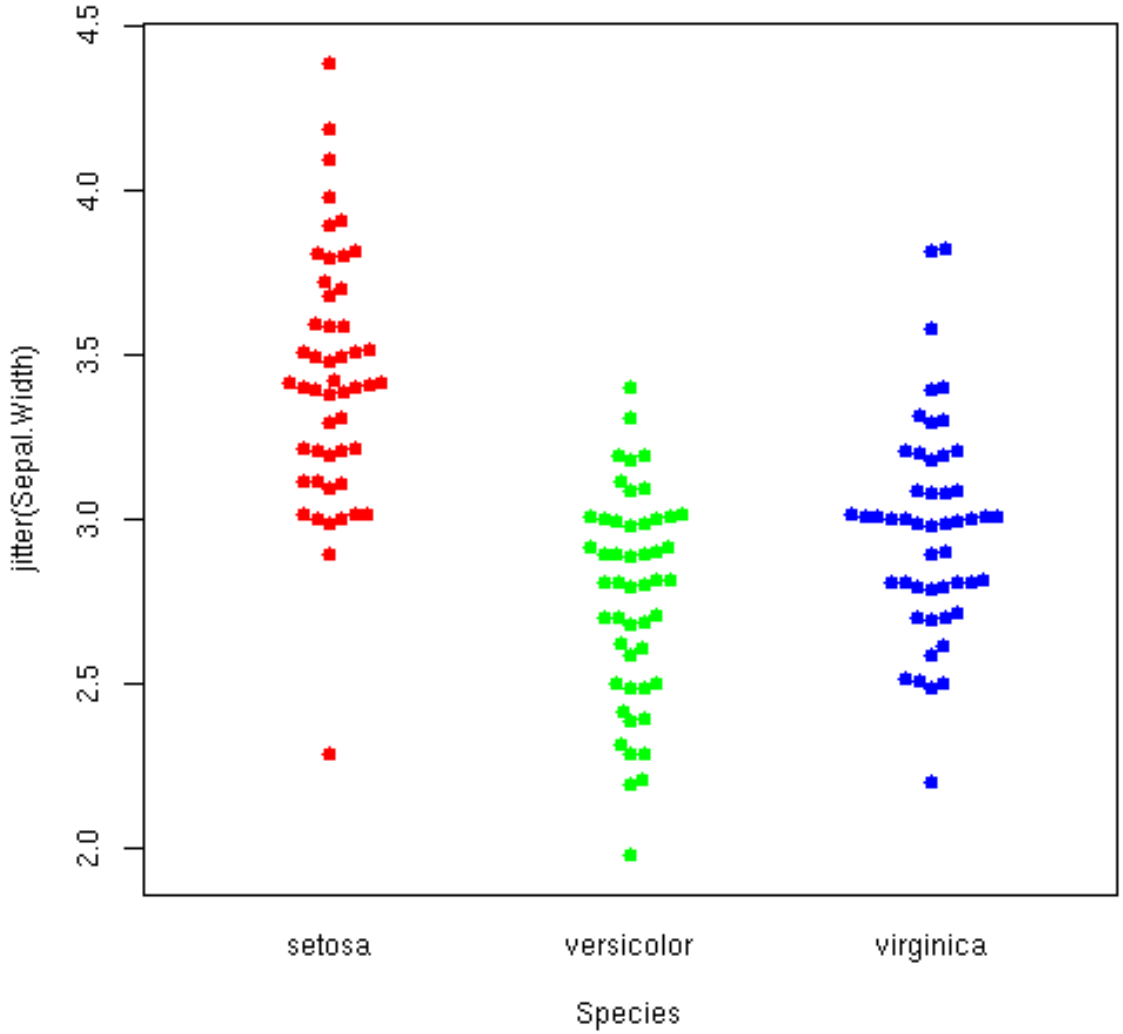


Figure 8: plot of chunk unnamed-chunk-29

Beeswarm and Boxplots Combined

```
boxplot(Sepal.Width~Species, data=iris_dat)
beeswarm(jitter(Sepal.Width)~Species, data=iris_dat,
         method='swarm',
         col=rainbow(3),
         pch=16,
         add=TRUE)
```

Beeswarm and Boxplots Combined

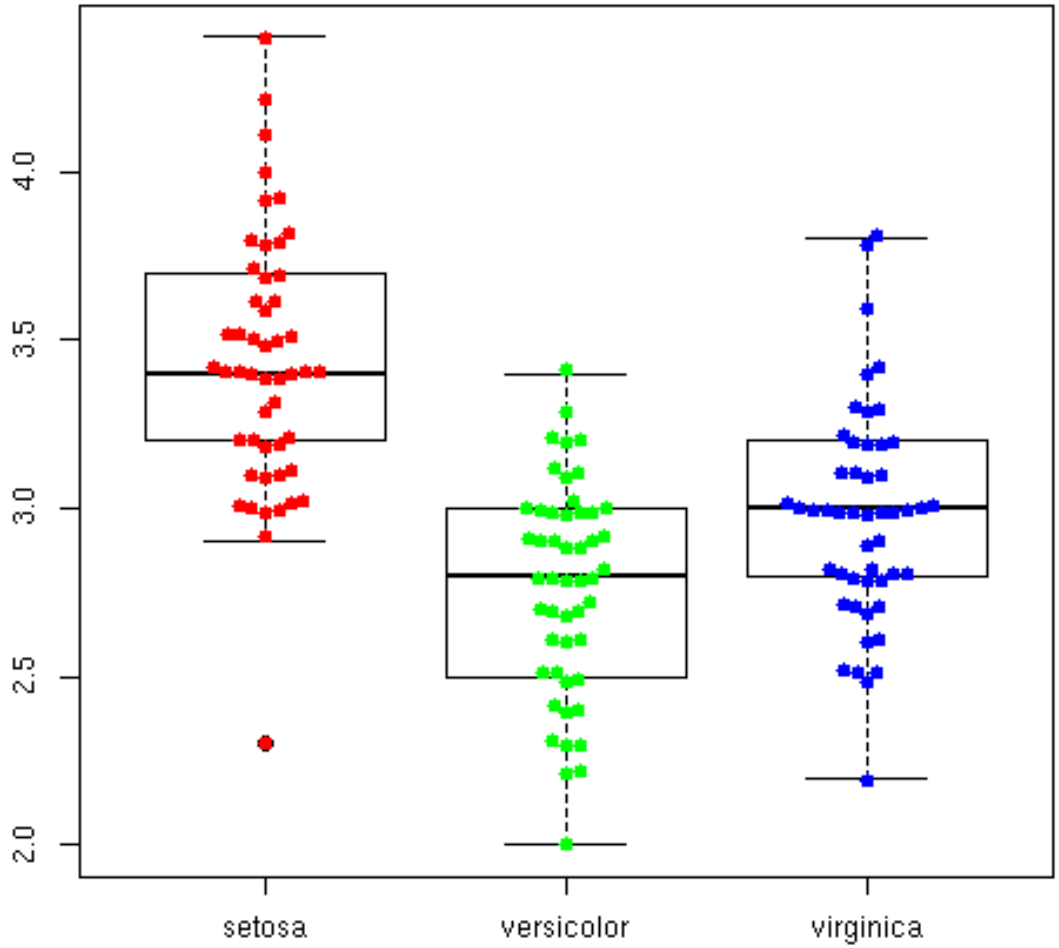


Figure 9: plot of chunk unnamed-chunk-31

Other Plotting Packages

Credits

This Workshop Brought to You By

Course Developers:

- Kieran O'Neill
- Eva Yap
- Alice Zhu (for next session)

Starting Material:

Much material was reused from Software Carpentry's Bootcamp workshops and from Andy Teucher's short R course, both under the terms of the Creative Commons Attribution License.

Pizza and Logistics:

- GraSPoDS (especially Eva Yap and Jessica Pilsworth)