VROOOOM

GETTING STARTED WITH R

The Quick Basics to Get you Going!

Ryan W. Benz • Southern California Power Platform Group • 2020-02-20

GOALS OF THIS MINI-WORKSHOP

- Understand **what R is** and why **it's a great programming language for working with data**
- Learn the basics of **working with R in RStudio**
- Learn the essential fundamentals
  - How to **get data into R**
  - How to **manipulate and transform data**
  - How to **make basic plots and visuals**
- *Note*: becoming proficient with R will take lots of practice, can be frustrating, but keep at it — it's worth it! 👍🏻👍🏻

A FEW NOTES ABOUT R

- R has been around for quite a while (20+ years), based on the S programming language
- R is a great language for working with data
- R is one of the top languages for data science
- R has really grown in the last 5+ years
  support, popularity, community, …

R IS POPULAR (THIS IS INTERESTING…)

TIOBE Index (https://www.tiobe.com/tiobe-index/)

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%
5	7	^	C#	5.927%	+3.08%
6	5	v	Visual Basic .NET	5.862%	-1.23%
7	6	v	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%
10	20	^^	Swift	1.460%	+0.54%
11	18	^^	Go	1.131%	+0.17%
12	11	v	Assembly language	1.111%	-0.27%
13	**15**	^	**R**	**1.005%**	**-0.04%**

R IS QUIRKY (TO START WITH)

- People with a traditional programming background might find R has a higher learning curve
- R has some strange syntax/operators:
  Uncommon use of `.` (dot), `<-`, `$`, `%>%`
- R often tries to be "helpful" (i.e. simplifying results when possible),
  can cause inconsistent results if you're not careful
- R has a factor data type, which makes sense for data analysis work, but can trip up more traditional programmers

…BUT R IS REALLY GREAT FOR DATA

- R was built by statisticians for statisticians (or data scientists)
- Development of R and its ecosystem is focused on data
- R is recognized as a key tool for data science and is integrated into lots of other tools now (MS tools for example)
- R has an amazing community that's supportive and inclusive

Part 0: Getting Started

FIRST THINGS FIRST… INSTALL R

1. Install R for your platform

https://cloud.r-project.org

2. Install dev tools

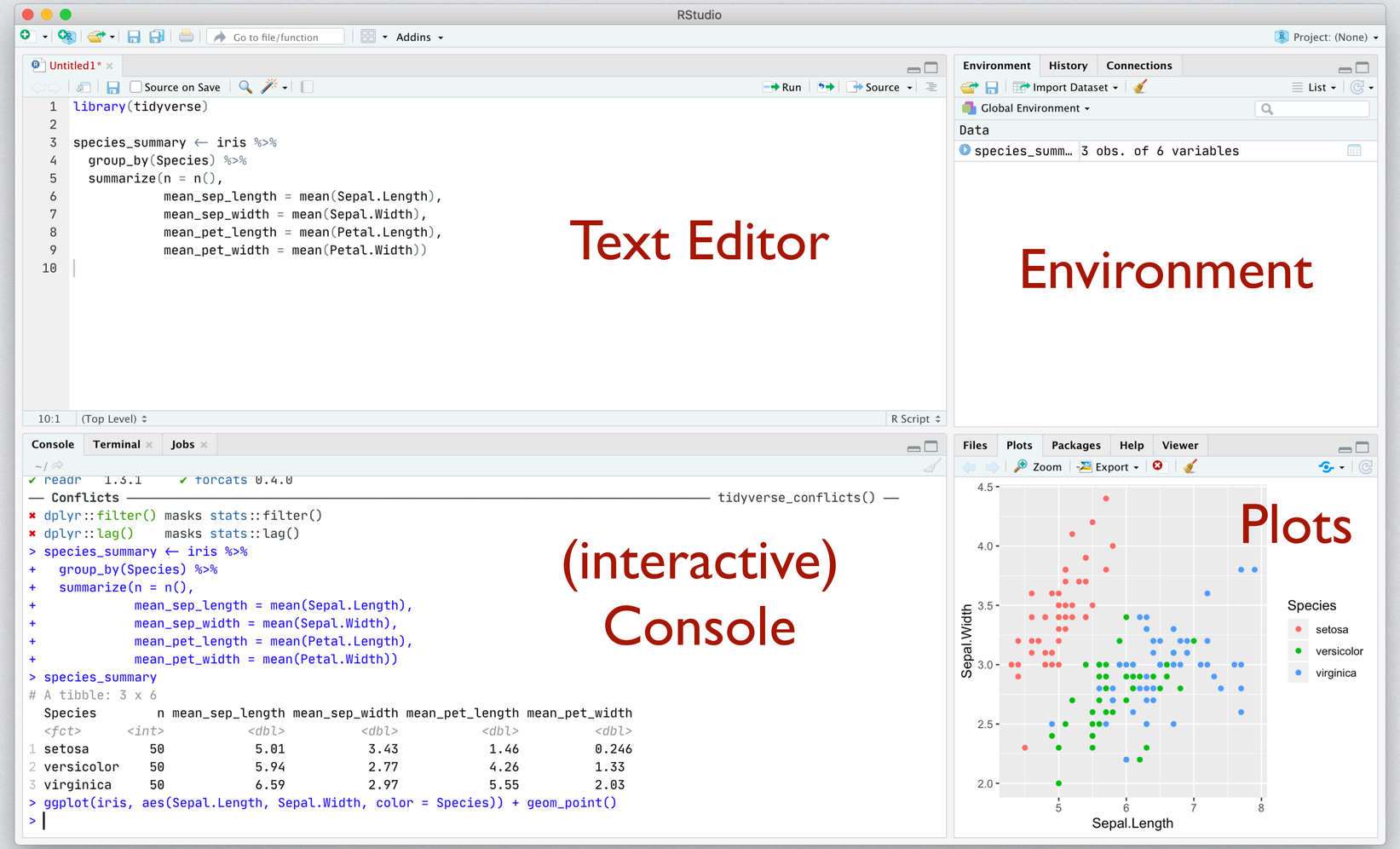
Windows: Rtools (same place as the base R download)

Mac: Developer Tools (from the Mac App Store)

3. Install the RStudio IDE

https://rstudio.com

RSTUDIO IS A REALLY GREAT (AND FREE) IDE FOR R



LIVE CODING EXAMPLE

(WORKING WITH RSTUDIO)

A FEW THINGS TO KNOW ABOUT R

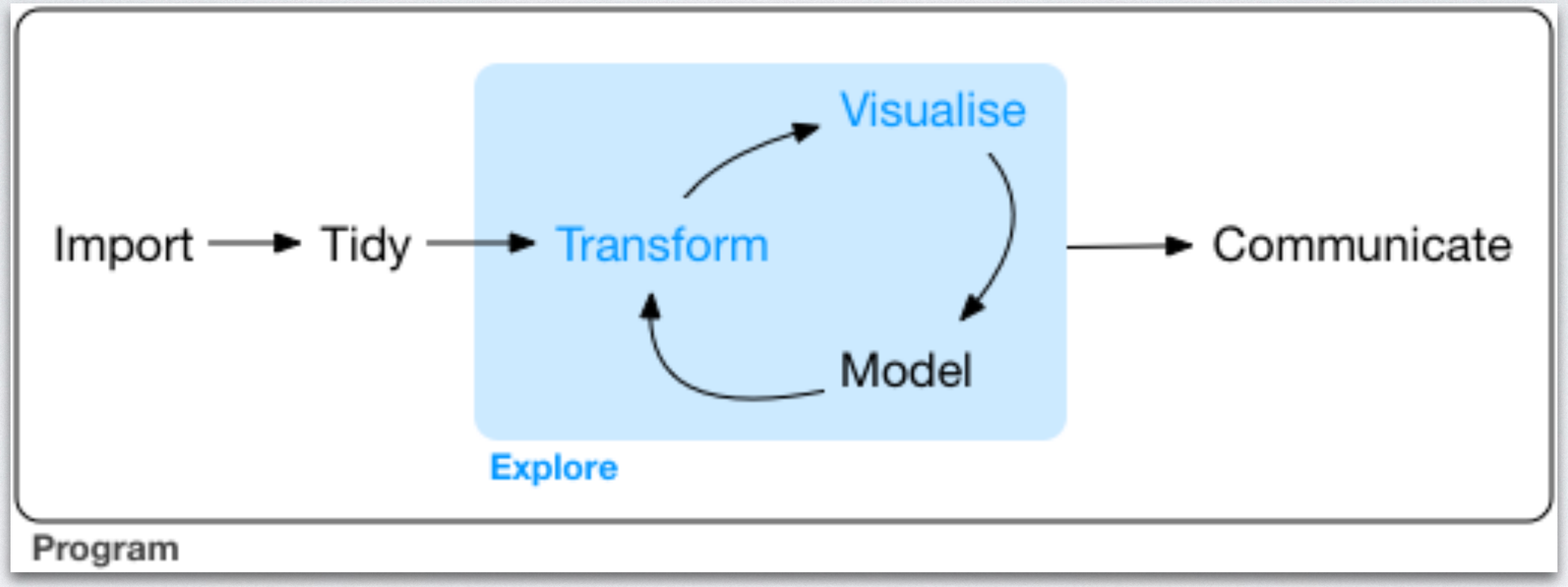
- You assign variables with the `<-` operator (kind of strange…)
  `my_var <- "hello"`

- R makes extensive use of *vectors*, lots of operations are vectorized
  `vec1 + vec2`
  will produce a new vector of the element wise sums (same length!)

- The dot operator, `.`, can be used in variable/function names
  `my.var` is an OK variable name, DOES NOT mean the `var` data member of object `my`

- R has all the main components of a typical programming language:
  data types, boolean expression, loops, functions, classes, etc

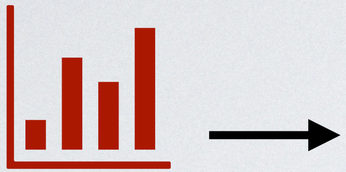
BASIC DATA ANALYSIS WORKFLOW



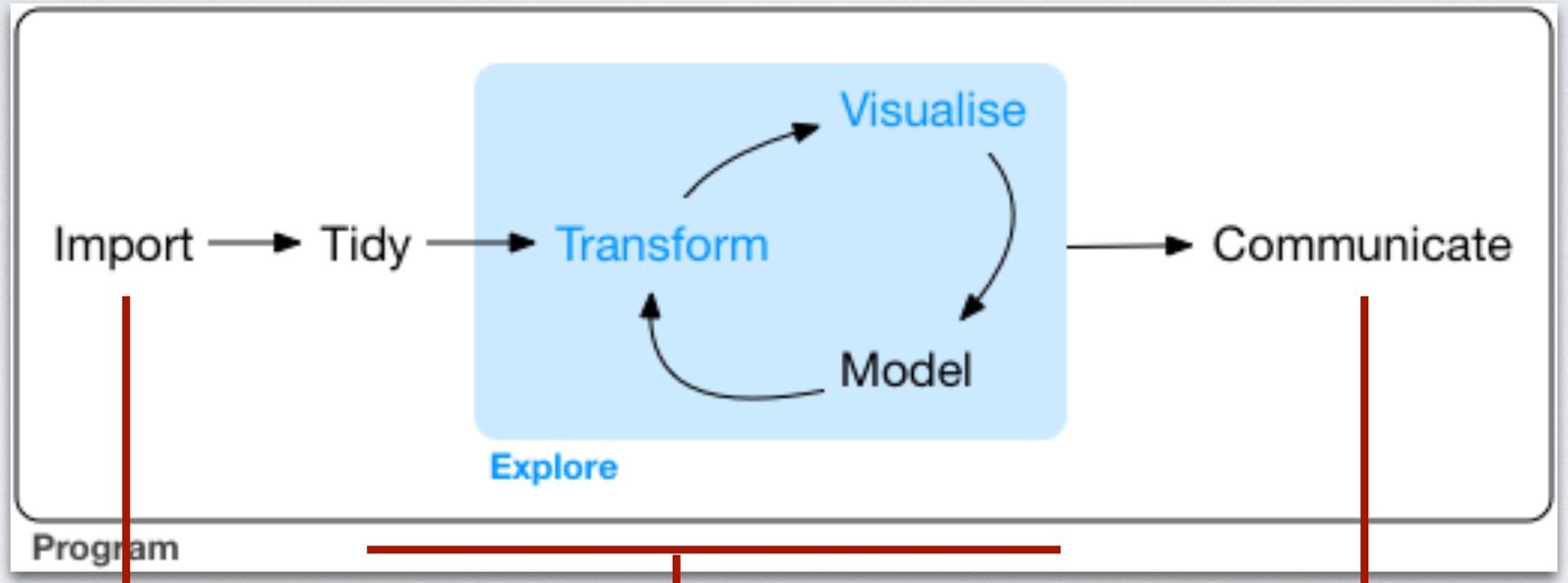
R for Data Science (Grolemund & Wickham)

https://r4ds.had.co.nz/explore-intro.html

BASIC DATA ANALYSIS WORKFLOW



Data out in the world



get data into R

manipulate & transform

make plots & visuals

or reports, presentations, etc.

Part 1: Getting Data into R

GETTING DATA INTO R

- R (and contributed packages) provide ways to read **lots of different types of data**, including:
  - **formatted text files**: csv, tsv, …
  - Excel files
  - Statistics formats: SPSS, Stata, SAS
  - Web formats: json, xml, html
  - Database connectors
  - Specific packages for reading specific types of files

- Sometimes it's easy to read data, sometime it's hard!
  (often more a property of the data, not the reading infrastructure)

TIPS FOR NEW R USERS

- Start practicing with formatted text files (e.g. csv) that are known to be in *good shape* (i.e. don't have formatting problems)

- Lots of "real world" data is not in good shape — if you get errors or warnings when you read a data file, try another dataset

- As you get better working with R (and data in general), you can start diagnosing and fixing data import problems, and reading data from more complex file types

THE TIDYVERSE PROVIDES FUNCTIONS FOR READING MANY DATA FORMATS

- Two key packages: `readr` and `readxl`

for formatted (tabular) text files

for Excel files

- `readr: read_csv, read_tsv, read_delim`, and more
- `readxl: read_excel`
- It's important to pay attention to any output (errors, messages, warning) you see when importing a file

LIVE CODING EXAMPLE

(READING DATA INTO R)

READING EXAMPLE # 1

- Try reading the provided .csv file, `01_read_example.csv`

- Questions to consider
  - What kind of file is this?
  - What read function should you use?
  - Did you see any messages? What did they say?

- We'll learn how to work with this data in Part II

READING EXAMPLE #2

- Try reading the provided .csv file, `02_read_example.tsv`

- Questions to consider
  - From the file name, how does this file differ from the first?
  - What read function should you use?
  - Did you see any messages? What did they say?

READING EXAMPLE #3

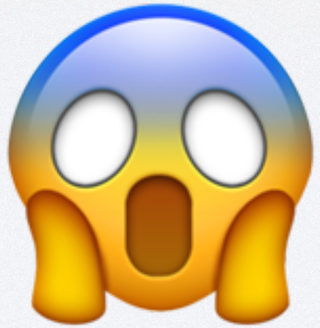
- Try reading the provided .csv file, `03_read_example.csv`

- Questions to consider
  - What read function should you use?
  - Did you see any messages? What did they say?

READING EXAMPLE #3 – TROUBLE SHOOTING

- The file, `03_read_example.csv`, has a problem!

- Can you figure out the issue from the read function output?



READING EXAMPLE #3 – TROUBLE SHOOTING

- `03_read_example.csv`, is actually a tsv file
  (tab delimited *not* comma)

- Real world data often has problems like this, and unfortunately, the problems can be much worse.

- Learning to spot and fix these problems is a critical skill for anyone who works with data, you'll get better with experience

SOME TYPES OF DATA FORMAT ISSUES YOU MIGHT ENCOUNTER

- A file type isn't (or isn't properly) specified from its extension
- No header row
- Inconsistent number of entries on each line
- Corrupted data
- Incomplete data

Sometimes, you just need to open the file in a text editor and see what's going on

PARTING THOUGHTS

- In theory, reading data into R should be the easiest step, but it often presents an initial roadblock 😡
- This step is really important — if there are problems with your source data, you need to know about it and handle appropriately

Part II: Basic Data Manipulation & Transformation

TWO KEY R DATA STRUCTURES: ATOMIC VECTORS & DATA FRAMES

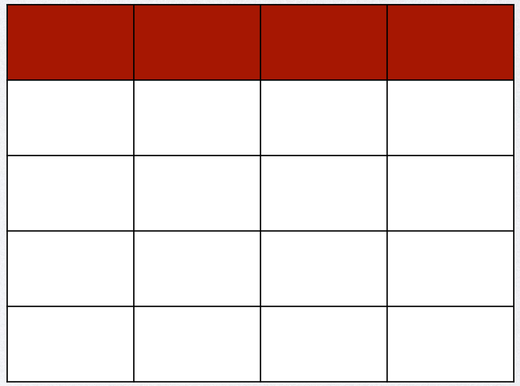
An atomic vector is just an array of values of a given type

1.2	3.4	2.5	8.1	7.7

A	ABC	hello	world	R

There aren't any scalars (single values) in R
A single value is just a vector of length 1

A data frame is essentially just a rectangular table, where each column is an atomic vector*



*not strictly true due to the support of list columns with tibbles

ATOMIC VECTORS CAN BE CREATED WITH THE `c()` OPERATOR

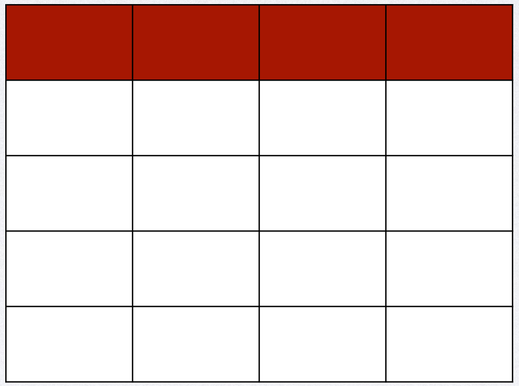
- c (lower case) stands for *combine*
- Example usage:
  `c(1,2,3,4)` or `c("hello", "world", "R is cool")`
- If you mix data types inside of c(), R will up-cast everything to common data type (important implications for reading data!), e.g.
  `c(1,2,"hello")` becomes `c("1", "2", "hello")`
- You index a vector with `[`
  `my_vec[3]` or `my_vec[c(4,5,6)]`

LIVE CODING EXAMPLE

(WORKING WITH VECTORS)

DATA FRAMES ARE TABLES OF DATA WHERE THE COLUMNS ARE ATOMIC VECTORS

When you read (tabular) data into R, the output will be a **data frame**



header row

a given column is an **atomic vector** with a specific type of data

but different columns can have different data types (i.e numbers and strings)

YOU GET DATA FRAMES WHEN READING (TABULAR) DATA

- Data frames can be created "by hand" with the `data.frame` function
- However, you'll likely more often get access to data frames by reading data into R, e.g. using the `read_csv` function
- Data frames are one of the most widely used data structures in R, so it's worth learning how to work with them

BASIC DATA FRAME OPERATIONS

Image you've read a csv file into R:

```
my_dat <- read_csv("data.csv")
```

Command	What it does
`my_dat`	Prints out the data frame
`head(my_dat)`	Prints out the first 6 rows
`dim(my_dat)`	Gets the # of rows & cols
`names(my_dat)`	Gets the column names
`my_dat$COLUMN_NAME`	Get the values of COLUMN_NAME
`my_dat$NEW_COLUMN <- new_vals`	Adds a new column with new_vals
`my_dat[c(1,2,3), c(3,4)]`	Gets rows 1 to 3 and columns 3 to 4

LIVE CODING EXAMPLE

(WORKING WITH DATA FRAMES)

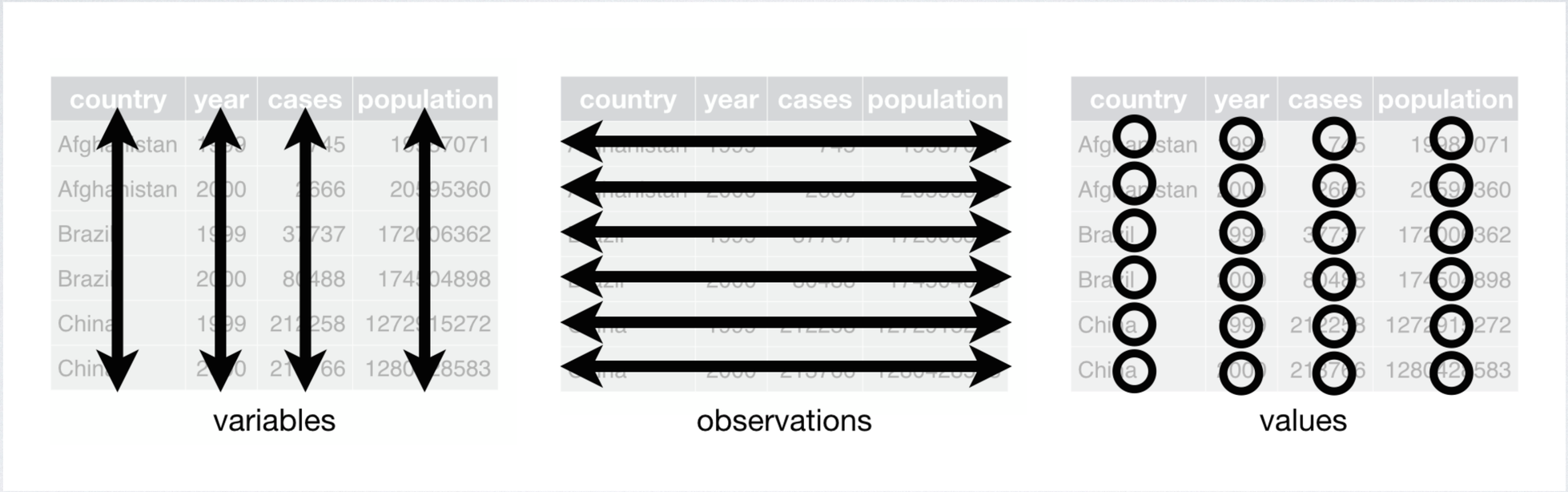
DATA MANIPULATION WITH DPLYR

- `dplyr` is an AMAZING R package that allows you create data processing and analysis pipelines

- Using the concept of data processing *verbs* that encompass fundamental data transformations that can be combined together

- dplyr is part of the tidyverse which further expands the possibilities for data analysis in R built around the concept of *tidy data*

WHAT IS TIDY DATA?



Every variable has its own column

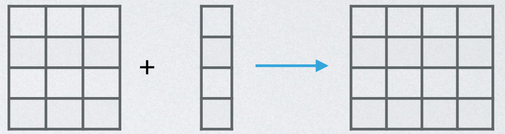
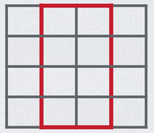
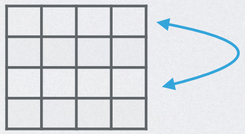
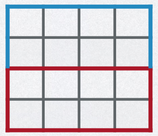
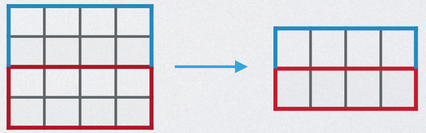
Every observation has its own row

Every value has its own cell

R for Data Science (Grolemund & Wickham)

https://r4ds.had.co.nz/tidy-data.html

DPLYR USES DATA MANIPULATION VERBS THAT COVER FUNDAMENTAL DATA OPERATIONS

`mutate`	Add a column	
`select`	Pick specific columns	
`filter`	Subset to specific rows	
`arrange`	Reorder/sort rows	
`group_by`	Group subsets	
`summarize`	Perform aggregated calculations	

YOU COMBINE THESE VERBS TOGETHER TO MAKE DATA PIPELINES

```
# dplyr examples using the 'storms' data set
library(dplyr)

# We'll use the dplyr built in 'storms' data set
storms

# Example dplyr pipeline
mean_stats <- storms %>%
  group_by(year) %>%
  summarize(mean_wind_speed = mean(wind),
            mean_air_pressure = mean(pressure)) %>%
  arrange(desc(mean_wind_speed))

mean_stats
```

This is the pipe operator (it looks weird, I know)

LIVE CODING EXAMPLE

(DPLYR PIPELINES)

Part III: Basic Plotting with ggplot2

GGPLOT2 IS AN AMAZING R PACKAGE FOR MAKING PLOTS

- ggplot2 is one of the most widely used R packages
- seriously… lots of people learn R just to use ggplot2
- Based on the *Grammar of Graphics* (L. Wilkinson) developed by H. Wickham
- Presents a foundation for systematically building almost any kind of plot

GGPLOT2 CHANGES THE WAY YOU THINKING ABOUT MAKING PLOTS

Many plotting programs give you a menu of plots to choose from

ggplot2 gives you the foundation to be your own plotting chef!

GGPLOT2 ALLOWS YOU TO BUILD-UP PLOTS, LAYER BY LAYER

- ggplot2 provides a variety of *geom*'s and *stat*'s including
  - points
  - lines
  - bars
  - boxplots
  - densities, counts
  - lots, lots, more…
- You build up plots by mapping your data to these graphical representations, to build virtually any type of 2-D plot
- Also provides scales (coordinates, color, size,…), positioning adjustments, annotations, faceting and others

GGPLOT2 GRAPHING TEMPLATE

a data frame, data you want to plot
should be in their own columns

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

the specific geom (or stat) function,
e.g. geom_point

the mapping of your data (column names) to
the parameters of the geom function,
e.g. the x and y position of the points

GGPLOT2 EXAMPLES

The infamous iris data set

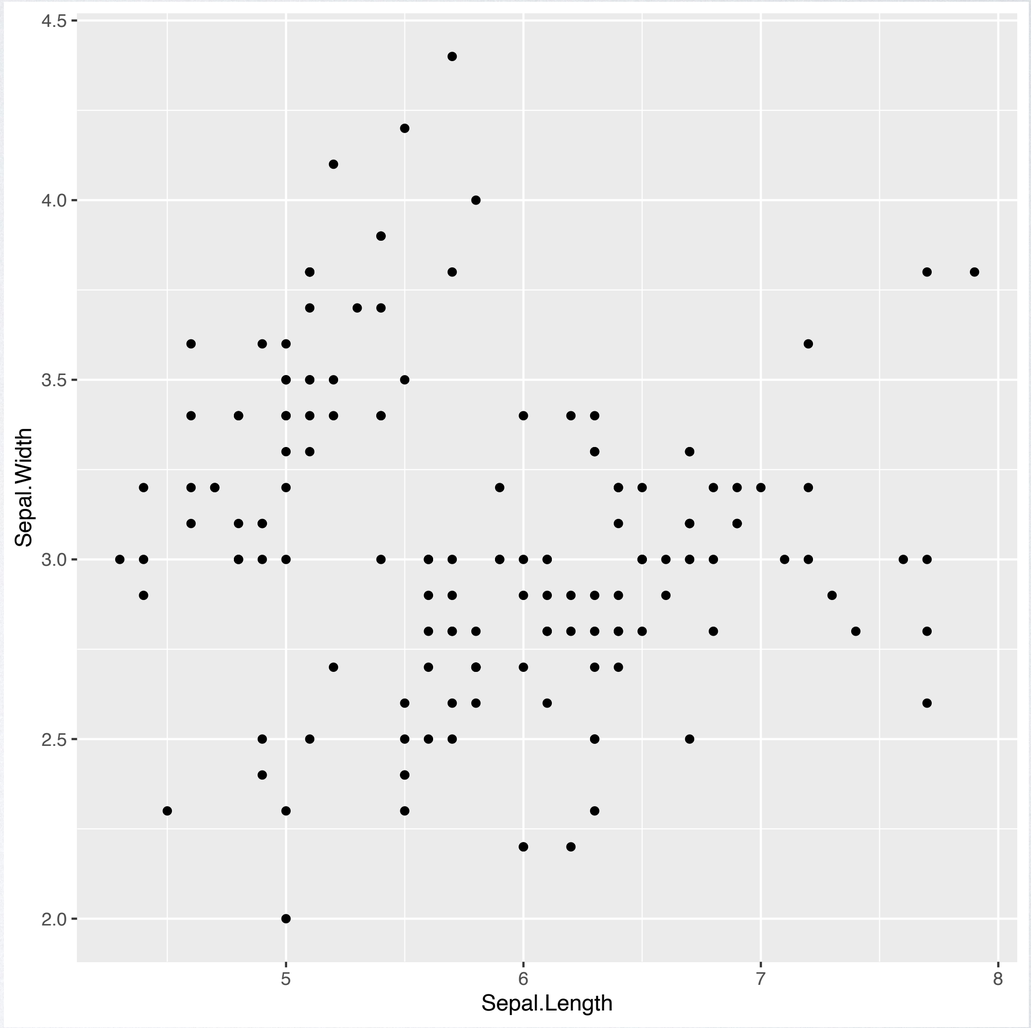
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

...

Measurements for iris flowers, 3 different species

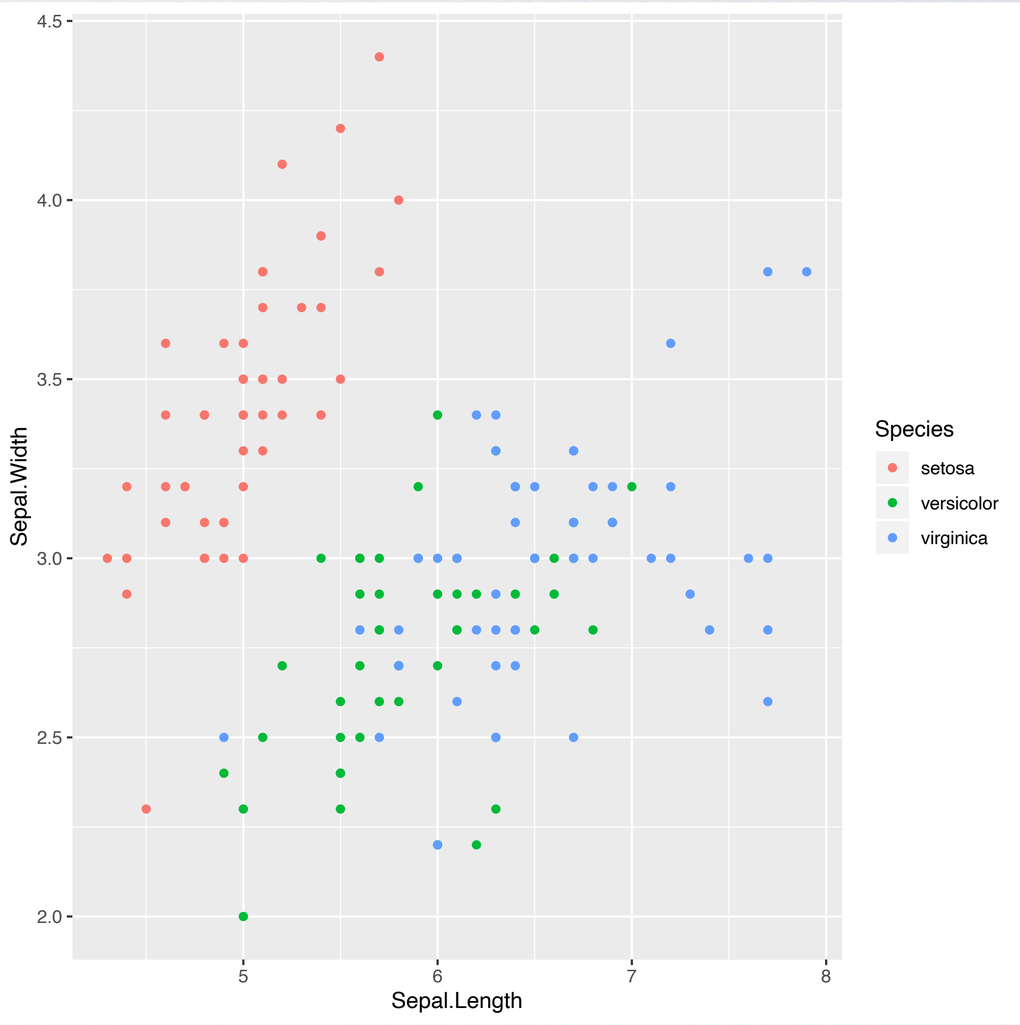
GGPLOT2 EXAMPLES

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



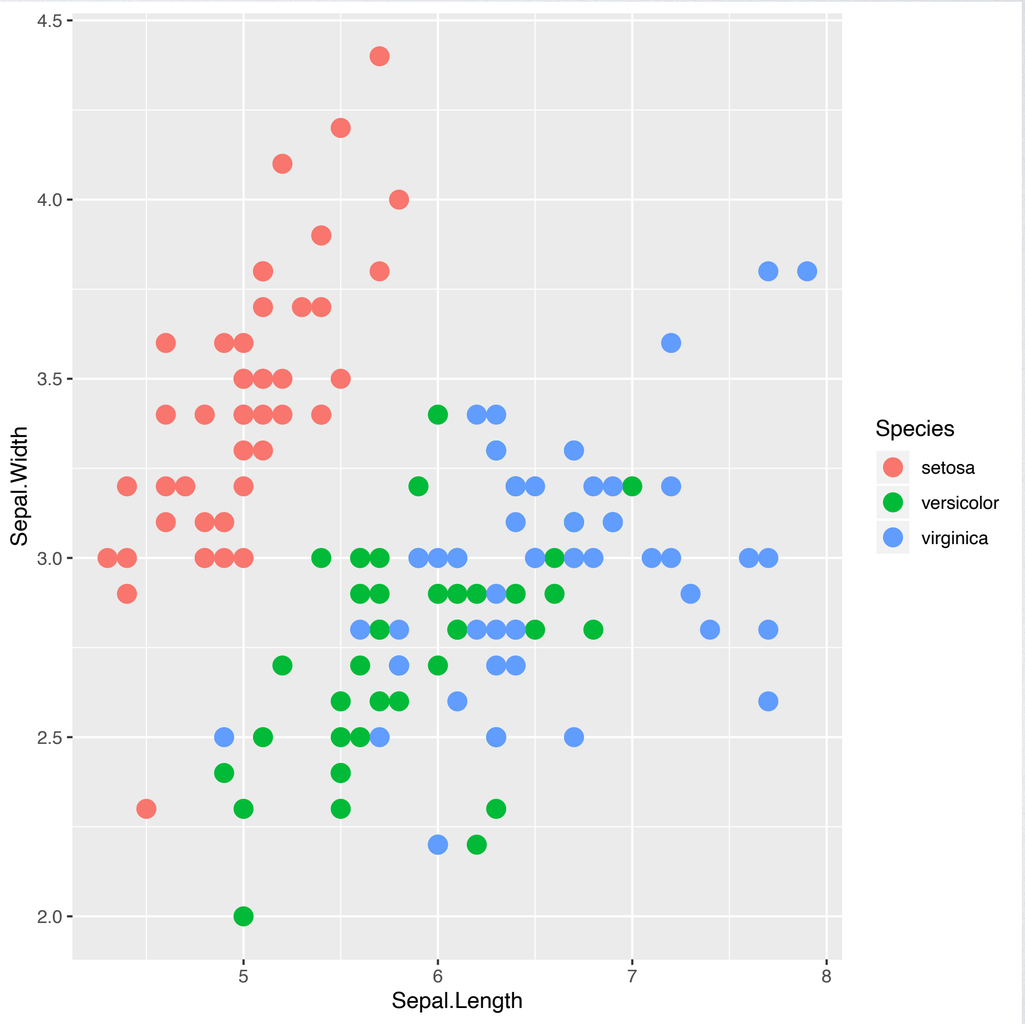
GGPLOT2 EXAMPLES

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



GGPLOT2 EXAMPLES

```
ggplot(iris,
  aes(x = Sepal.Length,
      y = Sepal.Width,
      color = Species)) +
  geom_point(size = 4)
```



LIVE CODING EXAMPLE

(GGPLOT2 EXAMPLES)

Part THE END
Next Steps

R IS CHALLENGING TO LEARN, PARTICULARLY AT THE BEGINNING

- A little practice every day goes a long way!
- Simply, but "complete" examples and practice are very effective
  — read in a data file, do some manipulations, make a plot
- Inspect, review, check your results — does the output make sense?
- Be kind to yourself! Learning R is challenging, but worth the effort!

RESOURCES

- **Books**
  - R for Data Science
  http://r4ds.had.co.nz

  - Hands on Programming with R
  https://rstudio-education.github.io/hopr/

  - ggplot2: Elegant Graphics for Data Analysis, 2nd Ed.
  https://ggplot2-book.org

  These are more advanced
  - Applied Predictive Modeling
  http://appliedpredictivemodeling.com

  - Advanced R
  https://adv-r.hadley.nz

- **Websites**
  - RStudio's Online Learning Resources
  https://education.rstudio.com

  - RStudio Community
  https://community.rstudio.com

  - Kaggle
  https://www.kaggle.com

  - R Bloggers
  https://www.r-bloggers.com

Thank you!

Any Questions?