Data Analytics
CS301
Chapter 2,
Intro to R

Week 3
Spring 2023
Oliver BONHAM-CARTER

Where To Now?

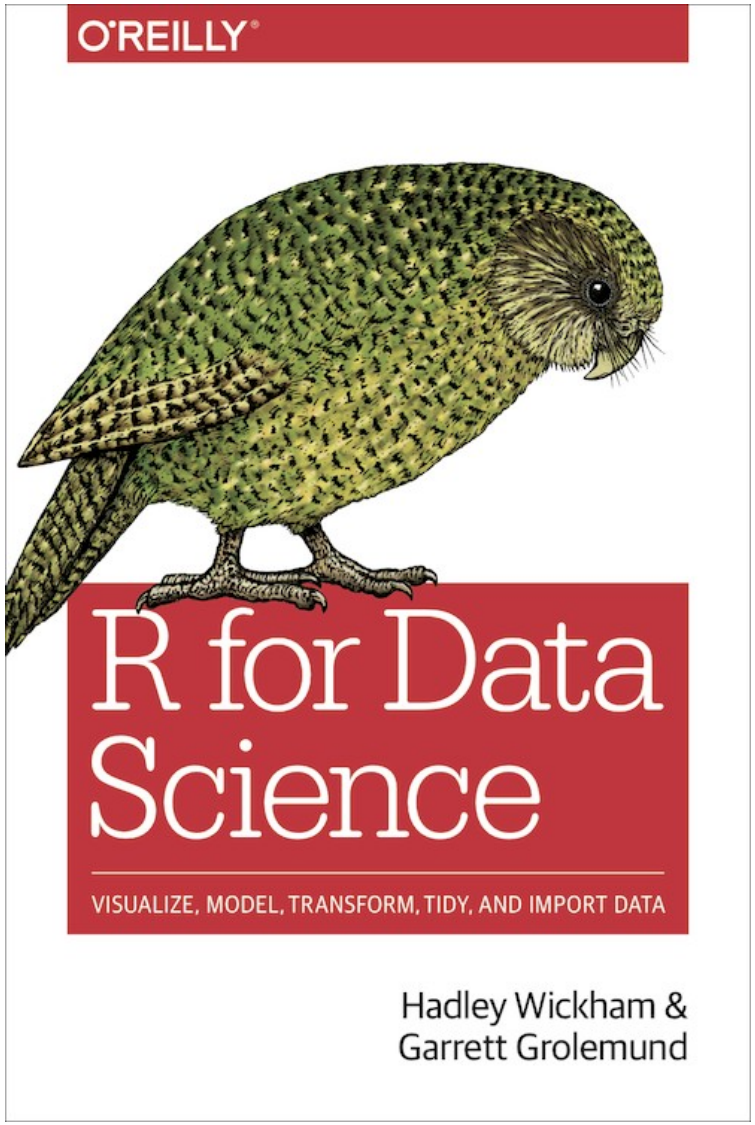
- Google Analytics is a tool allowing for convenient analysis of web sites
- The code was created by developers for a specified purpose (i.e., web site analysis).

For Your Own Analysis?

- **BUT! What if you are working on a project and no tools currently exist?!**

Develop Your Own Tools!!

We will be using the Book



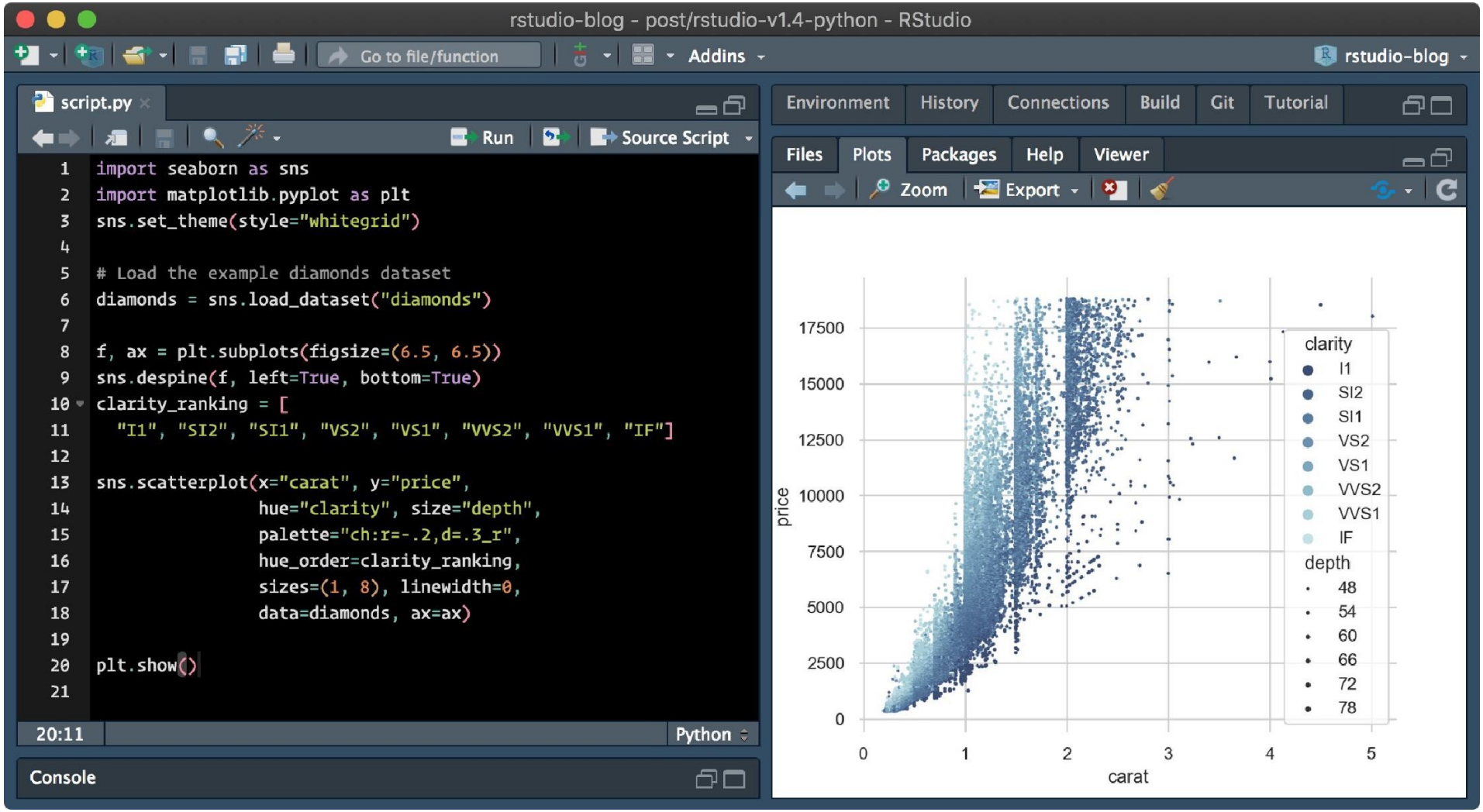
- Note the chapters between the book and the website are not numbered identically!
- Book:
  - Chap 1: Data Visualization with ggplot
  - **Chap 2: Workflow; Basics**
- On the web site:
  - http://r4ds.had.co.nz/
  - Chap 3: Data Visualization
  - **Chap 4: Workflow; Basics**

The R Programming Language

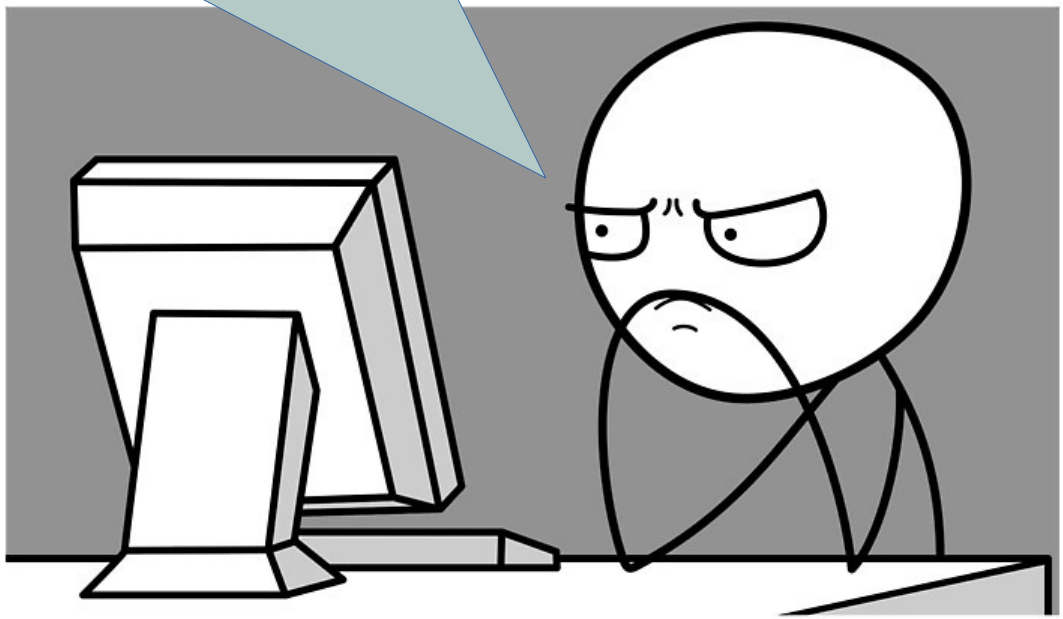
- https://www.r-project.org/
- What is the R language?
  - An open source, well-developed programming platform for work in statistics, mathematics and data analytics
  - Cross platform; runs on major OSs
  - Popular programming skill among Big Data analysts, and data scientists
- Community Blogs:
  - https://www.r-bloggers.com/
  - https://twitter.com/rstudiotips/
  - https://towardsdatascience.com/

RStudio

ALLEGHENY COLLEGE



Let's take a moment to install these software packages!



https://www.oliverbonhamcarter.com/resources/dataanalytics_resources/

Who uses R?

R applications are not enough until you don't know how people/companies are using the R programming language.

Facebook – Facebook uses R to update status and its social network graph. It is also used for predicting colleague interactions with R.

Ford Motor Company – Ford relies on Hadoop. It also relies on R for statistical analysis as well as carrying out data-driven support for decision making.

Google – Google uses R to calculate ROI on advertising campaigns and to predict economic activity and also to improve the efficiency of online advertising.

Foursquare – R is an important stack behind Foursquare's famed recommendation engine.

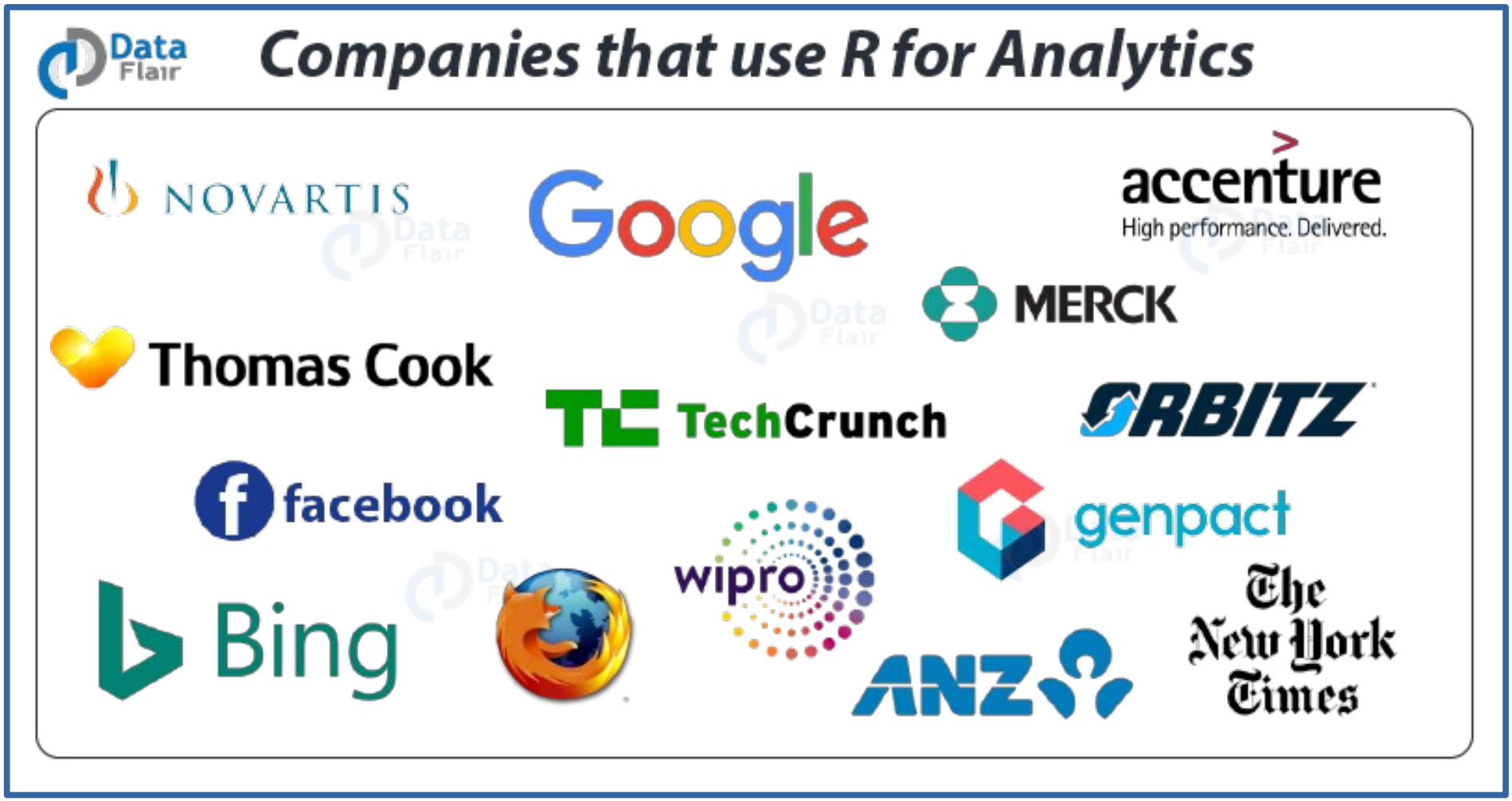
John Deere – Statisticians at John Deere use R for time series modeling and also geospatial analysis in a reliable and reproducible way. The results are then integrated with Excel and SAP.

Microsoft – Microsoft uses R for the Xbox matchmaking service and also as a statistical engine within the Azure ML framework.

Mozilla – It is the foundation behind the Firefox web browser and uses R to visualize web activity.

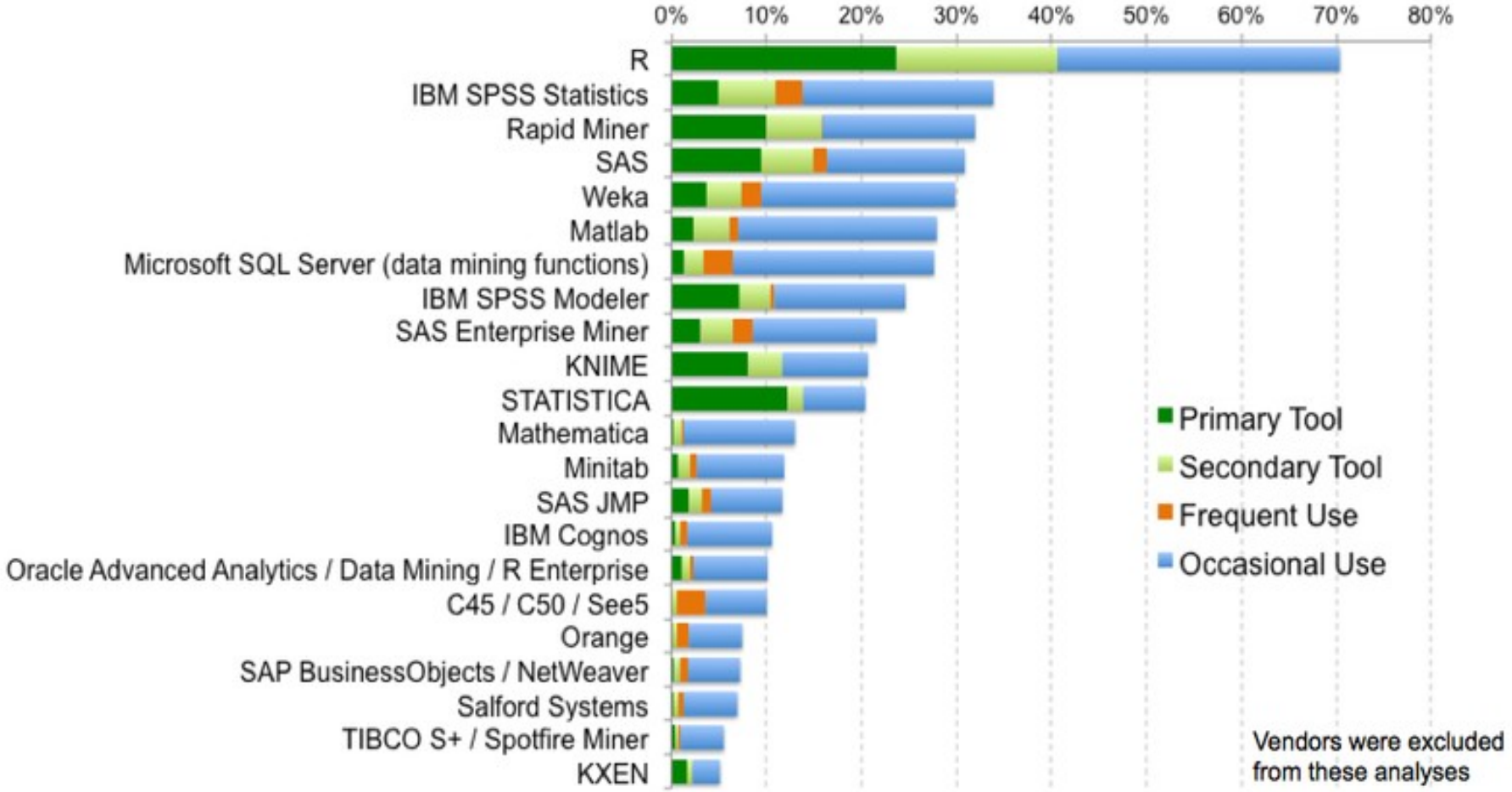
Ref: https://data-flair.training/blogs/r-applications/

Who uses R?



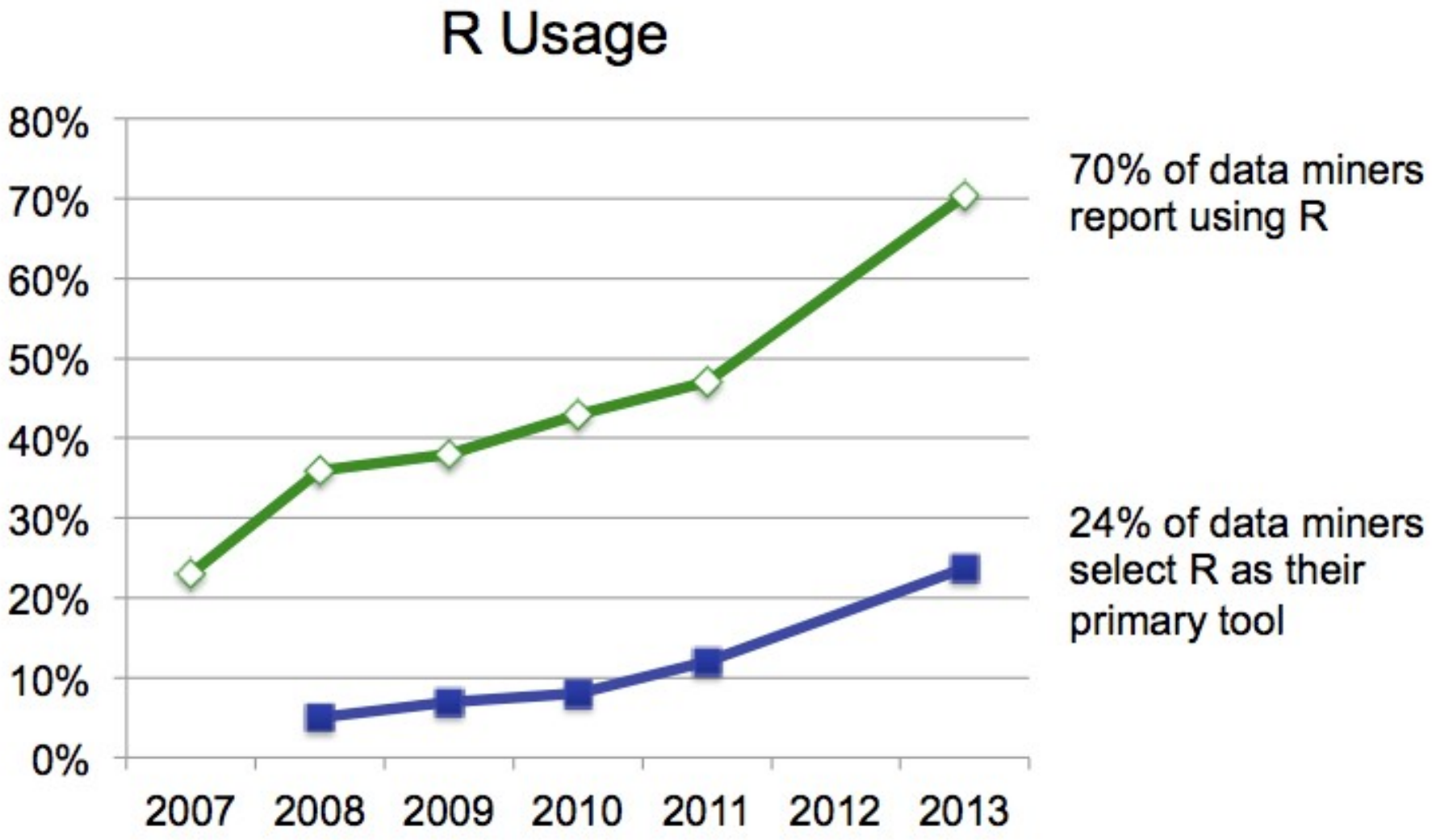
Ref: https://data-flair.training/blogs/r-applications/

R: The Most Popular Data Mining Tool



http://blog.revolutionanalytics.com/2013/10/r-usage-skyrocketing-rexer-poll.html

R is Exploding in Growth



http://blog.revolutionanalytics.com/2013/10/r-usage-skyrocketing-rexer-poll.html

NEWS | COMPUTING

Top Programming Languages 2022 >

Python's still No. 1, but employers love to see SQL skills

BY STEPHEN CASS | 23 AUG 2022 | 4 MIN READ

R is in the top Ten!

https://spectrum.ieee.org/top-programming-languages-2022

Ranking To Others: IEEE 2017

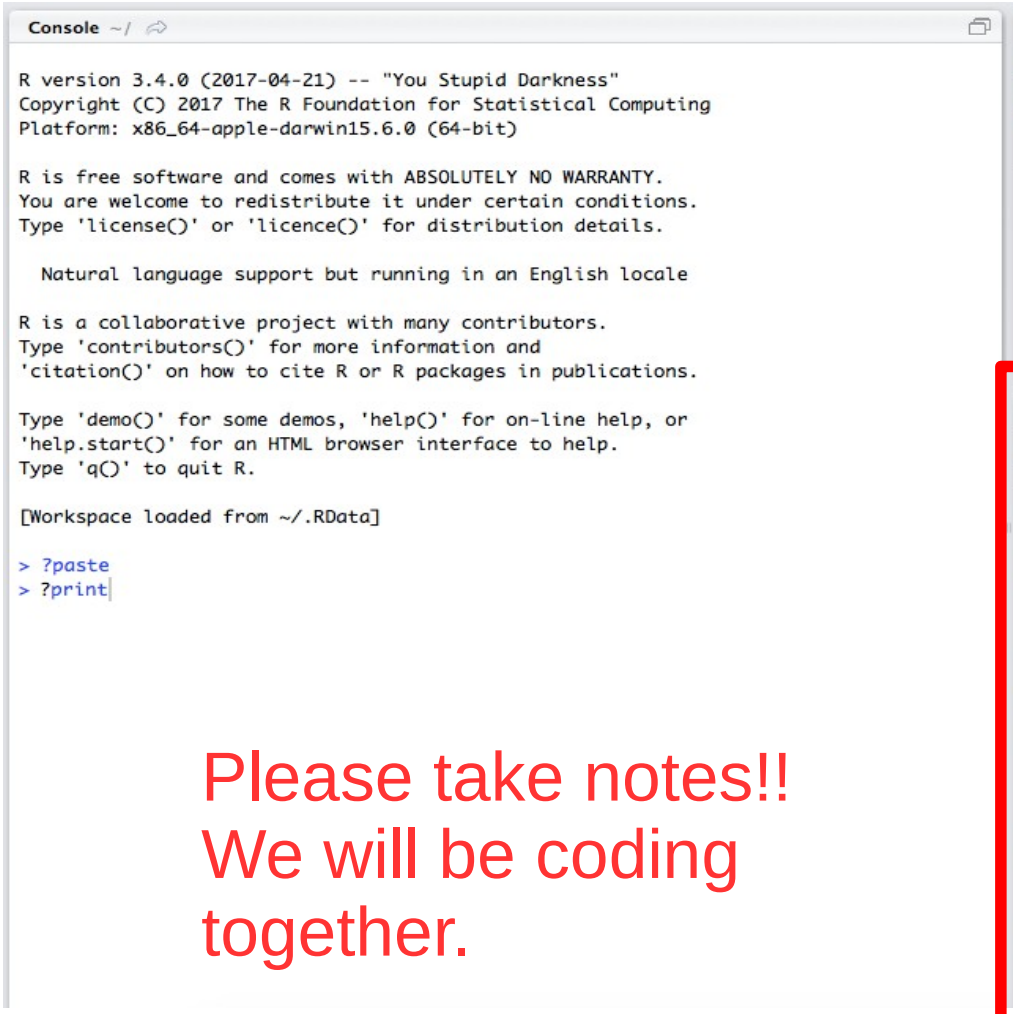
Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.4
4. C++		97.2
5. C#		88.6
6. R		88.1
7. JavaScript		85.5
8. PHP		81.4
9. Go		76.1
10. Swift		75.3

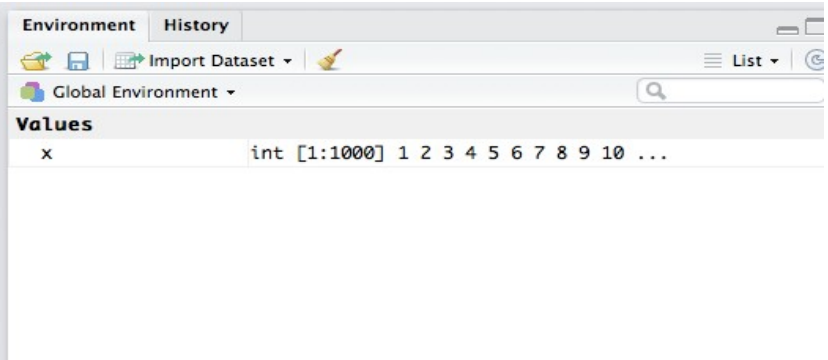
Find more amazing studies about R:

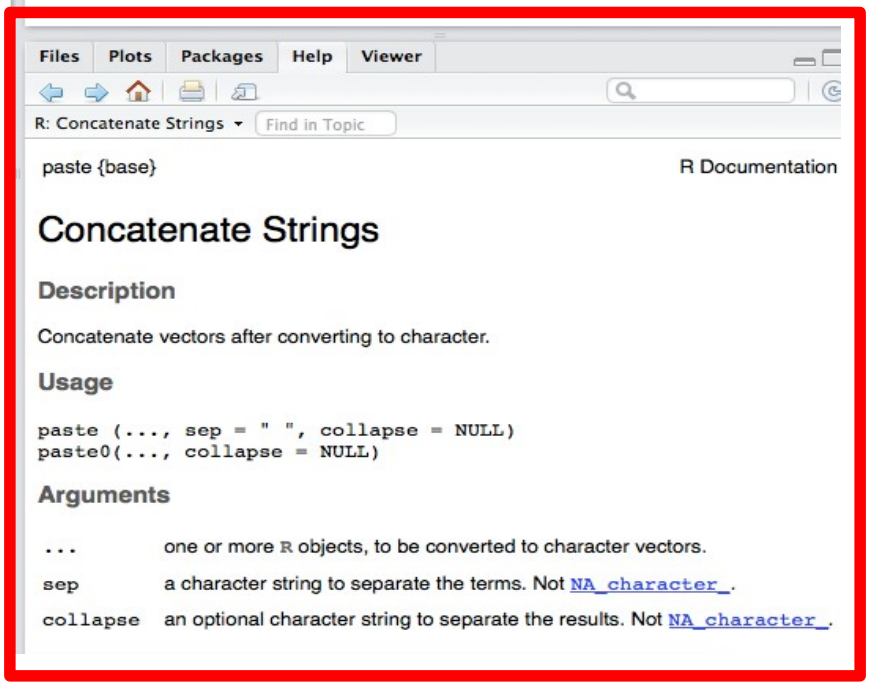
http://blog.revolutionanalytics.com/2018/06/pypl-programming-language-trends.html

Getting Help in R

- Online help: place a “?” in front of a keyword
  - Ex: ?print



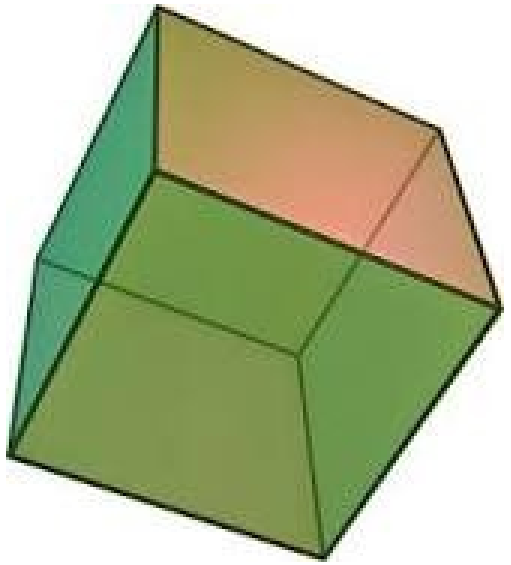




Please take notes!!
We will be coding together.

Variable Names

- Variable Names:
  - Begin with a letter, and can only include letters, numbers, periods and hyphens.
  - Hyphens: “-”
  - Periods: “.”
- SnakeCase (recommended by book)
  - val_of_height,
  - val_of_length,
  - val_of_width



Basic Math

- Mathematics
  - Addition: 1 + 1
  - Subtraction: 1 - 1
  - Multiplication: 3 * 7
  - Division: 1 / 4
- More complicated math, var assignments:
  - 4*(7+3)/10+1 Note: watch the order of operations!
  - Parameter of circle ($C = 2 * pi * r$)
    - *R* <- 4, Note the "<-" means *equal* in R.
    - *C* <- 2 * pi * R = 2 * 3.1415 * 4
    - *C* is 25.13274

Variable Names

- CamalCase:
  - valOfHeight,
  - valOfLength,
  - valOfWidth
- Period.Case
  - Val.of.height,
  - Val.of.length,
  - Val.of.width
- What-EVER.Case
  - Val.ofHEIGHT,
  - Val.Of_Length,
  - Val.oF.Width

Assigning Variables

- Assign a variable
  - *x* = 1, or
  - *x* <- 1
  - *y* = 3
  - *y* <- 3
  - Run:

    x + y

```
> x <- 1
> y <- 3
> x + y
[1] 4
```

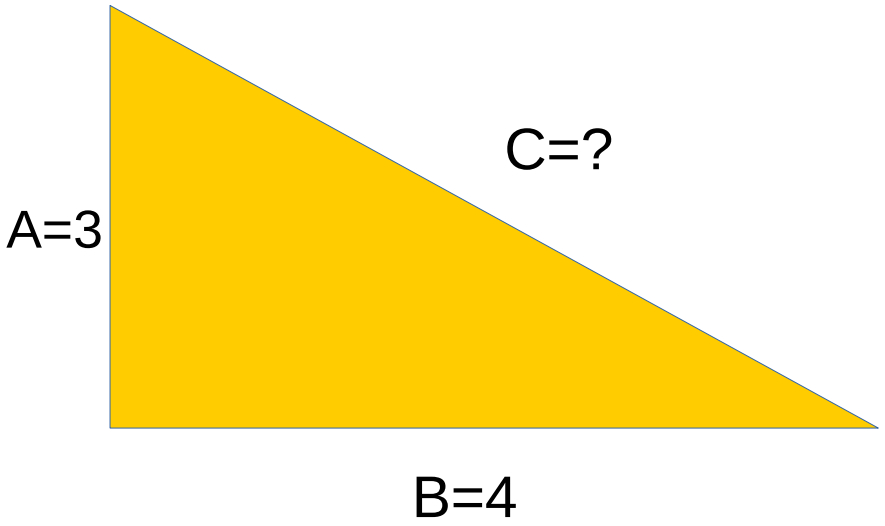
  - *myNum <- -2*
  - *myOtherNum <- -4*
  - Run:

    myNum + myOtherNum

```
> myNum <- -2
> myOtherNum <- -4
> myNum + myOtherNum
[1] -6
```

Variables and Assignments

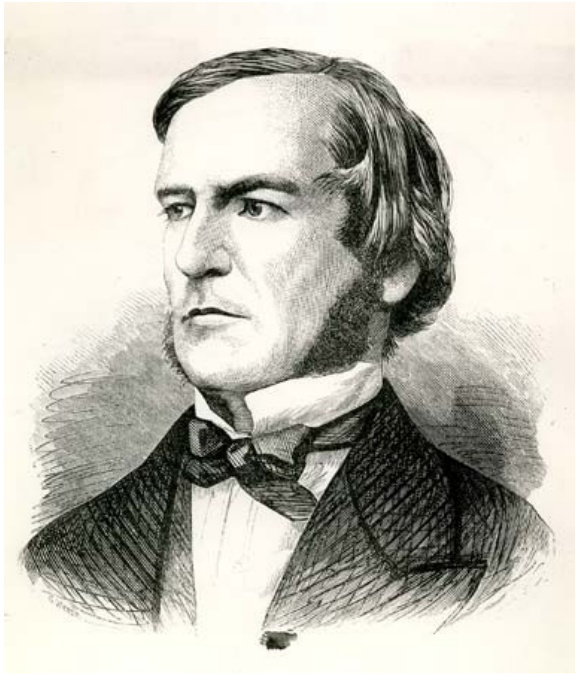
- *A* <- 3
- You could also use "*A*=3" (but this is not traditional programming in R)
- Hypotenuse (*C*) defined by *sqrt(A^2 + B^2)*
- A <- 3
- B <- 4
- C <- sqrt(A^2 + B^2)
- C is ??

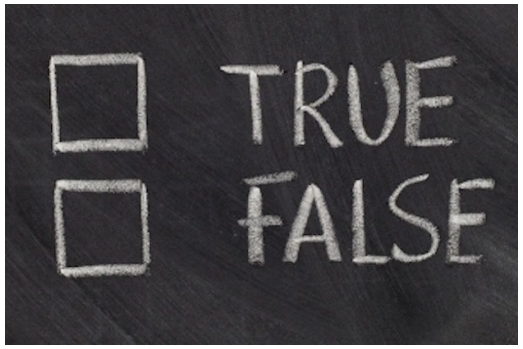


Logical Operations

- Booleans: Returning True or False:

```
3 > 4, 3 < 4,
2 + 4 == 6,
2 + 3 == 4 + 1
T == TRUE
F == FALSE
3 + 4 != 5
3 + 4 == 7
5 * 2 != 11
```





Try some of These in R!

- Logical **AND**
- (&&)

```
F && F is F
F && T is F
T && F is F
T && T is T
```

- Logical **OR**
- (||)

```
F || F is F
F || T is T
T || F is T
T || T is T
```

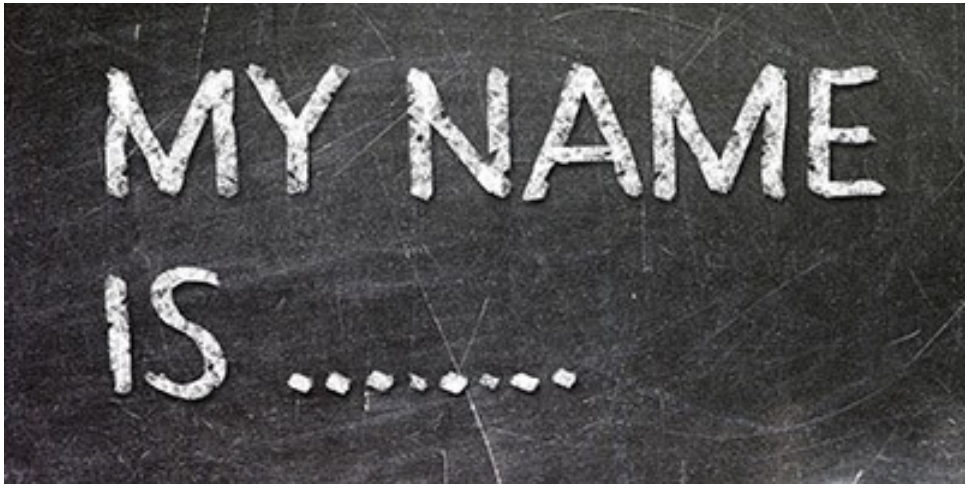
- Logical **NOT**
- (!)

```
!F is T
!T is F
```

Truth Tables:
https://en.wikipedia.org/wiki/Truth_table
De Morgan's Laws:
https://en.wikipedia.org/wiki/De_Morgan%27s_laws

Simple Strings

- Strings
  - “Hello World”
- Concatenation of strings
  - H <- “Hello”
  - W <- “world”
  - paste(H,W, sep = “ ”)
    - What is the result here??



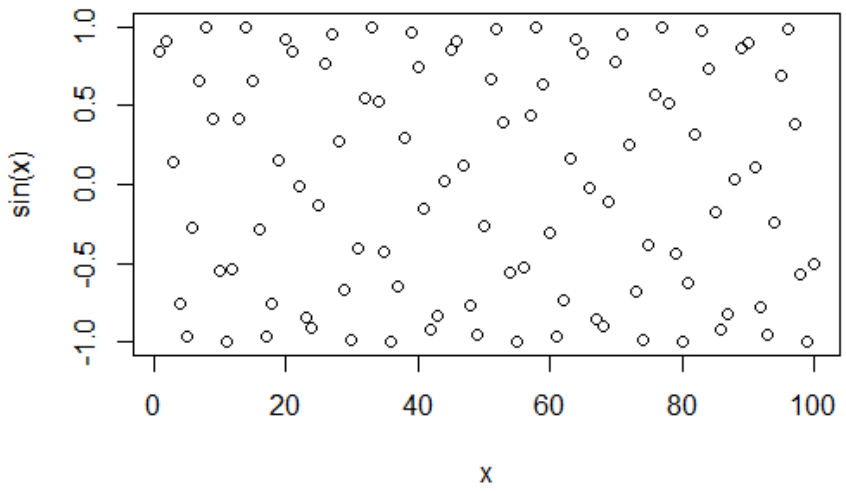
- You try: print your full name!
  - *first* <- "Sherlock"
  - *last* <- "Holmes"
  - paste(*first*,*last*, sep =" ")

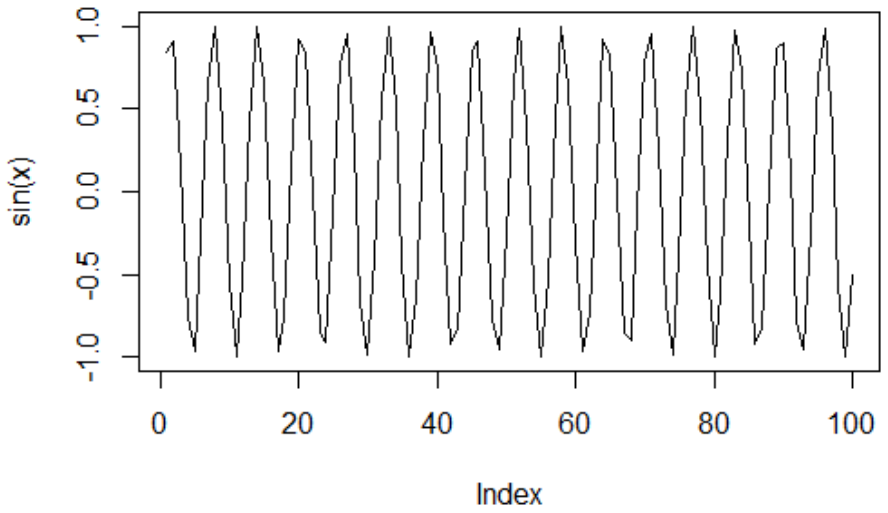
Built-in Functions

- R has a large collection of built-in functions:
  - function_name(arg1 = val1, arg2 = val2, …)
  - `seq(from, to), ex: seq(0,10)`
  - Gives a sequence, *S* ={0,…,10}
  - What happens when you press TAB after typing, “seq”?
- Use the `sum( )` function to add two numbers
  - sum(1,10)
  - Adds 1 and 10
- Add all elements in a vector, *v*
  - v <- 0:10
  - sum(v)
  - Adds: 0 + 1 + … + 9 + 10 = 55

Simple Plots

- x<- seq(1,100) # assign *x* to the sequence 1 to 100
- plot(x) # plot this sequence
- plot(sin(x)) or plot(x,sin(x)) # see left plot below
- plot(sin(x)) or plot(x,sin(x), type = “l”) # see right plot below





Now, You Try

- Use R to write a command that...
  - Finds the **sum** of all numbers, 0 through 100
  - Finds the **sum** of all numbers, 0 through 10000

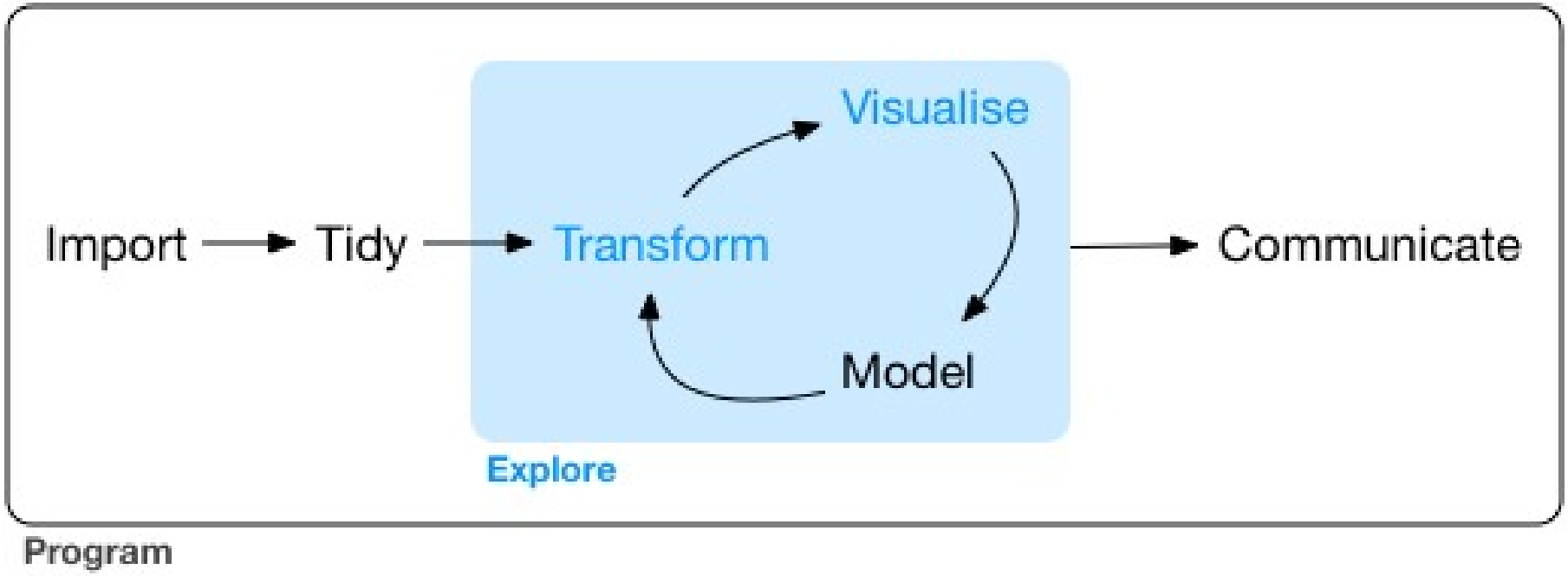
    (now, set a variable equal to the sequence first)
- Using the plot function, **plot(x,y,type = "l")** to plot a line of the function, f(*x*) = sin(*x*) for *x* in {0, …, 30}
  - x <- 0:10
  - plot(x, sin(x), type = "l")

Now try `cos()` and `tan()`!

Exiting R:
q()

Explore the Data Of Your World

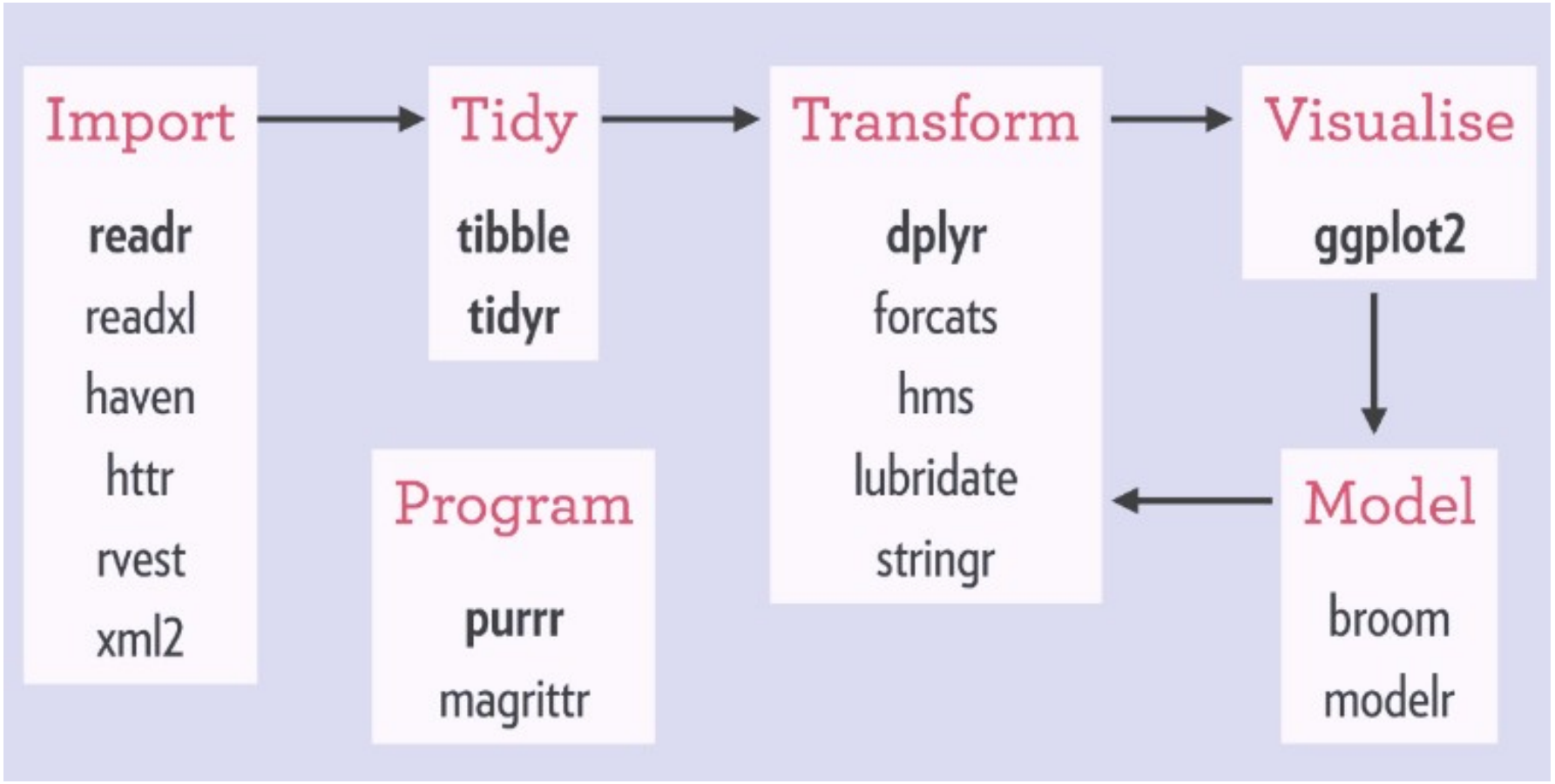
"Data exploration is the art of looking at your data, rapidly generating hypotheses, testing them, then repeating again and again..."



Import : Bringing in the raw data to work on it
Tidy: Cleaning it up so that numbers are numbers and etc.
Transform: Converting the data into something more *convenient* to use
Visualize: Finding general trends in data
Model: Testing phases, learning how to predict from the data.
Communicate: Publish and change the world!

Tidyverse's Packages

The steps of the Tidyverse canonical data science workflow, as well as, the individual packages that the steps involve.



Go Visual!

The Tidyverse library in R: a coherent system
of packages for data manipulation, exploration and visualization

Install the Library

```
> install.packages("tidyverse")
also installing the dependencies ‘colorspace’, ‘sys’
t’, ‘ps’, ‘sass’, ‘cachem’, ‘memoise’, ‘base64enc’,
st’, ‘fastmap’, ‘farver’, ‘labeling’, ‘munsell’, ‘RC
rewer’, ‘viridisLite’, ‘rematch’, ‘askpass’, ‘bit64’
ettyunits’, ‘processx’, ‘evaluate’, ‘highr’, ‘yaml’,
n’, ‘bslib’, ‘htmltools’, ‘jquerylib’, ‘tinytex’, ‘b
rts’, ‘ellipsis’, ‘generics’, ‘glue’, ‘assertthat’,
```

- For the first use, you need to **install** the library to your computer with,
  - *Install.packages(tidyverse)*

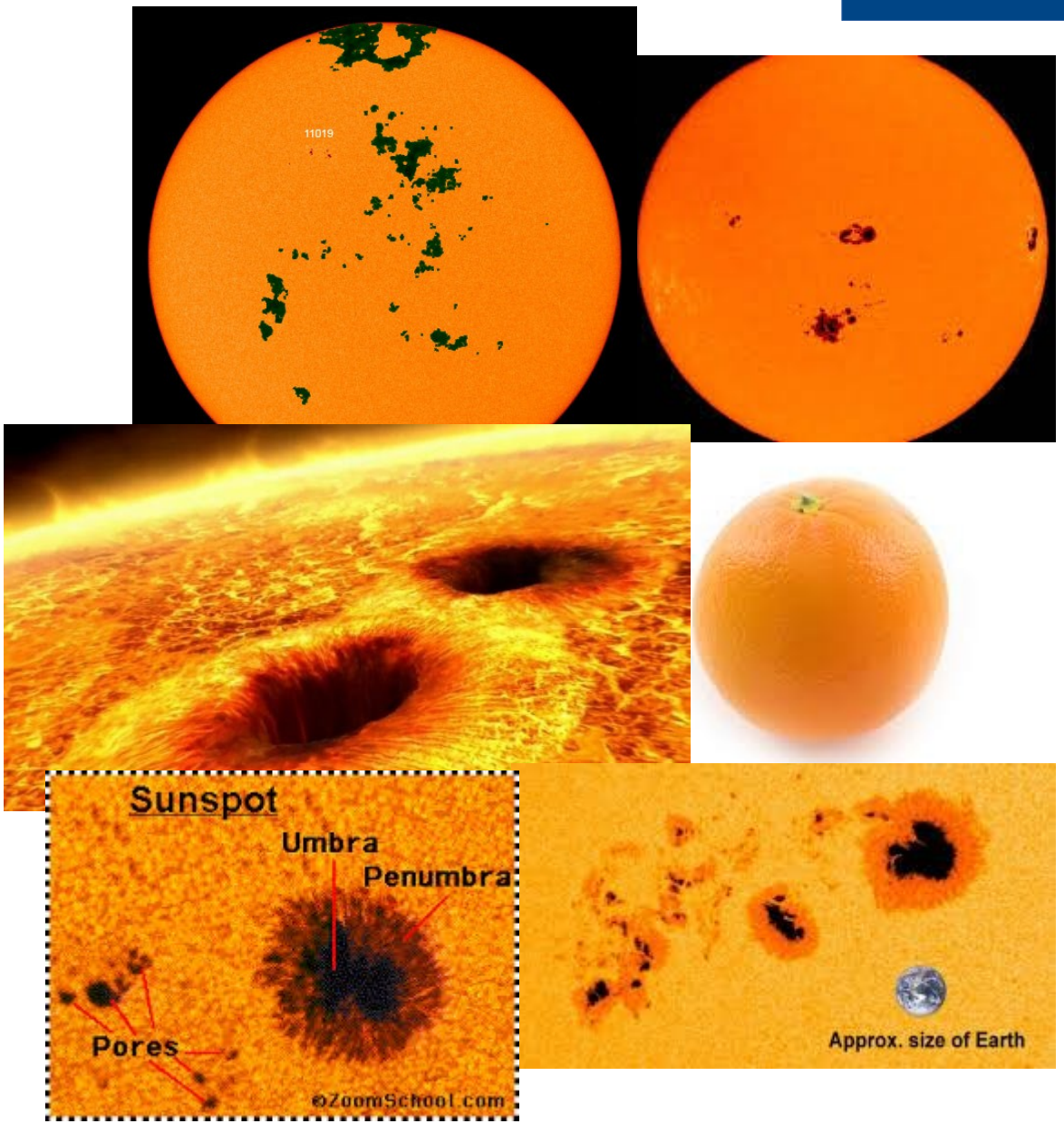
Load the Library

```
> library(tidyverse)
── Attaching packages ─────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.4.0      ✔ purrr   1.0.1
✔ tibble  3.1.8      ✔ dplyr   1.0.10
✔ tidyr   1.3.0      ✔ stringr 1.5.0
✔ readr   2.1.3      ✔ forcats 0.5.2
── Conflicts ──────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

- Once installed, you only need to **call** (or **load**) the library with,
  - *library(tidyverse)*

Exploring Sun-Spot Data

- Sunspots – magnetic disturbances on the sun that can be observed from Earth
- Spots cycles are noted to repeatedly increase and then decrease over time



Articulating the Research Question

- Is there something predictable about the sunspot data?
- Can we collect come evidence of a pattern in the data?
- Could we use this pattern to predict?
- What does a pattern look like in the data?

Load and Plot Sunspot Data

```
#Load library
library(tidyverse)

# find your sandbox file
sunData <- read.table(file.choose(), header =
TRUE, sep = ",")

# See what the data looks like
View(sunData)
```

```
# Plot the data:
ggplot(data = sunData) +  geom_point(mapping = aes(x =
fracOfYear, y = sunspotNum))

# Save a file to the Desktop/ (or wherever) if you
want...
ggsave("~/Desktop/myplot.png")
```

file: sandbox/sunspots.r

Code for a Simple GGPlot

- **install.packages("tidyverse") # install as necessary**
- **library(tidyverse) # call installed library**
- **ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy))**

- Establish the *canvas* (where the plot is shown)
- **ggplot()**

- Link to the data (set is called, 'mpg')
  - **ggplot(data = mpg)**

- Compute the geometery of point placement on canvas
  - **geom_point(mapping = …)**

- Compute the aesthetics of the plot (titles, color, point type, etc)
  - **aes(x = displ, y = hwy)**

Consider this ...

```
names(sunData)

ggplot(data = sunData) +  geom_point(mapping = aes(x = fracOfYear, y =
sunspotNum))

# Add a smooth line to see general trends
ggplot(data = sunData) +  geom_point(mapping = aes(x = fracOfYear, y =
sunspotNum)) + geom_smooth(mapping = aes(x = fracOfYear, y =
sunspotNum))

# Color by year
ggplot(data = sunData) +  geom_point(mapping = aes(x = fracOfYear, y =
sunspotNum, color = fracOfYear)) + geom_smooth(mapping = aes(x =
fracOfYear, y = sunspotNum))

# Color by month
ggplot(data = sunData) +  geom_point(mapping = aes(x = fracOfYear, y =
sunspotNum, color = month)) + geom_smooth(mapping = aes(x = fracOfYear,
y = sunspotNum, color = fracOfYear))
```

Run this code
to make other plots.
What do you see?

file: sandbox/sunspots.r

More Practice?!

```
#Load library
library(tidyverse)

# find and then choose a new dataset
data() # I choose ChickWeight

# See what the data looks like
View(ChickWeight)

# Make a plot!
ggplot(data = ChickWeight) +  geom_point(mapping = aes(x
= weight, y = Chick, color = Time))

# make more plots, then choose a new dataset,and repeat
```

Run this code
to make other plots.
What do you see?

file: sandbox/chickWeight.r