PO 399

Lecture 1

Instructor: Ahyoung Cho

Department of Political Science

Summer 2024

DATA SCIENCE FOR POLITICS

- Lots of hard political/policy problems in the world today
- Lots of data in the world today
- Problem: methods/approaches to data primarily built for engineering
    - Politics is not like engineering!
- Solution: combine data science skills with social science reasoning

GOALS OF THIS COURSE

- Develop technical skills
  - Programming
  - Probability/Statistics/Inference
- Combine these skills with social science thinking
  - Data and statistics are not a substitute for *thinking*.
  - Why do experts frequently make mistakes understanding data and statistics?
- Provide a framework for thinking about data and analysis
  - Correlation
  - Measurement
  - Causality
  - Prediction
  - Visualization

POLITICS AND DATA GO TOGETHER

- Political campaigns
- Political journalism
- Academic research
- Tech firms/startups
- Policy/government relations
- DC think tanks/non-profits
- Advocacy groups
- Social media, public relations
- Consulting and finance
- ...

WE'LL ANSWER IMPORTANT POLITICAL QUESTIONS

- As political scientists, we want to answer big questions about politics.
  - How can we measure ideology and predict voting in Congress?
  - How do assassinations affect national politics?
  - Can we predict presidential elections?
  - How does encouraging more women to run for office affect local politics?
  - Are peaceful protests more effective than violent protests?
  - Does broken windows policing work?

LEARNING GOALS

At the end of this course, students should:

1. Be comfortable using basic features of the R programming language.
2. Be able to combine political data with statistical concepts to answer political questions.
3. Know how to create visual depictions of statistical patterns in data.

SPECIFIC SKILLS WE'LL LEARN

We'll learn how to...

- Use and understand political data
- Link political questions to statistical solutions
- Write R code
- Import, clean, and manipulate data
- Calculate statistical quantities of interest
- Assess our confidence in these quantities
- Visualize data
- **Be an informed consumer of statistical evidence**

WHY WE'RE USING R

- It's relatively quick to learn the basics for data science
- It's FREE and open-source
- It's flexible
- R is also a full programming language; once you understand how to use it, you can learn other languages too.

HOW TO LEARN R: BE ACTIVE

- Traditional lecture approach bad for learning code
- You can only learn by doing
    - Lectures will introduce material and give you opportunities to practice.
- Programming takes practice.

LEARNING HOW TO PROGRAM

- Learning to program is like learning a language:
  - Vocabulary
  - Syntax
  - Grammar
  - Style
- There are many ways to do the same thing.
  - Not just one right answer.
  - Many different ways of communicating the same statement.

SYLLABUS / GRADING

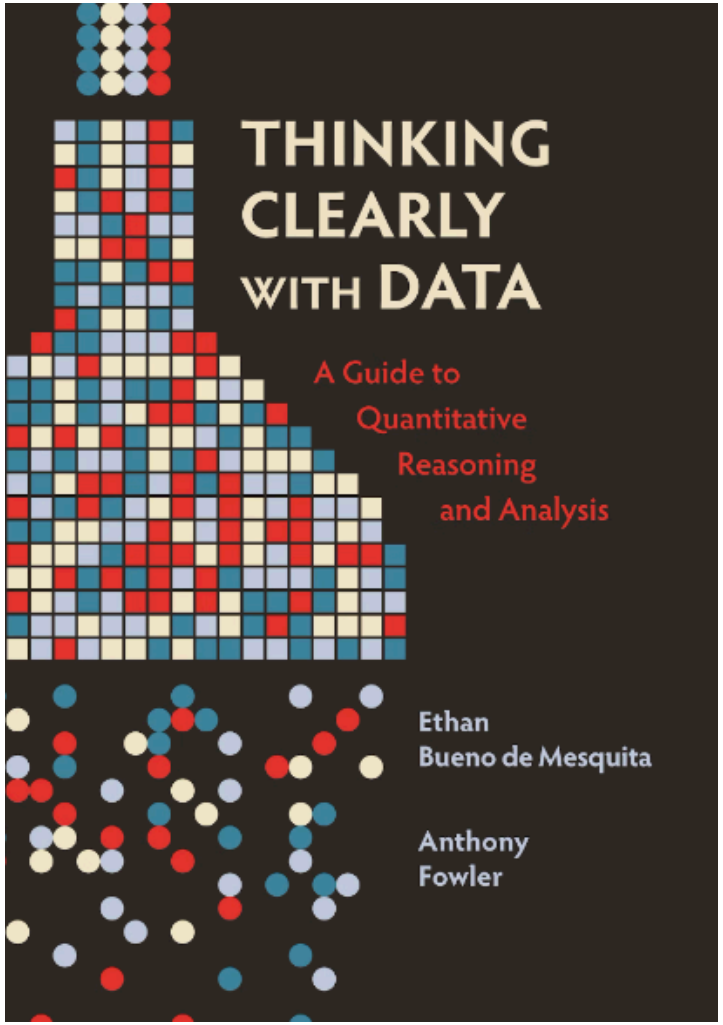
Syllabus

- Course website: Blackboard
- Office hours
  - Mondays and Wednesdays, 12:15 PM - 1:15 PM
  - Schedule other times via email
- Full syllabus on Blackboard. **Read it!**.

Grading

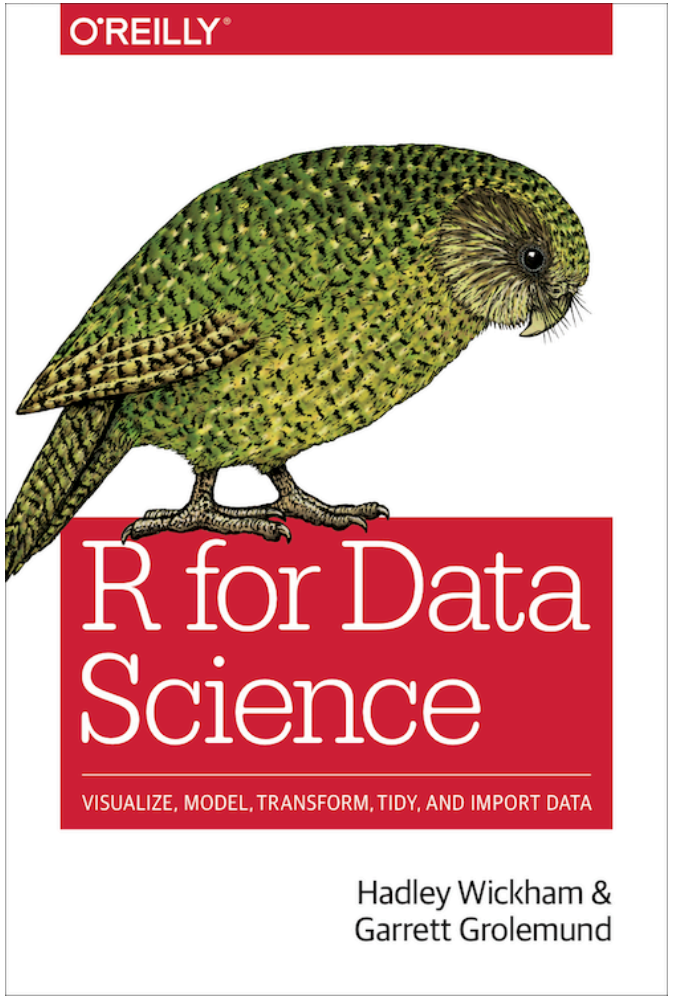
- Problem Sets (40%): Four problem sets. You can collaborate, but your submitted code and written answers must be your own.
- Take-home Midterm Exam (20%)
- Final Projects (35%)
  - Analyze a data set and present findings on a poster.
  - Work in pairs or on your own.
- Attendance and Participation (5%)

BOOKS



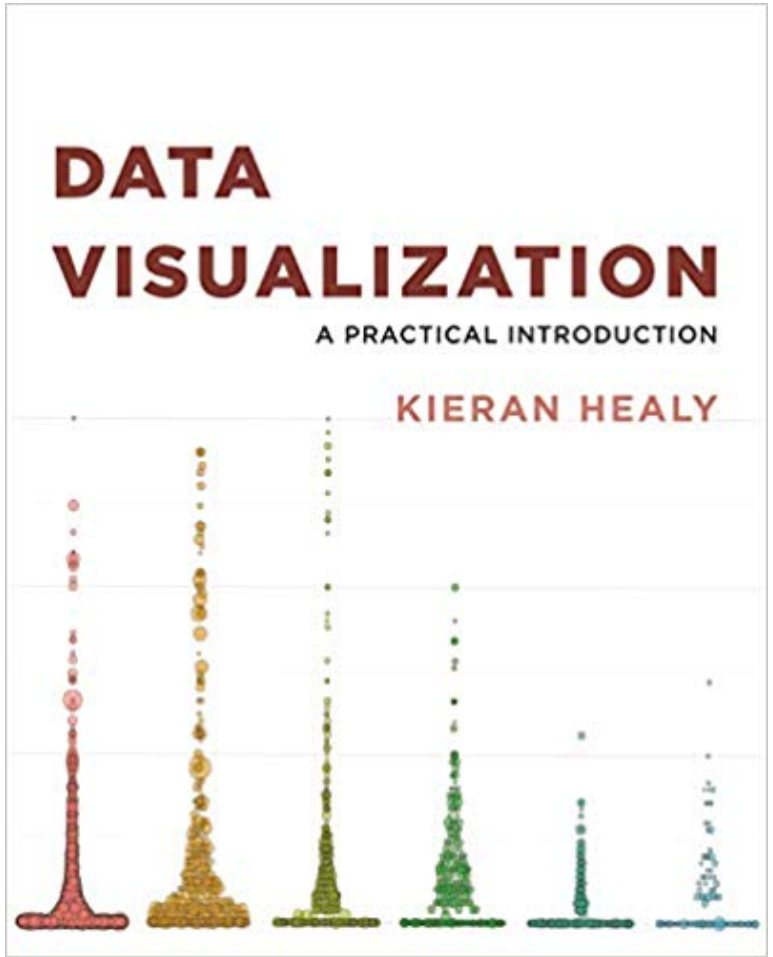
- Bueno de Mesquita, Ethan and Anthony Fowler. *Thinking Clearly With Data*. Princeton University Press, 2021.

BOOKS



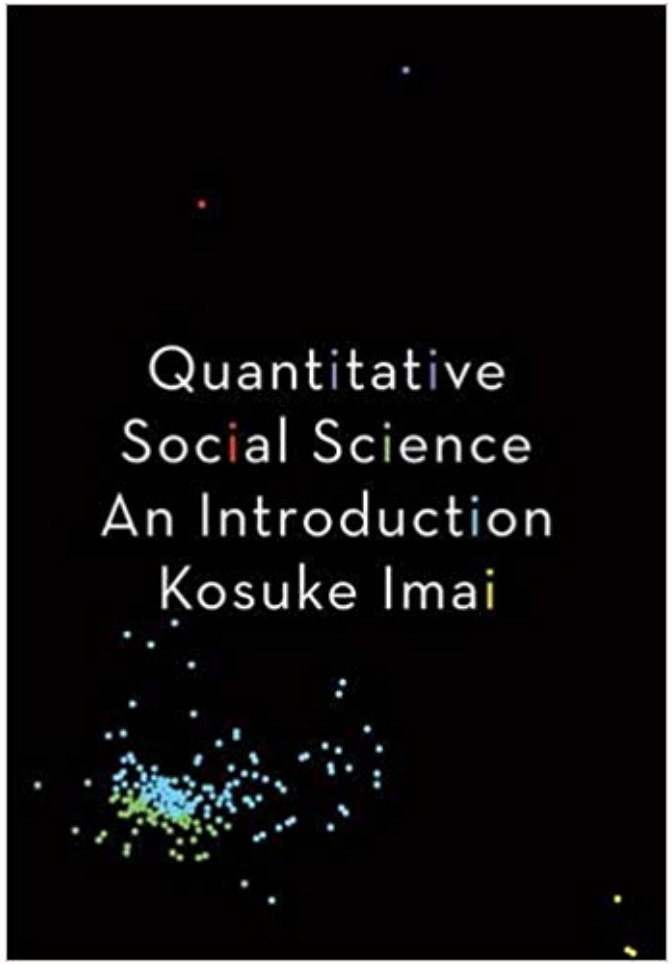
- Grolemund, Garrett and Hadley Wickham. *R for Data Science*. O'Reilly, 2017.
    - This book is free online at https://r4ds.had.co.nz/.

BOOKS



- Healy, Kieran. *Data Visualization: A practical introduction*. Princeton University Press, 2018.
    - This book is free online at https://socviz.co/.

OPTIONAL BOOK



- Imai, Kosuke. *Quantitative Social Science: An Introduction*, Princeton University Press, 2017.
    - There is also a new version, *Quantitative Social Science: An Introduction in Tidyverse*. Use this if possible.

HOW TO SUCCEED THIS COURSE

- This is a course for students with a wide range of backgrounds.
- You are not expected to know any programming or data analysis skills.
    - We will start on Friday with some basics of programming.
    - If you have programming experience, focus on mastering the analysis components and challenging yourself with more advanced topics.
- Success requires **time** and **effort**.
    - Learning new skills, language, and ways of thinking are not easy.
    - You will need to practice, beyond the course homework.
    - Textbooks include practice software and problems.
    - Google to answer programming questions.
    - R has great online resources and a very helpful community.
- Don't wait to ask for help.

TAKE THE FIRST CLASS SURVEY

- Take the first class survey (link)

INSTALL R AND R STUDIO

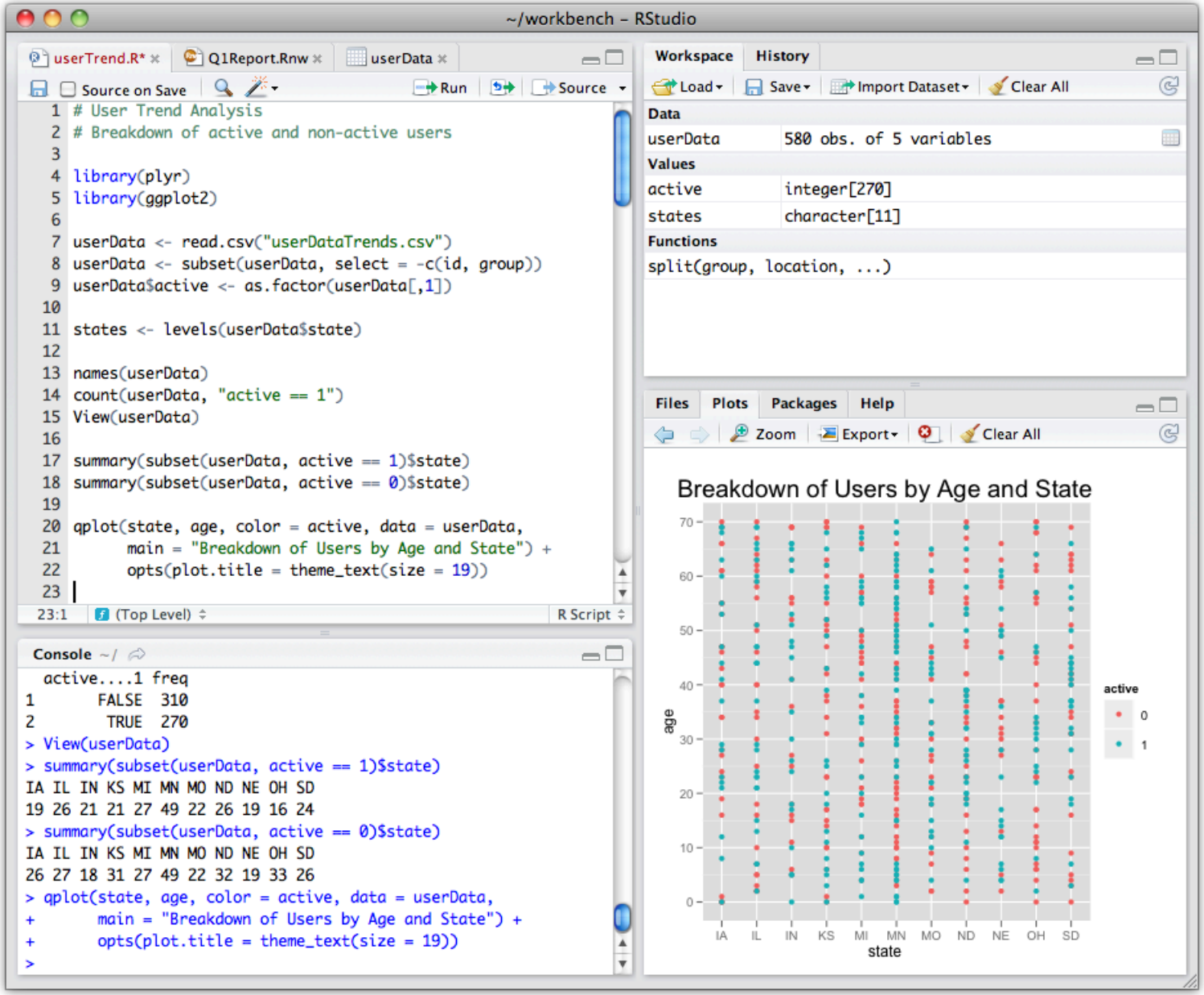
1. Install R (https://cran.r-project.org/)
2. Install RStudio (https://rstudio.com/products/rstudio/download/)

TODAY

- Using R Studio
- Lab Session
- Practice Problems

RSTUDIO

- RStudio is an IDE (integrated development environment).
- It includes all the tools you need to work in R.
    - Write code.
    - Run code and see results.
    - Create, load and save objects.
    - Make graphs and see them.
    - Search help files.
    - … and a lot more.


~/workbench – RStudio
userTrend.R*
Q1Report.Rnw
userData
Source on Save
Run
Source
1 # User Trend Analysis
2 # Breakdown of active and non-active users
3
4 library(plyr)
5 library(ggplot2)
6
7 userData <- read.csv("userDataTrends.csv")
8 userData <- subset(userData, select = -c(id, group))
9 userData$active <- as.factor(userData[,1])
10
11 states <- levels(userData$state)
12
13 names(userData)
14 count(userData, "active == 1")
15 View(userData)
16
17 summary(subset(userData, active == 1)$state)
18 summary(subset(userData, active == 0)$state)
19
20 qplot(state, age, color = active, data = userData,
21 main = "Breakdown of Users by Age and State") +
22 opts(plot.title = theme_text(size = 19))
23
23:1
(Top Level)
R Script
Console ~/
active....1 freq
1 FALSE 310
2 TRUE 270
> View(userData)
> summary(subset(userData, active == 1)$state)
IA IL IN KS MI MN MO ND NE OH SD
19 26 21 21 27 49 22 26 19 16 24
> summary(subset(userData, active == 0)$state)
IA IL IN KS MI MN MO ND NE OH SD
26 27 18 31 27 49 22 32 19 33 26
> qplot(state, age, color = active, data = userData,
+ main = "Breakdown of Users by Age and State") +
+ opts(plot.title = theme_text(size = 19))
>
Workspace
History
Load
Save
Import Dataset
Clear All
Data
userData 580 obs. of 5 variables
Values
active integer[270]
states character[11]
Functions
split(group, location, ...)
Files
Plots
Packages
Help
Zoom
Export
Clear All
Breakdown of Users by Age and State
age
state
IA IL IN KS MI MN MO ND NE OH SD
0 10 20 30 40 50 60 70
active
0
1

USING R AS A CALCULATOR


```
2+3
```

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

```
# order of operations is observed:
3*9+2
```

```
## [1] 29
```

```
# spaces don't matter:
2 + 3
```

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

```
4*  5
```

```
## [1] 20
```

```
(4^3+(3*5-9))/2
```

```
## [1] 35
```

creates comments: notes in your code that are *not* run.

BASIC OPERATIONS

Symbol	Use
<-	Assign value to object
==	Test equality
!=	Test not equal
&	And
|	Or
!	Not

```
x <- 1
y <- 2
```

```
x == 1
```

```
## [1] TRUE
```

```
x == 2
```

```
## [1] FALSE
```

```
x != 3
```

```
## [1] TRUE
```

```
x + y == 3
```

```
## [1] TRUE
```

```
x==1 & y==2
```

```
## [1] TRUE
```

```
x==1 | y==4
```

```
## [1] TRUE
```

OBJECTS

`<-` assigns values to a named object.

```
my_variable <- 5

another_variable <- "Max"

third_var <- TRUE
```

Objects are saved and can be referenced and modified.

```
my_variable + 3
```

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

```
another_variable == "John"
```

```
## [1] FALSE
```

OBJECTS

- Many ways to name variables.
  - `snake_case_uses_underscores`
  - `camelCaseUsesCapitalLetters`
  - `you.can.also.use.periods`
  - `thisisreallyhardtoread`
  - `thisIs_also.hardTOREAD`
- **Choose a system and be consistent.**

- Variable names must be *exact*:
  - Case sensitive
  - Identical punctuation
  - `my_variable` and `my_Variable` are two different objects.
  - If you get an error like this, check that the variable name is correct:

```
my.var
Error: object 'my.var' not found
```

OBJECT TYPES

- Numbers:
  - Integers: 1, 2, 10456
  - Doubles: 1.2, 3.99, -15.75769567
- Strings / characters:
  - Enclosed in single or double quotes.
    - "Joe Biden", 'Bernie Sanders'
    - Numbers in quotes are treated as strings:
- Booleans:
  - Object that can only have the value `TRUE` or `FALSE`.
  - Can be abbreviated to `T` or `F`.
- Groups of Objects:
  - The function `c()` creates vectors and lists:
  - Vectors are sets of the same data type.
  - Lists may have different types.

```
my_vector <- c(1, 2, 3, 4, 5)
```

PREPARING OBJECTS

- If you type the name of an object, R will print a version of it in the console.
- Simple objects (numbers, strings, lists...) will be printed.
- More complex objects may be printed differently (models.)

```
x <- "ABC"
x
```

```
## [1] "ABC"
```

```
my_vector
```

```
## [1] 1 2 3 4 5
```

FUNCTIONS

- Functions are blocks of code that do something.
- This function prints the alphabet:

```
print_alpha <- function() {
  print("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
}

print_alpha()
```

```
## [1] "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

- Functions often take arguments, which we list as variable names in the parentheses that follow the function name.
- The function then uses those arguments.
- Functions also *return* an object, which can be saved.

```
geometric_mean <- function(x1, x2) {
  y <- sqrt(x1*x2)
  return(y)
}

geometric_mean(5, 10)
```

```
## [1] 7.071068
```

FUNCTIONS

- R has many built in functions.
- Packages will add more functions.
- And we can also write our own functions.
    - We can use functions to save time.
    - Instead of writing the same code to do the same thing multiple times, write one function and then call it multiple times.
- Some uses of functions:
    - Create objects
    - Complex calculations
- Statistical models
- Graphs & Maps
- Generate output (e.g. tables)

FUNCTIONS

- Some useful starter functions:
  - `c()`
  - `length()` & `dim()`
  - `seq()` & `rep()`
  - `sum()`, `mean()`
  - `min()`, `max()`

```
sum(1, 2, 4, 5) # Adds numbers
```

```
## [1] 12
```

```
seq(1, 10) # Sequence of numbers
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(0, 10, 2) # Count by last argument
```

```
## [1]  0  2  4  6  8 10
```

```
c("A", "B", "C") # Create a list or ve
```

```
## [1] "A" "B" "C"
```

PACKAGES

- Packages extend R by adding new functionality.
- Install package with the code `install.packages("package_name")`
- After installing, load a package with `library(package_name)`

PACKAGES

- Tidyverse
  - A set of packages that make it easy to load, clean, and manipulate data.
  - Includes `ggplot`, a package for making graphs.
  - There are many other packages that do similar things.
    - We will use `tidyverse` because the syntax is relatively simple and readable.
    - There are faster options for more complex problems.

INSTALLING TIDYVERSE

- Install the `tidyverse` packages:

```
install.packages("tidyverse")
library(tidyverse)
```

Result should look like this:

```
> library(tidyverse)
── Attaching packages ─────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6     ✔ purrr   0.3.4
✔ tibble  3.1.7     ✔ dplyr   1.0.9
✔ tidyr   1.2.0     ✔ stringr 1.4.0
✔ readr   2.1.2     ✔ forcats 0.5.1
── Conflicts ──────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

- If you get an error:
  - Does it mention the `stringi` package?

```
install.packages("stringi")
install.packages("tidyverse")
library(tidyverse)
```

  - Something else?

GETTING HELP

- You can search the R help files by typing ? and a function name.

```
?sum
```

- The help files tell you what a function does, how to use it, and it's arguments.
- Help files often include sample code, which you can use to learn how a function works and then modify.

GRAPHS


```
library(tidyverse)
ggplot(mpg, aes(x=displ, y=hwy, color = class)) +
  geom_point()
```