Coding Lab: `Why code?` and getting situated

Ari Anisfeld

Summer 2020

Intro to coding lab

- Why are we here?
- What are we going to do?
- A quick introduction to R and R Studio and the `tidyverse`

Why coding?

Many public policy jobs and the Harris curriculum rely on programming

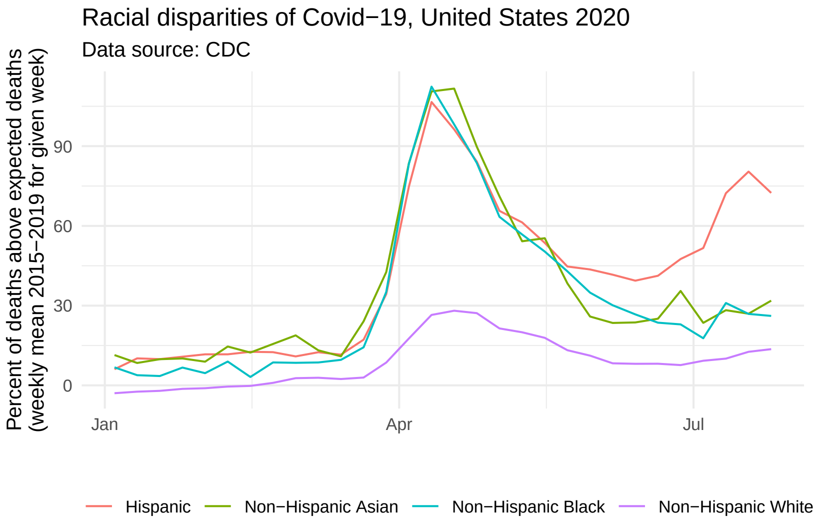
- to quickly engage with policy data
- to complete statistical analyses

Why R?

- Great data manipulation and visualization suite
- Strong statistical packages (e.g. program evaluation, machine learning)
- Complete programming language with low barriers to entry
- Open source and free

An example

I wanted to understand racial disparities of Covid-19.



What will we cover?

Foundations:

0. R, RStudio and packages
1. Reading files, and manipulating data with `dplyr`
2. Vectors and data types
3. If statements
4. Analyzing data with groups
5. Basic graph making (summer only)
6. Loops (in fall)
7. Functions (in fall)

In stats 1 and other courses, you will build off of these lessons:

- extend your capabilities with the functions we teach you
- introduce statistics functions
- introduce new packages etc. based on needs

Learning philosophy

- We learn coding by experimenting with code.
- Coding is requires a different modality of thinking
- Coding can be frustrating
- We develop self-sufficiency by learning where to get help and how to ask for help
- Coding lab is for you.

How will we progress?

1. Video lectures:
    - Have R open. Pause regularly.
    - Focus on main idea first.
2. Practice in labs (most important part):
    - You learn coding by coding.
    - Break up into small groups and work on problems with peer and TA support
3. Q and A (live session):
    - Please send me questions ahead of class
    - May include additional practice problems.
4. Final project: (see next slide)

Final project:

You'll know you're ready for policy school coding, if you can open a data set of interest to you and produce meaningful analysis. For the final project, you will:

- Pick a data set aligned with your policy interests (or not)
- Use programming skills to engage with data and make a data visualization showing something you learned from the data.

Getting help

- R's `?` documentation is very good, esp. for `tidyverse` code.
- Rstudio has useful cheatsheets for `dplyr` and `ggplot`
  - In the menu bar, select help > cheatsheets
- Get situated with R for Data Science https://r4ds.had.co.nz/
- google and stackoverflow are your friends for idiosyncratic problems
  - googling is its own skill
  - add "in R tidyverse" to your searches for better targeted help

A quick introduction to R and R Studio and the `tidyverse`

We will

- Discuss what Rstudio is
- Introduce minimal information to get started working with R
- Learn how to install and load packages
- Discuss what the `tidyverse` is

Getting started with R and R Studio

Please install R and R Studio. These are two distinct things!

We have provided information previously.

- On the first day, Harris IT will be available for troubleshooting installations.

What is RStudio?

R Studio is an "integrated developement environment" for R.

- It provides a console to access R directly.
- A text editor to write R scripts and work with Rmds
- An enviroment and history tab that provide useful information about what objects you have in your R session
- A help / plots / files / packages etc. section

Basic syntax: Variable assignment

We use <- for assigning variables in R.

```r
my_number <- 4
my_number
```

```
## [1] 4
```

Variable assignment

We can re-assign a variable as we wish. This is useful if we want to try the same math with various different numbers.

```r
my_number <- 2
my_output <- sqrt((12 * my_number) + 1)
```

Variable assignment

We assign all sorts of objects to names including data sets and statistical models so that we can refer to them later.

- use names that are meaningful

```r
model_fit <- lm(mpg ~ disp +  cyl + hp, mtcars)

summary(model_fit)
```

```
##
## Call:
## lm(formula = mpg ~ disp + cyl + hp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0889 -2.0845 -0.7745  1.3972  6.9183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.18492    2.59078  13.195 1.54e-13 ***
```

Using functions

Functions are procedures that take an input and provide an output.

```r
sqrt(4)
```

```
## [1] 2
```

```r
median(c(3, 4, 5, 6, 7 ))
```

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

Function arguments

Function inputs are called arguments.

Functions know what the argument is supposed to do based on

- name
- position

```
f <- function(x, y) {
      2 * x + y
}


f(7, 0)
```

```
## [1] 14
```

```
f(y = 7, x = 0)
```

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

Finding help with ?

```
?sum
```

- Description

```
sum returns the sum of all the values present in its
arguments.
```

- Usage (API)

```
sum(..., na.rm = FALSE)
```

- Arguments

```
...    numeric or complex or logical vectors.
```

- Examples (scroll down!)

```
sum(1, 2, 3, 4, 5)
```

what are packages?

A package makes a new set of functions available to you.

Benefits: - Don't need to code everything from scratch - Often functions are optimized using C or C++ code to speed up certain steps.

```
Analogy:
- base R comes with screw drivers and hand saws.
- packages give you power tools
```

installing and loading packages

To use a package we need to:

- install it once from the internet

```
install.packages("readxl")  # do this one time directly in
```

- load it *each time* we restart R

```
library(readxl) # add this to your script / Rmd everytime
read_xlsx("some_data.xls")
```

- `package::command()` lets you call a function without loading the library

```
readxl::read_xlsx("some_data.xls")
```

common package error

The package ‘haven’ provides a function to read dta files called `read_dta()`. What goes wrong here?

```
install.packages("haven")
our_data <- read_dta("my_file.dta")

Error in read_dta("my_file.dta") : could not find function
```

common package error

We need to load the package using `library()`!

```r
library(haven)
our_data <- read_dta("my_file.dta")
```

tidyverse: set of useful packages

Think of the tidyverse packages providing a new dialect for R.

```
library(tidyverse)

## -- Attaching packages -------------------------------
## v ggplot2 3.3.0  v purrr   0.3.4
## v tibble  2.1.3  v dplyr   0.8.5
## v tidyr   1.0.2  v stringr 1.4.0
## v readr   1.3.1  v forcats 0.5.0

## -- Conflicts ----------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Recap: Intro to R, RStudio and the `tidyverse`

After going through this video, you should understand how to

- navigate and use Rstudio's features
  - particularly, the console, the text editor and help
- assign objects to names with `<-`
- use functions by providing inputs and learn more with `?`
- `install.packages()` (once) and then load them with `library()` (each time you restart R)