Introduction / Thinking Like a Computer

Day One of Programming Workshop

Andie Creel (she/her)

January 2025

TF: Eliana Stone

My background

- Freshman in college: financial engineering (whatever that is) -> Java
  - “I am a magician”
- Majored in Econ, minored in CS
- TAed programming labs in college
- Pandemic Hobby: big data
- Master’s thesis: Parks use in pandemic (Cell Phone Data)
- Dissertation work: Environmental Econ -- value of local recreation and urban green space

My motivation for knowing how to program

- Sustainable development: "meets the needs of the present without compromising the ability of future generations to meet their own needs" -- Brundtland Report, 1987
- Needs: consumption of goods, health, political stability, culture
  - The environment affects this
  - That effect can be hard to observe/measure
- Explosion of data and computing power can help us observe and understand how nature affects sustainable development
- "What gets measured gets managed" – Peter Drucker (maybe?)

Eliana's background

- Majored in Environmental Studies and Economics
- Didn't start coding till junior year of college and felt a HUGE barrier to entry compared to peers
- Started with classes in R
- Now other languages like Python and Google Earth Engine JavaScript are easy to self-learn with online resources
- Master's thesis: Impacts of Agribusiness-aligned Politicians on Agriculture and Development in Brazil (remotely-sensed crop data and estimates of wealth, decades of municipal election data)
- Ongoing PhD research: Debiasing Remote Sensing for Casual Inference (remotely-sensed forest cover, machine learning models in R and Python)

spotify_reanalysis.R

```r
spotify_files <- c("Spotify Account Data/StreamingHistory0.json",
                   "Spotify Account Data/StreamingHistory1.json",
                   "Spotify Account Data/StreamingHistory2.json")

# Make a dataframe and remove most listened to podcasts
total_data <- map_df(spotify_files, ~fromJSON(read_file(.x))) %>%
  filter(artistName != "Up First" &
           artistName != "Ologies with Alie Ward" &
           artistName != "The Daily" &
           artistName != "Aware & Aggravated" &
           artistName != "Normal Gossip" &
           artistName != "True Crime & Cocktails")

# find top songs by number of milliseconds played
top_songs_time <- total_data %>%
  group_by(trackName, artistName) %>%
  summarise(total_msPlayed = sum(msPlayed, na.rm = TRUE))%>%
  arrange(desc(total_msPlayed))

# find top songs by number of times played
top_songs_count <- total_data %>%
  filter(msPlayed >= 30000) %>%  #spotify counts a play as more than 30 seconds
  group_by(trackName, artistName) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

# find top artists by number of milliseconds played
top_artists_time <- total_data %>%
  group_by(artistName) %>%
  summarise(total_msPlayed = sum(msPlayed, na.rm = TRUE)) %>%
  arrange(desc(total_msPlayed))

# find top artists by number of songs played
top_artists_count <- total_data %>%
  filter(msPlayed >= 30000) %>%
  group_by(artistName) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
```

7:36 (Top Level)

Introductions

- Name (pronouns)
- What you do at YSE
- Why you decided to take this workshop

Goal of next three days

- Rid the intimidation factor
- Build a foundation for other classes and opportunities to build on
- Accelerate the initial learning curve
  - I tried to build what I wish I had

- Advice: Ask questions
  - Believe that you're entitled to learn everything the next three days has to offer
  - Even when you don't know how to say it

Logistics

- 9:00 am to 12:30 every morning
  - Two 60 to 90-minute lectures per day
  - 30 min break between
  - Lunch!
- A mini problem set every day
  - Not graded, but Eliana will provide some feedback
  - I will release an answer key at 5pm
- Office hours
  - Location: Sage 8A
  - Time: 2 – 5 pm
  - Come to OH (even if you don’t know how to ask your question)!

Outline of Material

- Today
  - Thinking Like a Computer (pseudo code)
  - Base R
- Tomorrow – tidyverse packages
  - Data manipulation (dplyr, tidyr)
  - Data Management and visualization (ggplot2)
- Friday
  - Collaboration and Version Control (GitHub)
  - Programming is Programming (python)

Thinking Like A Computer

Skills to help you outline and (eventually) debug code

What is programming

1. Bits and bytes – what the computer actually speaks (0s and 1s)
2. Machine and Assembly languages -- translates our code to 0s and 1s
3. Code – R, Python, etc.
4. Software – Windows, iOS, Applications, Excel, R Studio

What is programming

1. Bits and bytes – what the computer actually speaks (0s and 1s)
2. Machine and Assembly languages -- translates our code to 0s and 1s
3. **Code – R, Python, etc.**
4. Software – Windows, iOS, Applications, Excel, R Studio

Some definitions

- Code: What your computer runs (different languages)
- Script: The text file where you write your code
- Comments: Notes you write yourself that the computer doesn’t read
- Run (aka compile and execute): when the computer executes your code
- IDE/GUI/Software
  - Where you write and run your scripts and comments
  - R Studio, Google Collab (Collaboratory), VS Code

Why should you think like a computer?

- Clear definition of “problems”
  - Cannot solve a problem that isn’t defined (how do you know it's solved?)
  - Forces precision and accuracy in problem definition
  - Collaboration: Once clear to you, can also be clear to team members
- Solutions to “problems”
  - Our field is filled with giant problems
  - To make progress, need to break them into solvable pieces
  - Learning to program is learning how to break big problems down
  - Steps are clear

Why should you learn to code?

- Efficiency and Speed: it’s so much faster
- Accurate: not reliant on copy-paste/find-replace
- Replicability: teams, peer review, new datasets
- Customize: write models for your data, clean and manipulate data for your models
- Employment: more jobs, better jobs, more money
- Collaboration: compared to Excel, GitHub on our last day

Pseudo Code – first step of coding

- What
  - Pseudo code is the outline of your code
  - Similar to the writing process
  - Not a coding language
  - I do it on scrap paper or in comments at the top of my scripts
- Why
  - Clarify your logic
  - Serves as road map and plan
    - Super important for multiple work sessions
  - Collaboration/Documentation
    - Can communicate what you've done/are doing to others

Exercise One: Coffee

Write instructions for how to brew a cup of coffee (3 min)

Exercise One: Coffee

Switch instructions with a partner

Have them “run” your code

Do you find any bugs? (5 min)

Exercise Two: Rock Paper Scissors

Write instructions for how to play Rock, Paper, Scissors

You will need multiple cases

(5 min)

Exercise Two: Rock Paper Scissors

Switch with a partner.

Play by the rules they just gave you.

Can you cheat??

(5 min)

My pseudo code (lazy)

Player: choose rock paper scissor

Computer: random generate rock paper scissor

Compare P & C

- If P = Rock
  - And C = Rock: tie
  - And C = paper: player loses
  - And C = scissors: player wins
- Same for if P = paper and P = Scissors

Not Lazy Pseudo Code

```
While play_again is true
    Prompt the player to select Rock, Paper, or Scissors
    Generate a random choice for the computer (Rock, Paper, or Scissors)

    If player's choice is the same as computer's choice
      Display "It's a tie!"
    Else If player chooses Rock and computer chooses Scissors
      Display "Player wins! Rock crushes Scissors."
    Else If player chooses Paper and computer chooses Rock
      Display "Player wins! Paper covers Rock."
    Else If player chooses Scissors and computer chooses Paper
      Display "Player wins! Scissors cut Paper."
    Else
      Display "Computer wins!"

    Ask the player if they want to play again
       set play_again to true or false
```

R Code

```r
# Function to get computer's choice
get_computer_choice <- function() {
  choices <- c("Rock", "Paper", "Scissors")
  sample(choices, size = 1)
}

# Function to determine the winner
determine_winner <- function(player_choice, computer_choice) {
  if (player_choice == computer_choice) {
    return("It's a tie!")
  } else if (player_choice == "Rock" && computer_choice == "Scissors") {
    return("Player wins! Rock crushes Scissors.")
  } else if (player_choice == "Paper" && computer_choice == "Rock") {
    return("Player wins! Paper covers Rock.")
  } else if (player_choice == "Scissors" && computer_choice == "Paper") {
    return("Player wins! Scissors cut Paper.")
  } else {
    return("Computer wins!")
  }
}
```

```r
while (play_again) {
  # Get player's choice
  player_choice <- readline(prompt = "Choose Rock, Paper, or Scissors:

  # Validate input
  while(!(player_choice %in% c("Rock", "Paper", "Scissors"))) {
    player_choice <- readline(prompt = "Invalid choice. Choose Rock, Pa
  }

  # Get computer's choice
  computer_choice <- get_computer_choice()
  cat("Computer chose:", computer_choice, "\n")

  # Determine and display the winner
  result <- determine_winner(player_choice, computer_choice)
  cat(result, "\n")

  # Ask if the player wants to play again
  play_again_input <- readline(prompt = "Play again? (yes/no): ")
  play_again <- tolower(play_again_input) == "yes"
}

cat("Game Over. Thanks for playing!")
```

Couple of notes on writing, debugging code

- “I minored in stack overflow”
- I still have to look things up constantly
- Knowing programming makes me better at:
  - Defining problems
  - Developing solution strategies
  - Solving harder and harder problems
- Documentation, Google, Stack Overflow, and ChatGPT can help you from there
- But if you don’t know how to define and solve problems, you won’t be able to take full advantage of these aids

Couple of notes on writing, debugging code

- “I minored in stack overflow”
- I still have to look things up constantly
- Knowing programming makes me better at:
  - Defining problems
  - Developing solution strategies
  - Solving harder and harder problems
- Documentation, Google, Stack Overflow, and ChatGPT can help you from there
- But if you don’t know how to define and solve problems, you won’t be able to take full advantage of these aids

ChatGPT and other AI tools

- Any of them could probably solve all of what we do in this class
- But they can’t solve everything you’ll eventually want to do
- You can’t get to the cutting-edge problems without basics
  - At the very least, you’ll be limited
- Learn basics so that you can get to hard problems, and use AI as a collaborative aid once there

What’s next

- 30 min break
- Base R