Programming

Background

Computer programming is hugely important for modern data analysis. It is also fun and nifty. It will extend your understanding of data analysis methods, allowing you to ask "What if?" and try methods out under various circumstances.

There are two ways to learn computer programming. You can read books or guides that take you step by step, or you can pick a goal and seek information as needed to accomplish that goal. Today's demo will take a goal-oriented approach, because it is fun and engaging. But lack of rigid structure can cause certain types of problems, so you have two responsibilities.

- If you get stuck, please say something! It will help you, and me, and any classmates stuck on the same issue.
    - One thing that might help is the solutions, but I ask that you do not copy-paste from these. Even if you write the same exact lines of code, type them out yourself. You'll learn more.
- Realize that today's session teaches you enough to be dangerous. If you decide to pursue programming further, you need to eventually become less dangerous by spending more time learning good practices. This is a great resource for scientific projects, and this is a great resource for the R programming language in particular.

Learning objectives

Today you will test-drive a programming language called "R". The goals:

- Use lists and dataframes in R.
- Encapsulate R code using functions.
- Test your functions.
- Use the R package `ggplot2` to display data.

Exercises

After doing the exercises below, email your chart and your R code as `LAST_FIRST_letter_counts.png` and `LAST_FIRST_fizzbuzz.R`.

1. It's easiest to use R within a dedicated code editor. My recommendation is Rstudio. Go to rstudio.cloud and sign up for an account. Start a new project in Rstudio. Or, if you have R or Rstudio installed on your computer already, you can use that instead.

2. Solve a variant of the infamous "FizzBuzz" exercise in R. You must write an R function called `FizzBuzz` that takes a number N as input and return a list of length N as output. (A list is an R data type that can hold different kinds of data, such as numbers, words, or matrices. Its nearest equivalent in some other programming languages might be called an array.) The list should contain the numbers 1 through N, mostly. But for each number divisible by 3, it should put "Fizz" instead of the number, and for each number divisible by 5, it should put "Buzz" instead of the number, and for numbers divisible by both 3 and 5, it should put "FizzBuzz" instead of the number. To do this, you'll need to learn some core features of R. Here's two approaches.
    - For a systematic introduction to R oriented towards statistical uses, check out Roger Peng's intro to R programming for data science. For today, focus on chapters 4, 13, and 14.
    - Another nice intro is here. It is less comprehensive but quicker to read. It gives an interesting perspective more oriented towards programming languages.
    - If you're impatient to get started, or if you have prior coding experience, just start trying things and use google to solve problems. For example, search "if-then statements in R" or "defining functions in R".
3. After you have a working or partly working solution:
    - Write out diverse test cases by hand. Check to make sure they are ok. If not, goto 2.
    - Compare your solution to another existing solution. Choose it from here or here.
4. This problem is optional. If you are running low on time, skip this and read Eric's results using this command.

    ```
    character_frequencies = read.csv("https://raw.githubusercontent.com/ekernf01/HEART_
    ```

    For the output of `FizzBuzz(1000)`, paste it all together using `paste`. Count up how many times each character appears using `strsplit` and `table`. Convert the results to a dataframe using `as.data.frame`. There's a great explanation of dataframes here.
5. Using the R package `ggplot2`, make a scatterplot displaying the character frequencies. Save the chart as a PDF or PNG. You can use this helpful cookbook or use the solutions to get started. The syntax of `ggplot2` can be counterintuitive, so 15 minutes before class ends, Eric will discuss some common problems.

Homework

- Read this post about FizzBuzz design choices.
- Browse through the collection of `ggplot2` charts here. Take a look at their names and ask yourself which types of comparison or analysis each chart is useful for.
- Do Exercise 4 if you didn't do it yet and if you still want to.