Accelerated Lab 1: Introduction to R

Harris Coding Camp

Summer 2023

The purpose of this exercise is two-fold.

1. to make sure you have R installed and running.
2. to start playing with code as soon as possible.

Experimenting and tinkering with code is how you hone these skills!

Note: some problems simply ask you to run code and think about the output, while others ask you to manipulate code or answer questions.

General Guidelines

You may encounter some functions we did not cover in the lectures. Try the help page first!

1. Type `?new_function` in your Console to open up the help page
2. Skim the Description, Usage and Arguments–sometimes these are confusing. It gets clearer with practice.
3. Scroll to the bottom to look at the examples! Run some of these in the Console!

Installation completion

If you have not yet, install R and R Studio. We have instructions on how to do so here.

1. If you have not yet, run the following code to install the `tidyverse`.

```
# download from the internet -- you only need to do this once.
install.packages("tidyverse")
```

2. This code returns an error. Why? Fix it and install the packages.

```
install.packages(haven)
install.packages(readxl)
```

Getting going

Open an R script so you can save your code for later!

3. In order to have access to `tidyverse` functions, you need to load the library. Run the following code.

```
library(tidyverse)
```

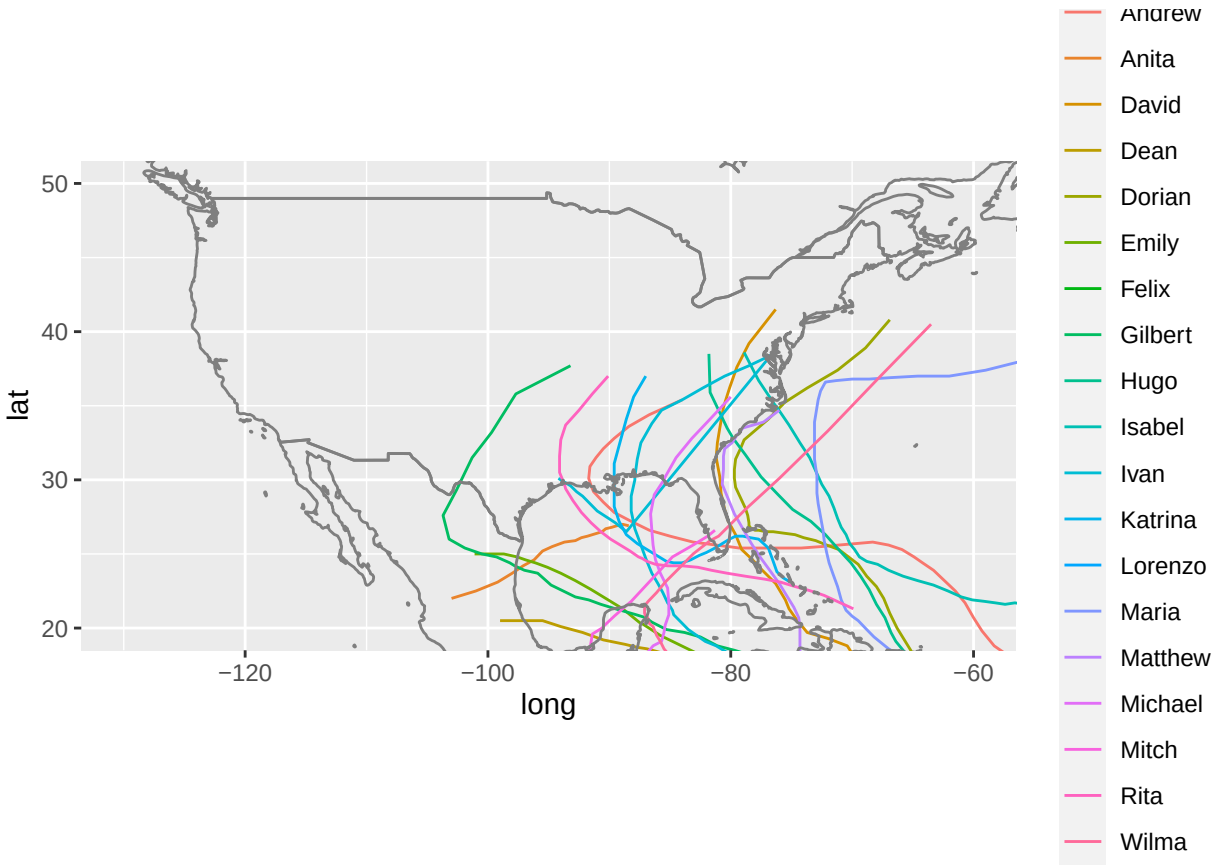
4. Run the following code to make sure you have installed successfully. Then, use the function `View()` to see the full dataset.

```
# storms is a dataset that comes with the tidyverse
# use View(storms) to see the full dataset
storms %>%
  group_by(name, year) %>%
  filter(max(category, na.rm = TRUE) == 5)
```

5. Assign the code above to the name `big_storms`. What is the difference between `storms` and `big_storms`?

6. This code makes the map seen below. If you run it you will get a message that says your code is missing a required package, R 4.1 will ask you to install the package. Choose yes. [This package is not essential – if the package fails to install, make a note of what error messages you get and then move on. You may consult with TAs after finishing the tasks below.]

```
# Heads up: copy and paste might mess up the quotes or other formatting
# look carefully if the code doesn't work.
ggplot(aes(x = long, y = lat, color = name), data = big_storms) +
       geom_path() +
       borders("world") +
       coord_quickmap(xlim = c(-130, -60), ylim = c(20, 50))
```



Warm-up

1. Which of these allow you to pull up the documentation for a command in R?

a. `*`
b. `?`
c. `help()`
d. `documentation()`

2. In the code block below, run code that will pull up documentation for the function `paste0()`.

```
?paste0()
```

What does this function do?

3. The second example in ?paste0 is

```
## If you pass several vectors to paste0, they are concatenated in a
## vectorized way.
nth <- paste0(1:12, c("st", "nd", "rd", rep("th", 9)))
```

This example uses a bunch of code/concepts we haven't covered yet!

a. Try to make sense of how the code works by running each input seperately. i.e. what does the code `1:12`, `c("st", "nd", "rd", rep("th", 9))` and `rep("th", 9)` do? And what does `paste0` do with it's inputs?
b. The function `paste0` is given two inputs here. What are they?
c. What does `paste0` do with two vector inputs?

4. *Assigning variables.* `score` is tracking your score in a game. The first round you got 3 points, the next round you got 2 points.

Your partner is keeping track of your score and wrote the following code.

```
score <- 0
# round 1
score + 3
# round 2
score + 2
```

a. What is `score` now?
b. Fix the code so that `score` is accurately tracking your score.

Comment: A common error for $_{\text{new}}$ programmers is to not assign output to a name!

Algebra

5. Guess the output of the following code:

```
a <- 3
b <- a^2 + 1

b
```

Now, run the code block to check your answer.

6. Guess the output of the following code:

```
a <- 10
b <- 3 %% a

b + 5
```

Hint: `%%` is the "remainder" or modulo function. If you are not sure what `%%` does, try to build intution with simple examples in the console.

7. Guess the output of the following code:

```
a <- c(1,2,3)
b <- a^2 + 1

b
```

Boolean/Logical

8. Guess the output of the following code:

```
25 >= 14
```

9. Guess the output of the following code:

```
10 != 100
```

10. Guess the output of the following code:

```
7 %% 5 == 2
```

11. Guess the output of the following code:

```
(5 > 7) & (7 * 7 == 49)
```

12. Guess the output of the following code:

```
sqrt(49) > mean(c(3, 5, 7, 9, 11))
```

13. Ok, let's try some logic! Try to figure out each one before running the code!

    a.

```
TRUE & FALSE
```

    b.

```
FALSE & FALSE
```

    c.

```
FALSE | FALSE
```

    d.

```
TRUE | (FALSE & TRUE)
```

    e.

```
(TRUE & (TRUE | FALSE)) | FALSE
```

Working with data and scripts

We recommend a file structure for coding lab. If you have your own preferred way of organizing code feel free to follow it.

Setting up working directory and coding environment

1. Do you have a folder on your computer for coding lab material?
   If not, create one and make sure you know the path to the folder.
2. We recommend creating a `problem_set` folder inside your coding lab folder.
3. Make folder called `data` inside the `problem_set` folder.

Putting your files in place

4. You've been working in a script! Save your script in the `problem_set` folder. For camp, when you start a script or `Rmd` save it there.
5. Download the first data set from `this link` and put the data in your `data` folder. Notice it is formated as an xlsx.

Tell R where to find files

- Local paths are like addresses on your computer.
- Use `getwd()` to see how your computer makes addresses.

6. Add a line at the *top of your script* where you `setwd()` to your problem set folder.

Usually we start our script by importing libraries and setting working directories. This allows others (and future you) to know what is necessary to work with the script.

7. Finally, we are using data in an excel format. We need the package `readxl` to process data of this type. Add code to load `readxl`.
8. You previously added code to load the `tidyverse`. Make sure that code is near the top of the script with your other library calls.
9. If you did everything correctly you should be able to run the following code:

```
fed_data <- read_xlsx("data/area_report_by_year.xlsx")
```

The path is *relative* to your working director. R looks for a `data` folder in your working directory and then for the data file in that folder. You could also give R an *absolute* file path, such as: `"/Users/John Doe/Coding Lab/problem_sets/data/area_report_by_year.xlsx"`.

However, note that this absolute path wouldn't work in someone else's computer, and also wouldn't work if John decides to move his Coding Lab files elsewhere, while the relative path will work just fine as long as the working directory is set.

10. Look at the data and notice it's a mess. If you can open the file in excel, you'll see that this is a workbook with many sheets of data. Here's some code to read in the student loan data.

```
fed_data <- read_xlsx("data/area_report_by_year.xlsx",
                      sheet = "studentloan",
                      skip = 3)
```

11. *Keep the good code.* Ideally, you want your code to reproduce the results from scratch. You should be able to restart R and then run the code in your script and have `fed_data` loaded.

    a. Check that your script has the code to setwd, load packages and then read in `fed_data` organized logically. (see example below).

    b. The other code you worked with today is fine, but calls to `View()`, `install.packages()` or anything that throws an error should be commented `#` out. You can comment many lines at the same time by highlighting them and then pressing the keys `shift + command + c` simultaneously.

    c. To restart R, go to `Session > Restart R` in the menu bar or push `shift + command + 0` simultaneously.

    d. Now run the script.

```
setwd(...) # your path instead of ...
library(tidyverse)
library(readxl)

fed_data <- read_xlsx("data/area_report_by_year.xlsx",
                      sheet = "studentloan",
                      skip = 3)

# other stuff from todays lab ...
# practice etc. ...
```