784 Lab 1: R Refresher

Eric Parajon

Code from today's lab was partially adapted from labs by Simon Hoellerbauer and Isabel Laterzo.

Today's lab will help you refresh some of what you learned in R in 783. It will also introduce you to some topics in R that you may not be familiar with, but are very useful for this semester.

For a refresher on how to use R Markdown, please see this cheat sheet: https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf.

Further see: https://posit.co/resources/cheatsheets/ For a list of all R studio cheatsheets (super helpful!)

Within your code chunks, comment our your code describing what you are doing in each step. This is an important habit to establish.

#R Review 1. First, it is important to save the document you are using in a central folder. Then, set your working directory to that folder. It's also important to clear your environment before working in a new document.

```
getwd() #what is your current working directory?

setwd(____) #in the parentheses, set the working directory to the folder where
#you save this file. Realistically, this isn't super relevant now because
#you're not reading in external data, but it's a good habit to get into!

rm(list=ls()) #clear global environ
```

2. First, create a vector of ten 5s using the function `rep()`. Second, create a vector that counts from 1 to 10 - do this in two ways, one using `c()` and one using `seq()`.

Third, create a vector of length 10 filled with NAs. Finally, uses `cbind()` to bind theses four vectors together to create a 10 x 14 (10 rows, 4 columns) matrix.

Remember, if you don't know how to use a function, type ? function name into your console to see the help file (e.g., `?seq()`).

Try binding the vectors together with `rbind()` instead. What happens?

3. Set your seed to 425. Simulate 100 draws from a Normal distribution with mean 2 and standard deviation 4, assigning the result to an object.

What proportion of the draws is above 3.1 in our sample? Think about using the functions `sum()` and `length()` to solve this, although there are other ways as well.

What is the expected proportion of draws above 3.1? Hint, look into the function `pnorm()`

Repeat this process, but this time take 10,000 draws from the same Normal distribution. How does our result compare with the expected proportion of draws above 3.1 now?

Thinking back to last semester, what did we just calculate using simulation?

4. Another important concept in R is a `for` loop. If you need a refresher on these, visit: https://www.r-bloggers.com/2015/12/how-to-write-the-first-for-loop-in-r/.

Using the 100 draws from the Normal distribution from Q3, create a new vector that is 1 when the vector of draws is greater than 3.1 and 0 when it is less than or equal to 3.1. First, do this using a `for` loop with an embedded `if else` sequence. What is the sum of this vector?

In addition to loops, `ifelse()` is also an important function. Look at the documentation of this function and repeat the above activity using `ifelse()`.

5. Practice: Load in the `swiss` dataset. A description for this data can be found here: https://stat.ethz.ch/R-manual/R-patched/library/datasets/html/swiss.html

How many variables does it have? How many observations?

The variable `Catholic` is the % of Catholic individuals, as opposed to Protestant in each province. First look at the summary of this variable. Then, create a new variable `Protestant` that is the percent of Protestant individuals in each province, just going with the assumption that all non-Catholics are Protestant here.

What is the mean percentage of this new variable?

6. Now, find out what type of object `swiss` is. Then, coerce `swiss` to a list by using `as.list()` and assign it to an objected named `swiss_list`. Look at the object with `View()` What happened? Use the `length()` function on `swiss`. What is the output? Why?

Create a matrix filled with random values that has 4 rows and 2 columns. Use `length()` on this matrix. Why might this be? Use `?` to read the documentation for `length()`.

7. Continuing to use the original `swiss` data, use a for loop to calculate the median of each of the variables. Store the result in a *named* vector (i.e., provide a name to each of the values so they are not just a bunch of random numbers with no context!).

Some useful functions here are `vector()`, `seq_along()`, and `names()`. Remember to use the help function `?` for further information about these functions.

8. Look into the three functions `apply()`, `sapply()`, and `lapply()`. What are the differences between these functions?

Repeat 7, but this time use the three above functions. Using each of these functions, examine what type of object they produce. How are they different from using a `for` loop? How are they different from each other in terms of what they produce? A useful function for examining different object types is `class()`.

9. Now we will explore `ggplot2`, which is a great package for plotting data and results. You will probably grow to have a love/hate relationship with `ggplot2`. For help with this package, reference this cheat sheet: https://res.cloudinary.com/dyd911kmh/image/upload/v1666806657/Marketing/Blog/ggplot2_cheat_sheet.pdf. Here is another useful link to a variety of graphs created using ggplot2: http://r-graph-gallery.com/

Reminder, load a package using library().

Using `ggplot`, create a scatterplot with the `swiss` dataset that plots `Education` against `Examination`. Add axes labels and a title.

Does it look like there might be a relationship between these two variables?

10. It is also useful to be able to write your own functions in R. Sometimes, R doesn't have a function for exactly what you are looking for. Doing so is pretty straightforward. For a quick review, see: https://swcarpentry.github.io/r-novice-inflammation/02-func-R/

For example, if I wanted to write a function that evaluated the equation

$$f(x, y) = xy$$

I could write one of the following options:

```
#example 1
fun_ex1 <- function(params) {
  x <- params[1]
  y <- params[2]

  result <- x*y

  return(result)
}

#evaluate
fun_ex1(c(3,4))


#example 2
fun_ex2 <- function(x, y) {

  result <- x*y

  return(result)
}

#evaluate
fun_ex2(3,4)
```

Now your turn. Write a function that evaluates the following equation:

$$f(x, y, z) = x^2 + 3xy + \sin(z)$$

Have this function takes a vector as its single argument (see example 1), made up of the three parameters (x, y, z). Evaluate the function at $x = 3/4, y = 13, z = 3.4$.

Math Review:

11. Let's try out some linear algebra. Below are 2 vectors, a and b, and two matrices A and B. Use them for this section.

Here is a link to a linear algebra cheat sheet in R (and other languages): https://github.com/scalanlp/breeze/wiki/Linear-Algebra-Cheat-Sheet

```
#a
a <- c(4, -5, 4, 1)
#b
b <- c(3, 2, -7, 17)

#A
A <- matrix(c(1, -3, 4, 5, 6, 8, 9, -10), ncol = 4)

#B
B <- matrix(c(4:1, 8, 14, -2, 1), ncol = 2)
```

What is the dot product of **a** and **b**? Does the result change if we take the dot product of **b** and **a**?

What are the dimensions for the matrix **AB**? Is this new matrix invertible?

What are the eigenvectors(vector whose direction remains unchanged when a linear transformation is applied to it) and eigenvalues (scalars associated with a linear system of equations) of this new matrix?

Verify that the values produced are actually eigenvalues.