Week 1 Lab problems

EEB 429
Bhaskar Kumawat

A. Working with paths

In the terminal for your operating system, do the following in order:

1. Go to the home directory
2. Print the current path
3. Print the list of files in the current directory
4. Create a directory called “EEB429” inside the home directory
5. Create a directory called “Week 1” inside the EEB429 directory
6. Go to the “Week 1” directory
7. Create a new file in this directory called “data.csv”
8. Print the current path
9. Print the list of files in the current directory

Either upload screenshots, or copy-paste the inputs and outputs from your terminal to receive full credit for this exercise.

(Optional extra reading on paths in R: https://www.r4epi.com/file-paths)

B. Can functions change their arguments?

Copy-paste the following code into an R-script, and

1. Annotate the code with comments, briefly describing what each line does.
2. Execute all the lines in the script.
3. What do you think the author of this code wanted to achieve? What ended up happening? (There is a hint in the title of this problem)

```
y <- 1
test_function <- function(x){
    x <- 2
    return(x)
}

y <- test_function(y)
y
```

4. Bonus: How would you fix this code to do what it was designed to do?

Copy-paste the annotated code, and write the answers to part 3 (and 4) to receive full credit for this exercise.

C. Vectors

In a new R-script file,

1. Create a new vector called “v” with elements 8,6,4,10,11.
2. List the objects that are currently in the working memory.
3. Create the following variables with given values: x=2, y=10, z=3
4. List the objects that are currently in the working memory.
5. Remove the objects “y” and “z” from the working memory in a single line of code.
6. Raise each element of vector “v” to the power of the value stored in “x” and store the resulting vector as a new object “z”
7. Divide each element of “v” by the respective element of “z”

Copy-paste your executed code and the outputs from the console to receive full credit for this exercise.

(Optional extra reading on vectors in R: https://bookdown.org/manishpatwal/bookdown-demo/vectors.html)

D.Tidy data and untidy data

A tidy dataset is defined as one where,

- Each **variable** has its own column.
- Each **observation** has its own row.
- Each **value** has its own cell.

Consider the following data:

Cases			**Population**		
	1999	**2000**		**1999**	**2000**
Afghanistan	745	2666	**Afghanistan**	19987071	20595360
Brazil	37737	80488	**Brazil**	172006362	174504898
China	212258	213766	**China**	1272915272	1280428593

This is how you might find data in a paper or an online article. Answer the following questions,

1. What are the variables in the data table?
2. What values do these variables take?
3. How many total observations are there in this data table?
4. Now, using the definition of tidy data, rewrite this table as a single **tidy** data table.

Write answers for questions 1-3 and submit either an excel screenshot or a picture of a handwritten table for question 4 to receive full credit for this exercise.

(Optional extra reading on tidy data: https://r4ds.had.co.nz/tidy-data.html)

E. Live annotation exercise

(Join the live-coding environment at: https://replit.com/join/tnfpzxnpuc-kmwtbhaskar)

I will write some code live to accomplish a task in R. Your job is to annotate lines and sections of the code to describe what these sections/lines do. Please also include your initials in the comment so I know you've annotated some part of the code. An example for annotating lines and blocks of code is given below. Try to be as concise yet informative about the code as possible. For example, "Check if an island is occupied" is both shorter and more readable than "A function that takes in an island dataframe and returns True or False depending on the value in the Occupied column"

```r
print("hello world") # Example of a line comment @Initials

test_function <- function(x){

  #--- Example of a block comment --- @Initials
  y <- x^2 # Square the argument x and assign it to variable y @BK
  z <- y/100 # Divide y by 100 and assign it to a variable z @BK

  return(z) # Return z back @BK
}

#--- Square 25, divide by 100, and print value --- @BK
test_var <- test_function(25)
print(test_var)

```