EDUC 641 Lab: Applied Statistics in Education and Human Services I

Lab 2: 10/4 and 10/5

## Importing data, exploring categorical variables, and exporting results

**Goals:**

1. Revise how to set-up an R-project and do it on your own
2. Data types in R
3. Import data
4. Identify the main features of a dataset
5. Describe categorical variables of your dataset (tables and plots)
6. ***Miscellaneous:*** Export tables and plots

**Note:**

1. Work in groups and discuss with your peers as you go.
2. Write down your questions as and when they arise.
3. Call for assistance.
4. Sit near the GE.
5. Everything inside the single quotes (‘) symbol below is a code. When typing the code in R, skip the quotes sign, and just type whatever is inside.
6. Run code through R Script line-by-line. To run a code, go anywhere on the line where you wrote the code and either click ‘Run’ using the mouse, or press Ctr/Command + Enter on your keyboard. The output will be printed on the console.

**Material aligned with the lab:** EDUC R Workshop - Modules 4, 5, 8, 9, 10

Tasks:

1. From last time: If not already, create an R-project for the lab. R project is basically making a folder to organize your work on your device through RStudio
2. First, decide where on your device you want to have a folder for the project (here, labs).
3. Then, from the RStudio application, click on Files -> New Project -> New Directory -> New Project.
4. Click on ‘browse’ and go to the location on your device where you want to make a new folder for the project. Note that a new folder will be made inside the location you choose.
5. Then, give a name to the Directory under the ‘Directory name:’ and click ‘Create Project.’
6. Now go to the location on your device and check if a new folder is created there. Also, check what is there inside the new folder. Do you see a file ending in ‘.Rproj’ in the folder?
7. Inside the project folder, now make the following new folders by left clicking on the mouse and selecting New Folder: “code”, “data”, “figure”, “table”. Now, you can save all the work related to the labs in this folder and in the respective folder.

*Note: You can open the project either by double-clicking the ‘.Rproj’ file in the project folder, or by clicking ‘File -> Open Project on RStudio’.*

1. Data types in R: There are four main datatypes in R - integer, character, numerical/double, and logical. This is **different** from the measurement scale you discussed in class. When you create an object in R, it will be one of these datatypes.
   1. In an R-script, create the following objects-

obj1 <- 2L  
obj2 <- "hi"  
obj3 <- FALSE  
obj4 <- 2

1. Check that the four objects are created under the Environment tab.
2. Use the ‘str()’ and ‘typeof()’ functions on the 4 objects and write their structures and types below -

i. obj1

1. obj2
2. obj3
3. obj4

Now, create some more objects and find their structures and types as well.

obj5 <- c(1,3,6,9)  
obj6 <- c("a","b", "c", "d")  
obj7 <- matrix(data = c(0,10,13,0,51,5,8,3,0),  
 nrow = 3,  
 ncol = 3,  
 byrow = TRUE)  
obj8 <- mtcars

1. obj5
2. obj6
3. obj7
4. obj8

*Take-home point: There are several types of objects you will work with in R. The most common in my experience are vectors (or arrays; obj5 and obj6) and data.frames.*

1. Import data set in R: We will use the data set you discussed in class.
2. First, download the dataset and the codebook from the class website and save it inside the *data* folder of your R-project,
3. Choose an object name for your data set. The general syntax for importing a data set is:

data <- read.csv("data/name\_of\_dataset.csv")

*Do this for your own data. Note that this will only work if you are in the correct R project and have the dataset in the data folder. This is also the benefit of using R-projects, you don’t have to use the full path.*

1. Make sure that the data set shows up on the environment.
2. Identify the key features of the dataset.
3. Use the following functions on the dataset object you created. Write down what each of the function is doing –

i. str()

* + 1. dim()
    2. colnames()
    3. rownames()
    4. head()

1. Based on your output above, answer the following -
2. How many columns are there?
3. How many rows are there?
4. Write the current datatype and measurement scale of each column (or, variable).
5. You can access specific rows and columns of the dataset as well. To do so, we use indexing with two arguments. The first part corresponds to the row and the second part corresponds to the column. Try the following code -

#Instead of data, you should write the object you created for the dataset.

data[1,]  
data[,1]  
data[1,1]

1. We can also access specific columns by using column names. Try this –

#Instead of data, you should write the object you created for the dataset. Instead of write\_column\_name, pick one of the columns from your dataset.  
data[,"write\_column\_name"]  
data$write\_column\_name

*The most common way to access columns is using the ‘$’ symbol as it means writing less code. This is how we will access columns from hereafter.*

1. Describing categorical variables
2. Identify the categorical columns of your data set using the code book.
3. Check the ‘str()’ of the categorical column (use the **data$write\_column\_name** syntax shown above as the argument). If the structure is anything other than a factor, we will convert the column to a factor (i.e., tell R that it is a categorical column) as shown in step d.
4. To convert the column into a factor, you need to know what are the unique values (levels) that appear in the column and what do they mean (labels). This should be checked in the codebook. Alternatively, run the ‘unique()’ function on the categorical function.
5. What are the levels of your categorical variable?
6. What are the labels?
7. Now, you are ready to convert the column into a categorical column. Remember the levels and labels identified above.

data$write\_column\_name <- factor(data$write\_column\_names,  
 levels = c(...), #write the levels from above  
 labels = c(...) #write the labels from above as characters separated by comma  
)

1. Summarize counts and proportions of the categorical column. Try the code –

table(data$write\_column\_name)  
prop.table(table(data$write\_column\_name))

1. Plot a bar chart to show the counts of your categorical column. Try the code. We will delve into ‘tidyverse’ in depth next week.

library(tidyverse)   
#If you don't have tidyverse installed, do install.packages("tidyverse") first  
  
ggplot(data) +  
 geom\_bar(aes(x = write\_column\_name), fill = "choose\_a\_color\_of\_your\_choice")+  
 labs(  
 x = "Give a meaningful x-axis label",  
 y = "Give a meaningful y-axis label",  
 title = "Give a meaningfule title"  
 ) +  
 theme\_minimal()

1. ***(Miscellaneous)*** Exporting your tables and figures: For assignment 1, you can either take a screenshot of your output, type out the tables, or export your results. Below, I’ll show you how to export your results.
2. First save what you want export into an object, like this,

counts <- table(data$write\_column\_name)

barplot <- ggplot(data) +  
 geom\_bar(aes(x = write\_column\_name), fill = "choose\_a\_color\_of\_your\_choice")+  
 labs(  
 x = "Give a meaningful x-axis label",  
 y = "Give a meaningful y-axis label",  
 title = "Give a meaningfule title"  
 ) +  
 theme\_minimal()

1. To export the table, use the following code. The .csv file will be saved into the table folder in the R project.

write.csv(counts, "table/counts.csv")

1. To export the plot, use the following code. The .jpeg/.png file will be saved into the figure folder in the R project.

ggsave(plot = barplot, "figure/barplot.jpeg")

**Summarize for yourself.**

Write all the functions you learned today to help yourself with the assignment!