BANA 7025
Data Wrangling with R
Coding Exercises for Week 1

Objective: Your group will use knowledge from today's class to clean a small data set! (cue air horn sounds)

Before we get started:

- Create a script called "coding_exercises_1_YYYYMMDD", where YYYMMDD is today's date.
- High five or fist bump everyone in your group. You're about to have fun! (Pro-tip: look at the other person's elbow while high fiving for a successful high five.)
- Ask your group members for help before running to Xiaorui or TA for any and every question. Yourgroup is very capable, intelligent, and supportive!

Let's get started!

1. Let's set up our RStudio session.
    a. Use the RStudio keyboard shortcut Ctrl + Shift + R (or whatever the Mac equivalent is) to create a section header in your script. An "Insert Section" dialog box should appear, asking you to input the name of the section header—call it "Session Setup" and press OK.
        - ➢ Notice the many dashes after the words "Session Setup". This section header organizes your code and allows you and others to quickly skim the structure of your code to understand what's happening.
        - ➢ Notice you can collapse the entire section if you want. (Figure out how to do that.)
        - ➢ Next to the *Source* button, you'll notice a ≡ button. Click it. What do you notice? Guess what happens as you create more section headers?

    b. Set your working directory to the folder where you downloaded today's course materials. Make sure this setwd() command is in the "Session Setup" section of code.
2. Now we need to import data and take a quick glance at it.
    a. Import the *customer_churn_data_cleaning.csv* file in your downloaded folder. Make sure you use the correct value for the *stringsAsFactors* argument!!!
    b. Look at the structure of your data set. What's good, what needs fixed?
    c. Look at the attributes of your data set. What do you see?
    d. Look at the first 5 rows of your data set. There are a couple of ways to do this but try using a function instead of subsetting.
3. You probably noticed two things after examining the structure of this data set: the variable names are not in proper R styling conventions, and one variable is not the correct data type. It's time to fix both issues.
    a. Change the variable names so that every letter is lowercase and variable names use snake_case. In other words, your variable names should be *customer_id*, *monthly_charges*, *total_charges*, *payment_method*, and *churn*.
    b. QA (quality assurance, or what some people call QC for quality check) every day! Check your variable names. Did your previous command work?
    c. The *total_changes* variable should be a numeric variable. Change its data type with the as.numeric() function, remembering to use the assignment operator to overwrite the variable.
    d. QA again: Look at the structure of the data set.

    e. Use a different function to find the number of rows and columns the data set has.

4. Missing values can ruin an analysis. Let’s take care of them.
    a. How many total missing values exist?
    b. How many missing values per column exist?
    c. After meeting with an important client—your boss—you decide to impute the median for missing values for each of the numeric variables. Do this in two commands (one for each variable).
    d. Your important client also wants you to replace empty strings (strings that look like “”) or strings that look like “—“ with the value “Not stated”. Use the | operator or the %in% operator to perform logical subsetting.
    e. Do any missing values remain?
5. Our data is just about clean but let’s check some quick numeric and visual summaries.
    a. Look at a summary of the entire table. Why do only two columns have useful information?
    b. Make a table of values for each character variable. Do these tables look good?
    c. Look at a numerical summary of just the *monthly_charges* variable. Do you notice anything?
    d. Look at a histogram and a boxplot of this variable. Do these visualizations support your numerical findings?
    e. Replace that large outlier with 0. (Maybe code something like “if any value is less than -10 then replace it with a 0”.)
    f. Look at the summary, histogram, and boxplot again. Is this better?
    g. Look at the summary, histogram, and boxplot for the *total_charges* variable. What, if anything, do you need to fix?
6. Your data is finally clean! (Well….pretty much. We *could* use some advanced regular expression functions to separate the *customer_id* variable into two separate variables but we’ll learn later some Tidyverse functions to do that.)

   But your important client has one final question: Is there a bivariate relationship between *monthly_charges* and *total_charges*? What could you look at?
7. Have one and only one person in your group send Xiaorui your .R script via Canvas. Don’t send thisto everyone else in the class (meaning don’t send your .R script in the general or week 1 channels).
8. You’re done for the day and are dismissed! Have a great week!