Homework 1: Introduction to R

Overview

For this first homework assignment, you will be creating your own data within R rather than using a data set. Starting next week, your assignments will use real-world data, but for now, familiarizing yourself with R will be most beneficial.

Coding Assignment

1. The goal of this question is to create your own tibble with three columns. Your tibble should look like Table 1.

   a) Create a vector named column_one which has five elements: 8, 10, 4, 1, and 30.
   b) Create a vector named column_two which has five elements: "hello", "welcome", "to", "Data", and "Hack". Recall that capitalization is essential to programming languages.
   c) Create a vector named column_three. It should have the following five elements: 0, 0, 17, NA, and 15.
   d) Using the `tibble` function, combine these vectors to so they look like Table 1 (Have you called all the relevant libraries?). Save this tibble as tibble_one. We will be utilizing this tibble later on in the homework assignment. Note the naming convention we have established as well: all variables have lower case letters, underscores for spaces, and do not start with numbers. This naming convention is called snake_case. Alternatively, there is also camelCase naming convention. There is no benefit to one over the other, although all of the assignments in this class will be using snake_case. The most important rule is to adopt one naming convention, and stick to it!
   e) Data wrangling requires extensive use of Google or R's in-house help functionality. Here, we will use a new function we have never seen before named `typeof`. You can type in the command `?typeof` and press return in your R console, or look up the function yourself in Google. Essentially, this function finds the data type of a column or vector. Note that the three most common data types we will see are "character", "double", and "logical". Characters (or "strings" in other languages) are interpretted by R as text, doubles (or "floats" in other languages) are interpretted by R as numbers and can have mathematical computations performed on them, and logicals (or Booleans) are logical values that have specific purpose for evaluating a condition (we will learn more about later in the course). Using quotation marks around text tells R that you want a character data type, using no quotation marks with numbers tells R that you want a double data type, and there are special logical values that R will recognize (e.g. `TRUE`, `FALSE`). Using tibble_one and the `typeof` function, assess the data type of column_one. Save the output as a variable named data_type_one.
   f) Using tibble_one and the `typeof` function, assess the data type of column_two. Save the output as a variable named data_type_two.

Table 1: Example of tibble_one

column_one	**column_two**	**column_three**
8	hello	0
10	welcome	0
4	to	17
1	Data	NA
30	Hack	15

2. This question will involve finding basic summary statistics of tibble_one that you created in Question 1.

    a) Find the mean, standard deviation, and variance of the `column_one` column. Store each of these results in a vector named summary_stats_column_one. The ordering that you put these values does not matter. **Hint: when creating vectors, you can put commands into the vector. For instance, you could type c(mean(variable_one), mean(variable_two)) and you would create a vector composed of two means.**
    b) Find the mean, standard deviation, and variance of the `column_three` column. Store each of these results in a vector named summary_stats_column_three. Recall the `na.rm` argument.

3. Most frequently, you will not be making your own tibbles, as data typically comes from an outside source where someone has hired you to analyze it for them. This question will introduce you to reading in an outside data source, and doing some very light cleaning.

    a) Before we begin cleaning, we want to load in an R package that we will be using throughout the course to help us with cleaning. R packages are a collection of functions that users have written to make life easier for everyone else. In other words, someone else solved a common problem, and decided to share their solution with everyone else. For this question, we will be loading in the `tidyverse` package, which is actually a collection of many packages. Use the `library` function to load in the `tidyverse` package. What packages are attached when this command is run? (look for the message "Attaching core tidyverse packages" in the Console) Create a new vector named tidyverse_packages where each element of the vector is the name of a package. Be sure that your vector is a "character" data type, and that you follow the exact capitalization of the package names.
    b) Notice that one of the packages attached is the `readr` package. This package contains functions for reading in data. Use the `read_delim` function to read in the data `assign_1.csv` and assign this data the name school_crime.
    c) Now that the data is loaded in, you can click on the `school_crime` in the upper right hand corner of your Environment in RStudio and browse. Observe that the data essentially looks like an Excel spreadsheet (and it was beforehand!). Now, notice that we can see the column names in this preview. However, we can also gather the column names with code. Use the `colnames` function to find the column names of school_crime and save the output as school_crime_colnames.
    d) Notice that the column names of school_crime are not in our snake-case convention (e.g. they have capital letters, spaces, and special characters). We want to change the column names to be snake-case. The `tidyverse` package, while incredibly versatile, does not have a function to do this. Instead, we will need to load in the `janitor` package and use the function `clean_names` on our school_crime data, and save the updated version. Hence, load in the janitor package, and use the `clean_names` function to clean the column names of school_crime to snake-case. Be sure to update the school_crime data and the school_crime_colnames output to reflect these changes (you can click on the preview in the Environment to see if the changes have taken place).
    e) The `is.na` function checks whether there are any NA values in a column. Perform this function on `location` of school_crime and save the output as na_location. You may need to use either Google or R's in-house help function `?`. How many NA values are there in the column `location`?