**Data Science for Social Scientists**

PSYC 546, Spring 2023

Week 2 – In-Class Assignment

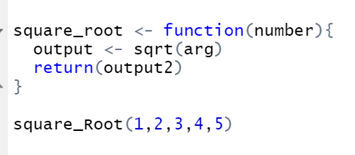
**Due Date**: February 1st (by 8:15 PM)

**Reminder**: See the assigned Week 1 readings and the lecture slides for tutorials on how to use R to perform the various functions included in the homework assignment below. **Once completed, you should submit a completed version of this document and your final R script file to the Homework Assignment 2– Submission Portal on Canvas**.

**R/RStudio**

Your submitted R script file should contain code to answer all of the questions below (except for Question 1). Please use comments (e.g., #Question 2) to label the code for each question.

1. Question 1 does not require any coding. Instead, it is to help gain training in another common task of a data scientist—namely, debugging someone else’s code that is not working. Below, is a custom function that simply takes the square root of a provided number(s). There are four errors/bugs in the code that are preventing it from working properly. Identify these problems and briefly describe each one in the table below. If you are having any trouble locating the errors by just looking at the function, you can always program the function in R yourself and use the error messages to guide identification of the bugs. [2 points overall]



|  |  |
| --- | --- |
| Problem 1 | The argument has been labeled differently. The “number” on the first line of code is not the same as “arg” on the second line. |
| Problem 2 | The output has similar issues. The “output” on the second line of code is not matching “output2” on the third line after return |
| Problem 3 | On the second line after output “ <- ” has been used instead of “ = ” |
| Problem 4 | The square\_root function cannot take several arguments at the same time unless it’s put into a vector |

1. You will need the **tidyverse** package loaded for this question. The data will be the **survey.csv** data set on Canvas that we have used before. Imagine you are annoyed with the fact that there are over 100 variables in the data set. For your analyses, you only plan on using the **age**, **educ** (education), **Mposaff** (positive affect scale score), and **Mnegaff** (negative affect scale score). Moreover, you want to restrict your analyses to people that are less than 40 years old. Your collaborator suggests making copies of the data file where you remove the columns and rows manually. However, in a single line of code (i.e., using the pipe operator), go about selecting the columns/variables you wish to retain and filter out the rows based on the age cutoff. You do not need to do any subsequent analyses on this reduced data set, but you should be able to confirm that everything was performed properly. [2 points]
2. You will need the **tidyverse** and **dslabs** packages loaded for this question. Use the data() function to load the **heights** data set from dslabs. This data set contains self-reported heights (in inches) for a sample of males and females. In a single line of code (i.e., using the pipe operator), go about grouping the data by sex and summarizing the heights. The following four descriptive statistics should be summarized and outputted together: the mean heights, the standard deviations, the minimum heights, and the maximum heights. As a default, the column names in the outputted tibble in R will not match the ones in the table below. See about finding a way (e.g., consult the readings) to make the column names match the names in the table below exactly. Finally, fill in the table below with all the outputted descriptive statistics. [2 points]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| sex | Average | SD | Min | Max |
| Female | 64.9 | 3.76 | 51 | 79 |
| Male | 69.3 | 3.61 | 50 | 82.7 |

1. In Question 4, we are going to combine our knowledge of for loops and if/else statements. Specifically, we will nest an if/else if/else statement in a for loop. Imagine that you are an instructor with 10 students in the class. The students take a test that is on a 0 to 100 scale. Scores >= 80 are considered above average, scores >= 60 but less than 80 are considered average, and scores below 60 are considered below average.

First, create an object called **test\_scores** that contains a vector of 10 student test scores. Feel free to come up with 10 scores of your choice. Create a for loop that iterates through each score in test\_scores. Then, within the body of the for loop, make an if/else if/else statement based on the above three score classifications. For each of the three possible outcomes, simply have the if/else if/else statement print out the classification descriptions included above. [2 points]

1. Let us get some more experience creating a custom function. Imagine you work on a personnel selection team in a mid-size sales company. The company has created an application test that measures knowledge of best sales practices and techniques. Based on past years, the company has found that scores on this application test significantly predict subsequent first-year sales performance in the company.

Imagine a simple linear regression equation is being used (i.e., Ŷ = mx + b). Past data suggests that the slope coefficient is 2.75 and the constant (y-intercept) is 37.253. Scores on the application test can range from 0 to 20 (with higher scores being better). The predicted performance scores are on a scale from 0-100% that is used by the company to rate the performance of sales representatives after their first year.

Your task is to create a function that simply takes a number(s) on the application test and returns the linear predicted value of subsequent sales performance. Name the function **performance\_prediction**. The function will have a single argument (arg) that will end up either being a single number (or a vector of numbers). In the body of the function, create an object that estimates the linear predicted performance score. Then, return this output object.

Call this custom function with the scores of five recent applicants to the company (see below for the scores). Put these predicted performance scores in the 3rd column. Finally, imagine that first-year sales performances are categorized as <70% (not acceptable), 70-79% (acceptable), and >=80% (good). Classify the predicted performance scores in the 4th column (this does not need to be done through code, just manually based on the scores estimated in column 3). [2 points]

|  |  |  |  |
| --- | --- | --- | --- |
| Applicant | Application Test Score | Predicted Performance  Score | Predicted Performance Classification |
| 1 | 14.5 | 77.128 | acceptable |
| 2 | 7 | 56.503 | not acceptable |
| 3 | 20 | 92.253 | good |
| 4 | 10.5 | 66.128 | not acceptable |
| 5 | 16 | 81.253 | good |