*\*Please save this file as “LAST NAME\_Assignment 2.docx”*

***Open-Ended Responses***

1. **II.1** Which of the following would **not** be an example of data wrangling? (Highlight correct answer)
   1. Filtering a tibble based on a condition
   2. Converting data to long form
   3. Computing a t-test statistic
   4. Adding new variables to a data.frame object
   5. Joining two tibbles together
2. **II.2** We discussed how tidy data is rectangular. Why doesn’t R allow you to have data.frames or tibbles that are non-rectangular? For example, the following will generate an error:

tibble(x = 1:10, y = 1:9) # error

|  |
| --- |
| R has to have rectangular data because there must be a y value for every x value, otherwise the data will repeat or produce NA values. This will throw an error when you attempt to make a dataframe with varying lengths of x and y. |

1. **II.3** Imagine my rawdata.csv file lives in a folder like Documents\Work\Dissertation\Survey. When importing this data, what is the advantage to running lines 2-3 as opposed to just line 1?

1 read\_csv(“Documents/Work/Dissertation/Survey/rawdata.csv”)

2 setwd(“Documents/Work/Dissertation/Survey”)

3 read\_csv(“rawdata.csv”)

|  |
| --- |
| Setting your working directory prior to importing data ensures that all future data gets exported to the same working directory that you are importing your current data from. It makes sure all of your data, imports, and exports are in the same folder and that R knows where that folder is. |

1. **II.4** Convert the following code into (a) full length sentence(s) that accurately and completely describe what is happening:

copus %>%

filter(!is.na(Size), Level == “100”) %>%

select(Size, Level, L) %>%

group\_by(Size) %>%

summarize(Max = max(L), Min = min(L)) %>%

mutate(Range = Max – Min)

|  |
| --- |
| From the copus dataset, this code filters out all NA values from the Size column and filters so that only columns with Level equal to 100 are utilized. It then selects this data so that only columns Size, Level and L are utilized. It then groups by size so that all large’s, smalls, etc are together in the data set. It then asks for the Max L and Min L to be printed out. Finally, it adds a column called Range that subtracts the min from the max and produces the range value in the copus dataset. |

1. **II.5** I collected 3 measurements from 3 different participants; the data are shown below. Highlight the word that correctly completes the sentence:

*Participant 1 has explicitly / implicitly / no missing data; Participant 2 has explicitly / implicitly / no Participant 3 has explicitly / implicitly / no missing data.*

|  |  |  |
| --- | --- | --- |
| **Participant** | **Measurement** | **Score** |
| 1 | 1 | 10 |
| 1 | 2 | 9 |
| 1 | 3 | 10 |
| 2 | 1 | 8 |
| 2 | 2 | 7 |
| 3 | 1 | 9 |
| 3 | 2 | NA |
| 3 | 3 | 10 |

1. **II.6** I have two tibbles, df1 and df2 that I’m looking to join by a common ID. I did this two ways, shown in the code below. Besides the fact that option #2 is more efficient (less code), what is an advantage to option #2?

df1 <- tibble(ID = 1:10, Score1 = rnorm(10))

df2 <- tibble(ID = sample(1:10), Score2 = rnorm(10))

# join option #1

df.joined <- df2 %>%

arrange(ID) %>%

mutate(Score1 = df1$Score1) %>%

select(ID, Score1, Score2)

# join option #2

df.joined <- df1 %>%

left\_join(df2, by = "ID")

|  |
| --- |
| Option 2 is better because you are not manipulating either of the original datasets, whereas in option 1 you are changing the datasets to only show ID, Score1, and Score2. Therefore, you no longer have your original datasets to run statistics on unless you reupload them. |

**II.9** Some people don’t like the default TukeyHSD() function in base R; they find the multcomp::glht() function more efficient. If I ran a general anova model and then wrapped it in a glht() call and stored that in an object called comp, how could I access a table of the covariance matrix?

|  |
| --- |
| You could simply run a summary of comp, and that would show you the tests for linear hypothesis and the covariance matrix. |

***Coding Section***

To complete this section, start a new script file with the following layout:

# YOUR NAME

# Assignment 1 Introduction to R

# #1 ---------------- (new section: CTRL + SHIFT + R)

here’s my code # with adequate commenting

# #2 ---------------- (new section: CTRL + SHIFT + R)

here’s my code # with adequate commenting

For the following questions, use the data from “df.csv” found in the data director on [GitHub](https://aub.ie/R22).

1. **II.3** Import the df.csv data to an object called “df”.
2. **II.4** In one continuous statement connected by piping operators, making the following changes to df and store it as df2:
   1. Rename “X1” as “ID”
   2. All negative values in Variables V2 though V6 should have been positive (all values will be positive for these variables); change this
   3. Any obs with V7 less than -0.9 is an outlier; replace these instances with NA, but do not remove the rest of their data. *Hint: If you are getting an error such as “`false` must be a logical vector, not a double vector,” check out the following and replace NA with NA\_real\_ (*[*https://stackoverflow.com/questions/53636644/r-if-else-assign-na-value/53636776*](https://stackoverflow.com/questions/53636644/r-if-else-assign-na-value/53636776)*).*
   4. Get rid of all observations where V1 is “D”
   5. Sort the data by V1 (A to C) and within each category, in decreasing V2
   6. Calculate the mean and standard deviation of V2 for each category of V1
3. **II.5** Ignoring all modifications in #2 and starting from a fresh import of df.csv, modify the data into an object called df.l that matches the screenshot here:

Text

Description automatically generated with medium confidence

1. **II.6** Create this dataset in R (call it “key”) and join it to df.l (from #3) so every obs in df.l that has V1 = “A” and Variable = “V2” should be assigned a NewValue = 1; every obs in df.l that has V1 = “A” and Variable = “V3” should be assigned a New Value = 2… and so on. Here is the desired, joined data. While you can do this with a LOT of chained if\_else statements… do you really want to? No, no you don’t.

A screenshot of a computer

Description automatically generated with low confidence

Question 5 is about a **different** df dataset (defined below):

1. **II.7** Study the code section below that has one line that has been redacted. What is the redacted line?

Text

Description automatically generated

Finally, Question 6 pertains back to the copus.csv data:

1. **II.8** One problem (of many) with the following plot is that the x-axis should be presented in the order of “Mostly lecture”, followed by “Transitioning”, and finally “High engagement”. Modify the copus data (Bcluster is on the x-axis) so that this variable will show up correctly in the plot.

Chart, box and whisker chart

Description automatically generated