*\*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

C. Computing a t-test statistic

1. **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

There needs to be an equal number of “X” and “Y” variables within tibbles in order to maintain the order within the dataset

|  |
| --- |
|  |

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”)

The “read\_csv” code allows you to access any folder and creates a tibble, making the data more available

|  |
| --- |
|  |

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)

This code filters out all of the NA’s within the size category then selects the Size, Level, and L. Grouping by size, then summarizes the data into the minimum and maximum of L to then mutate to subtract the maximum and minimum of L.

|  |
| --- |
|  |

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 |

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

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 Doesn’t affect the original data because not arranging, mutating, or selecting the scores and is more efficient by requiring less code.

**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?

Need to summarize the data to access the information by coding “Summary (comp)” to receive an output of the anova

|  |
| --- |
|  |

#Suzanne E. Tenison

#Assignment 2 Introduction to R

# 1 ------------------------------------------------------

#Imported df.csv data and called object "df"

library(tidyverse)

library(dplyr)

#2--------------------------------------------------------

df2 <- df %>% #rename "df" to be "df2"

#a.) Rename "X1" as "ID"

rename("ID" = "...1") %>% # There is no "X1" in this file

#b.)All negative values in Variables V2 - V6 should have been positive

#All values will be positive for these variables

mutate(V2 = abs(V2)) %>% # Take absolute value of V2, make values positive

mutate(V3 = abs(V3)) %>% # Take absolute value of V3, make values positive

mutate(V4 = abs(V4)) %>% # Take absolute value of V4, make values positive

mutate(V5 = abs(V5)) %>% # Take absolute value of V5, make values positive

mutate(V6 = abs(V6)) %>% # Take absolute value of V6, make values positive

#c.)Any obs with V7 < -0.09 is an outlier; replace with NA, not removing rest of data

mutate(V7 = replace(V7, which (V7 < -0.09), NA)) %>% #Any value less than -0.09 is an outlier, replaced with "NA"

#Had to separate less than "<" from the negative sign because code thought was "<-" denoted as

#d.) Get rid of all observations where V1 is "D"

filter(V1 == 'A' | V1 == 'B' | V1 == 'C') %>% #Filtered out all observations that weren't "D"

group\_by (V1) %>% #Grouped V1 excluding "D"

#e.) Sort the data by V1 (A-C) and within each category, in decreasing V2

arrange(V1, desc(V2)) %>% #Arranged V1 in descending order specific to each V1 category

#f.) Calculate the mean and standard deviation of V2 for each category of V1

mutate(V2.M = mean(V2)) %>% #Created V2.M to calculate the mean of V2

mutate(V2.stdev = sd(V2)) #Created V2.stdev to calculate the mean of V2

#view new variables in table within environment of df2 to see the mean (V2.M) and standard deviations (V2.stdev) of V1

#3---------------------------------------------------------

#Ignore all modifications in #2, fresh import of dr.csv, modify to fit photo on assignment

df.1 <- df %>% #Rename df to df.1

as\_tibble(df.1) %>% #create tibble of df.1

select(-...1) %>% #remove "...1" from the tibble

pivot\_longer(-V1, "Variable") %>% #pivot data to match screenshot

rename(Score = value) %>% #Rename the value to be set as "Score"

mutate(SignScore = if\_else(Score<0, "-" , "+")) #Create "SignScore" to show the positive and negative sign for each variable's score

df.1 #run df.1 to see tibble and view scores and correlated sign scores

#4---------------------------------------------------------

#Create new dataset in R called "key", join it to df.1 (from #3)

#Every obs in df.1 with V1=A and Variable=V2 should be assignmed as NewValue =1

#Every obs in df.1 with V1=A and Variable =V3 should be assigned a New Value =2

#Do not used chained if\_else statements

V1 <- rep(c("A", "B", "C", "D"), each = 10) #to match screenshot, each variable is assigned values in increments of 10 from V2-V11

NewValue <- 1:40 #New Value reaches 40 because A,B,C,D exist (increments of 10 with 4 values)

key <- data.frame(V1, NewValue)

key$Variable = c("V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "V10", "V11")

key.join <- inner\_join(df.1, key) #Joining data

key.join #Observe table

#5---------------------------------------------------------

df.n <- tibble(Variables = names(COPUS),

Type = rep(c("Demographic", "Behaviour", "Cluster"), c(8, 25, 2))) %>% #previous code given in assignment

mutate(NewVariable = str\_c(Type, Variables, sep = ".")) #combines the "Variables" and "Types" and separates the two with a "." within a string

#6---------------------------------------------------------

library(readxl)

copus <- read\_excel("~/Downloads/COPUS.xlsx") # Importing copus excel file

copus <- copus %>% # Bcluster levels

mutate(Bcluster = factor(Bcluster, levels = c("Mostly lecture", "Transitioning", "High engagement"))) #Mutating the data to be in the correct order

ggplot(copus, aes(x = Bcluster, y = Lec)) + geom\_boxplot() #Boxplot containing Bcluster (x) and Lec (y) with specific x values