Assignment 1

If you are defining an object called vec.x, you can do so by typing it into a script file   
first and then executing or by typing it directly into the console. Which way is better for   
reproducibility and why?

Typing “vec.x” to define the object into the script file is the better option because it will be saved and comments may be added using “#” to describe descriptions for why and what each row is defining, whereas in the console it will not be.

Explain how R came up with the following result:  
x <- 1:10  
y <- 1:3  
x-y  
 [1] 0 0 0 3 3 3 6 6 6 9   
Warning message:  
In x - y : longer object length is not a multiple of shorter object   
length

X is defined as 1 to 10, y is defined as 1 to 3. The command “x-y” is attempting to subtract the longer x in series with y, but is doing so in the pattern of 1-1, 2-2, 3-3, 4-1, 5-2, 6-3, 7-1, 8-2, 9-3, 10-1; Providing the error message incase that is not what the coder was attempting to pursue.

I.6 Explain the behavior of the round() function observed below where 0.5 is rounded   
down, but 1.5 is rounded up.  
round(.5)  
 [1] 0  
round(1.5)  
 [1] 2

The round() function rounds to the nearest even number which .5 is closer to 0 than to 2, and 1.5 is closer to 2.

4. I.7 A researcher wants to create a data set by sampling 100 integers ranging from 50 to   
75 with replacement, center those data (subtract the mean from each data point), and then   
calculate the centered mean divided by the centered standard deviation. Spot the silent   
error in the following code written to do this:  
1 data <- sample(50:75, 100, replace = TRUE) # sample the data  
2 data.cen <- data - mean(data) # center data on mean  
3 mean(data) / sd(data) # calculate mean / sd

The second row is unnecessarily defining a new vector as “data.cen” and not centering data on the mean as the comment states.

center\_mean(data) / center\_sd(data) #centered data / centered standard deviation

5. I.7 Why does the following generate an error? Special note: if you copy/paste from this   
word document, it brings what are called “smart quotes” which R can’t recognize. You   
might need to type the quotes manually.

x <- c(1, 5, 3, 4, “3”) # runs fine  
sum(x) # error generated

The “3” element in quotes does not allow the sum(x) function to run properly.

6. 1.8 Below are two sections of code that accomplish the same thing. Which one would you think is   
the “best” way? There isn’t necessarily a right answer here, I just want to hear your rationale.

x <- sample(-50:50)

# Code Section 1  
x.fil <- x[x > 0] # filter out all negative values  
x.fil.sq <- x.fil^2 # square results  
mean(x.fil.sq) # compute mean

# Code Section 1  
mean((x[x > 0])^2) # filter, square, compute mean

The first section of code is re-defining x as x.fil, which may be an unnecessary step that clutters the script pane and is breaking down each individual step of the desired tasks. This is easy to see each step individually, but at the cost of making the data messier than the alternative more succinct code assembly.

The second section is my choice for the best way because it can accomplish the task of filtering out negative values, square the results, and take the mean all in one line of code (reproducible, and concise code).

**Coding Section**   
To complete this section, start a new script file with the following layout:  
(ALSO SENT R FILE)

# #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

1. I.2 Suppose I wrote 2 lines of code and then said the following: “Here, I defined an   
object x as a numeric vector that contains 5 numeric elements, the numbers 1 through 5.   
Then I told R to add 1 to each element and print the result.” What are the two lines of   
code?

#SuzanneTenison

#Assignment 1 Introduction to R

x < - c(1, 2, 3, 4, 5)

[1] 1 2 3 4 5

x <- (x +1)

[1] 2 3 4 5 6

2. I.3 Cohen’s d is a metric that computes the effect size in a comparisons test (if you don’t   
know what I’m talking about, it’s okay). The formula is:

,where...  
• d is Cohen’s d  
• ME, MC are the means of experimental (E) and control (C) groups  
• Sample SD pooled is the pooled standard deviation:  
Calculate Cohen’s d in R comparing the two simulated groups below. I would   
recommend using mean() and sd() to compute means and standard deviations (as opposed   
to doing them “by hand”). Note: there are functions in other packages that compute   
Cohen’s d for you, but do not use one of these functions here.  
set.seed(42)  
exp <- rnorm(100, 1.1, .1) # experimental  
set.seed(42)  
con <- rnorm(100, 1, .1) # control

------------------------------

set.seed(42)

exp <- rnorm(100, 1.1, .1) #experimental

set.seed(42)

con <- rnorm(100, 1, .1) #control

m.exp <- mean(exp)

m.con <- mean(con)

sd.exp <- sd(exp)

sd.con <- sd(con)

sd.pooled <- sqrt (((sd.exp^2) + (sd.con^2))/2)

sd.pooled

d <- (m.exp - m.con)/ sd.pooled

d

[1] 0.1041357

> d <- (m.exp - m.con)/ sd.pooled

> d

[1] 0.9602855

3. I.5 Create an object called data and define it as a numeric sequence that starts at 0 and   
goes to 200 in increments of 0.5 (i.e. 0, 0.5, 1, 1.5... ...199, 199.5, 200). Then, take a   
random sample of 50 points without replacement (cannot sample the same set of points   
more than once) and assign it to an object called dat.sample. Set the seed to 42 prior to   
sampling so we get the same result. Calculate a 5-number summary of dat.sample   
(minimum, 1st quartile, median, 3rd quartile, and maximum).

set.seed(42)

data <- seq(0,200,0.5) #0 to 200, sequened by .5

dat.sample <- sample(data, 50, replace = FALSE) #created dat.sample, w/ 50 random samples

summary(dat.sample) #calculates a 5-number summary

4. I.8 Binning numbers is a pretty common task in research that entails taking a numeric   
vector and binning them into categories. The following will simulate this process   
commonly found in educational testing.   
a. Simulate a dataset that contains 200 students’ scores that follow a normal   
distribution (?rnorm) that have a class average of 80% and a standard deviation of   
20%. Use a seed of 42 so we get the same data. You will notice that many of the   
sampled scored go above 100, which is not possible. Replace any number over   
100 with 100.   
b. What is the average and standard deviation of just the students in the top third of   
the class?   
c. Create a vector that bins scores according to a common grade distribution (below)   
and tally how many of each grade were given  
•A >= 90%   
•B >= 80%, < 90%  
•C >= 70%, < 80%  
•D >= 60%, < 70%  
•F < 60%

#4A-------------------------------

data<- rnorm(200, mean= 80, sd = 20) #200 students, normally distributed, 80% average w/ sd of 20%

set.seed(42)

data[data > 100] <- 100 #get rid of all the numbers larger than 100, change to 100

data

#B---------------------------------

quantile(mean (data))

quantile(sd (data))

#C----------------------------------

library(dplyr)

df %>% data(points\_bin = ntile(points, n=100))

# Installed release version from CRAN

install.packages("rbin")

bins <- rbin\_manual(A=90, B=80:<90, C=70:<80, D=60:<70, F=60) #attempted to create vector that bins scores

table(df$bins) #didn't work, I tried!