Homework 1

Brady Lamson

2/02/2022

Problems 1-5:

Problem 1:

a) Try the commands pi, round(pi), round(pi, digits = 4), and trunc(pi), ceiling(pi), floor(pi). What are the results?

```
## The output of pi is 3.14159265358979

## The output of round(pi) is 3

## The output of round(pi, digits = 4) is 3.1416

## The output of trunc(pi) is 3

## The output of ceiling(pi) is 4

## The output of floor(pi) is 3
```

b) Try the commands sqrt(16), 16^0.5. Are the results the same?

```
## The output of sqrt(16) is 4

## The output of 16^0.5 is 4

## Are the two commands the same? TRUE
```

c) Write a command that computes 4^3

```
4^3
```

```
## [1] 64
```

d) Try the commands log10(1000), log(1000). Try the command log2(64). What are the results?

```
## The output of log10(1000) is 3
```

```
## The output of log(1000) is 6.90775527898214

## The output of log2(64) is 6
```

e) Does the text of the help file for log() match your observations?

- Yes it does! The number next to the log is the base, so log 10 uses a base of 10. The one thing to keep in mind is that log() uses a base of e by default (exp(1) in R).

Problem 2.

Manipulate the following character vector using square brackets [ ] to accomplish the following goals.

```
1) Barry arrives (and gets in the last position of the line)
2) Steve is served (and so he leaves)
3) Pam arrives and talks her way to the front of the line (with just one item)
4) Barry gets impatient and leaves
```

```
queue <- c("Steve", "Russell", "Alison", "Liam")
queue[length(queue) + 1] <- "Barry"
queue <- queue[-1]
queue <- c("Pam", queue)
queue <- queue[-length(queue)]
queue
```

```
## [1] "Pam"     "Russell" "Alison"  "Liam"
```

Problem 3.

a) Write a command that lists the objects in your Workspace.

b) Write a command that removes x from the Workspace.

c) Write a command that removes *all* the objects from your Workspace.

```
w <- 6
x <- 7
y <- 8
z <- 9

ls()
```

```
## [1] "queue" "w"     "x"     "y"     "z"
```

```
rm(x)
rm(list = ls())
```

Problem 4

Consider the below vector.

a) What is the output of x == 0

```
x <- c(3, 2, 0, 1, 4, 5, 9, 0, 6, 7, 2, 8)
x == 0
```

```
##  [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

b) Write a command involving sum() and the "logical" vector x == 0 that counts the number of elements of x that are equal to 0.

```
logical_vector <- x == 0
```

```
## The output of sum(logical_vector) is 2
```

c) Write a command that determines the *proportion* of elements of x that are equal to 0, assuming you *don't know* the number of elements in x.

```
proportion <- (sum(logical_vector) / length(logical_vector)) |>
    round(digits = 3)
```

```
## The proportion of elements of x that are equal to 0 is 0.167
```

Problem 5:

Using the following data frame:

```
numVec <- c(2, 4, 6, 5, 9, 8, 2, 4, 7, 8)
charVec <- c("a", "b", "c", "c", "b", "c", "a", "b", "b", "c")
myData <- data.frame(x1 = numVec, x2 = charVec, stringsAsFactors = FALSE)
```

a) The following commands do the same thing:

- myData$x1
- myData[[“x1”]]
- myData[[1]]

What do they do?

```
- These return the first column of the data set, which in this case is all of numVec.
```

b) What kind of object is returned by the commands in part a?

```
is.vector(myData$x1)
```

```
## [1] TRUE
```

If they return a *vector*, what type of vector is it?

```
glue::glue("
           Is myData$x1 a numeric vector or character vector?
           Numeric? {is.numeric(myData$x1)}
           Character? {is.character(myData$x1)}
           ")
```

```
## Is myData$x1 a numeric vector or character vector?
## Numeric? TRUE
## Character? FALSE
```

c) What do the following commands do?

```
myData[2, ]
```

```
##   x1 x2
## 2  4  b
```

```
myData[, 2]
```

```
##  [1] "a" "b" "c" "c" "b" "c" "a" "b" "b" "c"
```

myData[2,] returns the second row of the data frame. So (4, b)

myData[, 2] returns the second column of the data frame. This will be the full vector of characters.

d) What class of object is myData?

```
glue::glue("myData is of class {class(myData)}.")
```

```
## myData is of class data.frame.
```

e) What happens when you pass myData into the summary() command?

```
summary(myData)
```

```
##        x1            x2
##  Min.   :2.00   Length:10
##  1st Qu.:4.00   Class :character
##  Median :5.50   Mode  :character
##  Mean   :5.50
##  3rd Qu.:7.75
##  Max.   :9.00
```

This command provides the summary statistics, length, class and mode.

Textbook Exercises

Problem 1:

The following code chunk throws an error, why?

```
#mtcars %>%
#    select(mpg, cyl)
```

The library select() and the pipe %>% are being pulled from isn't specified or loaded. The following code works just fine.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
mtcars %>%
    select(mpg, cyl) %>%
    head()
```

```
##                   mpg cyl
## Mazda RX4         21.0   6
## Mazda RX4 Wag     21.0   6
## Datsun 710        22.8   4
## Hornet 4 Drive    21.4   6
## Hornet Sportabout 18.7   8
## Valiant           18.1   6
```

Problem 2:

Which of these kinds of names should be wrapped with quotation marks when used in R?

- function name (No)

- file name (Yes)

- the name of an argument in a named argument (No)

- object name (No)

Problem 3:

```
obj1 <- 2:10
obj2 <- c(2, 5)
obj3 <- c(TRUE, FALSE)
obj4 <- 42
```

```
obj1 * 10
```

```
## [1]  20  30  40  50  60  70  80  90 100
```

```
obj1[2:4]
```

```
## [1] 3 4 5
```

```
obj1[-3]
```

```
## [1]  2  3  5  6  7  8  9 10
```

```
obj1 + obj2
```

```
## Warning in obj1 + obj2: longer object length is not a multiple of shorter object
## length
```

```
## [1]  4  8  6 10  8 12 10 14 12
```

```
obj1 * obj3
```

```
## Warning in obj1 * obj3: longer object length is not a multiple of shorter object
## length
```

```
## [1]  2  0  4  0  6  0  8  0 10
```

```
obj1 + obj4
```

```
## [1] 44 45 46 47 48 49 50 51 52
```

```
obj2 + obj3
```

```
## [1] 3 5
```

```
sum(obj2)
```

```
## [1] 7
```

```
sum(obj3)
```

```
## [1] 1
```

Problem 4:

```
mylist <- list(x1 = "sally", x2 = 42, x3 = FALSE, x4 = 1:5)

is.list(mylist)
```

```
## [1] TRUE
```

```
names(mylist)
```

```
## [1] "x1" "x2" "x3" "x4"
```

```
length(mylist)
```

```
## [1] 4
```

```
mylist[[2]]
```

```
## [1] 42
```

```
mylist[["x1"]]
```

```
## [1] "sally"
```

```
mylist$x2
```

```
## [1] 42
```

```
length(mylist[["x4"]])
```

```
## [1] 5
```

```
class(mylist)
```

```
## [1] "list"
```

```
typeof(mylist)
```

```
## [1] "list"
```

```
class(mylist[[4]])
```

```
## [1] "integer"
```

```
typeof(mylist[[3]])
```

```
## [1] "logical"
```

Problem 5:

What's wrong with the below code

```
# help(NHANES, package <- "NHANES")
```

You typically aren't supposed to use the assignment operator within a function call like this. pacakage = "NHANES" would be better here.

Problem 6:

CPS, in this context, stands for **Current Population Survey**. This CPS is "used to supplement census information between census years".

Problem 7:

Why does this code throw an error?

```
#mtcars %>%
#    filter(cylinder == 4)
```

cyl is the column name to filter by, not cylinders.

```
mtcars %>%
    filter(cyl == 4) %>%
    head()
```

```
##                mpg cyl  disp hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0 93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7 62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8 95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7 66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7 52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.90  1  1    4    1
```

Problem 8:

The date() function takes no arguments and returns a string of the current system date and time. The result of Sys.time() is of class **POSIXct** and **POSIXt**.

Problem 9:

What do the following commands return? Describe the class of the object as well as its value.

```
a <- c(10, 15)
b <- c(TRUE, FALSE)
c <- c("happy", "sad")
```

```
## data.frame(a, b, c) is of class data.frame and returns:
```

```
##    a     b     c
## 1 10  TRUE happy
## 2 15 FALSE   sad
```

```
## cbind(a, b) is of class matrix
## cbind(a, b) is of class array
```

```
##       a b
## [1,] 10 1
## [2,] 15 0
```

```
## rbind(a, b) is of class matrix
## rbind(a, b) is of class array
```

```
##   [,1] [,2]
## a   10   15
## b    1    0
```

```
## cbind(a, b, c) is of class matrix
## cbind(a, b, c) is of class array
```

```
##      a    b       c
## [1,] "10" "TRUE"  "happy"
## [2,] "15" "FALSE" "sad"
```

```
## list(a, b, c)[[2]] is of class logical
```

```
## [1]  TRUE FALSE
```

Problem 10

For each of the following assignment statements, describe the error (or note why it does not generate an error).

```
# result1 <- sqrt 10
# This has no parentheses around 10

# result2 <-- "Hello to you!"
# This has one two many '-'s

# 3result <- "Hello to you"
# Variable names cannot start with a number.

# result4 <- "Hello to you
# Theres no second quotation mark

# result5 <- date()
# This is broken because of the lack of quotation mark from earlier.
```

Problem 11.

The following code chunk throws an error.

```
# mtcars %>%
#   filter(cyl = 4)
```

This is missing the extra equal sign. This would be assigning 4 to cyl which makes no sense. '==' is the comparison operator.

```
mtcars %>%
    filter(cyl == 4) %>%
    head()
```

```
##                 mpg cyl  disp hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0 93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7 62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8 95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7 66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7 52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.90  1  1    4    1
```

Problem 12:

Describe in words what computations are being done and then, using pipe notation, rewrite the code.

```
### Commenting this all out as the code takes forever to run.
#library(mosaic)
# ds <-
#   read.csv("http://nhorton.people.amherst.edu/r2/datasets/helpmiss.csv")
#  summarise(group_by(
#    select(filter(mutate(ds,
#      sex = ifelse(female == 1, "F", "M")
#    ), !is.na(pcs)), age, pcs, sex),
#    sex
#  ), meanage = mean(age), meanpcs = mean(pcs), n = n())
```

This is reading in a csv file and then the following queries are being called on it in order:

- mutate: Renamed the column "female" to "sex". Also changes the values of the cells in that column to "F" if the value in the cell is 1. If it isn't, the cell is assigned to "M".
- filter: This removes the rows with 'pcs' that are NA from our query.
- select: This selects **only** the columns age, pcs and sex.
- summarise: Shows us three things:
  - The mean age of the male and female samples.
  - The mean pcs of the male and female samples.
  - The count of male and female samples.

This block of code is an abomination and I rewrote it below.

```
ds <- read.csv("http://nhorton.people.amherst.edu/r2/datasets/helpmiss.csv")

ds %>%
    dplyr::mutate(
        sex = ifelse(
            female == 1, "F", "M"
            )
        ) %>%
    dplyr::filter(!is.na(pcs)) %>%
    dplyr::select(age, pcs, sex) %>%
    dplyr::group_by(sex) %>%
    dplyr::summarise(
        mean_age = mean(age),
        mean_pcs = mean(pcs),
        n = n()
    )
```

```
## # A tibble: 2 x 4
##   sex   mean_age mean_pcs     n
##   <chr>    <dbl>    <dbl> <int>
## 1 F         36.1     44.9   111
## 2 M         35.6     49.1   357
```

Problem 13:

The following concepts should have some meaning to you: package, function, command, argument, assignment, object, object name, data frame, named argument, quoted character string.

Construct an example of R commands that make use of at least four of these. Label which part of your example R command corresponds to each.

For this I'll use a helper function I wrote to help me with my Calc III homework! This function handles vector projection for me and returns a vector.

```
# vector_projection is the name of a function.
# Everything inside of the parentheses of function() is an argument.
# All of the arguments are also named here. vector_1 as an example.
# assignment is happening each time an arrow is used.
# vector_1_mag is an object, a numeric vector of length 1.
# It also has a name, vector_1_mag

vector_projection <- function(vector_1, vector_2, two_onto_one = TRUE) {
    ### Variables ----
    dot_product <- vector_1 %*% vector_2
    vector_1_mag <- vector_1^2 |>
        sum() |>
        sqrt()
    vector_2_mag <- vector_2^2 |>
        sum() |>
        sqrt()

    if (two_onto_one) {
        projection_scalar <- (dot_product / vector_1_mag^2) |>
            as.vector()
        return(projection_scalar * vector_1)
    }
    else {
        projection_scalar <- (dot_product / vector_2_mag)^2 |>
            as.vector()
        return(projection_scalar * vector_2)
    }
}
```