# BIOS 545 Spring 2020 Homework 1

Due 11:59 PM on February 17, 2020

## Instructions

There are 20 questions at 5 points each. Submit your answer sheet in a file named using the convention of `BIOS545_LastName_FirstName_HW1.Rmd` to Canvas. You should use RStudio to create the file. Late submissions are accepted with a penalty of 15% for each day late. Submissions made later than three days from the due date will not be accepted and a grade of F will be given for that homework assignment.

All of these problems can be solved using material presented in class. Unless otherwise indicated in the problem, you may use other R functions to help you find a solution although you cannot download additional packages to solve a problem. In some cases you might have to use the help mechanisms described in Week 1 to locate an appropriate function. We will run your Rmd file at the R console to verify the statements.

For questions that require you to provide a "one line statement" you should approach these problems by first sub dividing the problem into smaller parts. First, solve those individual components and then combine those smaller solutions into a single expression. If you try to do it all at once it can be confusing.

# Problems

**1)** Perform the following operations in R.

**a)** Create `x1` containing integers from 1 to 10, `x2` containing integers from 15 to 6, and `x3` as the vector sum of `x1` and `x2`.

**b)** Create `x4` containing all numbers from 0 to 3 in 0.25-unit increments, and calculate `x5` as each element of `x4` times 3 plus 5.

**c)** Use the `sum` function and elementwise vector multiplication to calculate the dot product $x1 \cdot x2 = \sum_{i=1}^{n}(x1_i)(x2_i)$. The answer is 495.

**d)** Use matrix multiplication to calculate the same dot product. (Note: R can multiply these two vectors even though neither is defined as a matrix. It assumes you want to consider `x1` a row vector and `x2` a column vector.)

**e)** Create `x6` as the elementwise product of `x1` and `x2`. Determine the maximum value of `x6` and find which element is the maximum (i.e. is it the 3rd number, the 4th, etc.). You may want to run `plot(x6)` to confirm your results.

**2)** Initialize the vector x using the following commands. Present a one line R statement that will count the total numbers that are evenly divisible by 15 and 25. The answer will be a single number.

```
set.seed(222)
x <- floor(rnorm(300, 200, 90))
```

**3)** Given the following vector provide a one line command that computes the sum of all element values that are within 5 units of the maximum number of this vector. The correct answer is 599.0835. You must present a one line R statement that issues this response. Hint, using the absolute function will be useful.

```
set.seed(123)
x <- rnorm(100, 100, 10)
```

**4-7)** In preparation for questions 4-7 execute the following commands from within an R session

```
u <- "https://www.dropbox.com/s/8q6z9t9r7i10q9k/worldfloras.txt?dl=1"
countries <- as.character(read.table(u,sep="\t")$V1)
```

For questions 4 through 7 please consult the material from Week 2 starting on slide 67 of the lecture deck relating to regular expressions. Use the `grep` function and/or vector bracket notation (or a combination thereof) to arrive at the correct answer. You may also refer to this online http://donovanh.com/pages/regex_list.html to help you answer the following questions:

**4)** Present a single R statement(s) that prints all country names that begin with an uppercase C, D, E, or F.

**5)** Present an R statement that prints all country names that do **not** contain a space in the name (e.g. you want to exclude country names like "Costa Rica", "Hong Kong", etc)

**6)** Present an R statement that finds all country names that begin with an "A" or ends in the letter "e"

**7)** Present a single R statement to find those countries with multiple names with uppercase "R" as the first letter of their second or subsequent names. For example, "Costa Rica" and "Dominican Republic" are two such cases.

**8-13)** The **quantmod** package allows you to download stock market prices from Yahoo! Finance. Run the code below to download **quantmod** from CRAN and load 3 years of stock prices for Netflix, then answer the following questions.

```
install.packages("quantmod")
library(quantmod)
netflix.prices <- as.matrix(getSymbols("NFLX", from = "2017-01-22",
                                       to = "2020-01-22", auto.assign = FALSE))
```

**8)** If you run `head(netflix.prices)` you will see the structure of the matrix. The 2nd column has daily high prices and the 3rd column has daily low prices. Figure out the highest high and lowest low, and then calculate the range of stock prices for this time period.

**9**) That range is huge considering the final-day closing price of 338.11. That suggests either Netflix is an extremely volatile stock, or it had a huge net gain or loss during the 3-year period. Using the `NFLX.Close` column, calculate the net gain as a price and as a percentage.

**10)** Notice that the `netflix.prices` matrix has dates for row names. You can see this when you run `head(netflix.prices)`. Using a certain function covered in lecture, extract these row names into a vector called `netflix.dates`.

**11)** If you run `class(netflix.dates)`, you will see that despite looking like dates, R is treating them as character strings. Convert the `netflix.dates` vector to date class, and call it `netflix.realdates`.

**12)** Determine the highest closing price for Netflix, on what date that price was achieved, and how many calendar days have passed from then until today (use Jan. 21, 2020 for "today"). (Well, isn't that result interesting!)

**13)** The daily percent gain for a stock is the closing price divided by the prior day's closing price, minus 1, times 100. Create a vector called `daily.gains` using the `NFLX.Close` column. Verify it has length 753 and then calculate the range of daily gains (i.e. worst 1-day loss and best 1-day gain).

**14)** Part of being a "good user of R" is being able to figure out how to use a function by reading its help file. The function `pnorm` lets you calculate probabilities for a normally distributed random variable. You can pull up the help file by running:

```
?pnorm
```

Learn how to use this function and then use it to calculate the following:

1. $P(X \leq 0)$ if $X \sim N(0, 1)$
2. $P(X > 16)$ if $X \sim N(10, 3)$
3. $P(-4 \leq X \leq -2)$ if $X \sim N(-2.5, 0.5)$

**15-16)** Body mass index (BMI) and waist circumference are two metrics commonly used to classify adiposity (fatness). BMI is calculated simply as body weight in kilograms divided by squared height in meters. BMI $>$ 30 and waist circumference $>$ 40 inches are two metrics commonly used classifications for excess adiposity.

Run the following code to simulation some data for 100 people.

```
set.seed(123)
sex <- sample(x = c("M", "F"), size = 100, replace = TRUE)
height.cm <- rnorm(n = 100, mean = ifelse(sex == "M", 175, 163), sd = 3)
weight.kg <- -110 + height.cm * 1.1 + rnorm(n = 100, sd = 7)
waist.in <- -20 + ifelse(sex == "M", 35, 33) + 0.5 * weight.kg + rnorm(n = 100, sd = 2.5)
```

**15)** Create a vector called `bmi` that contains BMI values for these people. Make sure your units are correct. Calculate the mean BMI overall and for males and females separately.

**16)** Create a logical vector called `bmi.obese` that is TRUE if BMI $>$ 30 and FALSE otherwise, and a logical vector called `waist.obese` that is TRUE if waist $>$ 40 and FALSE otherwise. Create a simple 2-by-2 table summarizing agreement between the two classifiers.

**17)** Given an N x N matrix find the two "off-diagonals" and return them as a matrix. Consider the following 4 x 4 matrix. The diagonal can easily be found:

```
(mymat <- matrix(1:16, 4, 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
diag(mymat)
```

```
## [1]  1  6 11 16
```

The off-diagonals for this or any N x N matrix would be the two (there are always two) "sub" diagonals located directly below and above the main diagonal. In this example those would be the numbers 2, 7, 12 and 5, 10, 15. Write a one-line statement that will return these numbers in the form of a matrix such that each off-diagonal is a column in the result matrix.

```
##      [,1] [,2]
## [1,]    2    5
## [2,]    7   10
## [3,]   12   15
```

The way to approach this is to first find a logical expression that can be used to locate the two off-diagonals. Next wrap that result into a call to the matrix function while using appropriate arguments to influence how the resulting matrix is created. Also remember to use functions you have learned about in class to help you generally figure out the dimensions (hint hint) of the resulting matrix. Here is another example matrix you can use to verify your work:

```
(mymat <- matrix(1:9, 3, 3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

The correct result would be:

```
##      [,1] [,2]
## [1,]    2    4
## [2,]    6    8
```

**18)** Suppose we are analyzing data from an epidemiological study, and one of the variables we are looking at is number of children in the household. Run the following code to simulate some data on number of children.

```
set.seed(1)
num <- rpois(n = 1000, lambda = 1.5)
```

Create a factor variable called `num.factor` with 3 groups: No children, 1-2 children, and 3 or more children. There is a natural ordering here, so your factor variable should be ordered. There are a few ways of doing it, but you will get credit as long as `num.factor` is an ordered factor, the classifications are correct, and there are meaningful labels (not just 0, 1, 2). (Note: factor variables are very convenient when analyzing data in R!)

**19)** Given any N x N matrix present a one line statement to find the Nth largest value of that matrix. As an example, for the first matrix given below, which is 4x4, the 4th largest value is 32.24 so your statement should present that number. In the second example, which is a 5x5 matrix, the 5th largest element is 27.01.

Explore using function(s) that help you to determine the size of a given matrix. This will help you achieve a general solution.

```
set.seed(123)
(mymat <- matrix(round(rnorm(16, 20, 10), 2), 4, 4))
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 14.40 21.29 13.13 24.01
## [2,] 17.70 37.15 15.54 21.11
## [3,] 35.59 24.61 32.24 14.44
## [4,] 20.71  7.35 23.60 37.87
```

```
set.seed(123)
(mymat <- matrix(round(rnorm(25, 20, 10), 2), 5, 5))
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 14.40 37.15 32.24 37.87  9.32
## [2,] 17.70 24.61 23.60 24.98 17.82
## [3,] 35.59  7.35 24.01  0.33  9.74
## [4,] 20.71 13.13 21.11 27.01 12.71
## [5,] 21.29 15.54 14.44 15.27 13.75
```

**20)** Given two vectors `veca` and `vecb` present a one line statement to find the maximum value of all values from `veca` that are also in `vecb`. Another way to think about it is to find the single largest value in `veca` that also occurs in `vecb`. This is easy using one or more of the vector functions mentioned in class in combination with the bracket notation. Your solution should work for any two vectors in general. Note that you don't have to worry about situations wherein `veca` does not contain any elements that are in `vecb`. As an example, consider the following two vectors.

```
set.seed(123)
(veca <- sample(1:50, 8))
```

```
## [1] 31 15 14  3 42 43 37 48
```

```
set.seed(123)
(vecb <- sample(1:15, 14))
```

```
##  [1] 15  3 14 10  2  6  5  4 12  7  1 11 13  8
```

The answer in this case is 15. Even though there are values in `veca` that are larger than 15 it turns out that 15 is the largest value in `veca` that also exists in `vecb`.