

# BISO545 Spring 2015 Homework 1

Due by 11:59 PM on January 30, 2015

## Instructions

There are 20 questions at 5 points each. Send responses via email in a file named using the convention of BIOS545\_LastName\_Firstname\_HW1.R. You should use RStudio to create the file. Email to **BOTH** [dvandom@emory.edu](mailto:dvandom@emory.edu) and [wsp@emory.edu](mailto:wsp@emory.edu). Submissions arriving after the indicated due date and time will incur a 10 percent penalty for each day late.

All of these problems can be solved using material presented in class and in the labs. 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 R commands at the R console to verify the statements.

**1-3)** Using the R “expression” command present expressions that represent the following formulae. Given the values  $x = 2$ ,  $a = 2$ ,  $b = 3$ ,  $n = 3$  then evaluate the formulae and present the result for each.

$z = x^b$	$z = (x^a)^b$	$z = \sqrt[n]{ab}$
$x = 2, a = 2, b = 3$	$x = 2, a = 2, b = 3,$	$a = 2, b = 3, n = 3$
$z = ?$	$z = ?$	$z = ?$

**4)** The *Jaccard Index* of two vectors can be used to determine how similar they are. There are other methods to determine similarity but this is one of the more fundamental ones.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

```
A <- c(0,1,2,5,6)
B <- c(0,2,3,5,7,9)
```

```
J(A,B) = 0.375
```

The Jaccard Index of A and B is 0.375. We compute this as follows. There are three elements in common to A and B which we then divide by the number of elements contained in the union of the two vectors. The union of the two vectors is every element in both sets but counting common elements only once. Thus,  $J(A,B) = 3/8 = 0.375$

Write R statement(s) that, given two numeric vectors A and B of arbitrary length, will print out the Jaccard index.

5) Given a vector with 1,000 elements, present an R statement(s) that produces the sum of all element *numbers*, (not values), that are evenly divisible by 3.5 and 2.

6) Create a vector T that contains all numbers greater than 0 and less than 100,000 that are evenly divisible by 19.

7) How many elements in T are greater than or equal to the 75th percentile of T ?

8) How many numbers in T end in a "1" ?

9) From within an R session run the following command. It will create a vector called "fruits" that contains 250 elements each of which is the name of a fruit. There are multiple repetitions of each fruit name.

```
source("http://stevie42.bitbucket.org/bios545r/SUPP.DIR/fruits.R")
```

What are the two least repeated fruit names ? (Provide R statements that give the answer.)

10) What is the mean value of the three top most repeated fruits ? (Provide R statements that give the answer.)

11) Issue the following command at an R prompt. It will create six vectors: *hours*, *minutes*, *seconds*, *years*, *months*, *days*, which represent what their respective names suggest. That is, the hours vector represents an hour between 0 and 24. The minutes and seconds vectors have values between 0 and 60, and so on.

```
source("http://stevie42.bitbucket.org/bios545r/SUPP.DIR/times.R")
```

Your job is to combine these individual vectors into a single vector of actual dates and times. The first six elements of the resulting vector should look like:

```
datesandtimes[1:6]
[1] "2008-07-14 11:04:16 EST" "2004-01-16 05:20:43 EST" "2012-08-03 16:01:37 EST"
[4] "2004-09-18 15:20:24 EST" "2009-03-23 11:47:10 EST" "2013-03-25 04:45:01 EST"
```

```
str(datesandtimes)
POSIXlt[1:50], format: "2008-07-14 11:04:16" "2004-01-16 05:20:43" "2012-08-03 16:01:37" ...
```

**HINT** Before attempting to convert these into a real date string, it will be easier to first combine dates and times into a character string.

In preparation for questions 12-14 execute the following commands from within an R session

```
url <- "http://nestor.sunderland.ac.uk/~cs0her/Statistics/therbook/worldfloras.txt"
countries <- as.character(read.table(url,sep="\t")$V1)
```

For questions 12 through 14 please consult the chart on slide number 67 for Week 1 Lecture 2 as well as this online chart [http://donovanh.com/pages/regex\\_list.html](http://donovanh.com/pages/regex_list.html) to help you answer the following questions:

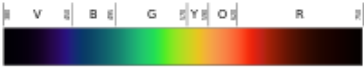
**12)** Present an R statement(s) that finds all country names that begin with an uppercase C,D,E, or F. You can use one statement to do this.

**13)** Present an R statement that finds 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)

**14)** Present an R statement that finds all country names that begin with an “A” or end in an “e”

**15)** Read the following from the Wikipedia article on Spectral Colors [http://en.wikipedia.org/wiki/Visible\\_spectrum](http://en.wikipedia.org/wiki/Visible_spectrum)

Colors that can be produced by visible light of a narrow band of wavelengths (monochromatic light) are called pure spectral colors. The various color ranges indicated in the diagram on the right are an approximation: The spectrum is continuous, with no clear boundaries between one color and the next.

			
Color	Wavelength	Frequency	Photon energy
violet	380–450 nm	668–789 THz	2.75–3.26 eV
blue	450–495 nm	606–668 THz	2.50–2.75 eV
green	495–570 nm	526–606 THz	2.17–2.50 eV
yellow	570–590 nm	508–526 THz	2.10–2.17 eV
orange	590–620 nm	484–508 THz	2.00–2.10 eV
red	620–750 nm	400–484 THz	1.60–2.00 eV

# Run the following commands to setup a vector with simulated  
# wavelengths

```
set.seed(123)
myvec <- runif(100,350,780)
```

Provide R command(s) that creates a factor that chops up the values into intervals corresponding to the wavelengths indicated in the chart. Each interval should be named accordingly. That is, values falling into the range of 380-450 nm should be labelled as "violet". Values matching the lower end of the range, in this case, 380, should be included in the interval whereas values at the top of range, in this case, 450, should not. Thus the resulting levels will look like this.

Levels: [380,450) [450,495) [495,570) [570,590) [590,620) [620,750)

Hint, create the intervals first without labels to check that the values are falling into the right intervals. After that add in the necessary argument to apply labels to each interval.

**16)** Some of the generated wavelength values in 16 do not fall into any interval. You can see which ones because they show up as NA. Provide an R command to count how many are missing.

**17)** Given a number  $n$ , create an  $n \times n$  matrix  $A$  so that  $A_{i,j} = 1$  where  $i + j = n + 1$  and zero otherwise. As an example, if  $n$  were 5 the resulting matrix would look like the following.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

Your answer will present R statements that, given “n”, will generate a matrix of size n x n and make element assignments according to the specification given above.

**HINTS:** 1) Practice first on a specific matrix. 2) To initialize a matrix you can fill it with whatever values you want (e.g. zeroes, ones, etc). 3) Check out the “rep” function that helps generate “n” repeated values. 4) The row and col functions will greatly simplify your work here.

**18)** There is a dice game wherein you roll a single die four times. If you get a 4 at least once then you have won that particular round. This exercise will use your knowledge of sampling to simulate a round, (4 rolls), of this game. Recall that rolling a single die will produce values of 1,2,3,4,5, or 6. Also recall that each roll of a dice is independent.

Present R statements that will replicate 10 rounds of the game with the results of each round being stored into a matrix. You saw examples in class on how to do this (check your slides).

**19)** Present R statement(s) that count how many times a 4 occurred. Depending on the results of your simulation there might not be any 4s present although your command to check would be the same. You can always rerun your simulation until you see some 4s in the output.

**20)** DNA nucleotide sequences are represented by combinations of A,C,G, and T. Use the sample function to create a sixty four element vector containing these letters. However, we would like A’s to occur 15 percent of the time and G’s to occur 25 percent of the time. The C’s and T’s each occur 30 percent of the time. Store the resulting sample into a vector called my.dna.

```
my.dna <- <your R statement>
```