Homework 8

1. (1 points) In the slide 21 of class 8, we used function `Vectorize()` to vectorize function `sample()` so that it takes a vector for the `size` argument. The output of the function `sample2()` is a list. Try to generate such a list with a `for` loop (the exact numbers sampled will be different given the randomness in the sampling process). Hint: to create a list of length n in R, we can use `list_name <- vector("list", length = n)`; to fill the ith element of this list, we can use `list_name[[i]] <- value_you_generated`.

```
q1 = vector("list", length = 3)
for (i in 1:3){
  q1[[i]] = sample(1:100, i)
}
q1
```

```
## [[1]]
## [1] 78
##
## [[2]]
## [1] 25 34
##
## [[3]]
## [1]  2 90 92
```

2. (2 points) Suppose we have a dataset A (see code below) where each column represents multiple measures of nitrogen concentration in a particular lake. We want to get the average value for each lake. Do this in two ways: a `for` loop and a vectorized function `colMeans()`.

```
set.seed(12) # to be reproducible
A = matrix(data = 1:500, nrow = 50, ncol = 10)
colnames(A) = paste("lake", 1:10, sep = "_")
```

```
A = matrix(data = 1:500, nrow = 50, ncol = 10)
colnames(A) = paste("lake", 1:10, sep = "_")
q2 = vector(length = ncol(A))
for(i in 1:ncol(A)){
  q2[i] = mean(A[, i])
}
q2
```

```
##  [1]  25.5  75.5 125.5 175.5 225.5 275.5 325.5 375.5 425.5 475.5
```

```
colMeans(A)
```

```
##  lake_1  lake_2  lake_3  lake_4  lake_5  lake_6  lake_7  lake_8  lake_9
##    25.5    75.5   125.5   175.5   225.5   275.5   325.5   375.5   425.5
## lake_10
##   475.5
```

```
all(q2 == colMeans(A))
```

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

3. (2 points) Create two matrices `T1` and `T2` using the code below. These matrices contain binary values showing the interaction pattern between individuals in two different time periods. In these matrices, the cell in row i and column j is equal to 1 if individual i interacted with individual j and zero otherwise.

To find out if individual i interacted with individual j in both time periods, use a `for` loop to create another matrix `T1_T2` so that the value of each cell is the product of the corresponding cells of `T1` and `T2`. In this case, a 1 indicates that individual `i` interacted with individual `j` in both time periods whereas we obtain a 0 if these individuals have never interacted or interacted only once. What is another simpler/vectorized way to do this?

```
set.seed(123) # to be reproducible
T1 = matrix(rbinom(n = 100, size = 1, prob = 0.5), nrow = 10, ncol = 10)
T2 = matrix(rbinom(n = 100, size = 1, prob = 0.5), nrow = 10, ncol = 10)
```

```
set.seed(123) # to be reproducible
T1 = matrix(rbinom(n = 100, size = 1, prob = 0.5), nrow = 10, ncol = 10)
T2 = matrix(rbinom(n = 100, size = 1, prob = 0.5), nrow = 10, ncol = 10)
T1_T2 = matrix(NA, nrow(T1), ncol(T2))
for(i in 1:nrow(T1)){
  for(j in 1:ncol(T2)){
    T1_T2[i, j] = T1[i, j] * T2[i, j]
  }
}
T1_T2
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    1    1    1    0    0    1    1    0     0
##  [2,]    0    0    0    1    0    0    0    0    0     0
##  [3,]    0    0    0    1    0    0    0    1    0     0
##  [4,]    1    1    0    1    0    0    0    0    0     1
##  [5,]    0    0    0    0    0    0    0    0    0     0
##  [6,]    0    0    1    0    0    0    0    0    0     0
##  [7,]    1    0    0    1    0    0    1    0    0     0
##  [8,]    1    0    0    0    0    0    0    1    1     0
##  [9,]    0    0    0    0    0    0    0    0    1     0
## [10,]    0    0    0    0    1    0    0    0    0     1
```

```
T1_T2_2 = T1 * T2
T1_T2_2
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    1    1    1    0    0    1    1    0     0
##  [2,]    0    0    0    1    0    0    0    0    0     0
##  [3,]    0    0    0    1    0    0    0    1    0     0
##  [4,]    1    1    0    1    0    0    0    0    0     1
##  [5,]    0    0    0    0    0    0    0    0    0     0
##  [6,]    0    0    1    0    0    0    0    0    0     0
##  [7,]    1    0    0    1    0    0    1    0    0     0
##  [8,]    1    0    0    0    0    0    0    1    1     0
##  [9,]    0    0    0    0    0    0    0    0    1     0
## [10,]    0    0    0    0    1    0    0    0    0     1
```

```
all(T1_T2 == T1_T2_2)
```

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

4. (2 points) The Fibonacci Sequence is the series of numbers that the next number is the sum of the previous two numbers: 0, 1, 1, 2, 3, 5, 8 . . . Use a `for` loop to get the first 30 numbers of the Fibonacci Sequence. This question should demonstrate the need for loops because there is no easy way to use vectorized functions in this case.

```r
fb = numeric(30)
fb[1] = 0
fb[2] = 1
for(i in 3:length(fb)){
  fb[i] = fb[i-1] + fb[i-2]
}
fb
```

```
##  [1]      0      1      1      2      3      5      8     13     21     34
## [11]     55     89    144    233    377    610    987   1597   2584   4181
## [21]   6765  10946  17711  28657  46368  75025 121393 196418 317811 514229
```

5. (3 points) Use the vector generated in Q4 (suppose you named it as `fb`) to generate another vector (`fb_diff`) with 29 elements so that the ith element of `fb_diff` is equal to `fb[i + 1] - fb[i]`. Do this with a `for` loop and a vectorized function in the base package of R. Note: use Google to find out what this vectorized function is.

```r
fb2 = numeric(29)
for(i in 1:length(fb2)){
  fb2[i] = fb[i + 1] - fb[i]
}
fb2
```

```
##  [1]      1      0      1      1      2      3      5      8     13     21
## [11]     34     55     89    144    233    377    610    987   1597   2584
## [21]   4181   6765  10946  17711  28657  46368  75025 121393 196418
```

```r
fb3 = diff(fb) # vectorized way
fb3
```

```
##  [1]      1      0      1      1      2      3      5      8     13     21
## [11]     34     55     89    144    233    377    610    987   1597   2584
## [21]   4181   6765  10946  17711  28657  46368  75025 121393 196418
```

```r
all(fb2 == fb3)
```

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