Homework3

Evelin Reyes

10/16/2021

Homework #3

Q1 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 = runif(n = 1:500), nrow = 50, ncol = 10)
colnames(A) = paste("lake", 1:10, sep = "_")
#Answer 1
for(i in 1:10){
  M = mean(A[,i])
  print (M)
}
```

```
## [1] 0.4601492
## [1] 0.4992815
## [1] 0.5987037
## [1] 0.4580486
## [1] 0.4719578
## [1] 0.4965216
## [1] 0.5110536
## [1] 0.4577936
## [1] 0.5193423
## [1] 0.4856413
```

```
colMeans(A)
```

```
##    lake_1    lake_2    lake_3    lake_4    lake_5    lake_6    lake_7    lake_8
## 0.4601492 0.4992815 0.5987037 0.4580486 0.4719578 0.4965216 0.5110536 0.4577936
##    lake_9   lake_10
## 0.5193423 0.4856413
```

Q2 (2 points) From the for loop lecture,

#we see the following example of using apply():

```
x = array(1:27, dim = c(3, 3, 3))
apply(X = x, MARGIN = c(1, 2),
      FUN = paste, collapse = ", ")
```

```
##      [,1]        [,2]        [,3]
## [1,] "1, 10, 19" "4, 13, 22" "7, 16, 25"
## [2,] "2, 11, 20" "5, 14, 23" "8, 17, 26"
## [3,] "3, 12, 21" "6, 15, 24" "9, 18, 27"
```

```
# Answer 2
y = array(0, dim = c(3, 3, 3))
for (i in 1:3){
  for (j in 1:3) {
    for (k in 1:3) {
      y[i,j,] = paste(x[i,j,],collapse = ",")
    }

  }
}
y
```

```
## , , 1
##
##      [,1]      [,2]      [,3]
## [1,] "1,10,19" "4,13,22" "7,16,25"
## [2,] "2,11,20" "5,14,23" "8,17,26"
## [3,] "3,12,21" "6,15,24" "9,18,27"
##
## , , 2
##
##      [,1]      [,2]      [,3]
## [1,] "1,10,19" "4,13,22" "7,16,25"
## [2,] "2,11,20" "5,14,23" "8,17,26"
## [3,] "3,12,21" "6,15,24" "9,18,27"
##
## , , 3
##
##      [,1]      [,2]      [,3]
## [1,] "1,10,19" "4,13,22" "7,16,25"
## [2,] "2,11,20" "5,14,23" "8,17,26"
## [3,] "3,12,21" "6,15,24" "9,18,27"
```

##Q3 (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.

```
#Answer 3
N=numeric()
for (i in 1:30) {
  N[1] = 0
  N[2] = 1
  N[i+2] = N[i]+N[i+1]
}
N
```

```
##  [1]       0       1       1       2       3       5       8      13      21
## [10]      34      55      89     144     233     377     610     987    1597
## [19]    2584    4181    6765   10946   17711   28657   46368   75025  121393
## [28]  196418  317811  514229  832040 1346269
```

##Q4 (2 points) In the example data below, extract those ranking numbers with regular expression. #The results should have the number(s) and . #if it follows after the numbers immediately (i.e., 1., 12., 105., 105.3, etc.). Remove empty strings from the final results. You should get 107 strings for your results.

```
top105 = readLines("http://www.textfiles.com/music/ktop100.txt")
top105 = top105[-c(64, 65)] # missing No. 54 and 55

#Answer 4

pattern <- "^[0-9]*\\.*[0-9]\\."
library("stringr")
A = str_extract(top105,pattern)
A
```

```
##   [1] NA     NA     NA     NA     NA     NA     NA     NA     NA     NA    
##  [11] "1."   "2."   "3."   "4."   "5."   "6."   "7."   "8."   "9."   "10." 
##  [21] "11."  "12."  "13."  "14."  "15."  "16."  "17."  "18."  "19."  "20." 
##  [31] "21."  "22."  "23."  "24."  "25."  "26."  "27."  "28."  "29."  "30." 
##  [41] "31."  "32."  "33."  "34."  "35."  "36."  "37."  "38."  "39."  "40." 
##  [51] "41."  "42."  "43."  "44."  "45."  "46."  "47."  "48."  "49."  "50." 
##  [61] "51."  "52."  "53."  "56."  "57."  "58."  "59."  "60."  "61."  "62." 
##  [71] "63."  "64."  "65."  "66."  "67."  "68."  "69."  "70."  "71."  "72." 
##  [81] "73."  "74."  "75."  "76."  "77."  "78."  "79."  "80."  "81."  "82." 
##  [91] "83."  "83."  "84."  "85."  "86."  "87."  "88."  "89."  "90."  "91." 
## [101] "91."  "92."  "93."  "94."  "95."  "96."  "97."  "97."  "98."  "99." 
## [111] "100." "101." "102." "103." "104." "105." "105." NA     NA     NA    
## [121] NA
```

```
B = grep("^[0-9]*\\.*[0-9]\\.*[0-9]", A, value = TRUE)
B
```

```
##  [1] "10." "11." "12." "13." "14." "15." "16." "17." "18." "19."
## [11] "20." "21." "22." "23." "24." "25." "26." "27." "28." "29."
## [21] "30." "31." "32." "33." "34." "35." "36." "37." "38." "39."
## [31] "40." "41." "42." "43." "44." "45." "46." "47." "48." "49."
## [41] "50." "51." "52." "53." "56." "57." "58." "59." "60." "61."
## [51] "62." "63." "64." "65." "66." "67." "68." "69." "70." "71."
## [61] "72." "73." "74." "75." "76." "77." "78." "79." "80." "81."
## [71] "82."  "83."  "83."  "84."  "85."  "86."  "87."  "88."  "89."  "90."
## [81] "91."  "91."  "92."  "93."  "94."  "95."  "96."  "97."  "97."  "98."
## [91] "99."  "100." "101." "102." "103." "104." "105." "105."
```

##Q5 (2 points) For the vector with length of 107 you got from question 4, remove all trailing .. (hint: ?sub). #Then convert it to a numeric vector and find out which numbers have duplications (i.e., a tie in ranking). Don't count by eyes, use R to find it out (hint: table(), sort(); or duplicated(), which(), [subsetting; there are more than one way to do so).

```
#Answer 5
C = as.numeric(B)
is.numeric(C)
```

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

```
D <- gregexpr(pattern = "(\\d{1,3})", text = C)
E = regmatches(C,D)
C[duplicated(C, incomparables = FALSE)]
```

```
## [1]  83  91  97 105
```