Worksheet 01 - Intro to R programming - NCBS MSc WL (Answers)

Akshay Surendra, Anand M Osuri

23 - September - 2020

Convert the following tasks to R commands and run them:

1. Create a vector `g` that stores even numbers from 10 to 20 in a sequence

```
g <- seq(from = 10,to = 20,by = 2)
```

2. Find the logarithm (base-2) of each element of vector `g` in just one line of code (Clue: *vectorize*)

```
logG <- log2(x = g) # built-in function specifically for log base-2
logG
```

```
## [1] 3.321928 3.584963 3.807355 4.000000 4.169925 4.321928
```

```
logG <-log(x = g,base = 2) # generic log function where you can change the base
```

3. Can you implement 1) and 2) above in one line of code? (Clue: nested functions)

```
logG <- log2(x = seq(from = 10,to = 20,by = 2))
```

4. Run the following code *snippet* (another word for a block of code) in your R script -

```
  matrix(data = rnorm(n = 32,mean = 0,sd = 2),nrow = 4,ncol = 8)
```

```
##           [,1]       [,2]       [,3]       [,4]     [,5]       [,6]     [,7]
## [1,] -2.378035  0.8698326  2.6184570  1.9521929 2.262009  1.3250382 1.279538
## [2,] -3.806625 -0.4306273 -2.0645031  0.6404607 1.669459 -1.4342297 2.891525
## [3,] -1.233222 -1.2110467  0.6159652  1.4323628 5.719429 -0.8192159 0.349111
## [4,]  2.721058 -0.4986965 -1.3831494 -1.1908689 6.104870 -0.5317686 1.518030
##            [,8]
## [1,]  4.1973439
## [2,] -1.1040018
## [3,] -0.0363777
## [4,] -1.0198469
```

Next class: read more about the `rnorm()` function and what it does - this is not required to answer the questions below

5. Save the above matrix in an object and use appropriate indices to extract and print the following:

```
obj1 <- matrix(data = rnorm(n = 32,mean = 0,sd = 2),nrow = 4,ncol = 8)#run once
obj1
```

```
##            [,1]       [,2]       [,3]       [,4]      [,5]      [,6]       [,7]
## [1,]  3.1493447 -1.1036315  2.9045310  2.9143978 -1.514198 -1.225386 -2.2492801
## [2,] -0.2786536 -0.9467363 -0.3996999 -0.6579772  3.472045 -5.618096 -3.4238582
## [3,] -1.4086235  3.6917189 -1.0374467  0.9662396 -1.145788 -1.600536 -1.4140400
## [4,] -0.2253820 -1.0728672 -4.4483844  1.9026299 -1.790314  0.167456  0.2431992
##           [,8]
## [1,] -1.095872
## [2,] -2.745408
## [3,]  3.597806
## [4,] -1.607672
```

a. even-numbered columns

```
# a
obj1[,c(2,4,6,8)]
```

```
##            [,1]       [,2]      [,3]      [,4]
## [1,] -1.1036315  2.9143978 -1.225386 -1.095872
## [2,] -0.9467363 -0.6579772 -5.618096 -2.745408
## [3,]  3.6917189  0.9662396 -1.600536  3.597806
## [4,] -1.0728672  1.9026299  0.167456 -1.607672
```

b. odd-numbered rows

```
# b
obj1[c(1,3),]
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]     [,7]
## [1,]  3.149345 -1.103632  2.904531 2.9143978 -1.514198 -1.225386 -2.24928
## [2,] -1.408623  3.691719 -1.037447 0.9662396 -1.145788 -1.600536 -1.41404
##           [,8]
## [1,] -1.095872
## [2,]  3.597806
```

c. both even-numbered columns AND odd-numbered rows

```
# c
obj1[c(1,3),c(2,4,6,8)]
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -1.103632 2.9143978 -1.225386 -1.095872
## [2,]  3.691719 0.9662396 -1.600536  3.597806
```

6. In the above matrix, *check* if the following statements are true or false (Clue- use one or more of these three operators: relational/logical/assignment):

a. Is the 1st element of row 1 less than the 8th element of row 1?

```r
obj1[1,1] < obj1[1,8]
```

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

b. Is the 4th element of column 1 more than the 6th element of column 1? (Report any result you get, error or otherwise)

```r
# obj1[4,1] > obj1[6,1] ## gives you an error - subscript out of bounds
```

7. Learn what the function `substr()` does using the help command (and also the internet) and answer the following questions:

a. Create a character vector object with 6 elements, and each element must hold the name of a bird native to your country (data type: ?)

```r
obj2 <- c("HoodedCrow","LaughingDove","Redwing","HazelGrouse","Gadwall","AmurFalcon")
obj2
```

```
## [1] "HoodedCrow"   "LaughingDove" "Redwing"      "HazelGrouse"  "Gadwall"
## [6] "AmurFalcon"
```

b. Identify and print the first 2 letters of each element in this vector

```r
first2_1 <- substr(x = obj2[1],start = 1,stop = 2) # includes the starting and ending index
first2_2 <- substr(x = obj2[2],start = 1,stop = 2)
first2_3 <- substr(x = obj2[3],start = 1,stop = 2)
first2_4 <- substr(x = obj2[4],start = 1,stop = 2)
first2_5 <- substr(x = obj2[5],start = 1,stop = 2)
first2_6 <- substr(x = obj2[6],start = 1,stop = 2)

first2_1
```

```
## [1] "Ho"
```

```r
first2_2
```

```
## [1] "La"
```

```r
first2_3
```

```
## [1] "Re"
```

```r
first2_4
```

```
## [1] "Ha"
```

```
first2_5
```

```
## [1] "Ga"
```

```
first2_6
```

```
## [1] "Am"
```

c. If you used 6 lines of code to implement 7-b, can you do it in 1 line? (clue - *vectorize*)

```
first2 <- c(NA,NA,NA,NA,NA,NA) # not required but good practice to define length first
first2 <- NULL # empty vector, akin to saying, d <- data.frame()
first2 <- substr(x = obj2,start = 1,stop = 2)
first2
```

```
## [1] "Ho" "La" "Re" "Ha" "Ga" "Am"
```