A-1.R

pradyuth

Fri Jan 26 16:59:35 2018

```
#Name : Pradyuth Vangur

# Problem 1: The seq() function

# Question 1
seq(from = 10, to = 15, by = 0.5)
```

```
##  [1] 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0
```

```
#from represents the starting point of the sequence of numbers to be generated.
#to represents the end point of the sequence.
#by represents the amount in which the consecutive number should differ from the previous.

seq(from = 10, to = 15, length.out = 6)
```

```
## [1] 10 11 12 13 14 15
```

```
#length.out represents the length of the sequence and the numbers will differ
#uniformly from each other.

#Question 2
seq(10)
```

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

```
#This represents the number of digits to be sequenced.
#It assumes the starting number to be one and ends up with the number inputted.

seq(3,10)
```

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

```
#3 represents the starting number of the sequence.
#10 represents the end of the sequence. Each number differs by 1.

#Question 3
vec_1 <- c(1,10,4,6,7)
seq(vec_1)
```

```
## [1] 1 2 3 4 5
```

```
#We observe that seq takes in the size of the vector and treats it like
#seq(n), where n is the size of vector and returns sequence of numbers
#starting from 1 and ending in n with a difference of 1

#Question 4
vec_2 <- c(4)
seq(vec_2)
```

```
## [1] 1 2 3 4
```

```r
#seq function is considering the first number of the vector as the size
#of the sequence.

#Question 5
seq_along(vec_1)
```

```
## [1] 1 2 3 4 5
```

```r
#We observe that result is same as seq(vec_1)
seq_along(vec_2)
```

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

```r
#seq_along takes in vector and treats it as a single sized vector.

#Question 6
2 + seq_len(8)
```

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

```r
#This returns a sequence from 3 to 10. This basically adds up 2
#to the vector created by seq_len(8)

#Question 7
#seq_len(-8)
#This provides an error because it cannot take in negative numbers.

-seq_len(8)
```

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

```r
#Instead we will have to provide -seq_len(8) to obtain the negative
# sequenced numbers.

seq_len(8)
```

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

```r
#This provides a sequence of length 8 with numbers from 1 to 8 with
#spacing of one.

#Problem 2: Vectors in R

#Question 1
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```r
summary(letters)
```

```
##    Length     Class      Mode
##        26 character character
```

```r
#Question 2
class(letters)
```

```
## [1] "character"
```

```r
#We observe that the datatype of letters is character. We make use
#of class() function.

#Question 3
length(letters)
```

```
## [1] 26
```

```r
#Length of letters is 26

#Question 4
letters[3]
```

```
## [1] "c"
```

```r
#To chooose the nth element from letters, we wrap n inside [] and
#place it adjacent to letters as letters[n]

#Question 5
letters_back <- rev(letters)
letters_back
```

```
##  [1] "z" "y" "x" "w" "v" "u" "t" "s" "r" "q" "p" "o" "n" "m" "l" "k" "j"
## [18] "i" "h" "g" "f" "e" "d" "c" "b" "a"
```

```r
#We make use of the function rev() to reverse it.

#Question 6
letters_back_alt_odd <- letters_back[seq(1, length(letters), 2)]
letters_back_alt_odd
```

```
##  [1] "z" "x" "v" "t" "r" "p" "n" "l" "j" "h" "f" "d" "b"
```

```r
#This code takes in letters which are placed in odd positions.

letters_back_alt_even <- letters_back[seq(2, length(letters), 2)]
letters_back_alt_even
```

```
##  [1] "y" "w" "u" "s" "q" "o" "m" "k" "i" "g" "e" "c" "a"
```

```r
#This code takes in letters which are placed in even positions.

#Question 7
letters_matrix <- matrix(data = letters[seq(1,16)], nrow = 4, ncol = 4)
letters_matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "a"  "e"  "i"  "m"
## [2,] "b"  "f"  "j"  "n"
## [3,] "c"  "g"  "k"  "o"
## [4,] "d"  "h"  "l"  "p"
```

```r
#This generates matrix with the first 16 elements of letters.

#Question 8
month.abb
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
```

```r
class(month.abb)
```

```
## [1] "character"
```

```r
#This represents the abbreviation of the months in character form.

month.name
```

```
##  [1] "January"   "February"  "March"     "April"     "May"
##  [6] "June"      "July"      "August"    "September" "October"
## [11] "November"  "December"
```

```r
class(month.name)
```

```
## [1] "character"
```

```r
#This represents the names of the months.

#Problem 3: Matrices in R

#Question 1
num <- runif(20, min = 0, max = 1)
num
```

```
##  [1] 0.216991469 0.676367763 0.208394637 0.670371957 0.883459471
##  [6] 0.459606897 0.437910749 0.009559304 0.763355482 0.786804910
## [11] 0.133524481 0.899276828 0.780348789 0.451556738 0.388357522
## [16] 0.979900575 0.543719207 0.263530557 0.800860302 0.890235787
```

```r
num_mat <- matrix(data = num, nrow = 4, ncol = 5)
num_mat
```

```
##           [,1]        [,2]      [,3]      [,4]      [,5]
## [1,] 0.2169915 0.883459471 0.7633555 0.7803488 0.5437192
## [2,] 0.6763678 0.459606897 0.7868049 0.4515567 0.2635306
## [3,] 0.2083946 0.437910749 0.1335245 0.3883575 0.8008603
## [4,] 0.6703720 0.009559304 0.8992768 0.9799006 0.8902358
```

```r
#Question 2
num_mat[3, ]
```

```
## [1] 0.2083946 0.4379107 0.1335245 0.3883575 0.8008603
```

```r
#The above code will index the third row

class(num_mat[3, ])
```

```
## [1] "numeric"
```

```r
#It returns numeric as the data-structure

dim(num_mat[3, ])
```

```
## NULL
```

```r
#Question 3
dim(num_mat[3, , drop = FALSE])
```

```
## [1] 1 5
```

```r
#Dimension is displayed here

#Question 4
sum(num_mat)
```

```
## [1] 11.24413
```

```r
#Question 5
num_mat_tf <- num_mat/sum(num_mat)
num_mat_tf
```

```
##            [,1]         [,2]       [,3]       [,4]       [,5]
## [1,] 0.01929819 0.0785707033 0.06788922 0.06940053 0.04835581
## [2,] 0.06015295 0.0408752618 0.06997470 0.04015932 0.02343716
## [3,] 0.01853363 0.0389457091 0.01187504 0.03453868 0.07122472
## [4,] 0.05961971 0.0008501592 0.07997742 0.08714772 0.07917336
```

```r
sum(num_mat_tf)
```

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

```r
#Question 6
row <- rowSums(num_mat)
row
```

```
## [1] 3.187874 2.637867 1.969048 3.449344
```

```r
col <- colSums(num_mat)
col
```

```
## [1] 1.772126 1.790536 2.582962 2.600164 2.498346
```

```r
#Question 7
num_mat_row <- num_mat / row
num_mat_row
```

```
##            [,1]        [,2]       [,3]      [,4]      [,5]
## [1,] 0.06806776 0.277131202 0.23945595 0.2447866 0.1705585
## [2,] 0.25640709 0.174234304 0.29827317 0.1711825 0.0999029
## [3,] 0.10583524 0.222397228 0.06781171 0.1972311 0.4067247
## [4,] 0.19434764 0.002771339 0.26070949 0.2840831 0.2580884
```

```r
rowSums(num_mat_row)
```

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

```r
#Question 8
t(num_mat)
```

```
##           [,1]      [,2]      [,3]        [,4]
## [1,] 0.2169915 0.6763678 0.2083946 0.670371957
## [2,] 0.8834595 0.4596069 0.4379107 0.009559304
## [3,] 0.7633555 0.7868049 0.1335245 0.899276828
## [4,] 0.7803488 0.4515567 0.3883575 0.979900575
## [5,] 0.5437192 0.2635306 0.8008603 0.890235787
```

```r
#t() transposes the matrix

#Question 9
num_mat_col <- t(num_mat) / col
```

```r
#Divide the transpose of num_mat with col

num_mat_colt <- t(num_mat_col)
#Transpose the matrix whose column sum is 1

num_mat_colt
```

```
##           [,1]        [,2]       [,3]      [,4]      [,5]
## [1,] 0.1224470 0.493404915 0.29553496 0.3001153 0.2176317
## [2,] 0.3816703 0.256686707 0.30461346 0.1736647 0.1054820
## [3,] 0.1175958 0.244569585 0.05169433 0.1493589 0.3205562
## [4,] 0.3782869 0.005338793 0.34815724 0.3768611 0.3563301
```

```r
colSums(num_mat_colt)
```

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

```r
#We see that sum of each column is one

#Question 10
#Case 1: Looking at the numbers vertically in each columns
mat <- matrix(runif(18, min = 0, max = 5), nrow = 2, ncol = 9)
mat
```

```
##           [,1]     [,2]     [,3]      [,4]     [,5]     [,6]     [,7]
## [1,] 0.3062087 3.274413 0.397337 0.2961476 1.984522 4.287635 3.848923
## [2,] 1.4200734 2.216787 2.121647 0.8814427 1.741560 3.931259 1.898664
##          [,8]     [,9]
## [1,] 3.019071 3.753238
## [2,] 3.641871 3.764224
```

```r
mat_new1 <- matrix(data = mat[2,], nrow = 3, ncol = 3)
#mat[2, ] represents the alternate elements to be considered
mat_new1
```

```
##          [,1]      [,2]     [,3]
## [1,] 1.420073 0.8814427 1.898664
## [2,] 2.216787 1.7415595 3.641871
## [3,] 2.121647 3.9312589 3.764224
```

```r
#Case 2: Moving along columns of the matrix
mat_new_t2 <- t(mat)
mat_new2 <- matrix(data = mat_new_t2[,1], nrow = 3, ncol = 3)
#mat[,1] represents the alternate elements of the column to be considered
mat_new2
```

```
##           [,1]      [,2]     [,3]
## [1,] 0.3062087 0.2961476 3.848923
## [2,] 3.2744130 1.9845221 3.019071
## [3,] 0.3973370 4.2876346 3.753238
```

```r
#Question 11
m <- cbind(1, 1:7) # the '1' (= shorter vector) is recycled
#A matrix with two columns and 7 rows is generated

m
```

```
##      [,1] [,2]
```

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

```
m <- cbind(m, 8:14)[, c(1, 3, 2)] # insert a column
#Now, another column is being added to m matrix and positioning
#is done accordingly by [, c(1,3,2)]
m
```

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