2020MCS120003_LabAssignment02

Amith C A

06/09/2020

DSC513 Lab Assignment 2

06/09/2020

1. What output will the following r code produce? Code:

```
x<-c(TRUE,FALSE,0L)
typeof(x)
```

Ans: The output will be:

```
## [1] "integer"
```

The typeof(),will print the type of c. The output will be “integer”.The data type of the c() is integer as ‘0L’ is an element. The L is used to specify the value of 0 as integer.

2. What output will the following r code produce? Code:

```
TRUE | NA
```

Ans: The output will be :

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

The | is logical pipe which performs the logical OR operation.Performing the logial OR operation on TRUE | TRUE and TRUE | FLASE will give TRUE.

3. Let x be defined as: Code:

```
x<-c(’0’,’10’,’5’,’20’,’15’,’10’,’0’,’5’)
```

Write an R function that would turn x into a factor whose ordering corresponds to the numerical ordering of x

```
x<-c(’0’,’10’,’5’,’20’,’15’,’10’,’0’,’5’)
 y<-factor(sort(as.numeric(x)),ordered = TRUE)
```

Ans: The **numeric()** converts the list x to numeric. The **sort()** sorts the elements in ascending order. The **factor()** function with **ordered=TRUE** creates the factor y in ordered form.

4. In R, if mtcars is a data frame, why does mtcars[1:20] return an error? How does it differ from the similar mtcars[1:20,]?

Ans: When slicing a dataframe using mtcars[1:20] there is condition to select only the rows from 1 to 20 but no mention of what to do with the column. In case of mtcars[1:20,] the rows 1 to 20 will get selected from the data frame mtcars. the **,** after 20 suggests that all the columns have to selected.So it does not throw error.

```
mtcars[mtcars$cyl = 4]
mtcars[-1:4]
mtcars[mtars$cyl <= 5]
mtcars[mtcars$cyl == 4| 6,]
```

5. Fix each of the following common data frame subsetting errors in R:

Ans:

- **mt[mtcars$cyl=4]** code is trying to get the records from mtcars which have value 4 for **cyl** column.

Corrected code: **mtcars[mtcars$cyl==4,]**

- **mtcars[-1:4,]** code is trying to print the first 4 rows and all columns excluding the first row. But -1 can only be used with 0,also there should be a **,** to get the correct syntax.So the desired code is.

Corrected code: **mtcars[2:4,]**

- **mtcars[mtars$cyl <= 5]** is trying to print all the records in mtcars data frame having the value of cyl column less than or equal to 5.

Corrected code: **mtcars[mtars$cyl <= 5,]**

- **mtcars[mtcars$cyl == 4| 6,]** is trying to extract the data from mtcars dataframe which have cyl as 4 or 6.For this we use **c()**.

Corrected code : **mtcars[mtcars$cyl == c(4,6),]**

6. Create the vector (20,19,. . . ,2,1) in R ?

Ans: The output will be :

```
20:1
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

7. Create a 6X10 matrix of random integers in R

```
x<-matrix(sample(1:100,60,replace=TRUE),nrow=6,ncol=10,byrow=TRUE)
x
```

Ans:

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    3   58   12   14   99   10   69   33    7    16
## [2,]   31   77   98   83   30   40   87   78   55     3
## [3,]   50   42   10   73   49   60   31   69   90    72
## [4,]   11   69   53   73   35   14   73   23   25    48
## [5,]   93   71   97   39   77   61   62   86   61    19
## [6,]   47   68   19   39   91   74   19   87   12    86
```

8. Write a function to find the number of entries in each row of a matrix that are greater than 4.

```
x<-matrix(sample(1:10,60,replace=TRUE),nrow=6,ncol=10,byrow=TRUE)
x
```

Ans:

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

```
m<-apply(x,1,function(a){sum(a>4)})
m
```

```
## [1] 5 4 6 8 4 6
```

9.In continuation of the previous question. write a function to find how many rows have exactly two instances of the number 7.

```r
x<-matrix(sample(1:10,60,replace=TRUE),nrow=6,ncol=10,byrow=TRUE)
x
```

Ans:

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

```r
y<-which(apply(x,1,function(a){sum(a==7)==2}))
y
```

```
## [1] 4
```

10. Create a vector of the values of e^xcos(x) at x=3,3.1,3.2,. . . ..,6.

```r
x<-seq(3,6,by=0.1)
x
```

Ans:

```
##  [1] 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8
## [20] 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0
```

```r
exp(x)*cos(x)
```

```
##  [1] -19.884531 -22.178753 -24.490697 -26.773182 -28.969238 -31.011186
##  [7] -32.819775 -34.303360 -35.357194 -35.862834 -35.687732 -34.685042
## [13] -32.693695 -29.538816 -25.032529 -18.975233 -11.157417  -1.362099
## [19]  10.632038  25.046705  42.099201  61.996630  84.929067 111.061586
## [25] 140.525075 173.405776 209.733494 249.468441 292.486707 338.564378
## [31] 387.360340
```

11. Create the following by writing code snippets:

Ans:

- $\sum_{i=1}^{100}(i^3+4i^2)$

```
i<-1:100
sum(i^3+4*i^2)
```

```
## [1] 26855900
```

- $\sum_{i=1}^{25}(\frac{2^i}{i}+\frac{3^i}{i^2})$

```
i<-1:25
sum(((2^i)/i)+(3^i)/(i^2))
```

```
## [1] 2129170437
```

```


set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
```

12. Execute the following lines which create two vectors of random integers which are chosen with replecement from the integers 0,1,. . . . . . , 9999. both vectors have length 250 . The code: Suppose x=(x_1,x_2,. . . .,x_n) denotes the vector xVec and y =(y_1,y_2,. . . .,y_n) denotes the vector yVec. a. Create the vector (y_2-x_1,. . . y_n-x_{n-1})

```


set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
yVec[-1]-xVec[-length(xVec)]
```

Ans:

```
##   [1] -359  692 -724   40 -626 -719 -809  527  -89 -829  248  144 -749 -352 -220
##  [16] -249  387 -492   85 -106  303  -97 -436  146  282 -206 -385  -96 -567 -757
##  [31]  287  277 -562  292  -89  -93 -847 -822 -203  679  309 -199 -273    4  -47
##  [46]  142  122  414 -602 -304 -674   -8 -662 -168 -349  -63 -221  115    1 -600
##  [61] -382 -487    2  375   19 -113 -634  107   60   47  214 -325  -49 -290  169
##  [76]  290 -624  457 -408  581 -189  204  -80  409  209 -410  461   37 -127  185
##  [91]  382 -446   44  -56 -270 -598 -378 -155  134 -187  109  316 -139  158  305
## [106]  -39 -119  182  441 -403 -107  615  614 -378 -464   31 -385  665  674 -217
## [121] -279 -406  -45 -489 -350 -451  -18  660  504   -6   60 -130 -379 -302 -219
## [136]  -21  438  129 -201 -275  131  694  -96 -176  117 -113  887 -439 -126 -148
## [151]  392 -158  444 -291  232  -12 -274  477 -510  336 -759 -363 -195 -220  160
## [166] -308 -333  302 -183  227  -12  428  665 -301   -8  222  -50 -444 -425 -650
## [181] -424  318  154  238 -727   71  472  908  265  654 -644 -754  657 -382 -313
## [196]  910 -381  394 -596  602  397 -572  378 -274 -271  601 -791 -378 -461   39
## [211]  163 -118 -332 -170  -94  262 -474  566 -273 -366 -400  374   42  100  135
## [226]  609 -527  580 -219  128 -524  620 -206  410 -280  -66  -50  252  279   48
## [241] -595  -59 -623  247  514   62 -102  475  287
```

b.Calculate $\sum_{i=1}^{n-1}\frac{e^{-x_i+1}}{x_i+10}$

```
set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
sum(exp(-xVec[-1])/(xVec[-length(xVec)]+10))
```

Ans:

```
## [1] 5.029496e-05
```

13. This question uses the vectors xVec and yVec created in the previous question and the functions sort,order, mean, sqrt, sum and abs.

a. Pick out the values in yVec which are > 600.

```
set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
yVec[yVec>600]
```

Ans:

```
##  [1] 702 901 617 726 915 723 941 906 782 681 721 929 827 653 839 800 869 692 840
## [20] 845 769 866 696 685 788 642 902 797 601 656 842 970 680 792 662 868 875 795
## [39] 880 700 665 699 979 796 772 836 974 990 954 846 943 658 655 628 623 629 989
## [58] 738 992 758 870 910 933 641 872 904 647 988 753 624 996 621 714 965 920 755
## [77] 783 856 927 759 700 764 666 667 790 654 959 868 963 698 686
```

b. What are the index positions in yVec of the values which are > 600?

```
set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
which(yVec>600)
```

Ans:

```
##  [1]   3   9  10  18  20  22  25  26  27  29  37  41  42  43  45  48  49  51  65
## [20]  67  71  74  79  81  84  85  88  95  98  99 103 106 108 109 110 113 114 119
## [39] 120 129 130 131 138 139 143 147 148 152 154 159 161 166 167 168 172 173 174
## [58] 176 177 183 187 188 189 190 191 194 196 197 201 202 204 206 207 211 212 219
## [77] 223 224 225 227 229 230 233 235 238 239 240 243 246 248 249
```

c. What are the values in xVec which correspond to the values in yVec which are > 600? (By correspond, we mean at the same index positions.)

```r
set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
xVec[yVec>600]
```

Ans:

```
##  [1] 819 706 903 761 439 481 624 988 473 568 926 518 852 593  86 455 773 935 398
## [20] 755 335 500 810 755 233 125 332 440 811 385 591 345 610 221 646 261 640 206
## [39] 388 161 705 319 667 286 605  87 895 561 777 576 778 963 961 212 201 324 387
## [58] 770 258 232 438  25 376 218 665 708  78 762 227 873 390 113 839 757 397 601
## [77] 814 827  79 566 983   3 317 523 402 680 512 687 398 211 139
```

d. How many values in yVec are within 200 of the maximum value of the terms in yVec?

```r
set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
 sum( yVec>max(yVec)-200 )
```

Ans:

```
## [1] 42
```

e. How many numbers in xVec are divisible by 2?

```r
set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
sum(xVec%%2==0)
```

Ans:

```
## [1] 117
```

f. Sort the numbers in the vector xVec in the order of increasing values in yVec.

```r
set.seed(50)
xVec<-sample(0:999,250,replace=T)
yVec<-sample(0:999,250,replace=T)
xVec[order(yVec)]
```

Ans:

```
##   [1] 271 725 957 151 374  10 919 996 325 120 216 978 997 409 474 261 607 979
##  [19] 814 271 905 362 692 746 777 793 130  94 257 840 892 435  68 703 862  23
##  [37] 949 853 250 986 813 669 996 441 504 975  49  46  98 239 274 358 598 799
##  [55] 159 885  94 150 114 611 650 339 988 778 881 344 764 189 247 391 180  43
##  [73] 541 487 635 868 180 865 215 830 465 521 253 609  78 440 618 799 259 835
##  [91] 960 921 420 581 927 711 752 257 346 102 966 272 665 640 563 104 887 510
## [109] 276 958 160 855 662 795  40 450 648 656  12 234 915 362 765 800 678 786
## [127] 769 485 251 598 926 805 161 449 310 924 369 777  17 765  59 795 367 499
## [145] 498 778 274 450 651 722 954  55 470 526 469 749 477  31 962 811 903 113
## [163] 201 873 212 324 218 125  78 593 680 961 385 963 646 705 317 523 610 568
## [181] 755 139 935 810 211 319 161 983 819 839 926 481 761 770 227 601 232 566
## [199]   3 335 605 473 814 233 402 221 206 286 440 455 852  87  86 398 591 755
## [217] 576 827 500 261 687 773 438 665 640 388 706 332 708 988  25 439 397  79
## [235] 518 376 624 778 777 512 398 757 345 895 667 762 387 561 258 390
```

14. Try exploring the function cumprod() or any other function of your choice to calculate: $1 + \frac{2}{3} + (\frac{2}{3}\ \frac{4}{5}) + (\frac{2}{3}\ \frac{4}{5}\ \frac{6}{7}\ ) + \ldots . + (\frac{2}{3}\ \frac{4}{5} . \ . \ . \frac{38}{39})$

```
1+sum(cumprod(seq(2,38,b=2)/seq(3,39,b=2)))
```

Ans

15. Consider the continuous function f(x). Write a function tmpFn which takes a single argument xVec. The function should return the vector of values of the function f(x) evaluated at the values in xVec.

```
tmpFn <- function(x)
{
ifelse(x < 0, x^2 + 2*x + 3, ifelse(x < 2, x+3, x^2 + 4*x - 7))
}
a <- seq(-3, 3, len=100)
plot(a, tmpFn(a), type="l")
```

Ans: