Practical File
Of
Data Analytics Using R
(PCC-CSEAI204-P)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU JAMBHESHWAR UNIVERSITY OF SCIENCE & TECHNOLOGY
Hisar – Haryana (India)

Submitted To:
Dr. Anupma Sangwan
Department of CSE

Submitted By: Aaditya Ranjan
Roll Number: 210010150002
Class: B.Tech (CSE-AI & ML)

INDEX

8	Install and load the MASS package and access the *Boston* dataset. Study the dataset from the resources available onthe internet and write what you can find relevant to the dataset.	16 - 17	
9	Write a script file to compute the following of the numeric variable in *Boston* dataset. (a). Sum (b). Range (c). Mean (d). Standard deviation	18	
10.	Create a vector x of all those values from 1:100 that are divisible by 5 and do the following operations on the vector: (a). Find the length of vector x. (b). Print the values stored at the fifth, tenth, and fifteenth location of vector x. (c). Find the sum mean range median and standard deviation of vector x. (d). Replace the fifth and tenth values with NA and NaN values, respectively and find the mean of modified vector. (e). Check if x contains any NA values and print the indices of NA values in vector x. (f). Remove NA values from vector x and use summary command on it. (g). Print the values of first and third quartile of vector x from the output of the summary command.	19 - 21	

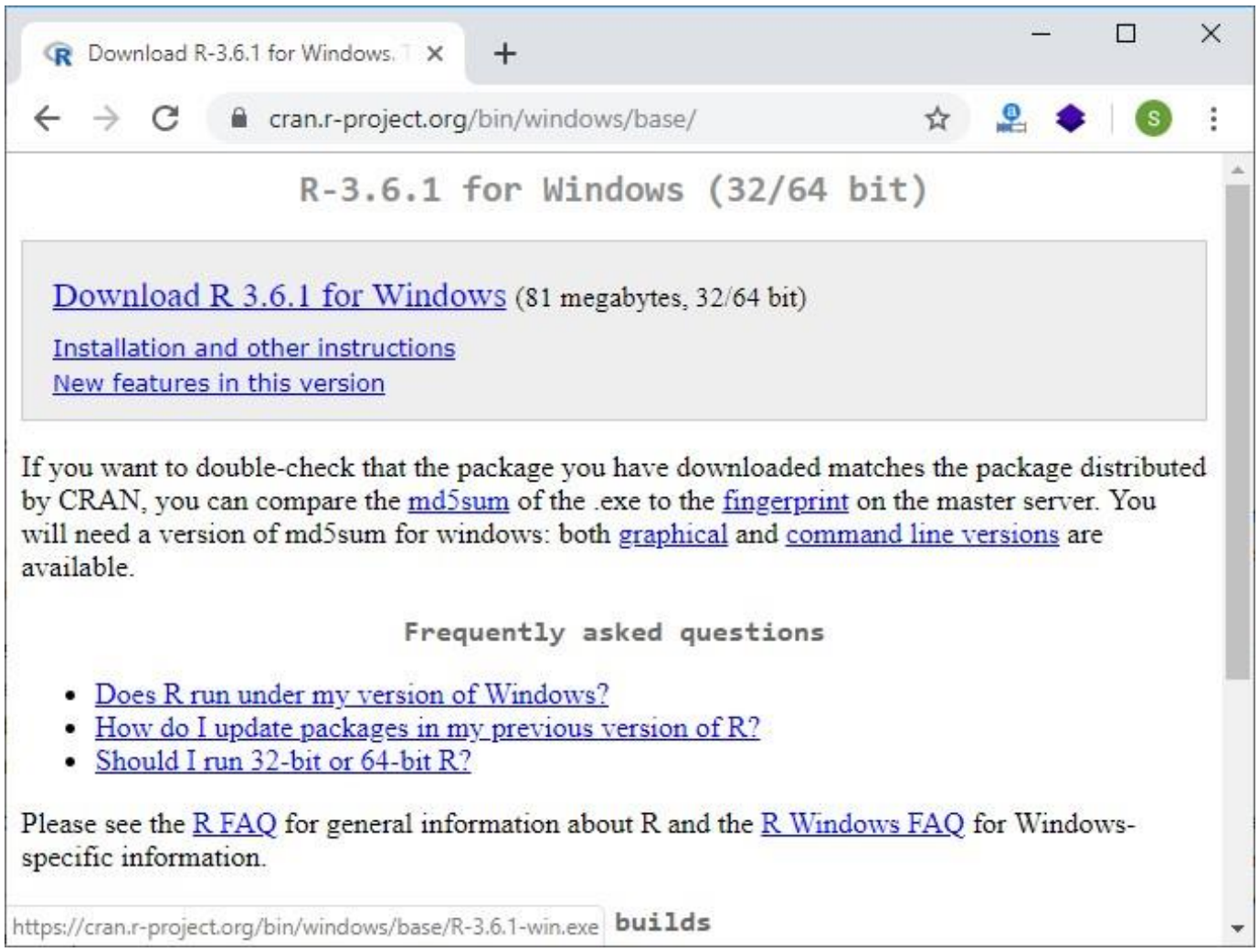
11.	Assume the given vectors and do the following operations:
	x = 1:12; y = 13:24;
	z = 1:6; a = 1:12;
	b = (13, 15, 17, 19, 20, 21, 23, 25, 27, 29, 31, 24); c = (5, 10, 15, NA, 25, NaN);
		22 - 25	

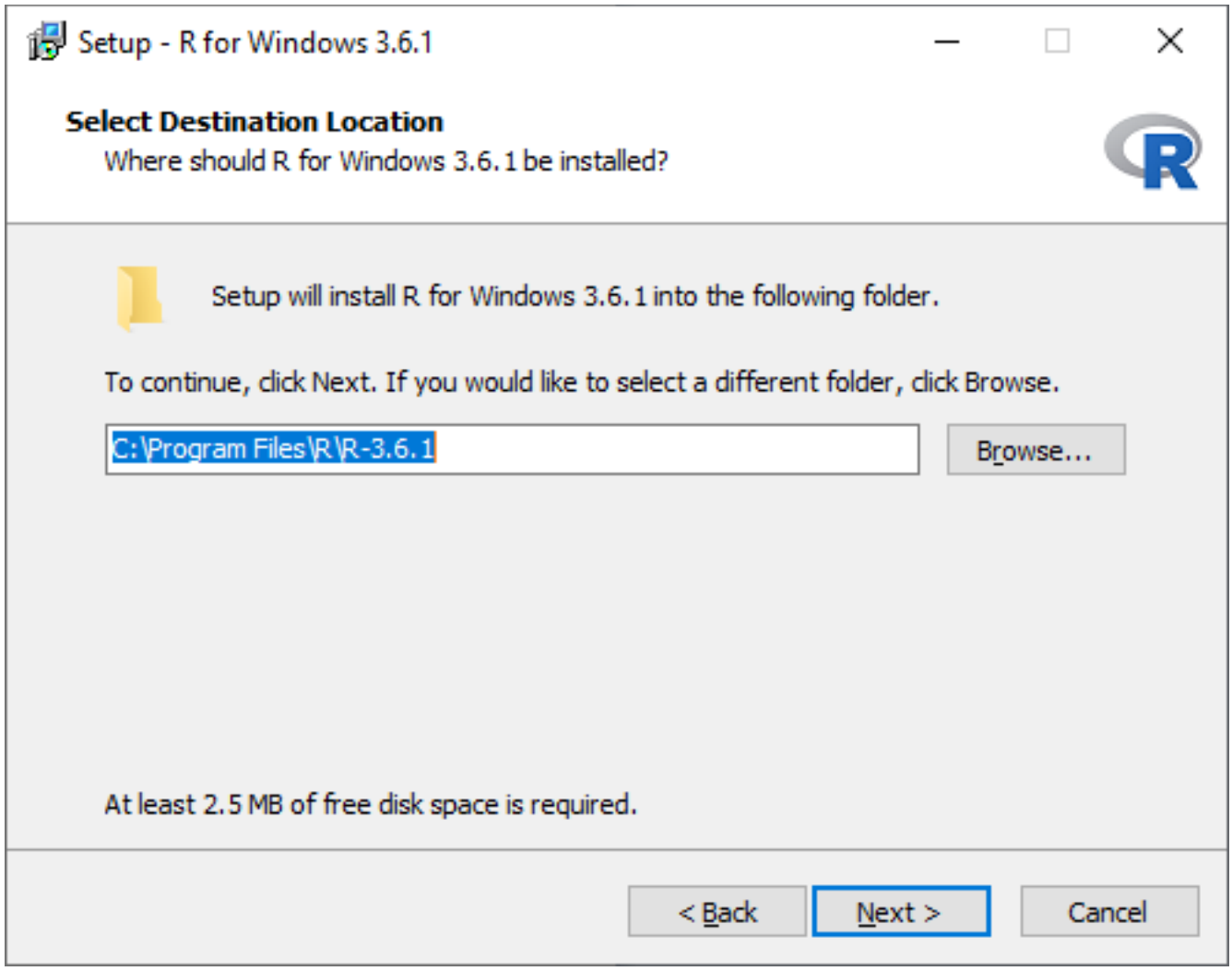
	v = (26, 21, 87, 56, 72, 60); k = (0, 2, 4, 6, 8, 16, 32) (a). Find x × y and x × y × z and interpret the output. (b). Do an element-wise comparison between x and a, andy and b. (c). Find all the elements that are greater than 6 of vectorx and store these elements into another vector p. (d). Check for NA and NaN values in vectors b and c. (e). Check if overall vector x is equal to vector a and vector b. (f). Why does *identical(x, z)* evaluate to FALSE? (g). What is the difference between *all()* and *all.equal()* functions? Illustrate with the help of an example.(h). Run any(x, z) function and interpret the out. (i). Create a new vector of the non-NA values of vector cusing a single line code. (j). Sort vector v in descending order and output theoriginal indices in order of the sorted elements. Find log to the base 2 of vector k.		
12	**.** Assuming the character vector cv = c("*sunita*", "*bimla*", "*kavita*", "*geeta*", "*anu*", "*dikshita*", "*susmita*", "*seema*"):(a). Find the character count in each name. (b). Find the *geeta* exist in vector cv	25 - 26	
13	Output the indices of the names that contain substring *ee*in vector cv of assignment 12	26	
14	Find out how many strings end with the letters ta invector cv of in assignment 12.	26	
15	Create a vector of factor type data for the hair color of ten people where values for hair colors are black, dark brown, grey, blonde.	27	
16	Apply class, str, and summary commands to the vector created in Q14).	27	
17	Create an empty vector of factor data type for the names of the first six months in a year. Remember to keep the levels of data in order of the months, from January to June.	27	
18	Create a vector to store the grades of 20 students for the first minor exam. Grades are given at four levels (A, B, C, D). Compute the modal grade. Further, store the grades of the same students for the second minor exam. Compare the grades for the two exams. Count	27 - 28	

	the number of students who have got a higher grade in the second minor.		
19	Create a matrix m of five rows in a row-major order of numbers from 1 to 100 incremented by a step of 5 units: a) Find row and column-wise mean of matrix m. b) Find the minimum value for each row and column. c) Find the transpose and sort the value in each column in decreasing order. d) Assign the row names R1 to R5 and column C1 to C4. e) Display all the elements of the second and forth column without using indices. f) Display all the elements of the first and third row without using indices. g) Create a new matrix by deleting the second and forth column of the matrix m using indices and column names. h) Replace elements at indices (2,3), (2,4), (3,3), and (3,4) with NA values. i) Replace element at index (1,3) with NaN. j) Check if matrix m contains any NA or NaN values and interpret the output. k) Create two matrices rm and cm by concatenating matrix m row-wise and column-wise with itself.	28 - 33	
20	Interpret the output of the following commands: a) n= matrix(rep(m,2),nrow=ncol(m),byrow=FALSE) b)n=matrix(rep(m,2),nrow=nrow(m),byrow=FALSE) c) n= matrix(rep(m,2),nrow=ncol(m), byrow= TRUE) d) m1= do.call(rbind,replicate(2,m,simplify=FALSE)) e) m2= do.call(cbind,replicate(2,m,simplify=FALSE)) f) Rename row and column names as per the requirements of matrix m1 and m2.	33 - 37	

1. Install R and then install R Studio. Get yourself acquainted with the GUI of various working windows of RStudio.
   Steps to install R and RStudio:

   i. Install R. Download the R installer from https://cran.r- -project.org/
   ii. Install RStudio. Download RStudio: https://www.rstudio.com/products/rstudio/download/
   iii. Check that R and RStudio are working. Open RStudio. ...
   iv. Install R packages required for the workshop.




Setup - R for Windows 3.6.1
Select Destination Location
Where should R for Windows 3.6.1 be installed?
Setup will install R for Windows 3.6.1 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\R\R-3.6.1
Browse...
At least 2.5 MB of free disk space is required.
< Back
Next >
Cancel

2. Create the following objects in R and then check their class

(a). A vector of strings

(b). A vector consisting of factor type data. For instance, a vector consisting of hair color of a few individuals.

(c). A list data type consisting of vectors of names of five students and a matrix of the marks of students in four courses.

(d). A data frame consisting of names of students, their age, total marks, and grades awarded

```
#2 (a)
t<-c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey")
class(t)

#2 (b)
hc<-c("Blonde", "Skin", "Black", "Yellow", "Brown")
g<-factor(hc)
g
class(g)

#2 (c)
s_name<-c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey")
r_name=c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey")
c_name=c("Maths", "COA", "History", "Sanskrit")

R<-matrix(c(80:99), nrow=5, dimnames=list(row_name, col_name))
print(R)

first<-list(s_name, R)
first
class(first)

#2 (d)
student<-data.frame(
  student_name=c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey"),
  student_age=c(19,19,20,21,18),
  marks=c(444,528,600,400,310),
  grades=c("D","B", "A", "C", "E")
)
```

student
class(student)

Output:

```
> #Aaditya Ranjan
> #210010150002 CSE(AI & ML)
>
> #2 (a)
> t<-c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey")
> class(t)
[1] "character"
>
> #2 (b)s
> hc<-c("Blonde", "Skin", "Black", "Yellow", "Brown")
> g<-factor(hc)
> g
[1] Blonde Skin   Black  Yellow Brown
Levels: Black Blonde Brown Skin Yellow
> class(g)
[1] "factor"
>
> #2 (c)
> s_name<-c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey")
> r_name=c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey")
> c_name=c("Maths", "COA", "History", "Sanskrit")
>
> R<-matrix(c(80:99), nrow=5, dimnames=list(r_name, c_name))
> print(R)
        Maths COA History Sanskrit
Aaditya    80  85      90       95
Rohit      81  86      91       96
Alex       82  87      92       97
Prateek    83  88      93       98
Pandey     84  89      94       99
>
> first<-list(s_name, R)
> first
[[1]]
[1] "Aaditya" "Rohit"   "Alex"    "Prateek" "Pandey"

[[2]]
        Maths COA History Sanskrit
Aaditya    80  85      90       95
Rohit      81  86      91       96
Alex       82  87      92       97
Prateek    83  88      93       98
Pandey     84  89      94       99

> class(first)
[1] "list"
```

```
>
> #2 (d)
> student<-data.frame(
+   student_name=c("Aaditya", "Rohit", "Alex", "Prateek", "Pandey"),
+   student_age=c(19,19,20,21,18),
+   marks=c(444,528,600,400,310),
+   grades=c("D","B", "A", "C", "E")
+ )
> student
  student_name student_age marks grades
1      Aaditya          19   444      D
2        Rohit          19   528      B
3         Alex          20   600      A
4      Prateek          21   400      C
5       Pandey          18   310      E
> class(student)
[1] "data.frame"
>
```

3. Apply str and summary commands to the object created in assignment 2.Interpret the output.

#3 (a)
str(t)
summary(t)

#3 (b)
str(g)
summary(g)

#3 (c)
str(first)
summary(first)

#3 (d)
str(student)
summary(student)

Output:

```
>
> #3 (a)
> str(t)
 chr [1:5] "Aaditya" "Rohit" "Alex" "Prateek" "Pandey"
> summary(t)
   Length     Class      Mode
        5 character character
>
> #3 (b)
> str(g)
 Factor w/ 5 levels "Black","Blonde",..: 2 4 1 5 3
> summary(g)
 Black Blonde  Brown   Skin Yellow
     1      1      1      1      1
>
> #3 (c)
> str(first)
List of 2
 $ : chr [1:5] "Aaditya" "Rohit" "Alex" "Prateek" ...
 $ : int [1:5, 1:4] 80 81 82 83 84 85 86 87 88 89 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5] "Aaditya" "Rohit" "Alex" "Prateek" ...
  .. ..$ : chr [1:4] "Maths" "COA" "History" "Sanskrit"
> summary(first)
     Length Class  Mode
[1,]  5     -none- character
[2,] 20     -none- numeric
>
> #3 (d)
> str(student)
'data.frame':   5 obs. of  4 variables:
 $ student_name: chr  "Aaditya" "Rohit" "Alex" "Prateek" ...
 $ student_age : num  19 19 20 21 18
 $ marks       : num  444 528 600 400 310
 $ grades      : chr  "D" "B" "A" "C" ...
> summary(student)
 student_name        student_age       marks          grades
 Length:5           Min.   :18.0   Min.   :310.0   Length:5
 Class :character   1st Qu.:19.0   1st Qu.:400.0   Class :character
 Mode  :character   Median :19.0   Median :444.0   Mode  :character
                    Mean   :19.4   Mean   :456.4
                    3rd Qu.:20.0   3rd Qu.:528.0
                    Max.   :21.0   Max.   :600.0
>
```

4. Check and justify the outcome of the followingexpressions:
(a). sqrt (3) ^2 ==3
(b). near (sqrt (3) ^2, 3)

```
#4 (a)
library(dplyr)
sqrt(3)^2==3

#4 (b)
 near(sqrt(3)^2,3)
```

Output :

```
> #4 (a)
> library("dplyr")
> sqrt(3)^2==3
[1] FALSE
> #4 (b)
> near(sqrt(3)^2,3)
[1] TRUE
>
```

5. Install, load package *'stringdist'* and run the followingcode:
my_string = c("Viraj", "Viraj", "Viraj", "Vikraj", "Viraji","Viroj", "Vroj", "Siroji")
name = "Viraj"
matched = (stringdist(my_string, name) ==0)matched = (stringdist(my_string, name) ==1)matched = (stringdist(my_string, name) ==2)

Interpret the output. [Hint: „*stringdist*" is a package to findthe distance between strings in term of replacement, insertion and declaration of letters.

```
#5 (a)
#install.packges(stringdist)
library("stringdist")
my_strings=c("Viraj", "Virat", "Vikraj", "Viraji", "Viroj", "Vroj", "Siroji")
name="Viraj"
matched=(stringdist(my_strings, name)==0)
matched
matched=(stringdist(my_strings, name)==1)
matched
matched=(stringdist(my_strings, name)==2)
matched
```

Output:

```
> #5 (a)
> #install.packges(stringdist)
> library("stringdist")
> my_strings=c("Viraj", "Virat", "Vikraj", "Viraji", "Viroj", "Vroj", "Siroji")
> name="Viraj"
> matched=(stringdist(my_strings, name)==0)
> matched
[1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
> matched=(stringdist(my_strings, name)==1)
> matched
[1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
> matched=(stringdist(my_strings, name)==2)
> matched
[1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
```

6. Apply summary command to iris dataset of the *'datasets'* package and interprets the output.

```
#6
install.packages("datasets")
library("datasets")
summary(iris)
```

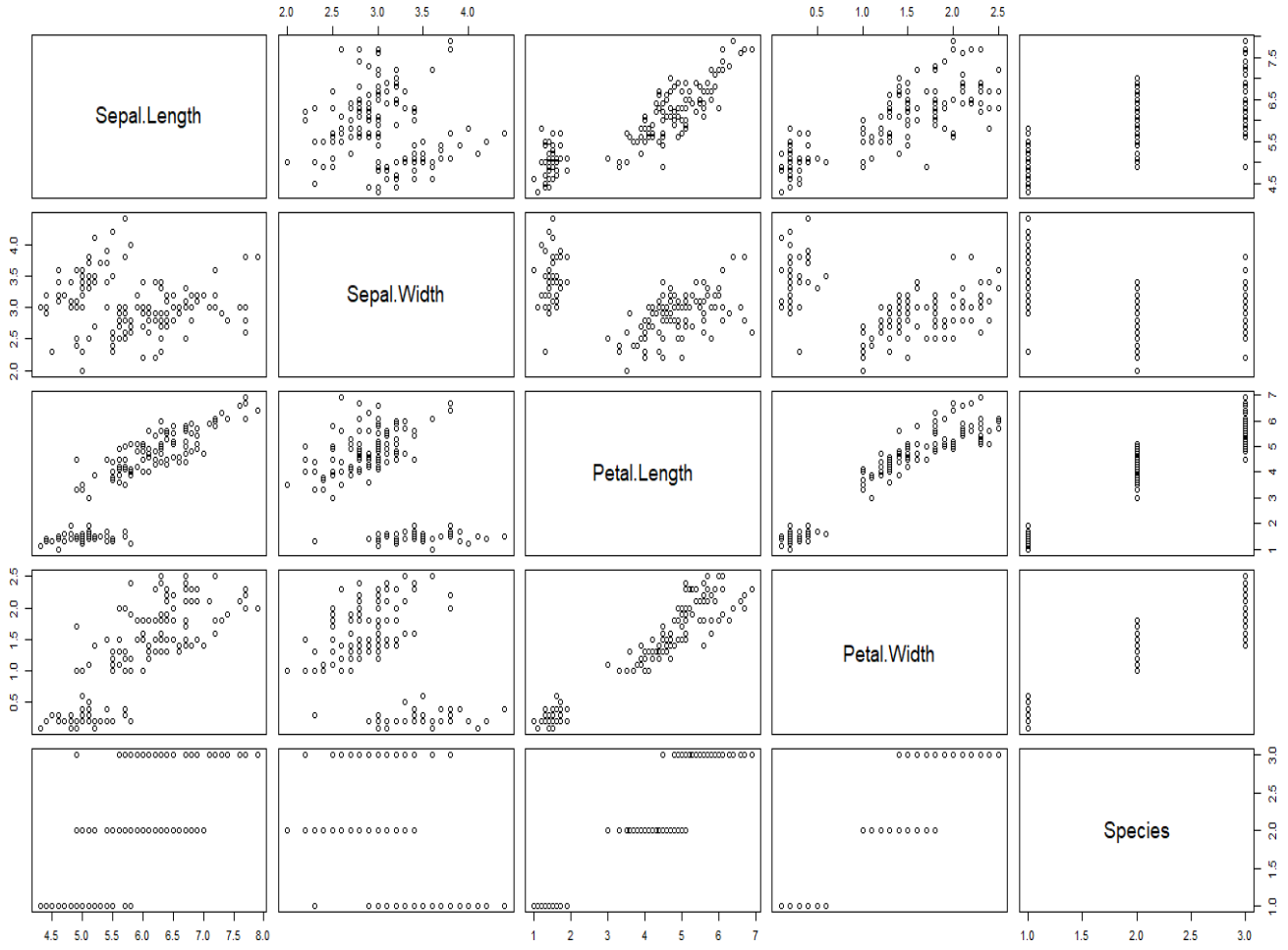
Output:

```
> #6
> #install.packages("datasets")
> library("datasets")
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

7. Use plot(iris) function and interpret the output. Writedown your finding about the dataset.

#7
plot(iris)

Output:



8. Install and load the MASS package and access the *Boston* dataset. Study the dataset from the resources available on the internet and write what you can find relevant to the dataset.

#8
library(MASS)
Boston

Output:

```
> #8
> library(MASS)
> Boston
      crim    zn indus chas    nox    rm   age    dis rad tax ptratio  black lstat medv
1  0.00632  18.0  2.31    0 0.5380 6.575  65.2 4.0900   1 296    15.3 396.90  4.98 24.0
2  0.02731   0.0  7.07    0 0.4690 6.421  78.9 4.9671   2 242    17.8 396.90  9.14 21.6
3  0.02729   0.0  7.07    0 0.4690 7.185  61.1 4.9671   2 242    17.8 392.83  4.03 34.7
4  0.03237   0.0  2.18    0 0.4580 6.998  45.8 6.0622   3 222    18.7 394.63  2.94 33.4
5  0.06905   0.0  2.18    0 0.4580 7.147  54.2 6.0622   3 222    18.7 396.90  5.33 36.2
6  0.02985   0.0  2.18    0 0.4580 6.430  58.7 6.0622   3 222    18.7 394.12  5.21 28.7
7  0.08829  12.5  7.87    0 0.5240 6.012  66.6 5.5605   5 311    15.2 395.60 12.43 22.9
8  0.14455  12.5  7.87    0 0.5240 6.172  96.1 5.9505   5 311    15.2 396.90 19.15 27.1
9  0.21124  12.5  7.87    0 0.5240 5.631 100.0 6.0821   5 311    15.2 386.63 29.93 16.5
10 0.17004  12.5  7.87    0 0.5240 6.004  85.9 6.5921   5 311    15.2 386.71 17.10 18.9
11 0.22489  12.5  7.87    0 0.5240 6.377  94.3 6.3467   5 311    15.2 392.52 20.45 15.0
12 0.11747  12.5  7.87    0 0.5240 6.009  82.9 6.2267   5 311    15.2 396.90 13.27 18.9
13 0.09378  12.5  7.87    0 0.5240 5.889  39.0 5.4509   5 311    15.2 390.50 15.71 21.7
14 0.62976   0.0  8.14    0 0.5380 5.949  61.8 4.7075   4 307    21.0 396.90  8.26 20.4
15 0.63796   0.0  8.14    0 0.5380 6.096  84.5 4.4619   4 307    21.0 380.02 10.26 18.2
16 0.62739   0.0  8.14    0 0.5380 5.834  56.5 4.4986   4 307    21.0 395.62  8.47 19.9
17 1.05393   0.0  8.14    0 0.5380 5.935  29.3 4.4986   4 307    21.0 386.85  6.58 23.1
18 0.78420   0.0  8.14    0 0.5380 5.990  81.7 4.2579   4 307    21.0 386.75 14.67 17.5
19 0.80271   0.0  8.14    0 0.5380 5.456  36.6 3.7965   4 307    21.0 288.99 11.69 20.2
20 0.72580   0.0  8.14    0 0.5380 5.727  69.5 3.7965   4 307    21.0 390.95 11.28 18.2
21 1.25179   0.0  8.14    0 0.5380 5.570  98.1 3.7979   4 307    21.0 376.57 21.02 13.6
22 0.85204   0.0  8.14    0 0.5380 5.965  89.2 4.0123   4 307    21.0 392.53 13.83 19.6
23 1.23247   0.0  8.14    0 0.5380 6.142  91.7 3.9769   4 307    21.0 396.90 18.72 15.2
24 0.98843   0.0  8.14    0 0.5380 5.813 100.0 4.0952   4 307    21.0 394.54 19.88 14.5
25 0.75026   0.0  8.14    0 0.5380 5.924  94.1 4.3996   4 307    21.0 394.33 16.30 15.6
26 0.84054   0.0  8.14    0 0.5380 5.599  85.7 4.4546   4 307    21.0 303.42 16.51 13.9
27 0.67191   0.0  8.14    0 0.5380 5.813  90.3 4.6820   4 307    21.0 376.88 14.81 16.6
28 0.95577   0.0  8.14    0 0.5380 6.047  88.8 4.4534   4 307    21.0 306.38 17.28 14.8
29 0.77299   0.0  8.14    0 0.5380 6.495  94.4 4.4547   4 307    21.0 387.94 12.80 18.4
30 1.00245   0.0  8.14    0 0.5380 6.674  87.3 4.2390   4 307    21.0 380.23 11.98 21.0
31 1.13081   0.0  8.14    0 0.5380 5.713  94.1 4.2330   4 307    21.0 360.17 22.60 12.7
32 1.35472   0.0  8.14    0 0.5380 6.072 100.0 4.1750   4 307    21.0 376.73 13.04 14.5
33 1.38799   0.0  8.14    0 0.5380 5.950  82.0 3.9900   4 307    21.0 232.60 27.71 13.2
34 1.15172   0.0  8.14    0 0.5380 5.701  95.0 3.7872   4 307    21.0 358.77 18.35 13.1
35 1.61282   0.0  8.14    0 0.5380 6.096  96.9 3.7598   4 307    21.0 248.31 20.34 13.5
36 0.06417   0.0  5.96    0 0.4990 5.933  68.2 3.3603   5 279    19.2 396.90  9.68 18.9
37 0.09744   0.0  5.96    0 0.4990 5.841  61.4 3.3779   5 279    19.2 377.56 11.41 20.0
38 0.08014   0.0  5.96    0 0.4990 5.850  41.5 3.9342   5 279    19.2 396.90  8.77 21.0
39 0.17505   0.0  5.96    0 0.4990 5.966  30.2 3.8473   5 279    19.2 393.43 10.13 24.7
40 0.02763  75.0  2.95    0 0.4280 6.595  21.8 5.4011   3 252    18.3 395.63  4.32 30.8
41 0.03359  75.0  2.95    0 0.4280 7.024  15.8 5.4011   3 252    18.3 395.62  1.98 34.9
42 0.12744   0.0  6.91    0 0.4480 6.770   2.9 5.7209   3 233    17.9 385.41  4.84 26.6
43 0.14150   0.0  6.91    0 0.4480 6.169   6.6 5.7209   3 233    17.9 383.37  5.81 25.3
44 0.15936   0.0  6.91    0 0.4480 6.211   6.5 5.7209   3 233    17.9 394.46  7.44 24.7
45 0.12269   0.0  6.91    0 0.4480 6.069  40.0 5.7209   3 233    17.9 389.39  9.55 21.2
46 0.17142   0.0  6.91    0 0.4480 5.682  33.8 5.1004   3 233    17.9 396.90 10.21 19.3
47 0.18836   0.0  6.91    0 0.4480 5.786  33.3 5.1004   3 233    17.9 396.90 14.15 20.0
48 0.22927   0.0  6.91    0 0.4480 6.030  85.5 5.6894   3 233    17.9 392.74 18.80 16.6
49 0.25387   0.0  6.91    0 0.4480 5.399  95.3 5.8700   3 233    17.9 396.90 30.81 14.4
50 0.21977   0.0  6.91    0 0.4480 5.602  62.0 6.0877   3 233    17.9 396.90 16.20 19.4
```

```
51 0.08873  21.0  5.64    0 0.4390 5.963  45.7 6.8147   4 243    16.8 395.56 13.45 19.7
52 0.04337  21.0  5.64    0 0.4390 6.115  63.0 6.8147   4 243    16.8 393.97  9.43 20.5
53 0.05360  21.0  5.64    0 0.4390 6.511  21.1 6.8147   4 243    16.8 396.90  5.28 25.0
54 0.04981  21.0  5.64    0 0.4390 5.998  21.4 6.8147   4 243    16.8 396.90  8.43 23.4
55 0.01360  75.0  4.00    0 0.4100 5.888  47.6 7.3197   3 469    21.1 396.90 14.80 18.9
56 0.01311  90.0  1.22    0 0.4030 7.249  21.9 8.6966   5 226    17.9 395.93  4.81 35.4
57 0.02055  85.0  0.74    0 0.4100 6.383  35.7 9.1876   2 313    17.3 396.90  5.77 24.7
58 0.01432 100.0  1.32    0 0.4110 6.816  40.5 8.3248   5 256    15.1 392.90  3.95 31.6
59 0.15445  25.0  5.13    0 0.4530 6.145  29.2 7.8148   8 284    19.7 390.68  6.86 23.3
60 0.10328  25.0  5.13    0 0.4530 5.927  47.2 6.9320   8 284    19.7 396.90  9.22 19.6
61 0.14932  25.0  5.13    0 0.4530 5.741  66.2 7.2254   8 284    19.7 395.11 13.15 18.7
62 0.17171  25.0  5.13    0 0.4530 5.966  93.4 6.8185   8 284    19.7 378.08 14.44 16.0
63 0.11027  25.0  5.13    0 0.4530 6.456  67.8 7.2255   8 284    19.7 396.90  6.73 22.2
64 0.12650  25.0  5.13    0 0.4530 6.762  43.4 7.9809   8 284    19.7 395.58  9.50 25.0
65 0.01951  17.5  1.38    0 0.4161 7.104  59.5 9.2229   3 216    18.6 393.24  8.05 33.0
66 0.03584  80.0  3.37    0 0.3980 6.290  17.8 6.6115   4 337    16.1 396.90  4.67 23.5
67 0.04379  80.0  3.37    0 0.3980 5.787  31.1 6.6115   4 337    16.1 396.90 10.24 19.4
68 0.05789  12.5  6.07    0 0.4090 5.878  21.4 6.4980   4 345    18.9 396.21  8.10 22.0
69 0.13554  12.5  6.07    0 0.4090 5.594  36.8 6.4980   4 345    18.9 396.90 13.09 17.4
70 0.12816  12.5  6.07    0 0.4090 5.885  33.0 6.4980   4 345    18.9 396.90  8.79 20.9
71 0.08826   0.0 10.81    0 0.4130 6.417   6.6 5.2873   4 305    19.2 383.73  6.72 24.2
 [ reached 'max' / getOption("max.print") -- omitted 435 rows ]
```

9. Write a script file to compute the following of the numeric variable in *Boston* dataset.

(a). Sum

(b). Range

(c). Mean

(d). Standard deviation

#9

sapply(Boston, sum)

sapply(Boston, range)

sapply(Boston, mean)

sapply(Boston, sd)

Output:

```
> #9
> sapply(Boston, sum)
      crim         zn      indus       chas        nox         rm        age        dis        rad        tax    ptratio
 1828.4429  5750.0000  5635.2100    35.0000   280.6757  3180.0250 34698.9000  1920.2916  4832.0000 206568.0000  9338.5000
     black      lstat       medv
180477.0600  6402.4500 11401.6000
> sapply(Boston, range)
        crim  zn indus chas   nox    rm   age     dis rad tax ptratio  black lstat medv
[1,]  0.00632   0  0.46    0 0.385 3.561   2.9  1.1296   1 187    12.6   0.32  1.73    5
[2,] 88.97620 100 27.74    1 0.871 8.780 100.0 12.1265  24 711    22.0 396.90 37.97   50
> sapply(Boston, mean)
       crim          zn       indus        chas         nox          rm         age         dis         rad         tax
 3.61352356 11.36363636 11.13677866  0.06916996  0.55469506  6.28463439 68.57490119  3.79504269  9.54940711 408.23715415
    ptratio       black       lstat        medv
18.45553360 356.67403162 12.65306324 22.53280632
> sapply(Boston, sd)
      crim         zn      indus       chas        nox         rm        age        dis        rad        tax    ptratio
 8.6015451 23.3224530  6.8603529  0.2539940  0.1158777  0.7026171 28.1488614  2.1057101  8.7072594 168.5371161  2.1649455
     black      lstat       medv
91.2948644  7.1410615  9.1971041
>
```

10. Create a vector x of all those values from 1:100 that aredivisible by 5 and do the following operations on the vector:

(a). Find the length of vector x.

(b). Print the values stored at the fifth, tenth, andfifteenth location of vector x.

(c). Find the sum mean range median and standard deviation of vector x.

(d). Replace the fifth and tenth values with NA and NaN values, respectively and find the mean of modified vector.

(f). Check if x contains any NA values and print the indices of NA values in vector x.

(g). Remove NA values from vector x and use summary command on it.

Print the values of first and third quartile of vector x from the output of the summary command.

```
#10 (a)
n=5*1:20
n
length(n)

#10 (b)
print(n[c(5,10,15)])

#10 (c)
sum(n)
mean(n)
range(n)
median(n)
sd(n)

#10 (d)
n[5]=NA
n[10]=NaN
n
mean(n)

#10 (e)
```

```
is.na(n)
which(is.na(n))

#10 (f)
x= na.omit(n)
print(x)

#10 (g)
x=c(seq(5,100,5))
   x
   summary(x)
   y=summary(x)
   y["1st Qu."]
   y["3rd Qu."]
```

Output:

```
> #10 (a)
> n=5*1:20
> n
 [1]   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95 100
> length(n)
[1] 20
>
> #10 (b)
> print(n[c(5,10,15)])
[1] 25 50 75
>
> #10 (c)
> sum(n)
[1] 1050
> mean(n)
[1] 52.5
> range(n)
[1]   5 100
> median(n)
[1] 52.5
> sd(n)
[1] 29.5804
>
> #10 (d)
> n[5]=NA
> n[10]=NaN
> n
 [1]   5  10  15  20  NA  30  35  40  45 NaN  55  60  65  70  75  80  85  90  95 100
> mean(n)
[1] NA
>
```

```
> #10 (e)
> is.na(n)
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> which(is.na(n))
[1]  5 10
>
> #10 (f)
> x= na.omit(n)
> print(x)
 [1]   5  10  15  20  30  35  40  45  55  60  65  70  75  80  85  90  95 100
attr(,"na.action")
[1]  5 10
attr(,"class")
[1] "omit"
>
> #10 (g)
> x=c(seq(5,100,5))
>     x
 [1]   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95 100
>     summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00   28.75   52.50   52.50   76.25  100.00
>     y=summary(x)
>     y["1st Qu."]
1st Qu.
  28.75
>     y["3rd Qu."]
3rd Qu.
  76.25
>
```

Q11) Assume the given vectors and do the following operations: x=1:12; y=13:24; z=1:6; a=1:12; b=(13,15,17,19,20,21,23,25,27,29,31,24); c=(5,10,15,NA,25,NAN); v=(26,21,87,56,72,60); k=(0,2,4,8,16,32)

a)Find x*y and x*y*z and interpret the output.
b)Do an element-wise comparison between x and a, and y and b.
c)Find all the elements that are greater than 6 of vector x and store these elements into another vector p.
d)Check for NA and NaN values in vectors b and c.
e)Check if overall vector x is equal to vector a and vector b.
f) Why does identical (x,z) evaluate to FALSE?
g)What is the difference between all() and all.equal() functions? Illustrate with the help of an example.
h)Run any(x,z) function and interpret the output.
i) Create a new vector of the non-NA values of vector c using a single line code.

j) Sort vector v in descending order and output the original indices in order of the sorted elements.

k)Find log to the base 2 of vector k.

```
#Aaditya Ranjan
#210010150002
#11
 x=1:12
 y=13:24
 z=1:6
a=1:12
 b=c(13,15,17,19,20,21,23,25,27,29,31,24)
c=c(5,10,15,NA,25,NaN)
 v=c(26,21,87,56,72,60)
 k=c(0,2,4,8,16,32)

# 11 (a)
 x*y
x*y*z

#11 ( b)
 x==a
 y==b

# 11 (c)
 x>6
 p=x[x>6] p

#11 (d)
 is.na(b)
 is.nan(b)
 is.na(c)
 is.nan(c)

#11 (e)
 all(x==a)
 all(x==b)

#11 (f)
```

```
identical(x,z)

#11 ( g)
all.equal(x,y)
all(x,y)

#11 (h)
any(x,z)

#11 ( i)
k1=na.omit(c)
k1

#11 (j)
sort(v,decreasing=TRUE)

#11 ( k)
log2(k)
```

OUTPUT :

```
> #Aaditya Ranjan
> #210010150002
> #11
> x=1:12
> y=13:24
> z=1:6
> a=1:12
> b=c(13,15,17,19,20,21,23,25,27,29,31,24)
> c=c(5,10,15,NA,25,NaN)
> v=c(26,21,87,56,72,60)
> k=c(0,2,4,8,16,32)
>
> #11 (a)
> x*y
 [1]  13  28  45  64  85 108 133 160 189 220 253 288
> x*y*z
 [1]   13   56  135  256  425  648  133  320  567  880 1265 1728
>
> #11 (b)
> x==a
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
> y==b
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
>
> #11 (c)
> x>6
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
> p=x[x>6]
> p
[1]  7  8  9 10 11 12
```

```
> #11 (d)
> is.na(b)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> is.nan(b)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> is.na(c)
[1] FALSE FALSE FALSE  TRUE FALSE  TRUE
> is.nan(c)
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
>
> #11 (e)
> all(x==a)
[1] TRUE
> all(x==b)
[1] FALSE
>
> #11 (f)
> identical(x,z)
[1] FALSE
>
> #11 (g)
> all.equal(x,y)
[1] "Mean relative difference: 1.846154"
> all(x,y)
[1] TRUE
>
> #11(h)
> any(x,z)
[1] TRUE
>
> #11 (i)
> k1=na.omit(c)
> k1
[1]  5 10 15 25
attr(,"na.action")
[1] 4 6
attr(,"class")
[1] "omit"
> #11 (j)
> sort(v,decreasing=TRUE)
[1] 87 72 60 56 26 21
>
> #11 (k)
> log2(k)
[1] -Inf    1    2    3    4    5
>
```

12. Assuming the character vector cv = c(“*sunita*”, “*bimla*”,“*kavita*”, “*geeta*”, “*anu*”, “*dikshita*”, “*susmita*”, “*seema*”):
(a). Find the character count in each name.
(b). Find the *geeta* exist in vector cv

#Aaditya Ranjan
#210010150002
#12
cv=c(“sunita”, “bimla”, “kavita”, “geeta”, “dikshita”, “sushmita”, “seema”)

#a

nchar(cv)

#b
Any(cv== “geeta”)

OUTPUT :

```
> #Aaditya Ranjan
> #210010150002
>
> #12
> cv<-c("Sunita", "Bimla", "Kavita", "Geeta", "Dikshita", "Sushmita", "Seema")
> cv
[1] "Sunita"   "Bimla"    "Kavita"   "Geeta"    "Dikshita" "Sushmita" "Seema"
>
> #a
> nchar(cv)
[1] 6 5 6 5 8 8 5
>
> #b
> any(cv=="Geeta")
[1] TRUE
>
```

13. Output the indices of the names that contain substring *ee* in vector cv of assignment 12.

#13
for(i in1:length(cv)){
if(grepl(‘ee’, cv[i] == T)){
print(i)
}

}

```
> #Aaditya Ranjan
> #210010150002
> #13
> for(i in 1:length(cv))
+ if(grepl("ee",cv[i])==TRUE)
+ print(i)
[1] 4
[1] 8
>
```

14. Find out how many strings end with the letters tainvector cv of in assignment 12.

#Aaditya ranjan
#210010150002
#14
endswith(cv, “ta”)

```
> #Aaditya Ranjan
> #210010150002
> #14
> endsWith(cv,"ta")
[1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
>
```

15 Create a vector of factor type data for the hair color of ten people where values for hair colors are black, dark brown, grey, blonde.

hair=factor(c("grey","darkbrown","black","blonde","black","black","blonde","dark brown","dark brown","blonde"))
levels(hair)
table(hair)
max(table(hair))

```
> #Aaditya Ranjan
> #210010150002
> #14
> endsWith(cv,"ta")
[1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
>
> #Aaditya Ranjan
> #210010150002
> #15
> hair=factor(c("grey","dark brown","black","blonde","black","black","blonde","dark brown","dark brown","blonde"))
> levels(hair)
[1] "black"      "blonde"     "dark brown" "grey"
> table(hair)
hair
     black     blonde dark brown       grey
         3          3          3          1
> max(table(hair))
[1] 3
>
```

16. Apply class, str, and summary commands to the vector created in Q12.

class(cv)
str(cv)
summary(cv)

```
> #Aaditya Ranjan
> #210010150002
> #16
> class(cv)
[1] "character"
> str(cv)
 chr [1:8] "sunita" "bimla" "kavita" "geeta" "anu" "dikshita" "sushmi\nta" "seema"
> summary(cv)
   Length     Class      Mode
        8 character character
```

17. Create an empty vector of factor data type for the names of the first six months in a year. Remember to keep the levels of data in order of the months, from January to June.

vec=factor(c(),levels=c("january","feb","march","april","may","june"),ordered = TRUE)
vec

```
>
> #Aaditya Ranjan
> #210010150002
> #17
> vec=factor(c(),levels=c("january","feb","march","april","may","june"),ordered = TRUE)
> vec
ordered(0)
Levels: january < feb < march < april < may < june
>
```

18. Create a vector to store the grades of 20 students for the first minor exam. Grades are given at four levels (A, B, C, D). Compute the modal grade. Further, store the grades of the same students for the second minor exam. Compare the grades for the two exams. Count the number of students who have got a higher grade in the second minor.

minor1=factor(c("A","B","C","D","C","B","C","D","B","B","C","D","B","B","B","D"), levels=c("A","B","C","D"), ordered=TRUE)
which.max(table(minor1))
minor2=factor(c("D","B","C","D","C","C","C","D","D","B","C","D","B","B","A","D"), levels=c("A","B","C","D"), ordered=TRUE)
minor1==minor2

sum(minor1>minor2)

```
> #Aaditya Ranjan
> #2100101500002
> #18
> minor1=factor(c("A","B","C","D","C","B","C","D","B","B","C","D","B","B","B","D"), levels=c("A","B","C","D"),ordered=TRUE)
> which.max(table(minor1))
B
2
> minor2=factor(c("D","B","C","D","C","C","C","D","D","B","C","D","B","B","A","D"), levels=c("A","B","C","D"),ordered=TRUE)
> minor1==minor2
 [1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
> sum(minor1>minor2)
[1] 1
>
```

Q19) Create a matrix m of five rows in a row-major order of numbers from 1 to 100 incremented by a step of 5 units:

a) Find row and column-wise mean of matrix m.
b) Find the minimum value for each row and column.
c) Find the transpose and sort the value in each column in decreasing order.
d) Assign the row names R1 to R5 and column C1 to C4.
e) Display all the elements of the second and forth column without using indices.
f) Display all the elements of the first and third row without using indices.
g) Create a new matrix by deleting the second and forth column of the matrix m using indices and column names.
h) Replace elements at indices (2,3), (2,4), (3,3), and (3,4) with NA values.
i) Replace element at index (1,3) with NaN.
j) Check if matrix m contains any NA or NaN values and interpret the output.
k) Create two matrices rm and cm by concatenating matrix m row-wise and column-wise with itself.

```
#Q19
m=matrix(seq(1,100,5),nrow=5)
 m
 #19 (a)
 rowMeans(m)
colMeans(m)
#19 (b)
 apply(m,MARGIN=1,min)
apply(m,MARGIN=2,min)
```

```
#19 (c)
 t(m)
 apply(t(m), MARGIN=2, function(x) sort(x,decreasing= TRUE))
 #19 (d)
rownames(m)=c("R1","R2","R3","R4","R5")
colnames(m)=c("C1","C2","C3","C4")
m
 #19 (e)
 m[,c("C2","C4")]
 #19 (f)
 m[c("R1","R3"),]
 #19 (g)
 m1=m[,-c(2,4)] m1
 #19 (h)
 m[(2:3),(3:4)]=NA
 m
#19 (i)
 m[1,3]=NaN
m
 #19 (j)
 is.na(m)
 is.nan(m)
 #19 (k)
 rm=matrix(rep(m,2),ncol=ncol(m),byrow=FALSE)
 rm
cm=matrix(rep(m,2),nrow=nrow(m),byrow=TRUE)
cm
```

```
> #Q19
> m=matrix(seq(1,100,5),nrow=5)
> m
     [,1] [,2] [,3] [,4]
[1,]    1   26   51   76
[2,]    6   31   56   81
[3,]   11   36   61   86
[4,]   16   41   66   91
[5,]   21   46   71   96
>
> #19 (a)
> rowMeans(m)
[1] 38.5 43.5 48.5 53.5 58.5
> colMeans(m)
[1] 11 36 61 86
>
> #19 (b)
> apply(m,MARGIN=1,min)
[1]  1  6 11 16 21
> apply(m,MARGIN=2,min)
[1]  1 26 51 76
>
> #19 (c)
> t(m)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]   26   31   36   41   46
[3,]   51   56   61   66   71
[4,]   76   81   86   91   96
> apply(t(m), MARGIN=2, function(x) sort(x,decreasing= TRUE))
     [,1] [,2] [,3] [,4] [,5]
[1,]   76   81   86   91   96
[2,]   51   56   61   66   71
[3,]   26   31   36   41   46
[4,]    1    6   11   16   21
>
> #19 (d)
> rownames(m)=c("R1","R2","R3","R4","R5")
> colnames(m)=c("C1","C2","C3","C4")
> m
   C1 C2 C3 C4
R1  1 26 51 76
R2  6 31 56 81
R3 11 36 61 86
R4 16 41 66 91
R5 21 46 71 96
>
```

```
R 4.2.1 · ~/
>
> #19 (e)
> m[,c("C2","C4")]
   C2 C4
R1 26 76
R2 31 81
R3 36 86
R4 41 91
R5 46 96
>
> #19 (f)
> m[c("R1","R3"),]
   C1 C2 C3 C4
R1  1 26 51 76
R3 11 36 61 86
>
> #19 (g)
> m1=m[,-c(2,4)]
> m1
   C1 C3
R1  1 51
R2  6 56
R3 11 61
R4 16 66
R5 21 71
>
> #19 (h)
> m[(2:3),(3:4)]=NA
> m
   C1 C2 C3 C4
R1  1 26 51 76
R2  6 31 NA NA
R3 11 36 NA NA
R4 16 41 66 91
R5 21 46 71 96
>
> #19 (i)
> m[1,3]=NaN
> m
   C1 C2  C3 C4
R1  1 26 NaN 76
R2  6 31  NA NA
R3 11 36  NA NA
R4 16 41  66 91
R5 21 46  71 96
>
> #19 (j)
> is.na(m)
      C1    C2    C3    C4
R1 FALSE FALSE  TRUE FALSE
R2 FALSE FALSE  TRUE  TRUE
R3 FALSE FALSE  TRUE  TRUE
R4 FALSE FALSE FALSE FALSE
R5 FALSE FALSE FALSE FALSE
```

```
R 4.2.1 · ~/
   C1 C2 C3 C4
R1  1 26 51 76
R2  6 31 NA NA
R3 11 36 NA NA
R4 16 41 66 91
R5 21 46 71 96
>
> #19 (i)
> m[1,3]=NaN
> m
   C1 C2  C3 C4
R1  1 26 NaN 76
R2  6 31  NA NA
R3 11 36  NA NA
R4 16 41  66 91
R5 21 46  71 96
>
> #19 (j)
> is.na(m)
      C1    C2    C3    C4
R1 FALSE FALSE  TRUE FALSE
R2 FALSE FALSE  TRUE  TRUE
R3 FALSE FALSE  TRUE  TRUE
R4 FALSE FALSE FALSE FALSE
R5 FALSE FALSE FALSE FALSE
> is.nan(m)
      C1    C2    C3    C4
R1 FALSE FALSE  TRUE FALSE
R2 FALSE FALSE FALSE FALSE
R3 FALSE FALSE FALSE FALSE
R4 FALSE FALSE FALSE FALSE
R5 FALSE FALSE FALSE FALSE
>
> #19 (k)
> rm=matrix(rep(m,2),ncol=ncol(m),byrow=FALSE)
> rm
      [,1] [,2] [,3] [,4]
 [1,]    1  NaN    1  NaN
 [2,]    6   NA    6   NA
 [3,]   11   NA   11   NA
 [4,]   16   66   16   66
 [5,]   21   71   21   71
 [6,]   26   76   26   76
 [7,]   31   NA   31   NA
 [8,]   36   NA   36   NA
 [9,]   41   91   41   91
[10,]   46   96   46   96
> cm=matrix(rep(m,2),nrow=nrow(m),byrow=TRUE)
> cm
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    6   11   16   21   26   31   36
[2,]   41   46  NaN   NA   NA   66   71   76
[3,]   NA   NA   91   96    1    6   11   16
[4,]   21   26   31   36   41   46  NaN   NA
[5,]   NA   66   71   76   NA   NA   91   96
>
```

Q20) Interpret the output of the following commands:

a)n= matrix(rep(m,2),nrow=ncol(m), byrow= FALSE)

b)n= matrix(rep(m,2),nrow=nrow(m), byrow= FALSE)

c)n= matrix(rep(m,2),nrow=ncol(m), byrow= TRUE)

d)m1= do.call(rbind, replicate(2, m, simplify= FALSE))

e)m2= do.call(cbind, replicate(2, m, simplify= FALSE))

f) Rename row and column names as per the requirements of matrix m1 and m2.

```
#20
m=matrix(seq(1,100,5),nrow=5)
 m
 #20 (a)
 n= matrix(rep(m,2),nrow=ncol(m), byrow= FALSE)
 n
#20 (b)
 n= matrix(rep(m,2),nrow=nrow(m), byrow= FALSE)
 n
 #20 (c)
 n= matrix(rep(m,2),nrow=ncol(m), byrow= TRUE)
 n
 #20 (d)
m1= do.call(rbind, replicate(2, m, simplify= FALSE))
m1
#20(e)
 m2= do.call(cbind, replicate(2, m, simplify= FALSE))
 m2
#20 (f)
 rownames(m1)= c("R1", "R2", "R3", "R4", "R5", "R6","R7", "R8", "R9",
"R10")
 m1
 colnames(m2)=c("C1", "C2", "C3", "C4", "C5", "C6","C7", "C8")
 m2
```

```
> #20
> m=matrix(seq(1,100,5),nrow=5)
> m
     [,1] [,2] [,3] [,4]
[1,]    1   26   51   76
[2,]    6   31   56   81
[3,]   11   36   61   86
[4,]   16   41   66   91
[5,]   21   46   71   96
>
> #20 (a)
> n= matrix(rep(m,2),nrow=ncol(m), byrow= FALSE)
> n
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1   21   41   61   81    1   21   41   61    81
[2,]    6   26   46   66   86    6   26   46   66    86
[3,]   11   31   51   71   91   11   31   51   71    91
[4,]   16   36   56   76   96   16   36   56   76    96
>
> #20 (b)
> n= matrix(rep(m,2),nrow=nrow(m), byrow= FALSE)
> n
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1   26   51   76    1   26   51   76
[2,]    6   31   56   81    6   31   56   81
[3,]   11   36   61   86   11   36   61   86
[4,]   16   41   66   91   16   41   66   91
[5,]   21   46   71   96   21   46   71   96
>
> #20 (c)
> n= matrix(rep(m,2),nrow=ncol(m), byrow= TRUE)
> n
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    6   11   16   21   26   31   36   41    46
[2,]   51   56   61   66   71   76   81   86   91    96
[3,]    1    6   11   16   21   26   31   36   41    46
[4,]   51   56   61   66   71   76   81   86   91    96
>
> #20 (d)
> m1= do.call(rbind, replicate(2, m, simplify= FALSE))
> m1
      [,1] [,2] [,3] [,4]
 [1,]    1   26   51   76
 [2,]    6   31   56   81
 [3,]   11   36   61   86
 [4,]   16   41   66   91
 [5,]   21   46   71   96
 [6,]    1   26   51   76
 [7,]    6   31   56   81
 [8,]   11   36   61   86
 [9,]   16   41   66   91
[10,]   21   46   71   96
>
> #20(e)
```

```
> #20(e)
> m2= do.call(cbind, replicate(2, m, simplify= FALSE))
> m2
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1   26   51   76    1   26   51   76
[2,]    6   31   56   81    6   31   56   81
[3,]   11   36   61   86   11   36   61   86
[4,]   16   41   66   91   16   41   66   91
[5,]   21   46   71   96   21   46   71   96
>
```

```
> #20 (f)
> rownames(m1)= c("R1", "R2", "R3", "R4", "R5", "R6","R7", "R8", "R9", "R10")
> m1
    [,1] [,2] [,3] [,4]
R1     1   26   51   76
R2     6   31   56   81
R3    11   36   61   86
R4    16   41   66   91
R5    21   46   71   96
R6     1   26   51   76
R7     6   31   56   81
R8    11   36   61   86
R9    16   41   66   91
R10   21   46   71   96
> colnames(m2)=c("C1", "C2", "C3", "C4", "C5", "C6","C7", "C8")
> m2
     C1 C2 C3 C4 C5 C6 C7 C8
[1,]  1 26 51 76  1 26 51 76
[2,]  6 31 56 81  6 31 56 81
[3,] 11 36 61 86 11 36 61 86
[4,] 16 41 66 91 16 41 66 91
[5,] 21 46 71 96 21 46 71 96
>
```

a) This code replicates the matrix m and give us a new code of 4x10. This new matrix n is a horizontal matrix formed by combining m matrix column-wise. The number of rows of new matrix n is equals to number of columns of old matrix m.

b) This code replicates the matrix m and give us a new code of 5x8. This new matrix n is a horizontal matrix formed by combining m matrix column-wise. The number of rows of the new matrix n is equals to number of rows of the old matrix m.

c) This code replicates the matrix m and give us a new code of 4x10. This new matrix n is a horizontal matrix formed by combining m matrix column-wise. The number of rows of new matrix n is equals to number of columns of old matrix m. Here byrow=TRUE will fill the attributes row wise which means that a row will be filled first before moving to the next column.

d) This code will bind or replicate the matrix m twice vertically and create a new matrix named m1 by repeating itself row wise and giving us a matrix of

10x4 and repeating the row name after R5.
e) This code will bind or replicate the matrix m twice horizontally and create a new matrix named m2 by repeating itself column wise and giving us a matrix of 5x8 and repeating the column name after C4.
f) This code will assign the row names to each row of m1 individually which were getting repeated after R5. This code will assign the column names to each column of m2 individually which were getting repeated after C4.