



# **SRI RAMACHANDRA**

**INSTITUTE OF HIGHER EDUCATION AND RESEARCH**

(Category - I Deemed to be University) Porur, Chennai

**SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY**

## **CSE 240 Data Science with R**

### **STUDENT WORK BOOK**

**Name** : M.Sathishkumar

**Unique ID** : E0119052

**Year** : II

**Quarter** : Q6

**Department** : B.Tech CSE (AI &ML)

**Faculty Name** : Prof.B.Nirmala

**Academic Year** : 2020-2021

**Date:**02/11/2020

### **Q.NO 1: Question**

Consider 2 vectors c(9,10,11,12) and c(13,14,15,16).

Create a 4 by 2 matrix from these two vectors

### **Program:**

```
#Matrix creation  
matrix<-matrix(c(c(9,10,11,12),c(13,14,15,16))),nrow = 4,ncol = 2)  
#combining two vertices and converting into 4 by 2 matrix  
print(matrix)
```

### **Output:**

	[ ,1]	[ ,2]
[1, ]	9	13
[2, ]	10	14
[3, ]	11	15
[4, ]	12	16

### **Explanation:**

- ★ Matrix is a two-dimensional data structure in R programming.
- ★ Matrix can be created using the matrix () function
- ★ Dimension of the matrix can be defined by passing appropriate value guments nrow and ncol

**Date:** 02/11/2020

## **Q.NO 2: Question**

### **Program:**

Write an R program to take input from the user (userID and Group/Branch) and display the values

```
#UserInput
Id = readline("Enter your UserID:")
#Prompts user for input(id)
Branch = readline(prompt="Enter you Branch/Group:")
#Prompts user for input(branch)
cat("Your UserID is",Id,"and you belong to",Branch,"group")
#Displaying Values
```

### **Output:**

Enter your UserID:E0119052

Enter you Branch/Group:AI & ML

Your UserID is E0119052 and you belong to AI & ML group

### **Explanation:**

- ★ readline reads a line from the terminal (in interactive use).
- ★ readline() lets the user enter a one-line string at the terminal.
- ★ The prompt argument is printed in front of the user input. It usually ends on ": ".

Date: 02/11/2020

### Q.NO 3: Question

Create a data frame Write a R program to create a data frame from four given vectors.

a name b. Subject C. Score d. Rank

#### Program:

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")

Score <- c(56, 76, 86, 96, 73, 87, 47)

Rank <- c(5,8,6,7,9,2,1)

df = data.frame(Name,Subject,Score,Rank)
print(df)
#Dataframe Created Successfully
```

#### Output:

	Name	Subject	Score	Rank
1	Jhon	Data Science	56	5
2	Lee	Machine Learning	76	8
3	Suzan	Deep Learning	86	6
4	Abhinav	Data Structures	96	7
5	Brain	Database Managemnt System	73	9
6	Emma	Operating Systems	87	2
7	David	Python Programming	47	1

#### Explanation:

- ★ A **data frame** is used for storing data tables. It is a list of vectors of equal length.
- ★ The top line of the table, called the **header**, contains the column names.
- ★ Each horizontal line afterward denotes a **data row**, which begins with header, and then followed by the actual data.
- ★ Each data member of a row is called a **cell**.

**Date: 02/11/2020**

### **Q.NO 4: Question**

Write a R program to get the statistical summary and nature of the data of a given data frame. ( use 3rd Question dataframe

### **Program:**

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")
Score <- c(56, 76, 86, 96, 73, 87, 47)
Rank <- c(5,8,6,7,9,2,1)
df = data.frame(Name,Subject,Score,Rank)
print(df)
#Dataframe Created Successfully
print(summary(df))
#Dataframe Stastical summary
```

### **Output:**

	Name	Subject	Score	Rank
Abhinav:1	Data Science	:1	Min. :47.00	Min. :1.000
Brain :1	Data Structures	:1	1st Qu.:64.50	1st Qu.:3.500
David :1	Database Managemnt System	:1	Median :76.00	Median :6.000
Emma :1	Deep Learning	:1	Mean :74.43	Mean :5.429
Jhon :1	Machine Learning	:1	3rd Qu.:86.50	3rd Qu.:7.500
Lee :1	Operating Systems	:1	Max. :96.00	Max. :9.000
Suzan :1	Python Programming	:1		

### **Explanation:**

Summary is a generic function used to produce result summaries of the results of various model fitting functions.

The function invokes particular methods which depend on the class of the first argument

The form of the value returned by summary depends on the class of its argument

**Date: 02/11/2020**

### **Q.NO 5: Question**

Write a R program to extract specific column from a data frame using column name

#### **Program:**

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")
Score <- c(56, 76, 86, 96, 73, 87, 47)
Rank <- c(5,8,6,7,9,2,1)
df = data.frame(Name,Subject,Score,Rank)
print(df)
#Dataframe Created Successfully
print(df["Subject"])
#Sliced column with column name
```

#### **Output:**

	Subject
1	Data Science
2	Machine Learning
3	Deep Learning
4	Data Structures
5	Database Managemnt System
6	Operating Systems
7	Python Programming

#### **Explanation:**

- ★ We retrieve a data frame column slice with the single square bracket "[" operator.
- ★ Here We can retrieve the same column slice by its name.
- ★ And we can pack the row names in an index vector in order to retrieve multiple rows

**Date: 02/11/2020**

### **Q.NO 6: Question**

.Write a R program to extract first two rows from a given data frame

#### **Program:**

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning",
             "Data Structures", "Database Managemnt System",
             "Operating Systems", "Python Programming")
Score <- c(56, 76, 86, 96, 73, 87, 47)
Rank <- c(5,8,6,7,9,2,1)
df = data.frame(Name,Subject,Score,Rank)
#Dataframe Created Successfully
print(df[1:2,])
#splicing Dataframe with respect to index
```

#### **Output:**

	Name	Subject	Score	Rank
1	Jhon	Data Science	56	5
2	Lee	Machine Learning	76	8

#### **Explanation:**

- ★ We retrieve rows from a data frame with the single square bracket operator, just like what we did with columns.
- ★ However, in addition to an index vector of row positions, we append an extra comma character.
- ★ This is important, as the extra comma signals a wildcard match for the second coordinate for column positions.



Date: 03/11/2020

### Q.NO 7: Question

Create an R script that calculates the square root of a given integer vector x of length one, if the value contained in x is negative it should return NA.

#### Program:

```
#Prompting user for input
x = c(as.integer(readline("Enter a number:")))
#Defining a function for finding square root
squareroot <- function(x){
  if (x < 0){
    print(NA)
  }else{
    print(paste("Square root of the given number :",sqrt(x)))
  }
}
squareroot(x)
#Invoking the function
```

#### Output:

```
Enter a number:57
[1] "Square root of the given number : 7.54983443527075"
```

```
Enter a number:-78
[1] NA
```

#### Explanation:

- ★ Missing values are represented by the symbol **NA** (not available)
- ★ If the numeric\_Expression is a positive value, the sqrt function returns the square root of a given value.
- ★ If the numeric\_Expression is a negative value, the sqrt function return **NaN**.
- ★ Numeric\_Expression is not a number (**NaN**), or Negative Infinity, then sqrt in R returns **NaN**.



**Date:** 03/11/2020

**Q.NO 8: Question**

Demonstrate and examine the output of letter and LETTER

**Program:**

```
a = "letter"  
b = "LETTER"  
print(a)           #printing letter  
print(b)           #printing LETTER
```

**Output:**

```
[1] "letter"  
[1] "LETTER"
```

**Explanation:**

- ★ This is useful because you can always view your data: just print it.
- ★ You needn't write special printing logic, even for complicated data structures.
- ★ The print function has a significant limitation, it prints only one object at a time.
- ★ Trying to print multiple items gives this mind-numbing error message

**Date:** 03/11/2020

**Q.NO 9: Question**

Create an R script that, given a numeric vector x with length 3, will print the elements by order from high to low.

**Program:**

```
desc <- function(x){  
  x<-c(5,7,8)  
  }  
print(sort(desc(x),decreasing = TRUE))
```

**Output:**

```
[1] 8 7 5
```

**Explanation:**

- ★ Sort (or *order*) a vector or factor (partially) into ascending or descending order. For ordering along more than one variable, e.g., for sorting data frame
- ★ To sort a data frame in R, use the **sort( )** function. By default, sorting is ASCENDING.
- ★ Prepend the sorting variable by a minus sign to indicate DESCENDING order

Date: 03/11/2020

### Q.NO 10: Question

Create an R script that returns the amount of values that are larger than the mean of a vector.  
You are allowed to use mean(). ( Use function)

### Program:

```
#function
nums<- function(vec)
{
  #returning values greater then mean value
  return(vec[vec>mean(vec)])
}
#Assigning values
nums(c(78,12,459,175,451,178,587,24,985))
```

### Output:

459 451 587 985

### Explanation:

- ★ It is calculated by taking the sum of the values and dividing with the number of values in a data series.
- ★ The function **mean()** is used to calculate and decision operator ">" is used to compare the giving condition and take decision

**Date: 03/11/2020**

### **Q.NO 11: Question**

Write a double for loop which prints 30 numbers (1:10, 2:11, 3:12). Those are three clusters of ten numbers each. The first loop determines the number of clusters (3) via its length; the second loop the numbers to be printed (1 to 10 at the beginning). Each cluster starts one number higher than the previous one

### **Program:**

```
#Nested looping
for(i in seq(1,3)){
  #for every single run in outside loop inside loop gets executed 10 times
  for(j in seq(i,i+10L)){
    print(j)
  }
}
```

### **Output:**

1 2 3 4 5 6 7 8 9 10 11 2 3 4 5 6 7 8 9 10 11 12 3 4 5 6 7 8 9 10 11 12 13

### **Explanation:**

- ★ In Nested for Loop, it makes use of the control structures to manage the execution of the expression, one such control structure is Nested for Loop a similar to basic 'for' loop executes.
- ★ It can be defined as placing one 'for' loop inside the first 'for' loop is called as nesting or loop of loops in some terms, which takes the responsibility of two loops such that the outer loop controls the number of repetition of the whole inner detailed information until it is false,
- ★ in other words, the inner loop executes n-times of every execution of the outer for loop and also, it's a great tool to work with R Programming Language.

**Date: 03/11/2020**

### **Q.NO 12: Question**

a. You have the data.frame 'mydf' with four columns like below

a = c(3,7,NA, 9) b = c(2,NA,9,3) f = c(5,2,5,6) d = c(NA,3,4,NA)

You want to add another column '5': the 5th column contains the value of col 2 if col 1 is NA; the 5th column contains the value of col 4 if col 2 is NA; the 5th column contains the value of col 3 in all other cases.

### **Program:**

```
#Creating a dataframe
mydf = data.frame(a=c(3,7,NA, 9),b=c(2,NA,9,3),f=c(5,2,5,6),d=c(NA,3,4,NA))
#Applying conditions as required to create 5th column
mydf[,5] <- ifelse(is.na(mydf[,1]) & !is.na(mydf[,2]),mydf[,2],
                  ifelse(is.na(mydf[,2]) & !is.na(mydf[,4]),mydf[,4], mydf[,3]))

print(mydf)
```

### **Output:**

	a	b	f	d	V5
1	3	2	5	NA	5
2	7	NA	2	3	3
3	NA	9	5	4	9
4	9	3	6	NA	6

### **Explanation:**

- ★ An if statement can be followed by an optional else statement which executes when the Boolean expression is false.
- ★ An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement

Date: 03/11/2020

### Q.NO 13: Question

Write a while loop starting with x = 0. The loop prints all numbers up to 35 but it skips number 7.  
Condition: If x== 7 next

### Program:

```
#While looping
x<-0
while (x<=35) {
  if(x==7){
    x<-x+1
    #Applying next function to bypass if x equals 7
    next
  }
  cat(x,"")
  x=x+1
}
```

### Output:

0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

### Explanation:

- ★ The **next** statement is useful when we want to skip the current iteration of a loop without terminating it.
- ★ On encountering next, the R parser skips further evaluation and starts next iteration of the loop. In other words, the inner loop executes n-times of every execution of the outer for loop.

**Date:** 03/11/2020

**Q.NO 14: Question**

Examine the difference between typeof and class () method using R program

**Program:**

```
#Variable  
num<-458  
class(num)    #It gives numeric  
typeof(num)   #It gives double
```

**Output:**

'numeric'

'double'

**Explanation:**

- ★ 'class' is a property assigned to an object that determines how generic functions operate with it. It is not a mutually exclusive classification. If an object has no specific class assigned to it, such as a simple numeric vector, its class is usually the same as its mode, by convention.
- ★ Typeof determines the (R internal) type or storage mode of any object



Date: 03/11/2020

### Q.NO 15: Question

Create a function and demonstrate their features like required, keyword, default.

#### Program:

```
#Keyword Argument
keyword<-function(a,b){
  return(a+b)
}
keyword(b=1,a=3)

#default Argument
def <- function(a=0) {
  return(a+1)
}
def(5)

#required Argument
norma<-function(x){
  return(x+1)
}
norma(1)
```

#### Output:

4  
6  
2

#### Explanation:

- ★ *Arguments are always named when you define a function.* When you call a function, you do not have to specify the name of the argument.
- ★ Arguments are optional; you do not have to specify a value for them. They can have a default value, which is used if you do not specify a value for that argument yourself.
- ★ You can use as many arguments as you like, there is no limit to the number of arguments. An argument list comprises of comma-separated values that contain the various formal arguments.

Date: 03/11/2020

## Q.NO 16: Question

Create a dataframe and delete the row and column. ( Use the own data values to create frame)

### Program:

```
#Dataframe creation
Name <- c("Jhon", "Lee", "Suzan", "Abhinav", "Brain", "Emma", "David")
#Variables
Subject <- c("Data Science", "Machine Learning", "Deep Learning", "Data Structures",
             "Database Managemnt System", "Operating Systems", "Python Programming")

Score <- c(56, 76, 86, 96, 73, 87, 47)

Rank <- c(5,8,6,7,9,2,1)

df = data.frame(Name,Subject,Score,Rank)
del_row_df<-df[-c(2),] # row deletion
del_col_df<-within(df, rm("Subject")) #Colmn deletion
print("After deleting Row 2")
del_row_df
print("After Deleting Subject Column")
del_col_df
```

### Output:

[1] "After deleting Row 2"

	Name	Subject	Score	Rank
1	Jhon	Data Science	56	5
3	Suzan	Deep Learning	86	6
4	Abhinav	Data Structures	96	7
5	Brain	Database Managemnt System	73	9
6	Emma	Operating Systems	87	2
7	David	Python Programming	47	1

[1] "After Deleting Subject Column"

Name	Score	Rank
Jhon	56	5
Lee	76	8
Suzan	86	6
Abhinav	96	7
Brain	73	9
Emma	87	2
David	47	1

### Explanation:

- ★ -c(rowNum) is used to return a slice of a dataframe without the mentioned row i.e. row deletion
- ★ The within function returns a subset of a dataframe with the second argument and a function to apply to all the column rm function removes the "Subject" column out of the data frame Typeof determines the (R internal) type or storage mode of any object.