**Week-1: Built in functions.**

**1.1) Calculate the cumulative sum(“running total”)of the numbers 2,3,4,5,6**

cumsum() function in R Language is used to calculate the cumulative sum of the vector passed as argument.

**Syntax:** cumsum(x)

**Parameters:** x: Numeric Object

**Code:**

>cumsum(2:6)

**Output:**

2 5 9 14 20

**Code:**

cumsum(1:4)

**Output:**

1 3 6 10

**1.2) Print the 1 to 10 numbers in reverse order.**

rev is used to reverse the input given

**Code:**

>rev(1:10)

**Output:**

10 9 8 7 6 5 4 3 2 1

**Week-2:** **Basic Programs.**

**2.1) Write a R program to take input from the user (name and age) and display the**

**values. Also print the version of R installation.**

In R, it’s also possible to take input from the user. For doing so, there are two methods in R.

 Using readline() method

 Using scan() method

In R language readline() method takes input in string format. If one inputs an integer then it is inputted as a string, lets say, one wants to input 255, then it will input as “255”, like a string. So one needs to convert that inputted value to the format that he needs. In this case, string “255” is converted to integer 255

**Code:**

name = readline(prompt="Input your name: ")

age = readline(prompt="Input your age: ")

print(paste("My name is",”aditya”, "and I am",20,"years old."))

print(R.version.string)

**Output:**

Input your name:

Input your age:

[1] "My name is aditya and I am 20 years old."

[1] "R version 4.0.3 (2020-10-10)"

**2.2) Write a R program to get the details of the objects in memory.**

To create an object in R, one needs to give it a name followed by the assignment operator <- (An equal sign, =, can also be used), and the value he wants to give it:

x <- 5

**Code:**

name = "R programming";

n1 = 1;

n2 = 1.5

nums = c(1, 2, 30, 4, 5, 60)

print(ls())

print("Details of the objects in memory:")

print(ls.str())

**Output:**

[1] "n1" "n2" "name" "nums"

[1] "Details of the objects in memory:"

n1 : num 1

n2 : num 1.5

name : chr "R programming"

nums : num [1:6] 1 2 30 4 5 60

**2.3) Write a R program to create a sequence of numbers from 20 to 50 and find the**

**mean of numbers from 20 to 60 and sum of numbers from 51 to 91.**

seq() is used to print numbers in sequence order

The simplest way to create a sequence of numbers in R is by using the : operator by seq.

**Code:**

print("Sequence of numbers from 20 to 50:")

print(seq(20,50))

print("Mean of numbers from 20 to 60:")

print(mean(20:60))

print("Sum of numbers from 51 to 91:")

print(sum(51:91))

**Output:**

[1] "Sequence of numbers from 20 to 50:"

[1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44

[26] 45 46 47 48 49 50

[1] "Mean of numbers from 20 to 60:"[1] 40

[1] "Sum of numbers from 51 to 91:"

[1] 2911

**Week-3: Graphics Write a R program to create a simple bar plot of five subject marks.**

R language is mostly used for statistics and data analytics purposes to represent the data graphically in the software. To represent those data graphically, charts and graphs are used in R.

**R – graphs**

There are hundreds of charts and graphs present in R. For example, bar plot, box plot, mosaic plot, dot chart, coplot, histogram, pie chart, scatter graph, etc.

**Bar Plot or Bar Chart**

Bar plot or Bar Chart in R is used to represent the values in data vector as height of the bars. The data vector passed to the function is represented over y-axis of the graph. Bar chart can behave like histogram by using table() function instead of data vector.

***Syntax:*** *barplot(data, xlab, ylab)*

*where:*

 data is the data vector to be represented on y-axis

 xlab is the label given to x-axis

 *ylab is the label given to y-axis*

**Code:**

marks = c(70, 95, 80, 74)

barplot(marks,

main = "Comparing marks of 5 subjects",

xlab = "Marks",

ylab = "Subject",

names.arg = c("English", "Science", "Math.", "Hist."),

col = "darkred",

horiz = FALSE)

**Output:**



**Week-4: Vectors.**

**4.1 ) Write a R program to get the unique elements of a given string and unique**

**numbers of vector.**

Vectors are the most basic R data objects and there are six types of atomic vectors. They are logical, integer, double, complex, character and raw**.**

**Vector Creation**

**Single Element Vector**

Even when you write just one value in R, it becomes a vector of length 1 and belongs to one of the above vector types.

# Atomic vector of type character.

print("abc");

# Atomic vector of type double.

print(12.5)

**Code:**

str1 = "The quick brown fox jumps over the lazy dog."

print("Original vector(string)")

print(str1)

print("Unique elements of the said vector:")

print(unique(tolower(str1)))

nums = c(1, 2, 2, 3, 4, 4, 5, 6)

print("Original vector(number)")

print(nums)

print("Unique elements of the said vector:")

print(unique(nums))

**Output:**

[1] "Original vector(string)"

[1] "The quick brown fox jumps over the lazy dog."

[1] "Unique elements of the said vector:"

[1] "the quick brown fox jumps over the lazy dog."

[1] "Original vector(number)"

[1] 1 2 2 3 4 4 5 6

[1] "Unique elements of the said vector:"

[1] 1 2 3 4 5 6

**4.2) Write a R program to create three vectors a,b,c with 3 integers. Combine the three**

**vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix.**

The cbind function – short for column bind – is a merge function that can be used to combine two data frames with the same number of multiple rows into a single data frame.

**Code:**

a<-c(1,2,3)

b<-c(4,5,6)

c<-c(7,8,9)

m<-cbind(a,b,c)

print("Content of the said matrix:")

print(m)

**Output:**

[1] "Content of the said matrix:"

a b c

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

**4.3) Write a R program to create a matrix from a list of given vectors.**

**Code:**

l = list()

for (i in 1:5)

l[[i]] <- c(i, 1:4)

print("List of vectors:")

print(l)

result = do.call(rbind, l)

print("New Matrix:")

print(result)

**Output:**

[1] "List of vectors:"

[[1]]

[1] 1 1 2 3 4

[[2]]

[1] 2 1 2 3 4

[[3]]

[1] 3 1 2 3 4

[[4]]

[1] 4 1 2 3 4

[[5]]

[1] 5 1 2 3 4

[1] "New Matrix:"

[,1] [,2] [,3] [,4] [,5]

[1,] 1 1 2 3 4

[2,] 2 1 2 3 4

[3,] 3 1 2 3 4

[4,] 4 1 2 3 4

[5,] 5 1 2 3 4

**Week-5: Vectors-continued.**

**5.1) Write a R program to append value to a given empty vector.**

Using append()

We can add data by using the append() function.

**Syntax:** *append(vector\_name,value)*

**Code:**

# create an empty numeric

# vector a

a=c()

# display it

print(a)

# append 10 using append()

# function

a=append(a,10)

# display

print(a)

**Output:**

NULL

[1] 10

**5.2) Write a R program to multiply two vectors of integers type and length 3.**

Vector() function is also used to create empty vector.

**Syntax:** *vector(class of the data object, length of the vector)*

There is a very straightforward approach for this.

Steps –

 Create vector of required type

 Also pass the size to it

 Here we can also check the type and size of the vector so created

**Code:**

x = c(10, 20, 30)

y = c(20, 10, 40)

print("Original Vectors:")

print(x)

print(y)

print("Product of two Vectors:")

z = x \* y

print(z)

**Output:**

[1] "Original Vectors:"

[1] 10 30 6

[1] 20 10 40

[1] "Product of two Vectors:"

[1] 200 300 240

**5.3) Write a R program to find Sum, Mean and Product of a Vector, ignore element like A or NaN.**

sum(), mean(), and prod() methods are available in R which are used to compute the specified operation over the arguments specified in the method. In case, a single vector is specified, then the operation is performed over individual elements, which is equivalent to the application of for loop.

**Function Used:**

 mean() function is used to calculate mean

***Syntax: mean(x, na.rm)***

***Parameters:***

 x: Numeric Vector

 na.rm: Boolean value to ignore NA value

 sum() is used to calculate sum

***Syntax: sum(x)***

***Parameters:***

 x: Numeric Vector

 prod() is used to calculate product

***Syntax: prod(x)***

***Parameters:***

 x: Numeric Vector

**Code:**

x = c(10, NULL, 20, 30, NA)

print("Sum:")

#ignore NA and NaN values

print(sum(x, na.rm=TRUE))

print("Mean:")

print(mean(x, na.rm=TRUE))

print("Product:")

print(prod(x, na.rm=TRUE))

**Output:**

[1] "Sum:"

[1] 60

[1] "Mean:"

[1] 20

[1] "Product:"

[1] 6000

**Week-6: Matrices.**

**6.1) Write a R program to create a 5 x 4 matrix , 3 x 3 matrix with labels and fill the**

**matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.**

Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types. Though we can create a matrix containing only characters or only logical values, they are not of much use. We use matrices containing numeric elements to be used in mathematical calculations.

A Matrix is created using the **matrix()** function.

**Syntax:** matrix(data, nrow, ncol, byrow, dimnames)

**Code:**

m1 = matrix(1:20, nrow=5, ncol=4)

print("5 × 4 matrix:")

print(m1)

cells = c(1,3,5,7,8,9,11,12,14)

rnames = c("Row1", "Row2", "Row3")

cnames = c("Col1", "Col2", "Col3")

m2 = matrix(cells, nrow=3, ncol=3, byrow=TRUE, dimnames=list(rnames, cnames))

print("3 × 3 matrix with labels, filled by rows: ")

print(m2)

cells = c(1,3,5,7)

rnames = c("Row1", "Row2")

cnames = c("Col1", "Col2")

print("2 × 2 matrix with labels, filled by columns: ")

m3 = matrix(cells, nrow=2, ncol=2, byrow=FALSE, dimnames=list(rnames, cnames))

print(m3)

**Output:**

[1] "5 × 4 matrix:"

[,1] [,2] [,3] [,4]

[1,] 1 6 11 16

[2,] 2 7 12 17

[3,] 3 8 13 18

[4,] 4 9 14 19

[5,] 5 10 15 20

[1] "3 × 3 matrix with labels, filled by rows: "

Col1 Col2 Col3

Row1 1 3 5

Row2 7 8 9

Row3 11 12 14

[1] "2 × 2 matrix with labels, filled by columns: "

Col1 Col2

Row1 1 5

Row2 3 7

**6.2) Write a R program to create a two-dimensional 5x3 array of sequence of even**

**integers greater than 50.**

**Code:**

a <- array(seq(from = 50, length.out = 15, by = 2), c(5, 3))

print("Content of the array:")

print("5×3 array of sequence of even integers greater than 50:")

print(a)

**Output:**

[1] "Content of the array:"

[1] "5×3 array of sequence of even integers greater than 50:"

[,1] [,2] [,3]

[1,] 50 60 70

[2,] 52 62 72

[3,] 54 64 74

[4,] 56 66 76

[5,] 58 68 78

**6.3) Write a R program to find row and column index of maximum and minimum value**

**in a given matrix.**

**Code:**

m = matrix(c(1:16), nrow = 4, byrow = TRUE)

print("Original Matrix:")

print(m)

result = which(m == max(m), arr.ind=TRUE)

print("Row and column of maximum value of the said matrix:")

print(result)

result = which(m == min(m), arr.ind=TRUE)

print("Row and column of minimum value of the said matrix:")

print(result)

**Output:**

[1] "Original Matrix:"

[,1] [,2] [,3] [,4]

[1,] 1 2 3 4

[2,] 5 6 7 8

[3,] 9 10 11 12

[4,] 13 14 15 16

[1] "Row and column of maximum value of the said matrix:"

row col

[1,] 4 4

[1] "Row and column of minimum value of the said matrix:"

row col

[1,] 1 1

**Week-7: Arrays.**

**7.1) Write a R program to combine three arrays so that the first row of the first array is followed by the first row of the second array and then first row of the third array.**

Arrays are the R data objects which can store data in more than two dimensions. For example − If we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only data type. An array is created using the array() function. It takes vectors as input and uses the values in the dim parameter to create an array**.**

**Code:**

num1 = rbind(rep("A",3), rep("B",3), rep("C",3))

print("num1")

print(num1)

num2 = rbind(rep("P",3), rep("Q",3), rep("R",3))

print("num2")

print(num2)

num3 = rbind(rep("X",3), rep("Y",3), rep("Z",3))

print("num3")

print(num3)

a = matrix(t(cbind(num1,num2,num3)),ncol=3, byrow=T)

print("Combine three arrays, taking one row from each one by one:")

print(a)

**Output:**

[1] "num1"

[,1] [,2] [,3]

[1,] "A" "A" "A"

[2,] "B" "B" "B"

[3,] "C" "C" "C"

[1] "num2"

[,1] [,2] [,3]

[1,] "P" "P" "P"

[2,] "Q" "Q" "Q"

[3,] "R" "R" "R"

[1] "num3"

[,1] [,2] [,3]

[1,] "X" "X" "X"

[2,] "Y" "Y" "Y"

[3,] "Z" "Z" "Z"

[1] "Combine three arrays, taking one row from each one by one:"

[,1] [,2] [,3]

[1,] "A" "A" "A"

[2,] "P" "P" "P"

[3,] "X" "X" "X"

[4,] "B" "B" "B"

[5,] "Q" "Q" "Q"

[6,] "Y" "Y" "Y"

[7,] "C" "C" "C"

[8,] "R" "R" "R"

[9,] "Z" "Z" "Z"

**7.2)Write a R program to create an array using four given columns, three given rows, and two given tables and display the content of the array.**

**Code:**

array1 = array(1:30, dim=c(3,5,2))

print(array1)

**Output:**

, , 1

[,1] [,2] [,3] [,4] [,5]

[1,] 1 4 7 10 13

[2,] 2 5 8 11 14

[3,] 3 6 9 12 15

, , 2

[,1] [,2] [,3] [,4] [,5]

[1,] 16 19 22 25 28

[2,] 17 20 23 26 29

[3,] 18 21 24 27 30

**Week-8: Data frame-I**

**8.1) Write a R program to create an empty data frame.**

**Code:**

df = data.frame(Ints=integer(),

Doubles=double(),

Characters=character(),

Logicals=logical(),

Factors=factor(),

stringsAsFactors=FALSE)

print("Structure of the empty dataframe:")

print(str(df))

**Output:**

[1] "Structure of the empty dataframe:"

'data.frame': 0 obs. of 5 variables:

$ Ints : int

$ Doubles : num

$ Characters: chr

$ Logicals : logi

$ Factors : Factor w/ 0 levels:

NULL

**8.2) Write a R program to create a data frame from four given vectors.**

**Code:**

name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas')

score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19)

attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1)

qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')

print("Original data frame:")

print(name)

print(score)

print(attempts)

print(qualify)

df = data.frame(name, score, attempts, qualify)

print(df)

**Output:**

[1] "Original data frame:"

[1] "Anastasia" "Dima" "Katherine" "James" "Emily" "Michael"

[7] "Matthew" "Laura" "Kevin" "Jonas"

[1] 12.5 9.0 16.5 12.0 9.0 20.0 14.5 13.5 8.0 19.0

[1] 1 3 2 3 2 3 1 1 2 1

[1] "yes" "no" "yes" "no" "no" "yes" "yes" "no" "no" "yes"

name score attempts qualify

1 Anastasia 12.5 1 yes

2 Dima 9.0 3 no

3 Katherine 16.5 2 yes

4 James 12.0 3 no

5 Emily 9.0 2 no

6 Michael 20.0 3 yes

7 Matthew 14.5 1 yes

8 Laura 13.5 1 no

9 Kevin 8.0 2 no

10 Jonas 19.0 1 yes

**Week-9: Data frame-II**

**9.1) Write a R program to create a data frame using two given vectors and display the duplicated elements and unique rows of the said data frame.**

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

Following are the characteristics of a data frame.

 The column names should be non-empty.

 The row names should be unique.

 The data stored in a data frame can be of numeric, factor or character type.

 Each column should contain same number of data items

**Code:**

a = c(10,20,10,10,40,50,20,30)

b = c(10,30,10,20,0,50,30,30)

print("Original data frame:")

ab =data.frame(a,b)

print(ab)

print("Duplicate elements of the said data frame:")

print(duplicated(ab))

print("Unique rows of the said data frame:")

print(unique(ab))

**Output:**

[1] "Original data frame:"

a b

1 10 10

2 20 30

3 10 10

4 10 20

5 40 0

6 50 50

7 20 30

8 30 30

[1] "Duplicate elements of the said data frame:"

[1] FALSE FALSE TRUE FALSE FALSEFALSE TRUE FALSE

[1] "Unique rows of the said data frame:"

a b

1 10 10

2 20 30

4 10 20

5 40 0

6 50 50

8 30 30

**9.2) Write a R program to save the information of a data frame in a file and display the information of the file.**

**Code:**

exam\_data=data.frame(

name = c('Anastasia','Dima','Katherine','James','Emily','Michael','Matthew','Laura','Kevin','Jonas'),

score = c(12.5,9,16.5,12,9,20,14.5,13.5,8,19),

attempts = c(1,3,2,3,2,3,1,1,2,1),

qualify = c('yes','no','yes','no','no','yes','yes','no','no','yes'))

print("Original dataframe:")

print(exam\_data)

save(exam\_data,file="data.rda")

load("data.rda")

file.info("data.rda")

**Output:**

[1] "Original dataframe:"

name score attempts qualify

1 Anastasia 12.5 1 yes

2 Dima 9.0 3 no

3 Katherine 16.5 2 yes

4 James 12.0 3 no

5 Emily 9.0 2 no

6 Michael 20.0 3 yes

7 Matthew 14.5 1 yes

8 Laura 13.5 1 no

9 Kevin 8.0 2 no

10 Jonas 19.0 1 yes

size isdir mode mtimectime

data.rda 303 FALSE 666 2022-03-25 11:24:32 2022-03-25 11:24:32

atime exe

data.rda 2022-03-25 11:24:32 no

**Week-10: Lists**

**10.1) Write a R program to create a list containing a vector, a matrix and a list and give names to the elements in the list. Access the first and second element of the list.**

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements. List is created using **list()** function.

**Creating a List**

Following is an example to create a list containing strings, numbers, vectors and a logical values.

# Create a list containing strings, numbers, vectors and a logical

# values.

list\_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)

print(list\_data)

**Code:**

list\_data<-list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11),nrow=2),

list("Python","PHP","Java"))

print("List:")

print(list\_data)

names(list\_data)= c("Color","Odd numbers","Language(s)")

print("List with column names:")

print(list\_data)

print('1st element:')

print(list\_data[1])

print('2nd element:')

print(list\_data[2])

**Output:**

[1] "List:"

[[1]]

[1] "Red" "Green" "Black"

[[2]]

[,1] [,2] [,3]

[1,] 1 5 9

[2,] 3 7 11

[[3]]

[[3]][[1]]

[1] "Python"

[[3]][[2]]

[1] "PHP"

[[3]][[3]]

[1] "Java"

[1] "List with column names:"

$Color

[1] "Red" "Green" "Black"

$`Odd numbers`

[,1] [,2] [,3]

[1,] 1 5 9

[2,] 3 7 11

$`Language(s)`

$`Language(s)`[[1]]

[1] "Python"

$`Language(s)`[[2]]

[1] "PHP"

$`Language(s)`[[3]]

[1] "Java"

[1] "1st element:"

$Color

[1] "Red" "Green" "Black"

[1] "2nd element:"

$`Odd numbers`

[,1] [,2] [,3]

[1,] 1 5 9

[2,] 3 7 11

**10.2) Write a R program to create a list containing a vector, a matrix and a list and remove the second element.**

**Code:**

list\_data<-list(c("Red","Green","Black"), matrix(c(1,3,5,7,9,11),nrow=2),

list("Python","PHP","Java"))

print("List:")

print(list\_data)

print("Remove the second element of the list:")

list\_data[2]= NULL

print("New list:")

print(list\_data)

**Output:**

[1] "List:"

[[1]]

[1] "Red" "Green" "Black"

[[2]]

[,1] [,2] [,3]

[1,] 1 5 9

[2,] 3 7 11

[[3]]

[[3]][[1]]

[1] "Python"

[[3]][[2]]

[1] "PHP"

[[3]][[3]]

[1] "Java"

[1] "Remove the second element of the list:"

[1] "New list:"

[[1]]

[1] "Red" "Green" "Black"

[[2]]

[[2]][[1]]

[1] "Python"

[[2]][[2]]

[1] "PHP"

[[2]][[3]]

[1] "Java"

**10.3) Write a R program to select second element of a given nested list.**

**Code:**

x =list(list(0,2),list(3,4),list(5,6))

print("Original nested list:")

print(x)

e =lapply(x,'[[',2)

print("Second element of the nested list:")

print(e)

**Output:**

[1] "Original nested list:"

[[1]]

[[1]][[1]]

[1] 0

[[1]][[2]]

[1] 2

[[2]]

[[2]][[1]]

[1] 3

[[2]][[2]]

[1] 4

[[3]]

[[3]][[1]]

[1] 5

[[3]][[2]]

[1] 6

[1] "Second element of the nested list:"

[[1]]

[1] 2

[[2]]

[1] 4

[[3]]

[1] 6

**Week-11: Lists-continued**

**11.1) Write a R program to merge two given lists into one list.**

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements. List is created using list() function.

Creating a List

Following is an example to create a list containing strings, numbers, vectors and a logical values.

# Create a list containing strings, numbers, vectors and a logical

# values.

list\_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)

print(list\_data)

**Code:**

n1 =list(1,2,3)

c1 =list("Red","Green","Black")

print("Original lists:")

print(n1)

print(c1)

print("Merge the said lists:")

mlist= c(n1, c1)

print("New merged list:")

print(mlist)

**Output:**

[1] "Original lists:"

[[1]]

[1] 1

[[2]]

[1] 2

[[3]]

[1] 3

[[1]]

[1] "Red"

[[2]]

[1] "Green"

[[3]]

[1] "Black"

[1] "Merge the said lists:"

[1] "New merged list:"

[[1]]

[1] 1

[[2]]

[1] 2

[[3]]

[1] 3

[[4]]

[1] "Red"

[[5]]

[1] "Green"

[[6]]

[1] "Black"

**11.2) Write a R program to create a list named s containing sequence of 15 capital letters, starting from ‘E’.**

**Code:**

s = LETTERS[match("E", LETTERS):(match("E", LETTERS)+15)]

print("Sequence of 15 capital letters, starting from ‘E’-")

print(s)

**Output:**

[1] "Sequence of 15 capital letters, starting from ‘E’-"

[1] "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T"

**11.3) Write a R program to assign new names "a", "b" and "c" to the elements of a given list.**

**Code:**

list1 = list(g1 = 1:10, g2 = "R Programming", g3 = "HTML")

print("Original list:")

print(list1)

names(list1) = c("a", "b", "c")

print("Assign new names 'a', 'b' and 'c' to the elements of the said list")

print(list1)

**Output:**

[1] "Original list:"$g1

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

$g2

[1] "R Programming"

$g3

[1] "HTML"

[1] "Assign new names 'a', 'b' and 'c' to the elements of the said list"

$a

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

$b

[1] "R Programming"

$c

[1] "HTML"

**Week-12: Factors**

**12.1) Write a R program to find the levels of factor of a given vector.**

Arrays are the R data objects which can store data in more than two dimensions. For example − If we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only data type.

An array is created using the array() function. It takes vectors as input and uses the values in the dim parameter to create an array.

**Code:**

v = c(1,2,3,3,4, NA,3,2,4,5, NA,5)

print("Original vector:")

print(v)

print("Levels of factor of the said vector:")

print(levels(factor(v)))

**Output:**

[1] "Original vector:"

[1] 1 2 3 3 4 NA 3 2 4 5 NA 5

[1] "Levels of factor of the said vector:"

[1] "1" "2" "3" "4" "5"

**12.2) Write a R program to create an ordered factor from data consisting of the names of months.**

**Code:**

mons\_v= c("March","April","January","November","January",

"September","October","September","November","August","February",

"January","November","November","February","May","August","February",

"July","December","August","August","September","November","September",

"February","April")

print("Original vector:")

print(mons\_v)

f = factor(mons\_v)

print("Ordered factors of the said vector:")

print(f)

print(table(f))

**Output:**

[1] "Original vector:"

[1] "March" "April" "January" "November" "January"

[6] "September" "October" "September" "November" "August"

[11] "February" "January" "November" "November" "February"

[16] "May" "August" "February" "July" "December"

[21] "August" "August" "September" "November" "September"

[26] "February" "April"

[1] "Ordered factors of the said vector:"

[1] March April January November January September

[7] October September November August February January

[13] November November February May August February

[19] July December August August September November

[25] September February April

11 Levels: April August December February January July March ... September

April August December February January July March

2 4 1 4 3 1 1

May November October September

1 5 1 4

**12.3) Write a R program to concatenate two given factors in a single factor.**

**Code:**

f1 <- factor(sample(LETTERS, size=6, replace=TRUE))

f2 <- factor(sample(LETTERS, size=6, replace=TRUE))

print("Original factors:")

print(f1)

print(f2)

f = factor(c(levels(f1)[f1], levels(f2)[f2]))

print("After concatenate factor becomes:")

print(f)

**Output:**

[1] "Original factors:"

[1] F E P C B I

Levels: B C E F I P

[1] Z J I J G E

Levels: E G I J Z

[1] "After concatenate factor becomes:"

[1] F E P C B I Z J I J G E

Levels: B C E F G I J P Z

1

**AUGMENTED EXPERIMENTS:**

**Experiment-15:** **Consider A=matrix(c(2,0,1,3),ncol=2)and B=matrix(c(5,2,4,-1),ncol=2) a) find A+B**

**b) find A-B.**

**Program:**

A=matrix(c(2,0,1,3),ncol=2)

B=matrix(c(5,2,4,-1),ncol=2)

A+B

A-B

**Output:**

[,1] [,2]

[1,] 7 5

[2,] 2 2

[,1] [,2]

[1,] -3 -3

[2,] -2 4

**Experiment-16:** **Consider a vector 1:K,where K is a positive integer.Write an R command that determines how many elements in the vector are exactly divisible by 3.**

**Program:**

v<-c(1:100)

j<-ifelse(v%%3==0,TRUE,FALSE)

print(v[j])

**Output:**

[1] 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75

[26] 78 81 84 87 90 93 96 99