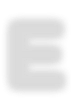
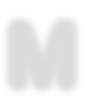
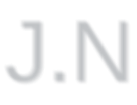
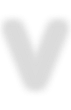
:

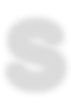
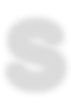


NAME

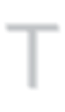
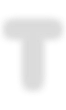


M.J.N.V. SAI

:

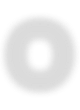


CLASS

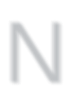
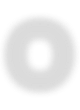


IT –

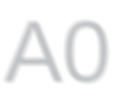
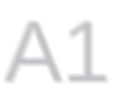
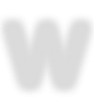
:



ROLL

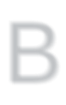


NO



208W1A12A0

**TASK – 1 :**



B

## R LAB – 01

Performing basic mathematical operations and Different Datatypes in R studio..

**PROGRAM :**

1+1

1+2+3+4+5

2\*4\*5

12\*5

4/2

2/4

4\*6+5 (4\*6)+5 4\*(6+5)

a=2 a

b<-10

b

30->c

c

e<-f<-20

e

f assign("z",50) z

rm(z) theVariable <- 17 theVariable THEVARIABLE

class(a) is.numeric(b) i <- 9L

i is.numeric(i) is.integer(i)

class(4L) class(2.8) s = 4L\*2.8

class(s) class(5L) s1 = 5L/2L

class(s1)

x1 <- "data set" x1

nchar(x1) nchar("welcome") nchar(5)

nchar(1234)

date1 <- as.Date("2022-03-31") date1

class(date1) as.numeric(date1)

date2 <- as.POSIXct("2022-03-31 17:42")

class(date2) date2 as.numeric(date2)

class(as.numeric(date1))

TRUE\*5 FALSE\*10 k <- TRUE

k class(k)

is.logical(k)

T

class(T) T = 10 T

class(T)

99 == 91

5!=8

9 < 10

9 <= 9

# what is a variable "data" == "stats" "data" < "stats" "data" == "hell" plot(a,b)

**Output :**

> 1+1

[1] 2

> 1+2+3+4+5

[1] 15

> 2\*4\*5 [1] 40

> 12\*5 [1] 60

> 4/2

[1] 2

> 2/4 [1] 0.5

> 4\*6+5 [1] 29

> (4\*6)+5 [1] 29

> 4\*(6+5) [1] 44

> a=2

* a

[1] 2

> b<-10

* b [1] 10

> 30->c

* c [1] 30
* e<-f<-20
* e [1] 20
* f [1] 20
* assign("z",50)
* z [1] 50
* rm(z)
* theVariable <- 17
* theVariable [1] 17
* THEVARIABLE

Error: object 'THEVARIABLE' not found

* class(a)

[1] "numeric"

* is.numeric(b)

[1] TRUE

* i <- 9L
* i

[1] 9

* is.numeric(i)

[1] TRUE

* is.integer(i)

[1] TRUE

* class(4L)

[1] "integer"

* class(2.8)

[1] "numeric"

* s = 4L\*2.8
* class(s)

[1] "numeric"

* class(5L)

[1] "integer"

* s1 = 5L/2L
* class(s1)

[1] "numeric"

* x1 <- "data set"
* x1

[1] "data set"

* nchar(x1)

[1] 8

* nchar("welcome")

[1] 7

* nchar(5)

[1] 1

* nchar(1234)

[1] 4

* date1 <- as.Date("2022-03-31")
* date1

[1] "2022-03-31"

* class(date1)

[1] "Date"

* as.numeric(date1) [1] 19082
* date2 <- as.POSIXct("2022-03-31 17:42")
* class(date2)

[1] "POSIXct" "POSIXt"

* date2

[1] "2022-03-31 17:42:00 IST"

* as.numeric(date2) [1] 1648728720
* class(as.numeric(date1))

[1] "numeric"

* TRUE\*5

[1] 5

* FALSE\*10

[1] 0

* k <- TRUE
* k

[1] TRUE

* class(k)

[1] "logical"

* is.logical(k)

[1] TRUE

* T

[1] TRUE

* class(T)

[1] "logical"

* T = 10
* T [1] 10
* class(T)

[1] "numeric"

* 99 == 91

[1] FALSE

* 5!=8

[1] TRUE

* 9 < 10

[1] TRUE

* 9 <= 9

[1] TRUE

* # what is a variable
* "data" == "stats"

[1] FALSE

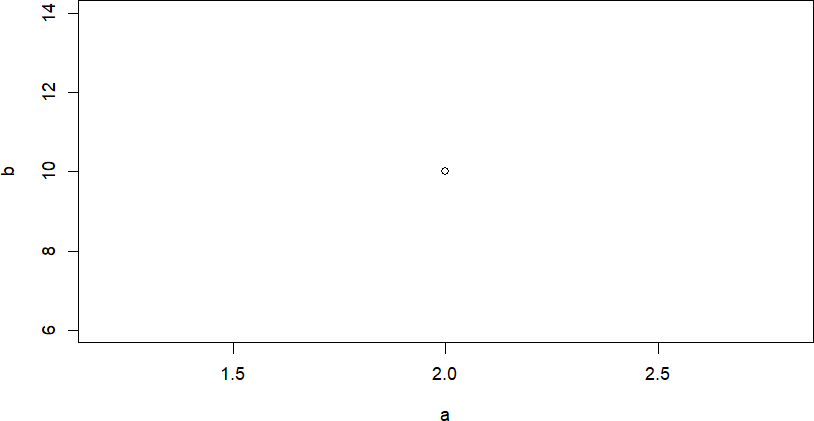
* "data" < "stats"

[1] TRUE

* "data" == "hell"

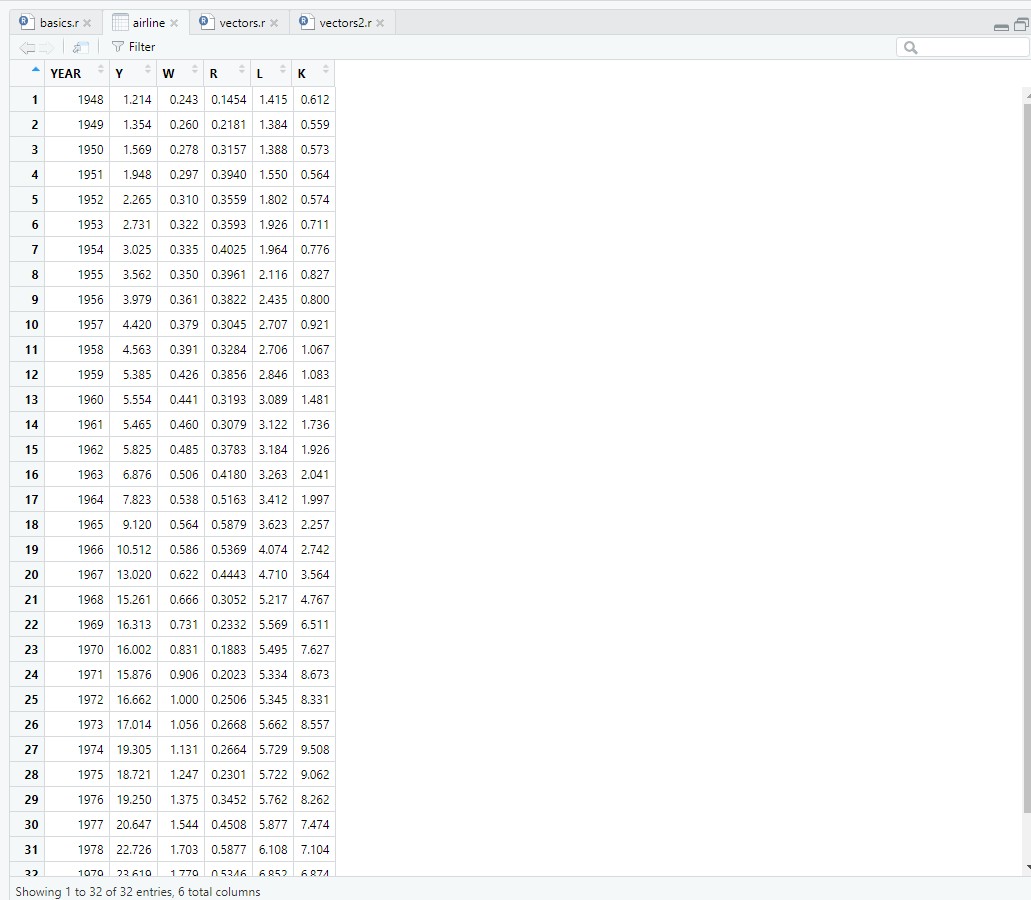
[1] FALSE

* plot(a,b)



**TASK – 02 :**

Importing Random Excel Dataset From Internet



**TASK – 01 :**

**R LAB – 02**

Applying Different Operations Vectors

**Program :**

x <- c(1,2,3,4,5,6,7,8,9,10)

x

x + 2

x - 5

x \* 5

x / 2

x ^ 2 sqrt(x) class(x)

1:10

10:1

-2:5

5:-2

a = 1:10

b = -5:4

a b

a + b a - b a \* b a / b a ^ b

length(a) length(b) length(a + b)

z = a + b \* x / 2 z

a + c(1,2)

a + c(1,2,3)

a <= 5

a < b

a1 = 10:1

b1 = -4:5

any(a1 < b1) all(a1 < b1)

q = c("Hockey","Football","Baseball","Curling","Rugby","Lacrosse","Basketball","Tennis"," Cricket","Soccer")

q

nchar(q) length(q)

a1[1] a1[1:3] a1[c(1,4)]

d1 = c(One = "x", Two = "Y", Three = "Z") d1

w = 1:3

w

names(w) = c("col1","col2","col3") w

q2 = c(q,"Hockey","Lacrosse","Hockey","Waterpolo","Hockey","Lacrosee") q2fact = as.factor(q2)

q2fact

as.numeric((q2fact))

abs(-356) ceiling(3.56) floor(3.56) round(3.56)

**Output :**

* x <- c(1,2,3,4,5,6,7,8,9,10)
* x

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

* x + 2

[1] 3 4 5 6 7 8 9 10 11 12

* x - 5

[1] -4 -3 -2 -1 0 1 2 3 4 5

* x \* 5

[1] 5 10 15 20 25 30 35 40 45 50

* x / 2

[1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

* x ^ 2

[1] 1 4 9 16 25 36 49 64 81 100

* sqrt(x)

[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427

3.000000 3.162278

* class(x)

[1] "numeric"

* 1:10

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

* 10:1

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

* -2:5

[1] -2 -1 0 1 2 3 4 5

* 5:-2

[1] 5 4 3 2 1 0 -1 -2

* a = 1:10
* b = -5:4
* a

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

* b

[1] -5 -4 -3 -2 -1 0 1 2 3 4

* a + b

[1] -4 -2 0 2 4 6 8 10 12 14

* a - b

[1] 6 6 6 6 6 6 6 6 6 6

* a \* b

[1] -5 -8 -9 -8 -5 0 7 16 27 40

* a / b

[1] -0.2 -0.5 -1.0 -2.0 -5.0 Inf 7.0 4.0 3.0 2.5

* a ^ b

[1] 1.000000e+00 6.250000e-02 3.703704e-02 6.250000e-02 2.000000e-01 1.000000e+00 7.000000e+00 6.400000e+01

[9] 7.290000e+02 1.000000e+04

* length(a) [1] 10
* length(b) [1] 10
* length(a + b) [1] 10
* z = a + b \* x / 2
* z

[1] -1.5 -2.0 -1.5 0.0 2.5 6.0 10.5 16.0 22.5 30.0

* a + c(1,2)

[1] 2 4 4 6 6 8 8 10 10 12

* a + c(1,2,3)

[1] 2 4 6 5 7 9 8 10 12 11

Warning message:

In a + c(1, 2, 3) :

longer object length is not a multiple of shorter object length

* a <= 5

[1] TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE

* a < b

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

* a1 = 10:1
* b1 = -4:5
* any(a1 < b1)

[1] TRUE

* all(a1 < b1)

[1] FALSE

* q = c("Hockey","Football","Baseball","Curling","Rugby","Lacrosse","Basketball","Tennis"," Cricket","Soccer")
* q

[1] "Hockey" "Football" "Baseball" "Curling" "Rugby" "Lacrosse" "Basketball" "Tennis"

[9] "Cricket" "Soccer"

* nchar(q)

[1] 6 8 8 7 5 8 10 6 7 6

* length(q) [1] 10
* a1[1]

[1] 10

* a1[1:3] [1] 10 9 8
* a1[c(1,4)] [1] 10 7
* d1 = c(One = "x", Two = "Y", Three = "Z")
* d1

One Two Three "x" "Y" "Z"

* w = 1:3
* w

[1] 1 2 3

* names(w) = c("col1","col2","col3")
* w

col1 col2 col3 1 2 3

* q2 = c(q,"Hockey","Lacrosse","Hockey","Waterpolo","Hockey","Lacrosee")
* q2fact = as.factor(q2)
* q2fact

[1] Hockey Football Baseball Curling Rugby Lacrosse Basketball Tennis Cricket

[10] Soccer Hockey Lacrosse Hockey Waterpolo Hockey Lacrosee

12 Levels: Baseball Basketball Cricket Curling Football Hockey Lacrosee Lacrosse Rugby Soccer ... Waterpolo

* as.numeric((q2fact))

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

* abs(-356) [1] 356
* ceiling(3.56)

[1] 4

* floor(3.56)

[1] 3

* round(3.56)

[1] 4

**R LAB – 03**

**Task – 01 :**

**Aim :** Apply different operations on Vectors.

1. Write R code to create a vector of a specified type and length. Create vector of numeric, complex, logical and character types of length 6.

### Program :

numbers = c(1,2,3,4,5,6) numbers class(numbers)

alpha = c('A','B','C','D','E','F')

alpha class(alpha)

comp = c(1+2i,3+4i,5+6i,7+8i,9+3i,5+3i)

comp

class(comp)

logar = c(TRUE,FALSE,TRUE,FALSE,TRUE,FALSE)

logar class(logar) **Output :**

* + numbers = c(1,2,3,4,5,6)
  + numbers [1] 1 2 3 4 5 6
  + class(numbers)

[1] "numeric"

* + alpha = c('A','B','C','D','E','F')
  + alpha

[1] "A" "B" "C" "D" "E" "F"

* + class(alpha)

[1] "character"

* + comp = c(1+2i,3+4i,5+6i,7+8i,9+3i,5+3i)
  + comp

[1] 1+2i 3+4i 5+6i 7+8i 9+3i 5+3i

* + class(comp)

[1] "complex"

* + logar = c(TRUE,FALSE,TRUE,FALSE,TRUE,FALSE)
  + logar

[1] TRUE FALSE TRUE FALSE TRUE FALSE

* + class(logar)

[1] "logical"

1. Write R code to add two vectors of integer’s type and length 3

### Program :

vect1 = c(1,2,3) vect1

vect2 = c(4,5,6) vect2

sum = vect1 + vect2 sum

### Output :

* + vect1 = c(1,2,3)
  + vect1 [1] 1 2 3
  + vect2 = c(4,5,6)
  + vect2 [1] 4 5 6
  + sum = vect1 + vect2
  + sum [1] 5 7 9

1. Write R code to append value to a given empty vector

### Program :

x = c()

k = append(x,1:5) k

### Output :

* + x = c()
  + k = append(x,1:5)
  + k

[1] 1 2 3 4 5

1. Write R code to multiply two vectors of integer’s type and length 3.

### Program :

mul1 = c(1,2,3)

mul2 = c(6,7,8) mul3 = mul1\*mul2 mul3

### Output :

* + mul1 = c(1,2,3)
  + mul2 = c(6,7,8)
  + mul3 = mul1\*mul2
  + mul3

[1] 6 14 24

1. Write R code to divide two vectors of integer’s type and length 3.

### Program :

div1 = c(4,6,8)

div2 = c(2,2,2) div3 = div1 /div2 div3

### Output :

* + div1 = c(4,6,8)
  + div2 = c(2,2,2)
  + div3 = div1 /div2
  + div3 [1] 2 3 4

1. Write R code to find Sum, Mean and Product of a Vector

### Program :

vect = c(1,2,3,4,5)

vect

addv = sum(vect) addv

avg = mean(vect) avg

mul = prod(vect) mul

### Output :

* + vect = c(1,2,3,4,5)
  + vect

[1] 1 2 3 4 5

* + addv = sum(vect)
  + addv [1] 15
  + avg = mean(vect)
  + avg

[1] 3

* + mul = prod(vect)
  + mul [1] 120

1. Write R code to find Sum, Mean and Product of a Vector, ignore element like NA or NaN.

### Prorgam :

temp = c(1,NA,5,NA,2,3)

vadd = sum(temp,na.rm = TRUE) vmean = mean(temp,na.rm = TRUE) vmul = prod(temp,na.rm = TRUE) vadd

vmean vmul **Output :**

* + temp = c(1,NA,5,NA,2,3)
  + vadd = sum(temp,na.rm = TRUE)
  + vmean = mean(temp,na.rm = TRUE)
  + vmul = prod(temp,na.rm = TRUE)
  + vadd [1] 11
  + vmean [1] 2.75
  + vmul [1] 30

1. Write R code to find the minimum and the maximum of a Vector.

### Program :

large = c(2,4,64,56,43,1)

small = min(large) big = max(large) small

big

### Output :

* + large = c(2,4,64,56,43,1)
  + small = min(large)
  + big = max(large)
  + small

[1] 1

* + big [1] 64

1. Write R code to sort a Vector in ascending and descending order.

### Program :

unsort = c(45,1,67,23,98,43,66)

ascen = sort(unsort,decreasing = FALSE) desen = sort(unsort, decreasing = TRUE) ascen

desen

### Output :

* + unsort = c(45,1,67,23,98,43,66)
  + ascen = sort(unsort,decreasing = FALSE)
  + desen = sort(unsort, decreasing = TRUE)
  + ascen

[1] 1 23 43 45 66 67 98

* + desen

[1] 98 67 66 45 43 23 1

1. Write R code to test whether a given vector contains a specified element.

### Program :

lins = c(2,4,5,6,7)

res = match(4,lins) res

### Output :

* + lins = c(2,4,5,6,7)
  + res = match(4,lins)
  + res

[1] 2

1. Write R code to count the specific value in a given vector..

### Program :

rep = c(1,2,2,2,3,4,1)

result = sum(rep == 2) result

### Output :

* + rep = c(1,2,2,2,3,4,1)
  + result = sum(rep == 2)
  + result

[1] 3

1. Write R code to access the last value in a given vector.

### Program :

org = c(1,2,3,4,5)

orgres = tail(org,n = 1) orgres

### Output :

* + org = c(1,2,3,4,5)
  + orgres = tail(org,n = 1)
  + orgres

[1] 5

1. Write R code to find second highest value in a given vector.

### Program :

sh = c(15,3,10,1,7,9)

sh

sh2 = sort(sh,decreasing = TRUE) sh3 = sh2[2]

sh3

### Output :

* sh = c(15,3,10,1,7,9)
* sh

[1] 15 3 10 1 7 9

* sh2 = sort(sh,decreasing = TRUE)
* sh3 = sh2[2]
* sh3 [1] 10

1. Write R code to find nth highest value in a given vector.

### Program :

n1 = c(15,3,10,1,7,9)

n2 = readline()

n3 = sort(n1,decreasing = TRUE) n4 = n3[n2]

n4

### Output :

* n1 = c(15,3,10,1,7,9)
* n2 = readline() 80

n3 = sort(n1,decreasing = TRUE)

* n4 = n3[n2]
* n4
  1. NA

1. Write R code to find common elements from multiple vector.

### Program :

cv = c(2,1,3,4,5)

cv2 = c(2,6,7,1,9)

cv3 = c(3,7,8,2,10)

cv4 = intersect(intersect(cv,cv2),cv3) cv4

### Output :

* cv = c(2,1,3,4,5)
* cv2 = c(2,6,7,1,9)
* cv3 = c(3,7,8,2,10)
* cv4 = intersect(intersect(cv,cv2),cv3)
* cv4

[1] 2

1. Write R code to convert given dataframe column(s) to a vector.

### Program :

df1 = c(1,2,3,4,5)

df2 = c(6,7,8,9,10)

df3 = c(11,12,13,14,15) df4 = c(16,17,18,19,20)

df <- data.frame(df1 = 1:5, df2 = 6:10, df3 = 11:15, df4 = 16:20)

df

### Output :

* df1 = c(1,2,3,4,5)
* df2 = c(6,7,8,9,10)
* df3 = c(11,12,13,14,15)
* df4 = c(16,17,18,19,20)
* df <- data.frame(df1 = 1:5, df2 = 6:10, df3 = 11:15, df4 = 16:20)
* df

df1 df2 df3 df4 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. Write R code to extract every nth element of a given vector.

### Program :

gv = 1:30

gv1 = gv[seq(1, length(gv), 5)] gv1

### Output :

* gv = 1:30
* gv1 = gv[seq(1, length(gv), 5)]
* gv1

[1] 1 6 11 16 21 26

1. Write R code to list the distinct values in a vector from a given vector.

### Program :

repv <- c(10,10,10,20,20,30,40,50,89,89)

repv1 <- unique(repv) repv1

### Output :

* repv <- c(10,10,10,20,20,30,40,50,89,89)
* repv1 <- unique(repv)
* repv1

[1] 10 20 30 40 50 89

1. Write R code to find the elements of a given vector that are not in another given vector.

### Program :

de = c(1,2,3,3,3,4)

de1 = c(5,6,6,6,8)

de2 = setdiff(de,de1) de2

### Output :

* de = c(1,2,3,3,3,4)
* de1 = c(5,6,6,6,8)
* de2 = setdiff(de,de1)
* de2

[1] 1 2 3 4

1. Write R code to reverse the order of given vector.

### Program :

v1 = c(1,2,3,4,5)

v2 = rev(v1) v2

### Output :

* v1 = c(1,2,3,4,5)
* v2 = rev(v1)
* v2

[1] 5 4 3 2 1

1. Write R code to concatenate a vector.

**Program :** vcon1 = c(1,2,3) vcon2 = c(4,5,6)

vcon3 = c(vcon1,vcon2) vcon3

### Output :

* vcon1 = c(1,2,3)
* vcon2 = c(4,5,6)
* vcon3 = c(vcon1,vcon2)
* vcon3

[1] 1 2 3 4 5 6

1. Write R code to count number of values in a range in a given vector.

### Program :

r1 = c(0,1,2,3,4,5,6,7,8,9,10)

r2 = sum(r1 > 2 & r1 < 9) r2

### Output :

* r1 = c(0,1,2,3,4,5,6,7,8,9,10)

> r2 = sum(r1 > 2 & r1 < 9)

* r2

[1] 6

1. Write R code to convert two columns of a data frame to a named

vector.

### Program :

plang = c(' c language ',' python ',' c++ ',' HTMl ',' java ')

ide = c('Turboc++', 'pycharm', 'online', 'notepad', 'netbeans')

tab = data.frame(languages = plang, IDE = ide) tab

setNames(as.character(tab$languages), as.character(tab$IDE))

### Output :

* plang = c(' c language ',' python ',' c++ ',' HTMl ',' java ')
* ide = c('Turboc++', 'pycharm', 'online', 'notepad', 'netbeans')
* tab = data.frame(languages = plang, IDE = ide)
* tab

languages IDE

1. c language Turboc++
2. python pycharm
3. c++ online
4. HTMl notepad
5. java netbeans

* setNames(as.character(tab$languages), as.character(tab$IDE))

Turboc++ pycharm online notepad netbeans

" c language " " python " " c++ " " HTMl " " java "

1. Write R code to create a vector and find the length and the dimension of the vector.

### Program :

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

length(real) dim(real) **Output :**

* real = c(1,2,3,4,5,6)
* length(real)

[1] 6

* dim(real) NULL

1. Write R code to test whether the value of the element of a given vector greater than 10 or not. Return TRUE or FALSE.

### Program :

whole = c(0,1,2,3,25,15,99,100)

whole > 10

### Output :

* whole = c(0,1,2,3,25,15,99,100)
* whole > 10
  1. FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE

1. Write R code to add 3 to each element in a given vector. Print the original and new vector.

### Program :

ov = c(1,2,3,4)

nv = ov + 3 ov

nv

### Output :

* ov = c(1,2,3,4)
* nv = ov + 3
* ov

[1] 1 2 3 4

* nv

[1] 4 5 6 7

1. Write a R code to create a vector using: operator and seq() function.

### Program :

sap = seq(from = 1, to = 30) sap

### Output :

* sap = seq(from = 1, to = 30)
* sap

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30

**Result :** Successfully completed the Vectors task.

**Task – 01 :**

**R LAB -- 04**

**Aim :** Create a data frame that stores the name, age, designation of the

employee. Find how many employees are working in each designation?

**Program :**

name = c(' Mounav ', ' Rizwan ', ' Ajay ', ' Charan ', ' Vamsi ', ' M.sai ')

age = c(19, 19, 20, 30, 18, 20)

desig = c(' Manager ', ' webprogramer ', ' cyber-manager ', ' Manager ', ' webprogramer ', ' Boss ')

df = data.frame(Employee = name, Age = age, Designation = desig)

df

ind = unique(desig)

des = c() no = c()

for(i in 1 : length(ind))

{

count = 0

for(j in 1 : length(desig))

{

if(ind[i] == desig[j])

{

count = count + 1

}

}

des = append(des,ind[i]) no = append(no,count)

}

des no

result = data.frame("Designation" = ind, "Count" = no) result

**Output :**

* name = c(' Mounav ', ' Rizwan ', ' Ajay ', ' Charan ', ' Vamsi ', ' M.sai ')
* age = c(19, 19, 20, 30, 18, 20)
* desig = c(' Manager ', ' webprogramer ', ' cyber-manager ', ' Manager ', ' webprogramer ', ' Boss ')
* df = data.frame(Employee = name, Age = age, Designation = desig)
* df

Employee Age Designation

1. Mounav 19 Manager
2. Rizwan 19 webprogramer
3. Ajay 20 cyber-manager
4. Charan 30 Manager
5. Vamsi 18 webprogramer
6. M.sai 20 Boss

* ind = unique(desig)
* des = c()
* no = c()
* for(i in 1 : length(ind))

+ {

+ count = 0

+ for(j in 1 : length(desig))

+ {

+ if(ind[i] == desig[j])

+ {

+ count = count + 1

+ }

+ }

+ des = append(des,ind[i])

+ no = append(no,count)

+ }

* des

[1] " Manager " " webprogramer " " cyber-manager " " Boss "

* no

[1] 2 2 1 1

* result = data.frame("Designation" = ind, "Count" = no)
* result

Designation Count

1. Manager 2
2. webprogramer 2
3. cyber-manager 1
4. Boss 1

**Aim :** Create two vectors that stores the details of name and gender of the employees. Find how many ‘male’ and ‘female’ employees are present?

**Program :**

name1 <- c('sai','geetha','ajay','Madhu','vamsi','Parveen')

gender <- c('male','female','male','female','male','female') mc = sum(gender == 'male')

fc = sum(gender == 'female') print("No.of male employess : ")

mc

print("No.of female employess : ") fc

**Output :**

* name1 <- c('sai','geetha','ajay','Madhu','vamsi','Parveen')
* gender <- c('male','female','male','female','male','female')
* mc = sum(gender == 'male')
* fc = sum(gender == 'female')
* print("No.of male employess : ")

[1] "No.of male employess : "

* mc

[1] 3

* print("No.of female employess : ")

1. "No.of female employess : "

* fc

[1] 3

**Result :** Successfully completed the Both Aims.

# R LAB – 05

**Task – 01 :**

**Aim :** Apply Different operations On Matrices in R.

* 1. Write a R program to create a matrix taking a given vector of numbers as input. Display the matrix

**Program :**

num = c(1,2,3,4,5,6,7,8,9,10)

ans = matrix(num, nrow = 5) ans

**Output :**

* num = c(1,2,3,4,5,6,7,8,9,10)
* ans = matrix(num, nrow = 5)
* ans

[,1] [,2]

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

* 1. Write a R program to create a matrix taking a given vector of numbers as input and define the column and row names. Display the matrix

**Program :**

vect = c(1,2,3,4,5,6,7,8,9)

ans2 = matrix(vect, nrow = 3) ans2

rownames(ans2) colnames(ans2)

rownames(ans2) = c("First","Second","Third") colnames(ans2) = c("col1","col2","col3") rownames(ans2)

colnames(ans2) ans2

**Output :**

* vect = c(1,2,3,4,5,6,7,8,9)
* ans2 = matrix(vect, nrow = 3)
* ans2

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

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | 1 | 4 | 7 |
| [2,] | 2 | 5 | 8 |
| [3,] | 3 | 6 | 9 |

* rownames(ans2)

NULL

* colnames(ans2) NULL
* rownames(ans2) = c("First","Second","Third")
* colnames(ans2) = c("col1","col2","col3")
* rownames(ans2)
  + 1. "First" "Second" "Third"
* colnames(ans2)

[1] "col1" "col2" "col3"

* ans2

col1 col2 col3 First 1 4 7

Second 2 5 8

Third 3 6 9

* 1. Write a R program to access the element at 3rd column and 2nd row, only the 3rd row and only the 4th column of a given matrix

**Program :**

vect3 = c(9,8,7,6,5,4,3,2,1)

ans3 = matrix(vect3, nrow = 3) ans3

res = ans3[c(2),c(3)] res

**Output :**

* vect3 = c(9,8,7,6,5,4,3,2,1)
* ans3 = matrix(vect3, nrow = 3)
* ans3

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

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | 9 | 6 | 3 |
| [2,] | 8 | 5 | 2 |
| [3,] | 7 | 4 | 1 |

* res = ans3[c(2),c(3)]
* res

[1] 2

* 1. Write a R program to create two 2x3 matrix and add, subtract, multiply and divide the matrixes.

**Program :**

arth = matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3) arth

arth2 = matrix(c(7,8,9,10,11,12), nrow = 2, ncol = 3) arth2

arth3 = t(arth2) arth3

add = arth + arth2 add

sub = arth - arth2 sub

mul = arth \* arth2 mul

div = arth / arth2 div

**Output :**

* arth = matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3)
* arth

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

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | 1 | 3 | 5 |
| [2,] | 2 | 4 | 6 |

* arth2 = matrix(c(7,8,9,10,11,12), nrow = 2, ncol = 3)
* arth2

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

|  |  |  |
| --- | --- | --- |
| [1,] | 7 | 9 11 |
| [2,] | 8 | 10 12 |

* arth3 = t(arth2)
* arth3

[,1] [,2]

|  |  |  |
| --- | --- | --- |
| [1,] | 7 | 8 |
| [2,] | 9 | 10 |
| [3,] | 11 | 12 |

* add = arth + arth2
* add

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

[1,] 8 12 16

[2,] 10 14 18

* sub = arth - arth2
* sub

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

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | -6 | -6 | -6 |
| [2,] | -6 | -6 | -6 |

* mul = arth \* arth2
* mul

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

[1,] 7 27 55

[2,] 16 40 72

* div = arth / arth2
* div

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

[1,] 0.1428571 0.3333333 0.4545455

[2,] 0.2500000 0.4000000 0.5000000

* 1. Write a R program to extract the submatrix whose rows have column value > 7 from a given matrix

**Program :**

rname = c("r1","r2","r3","r4")

cname = c("c1","c2","c3","c4")

vect5 = matrix(1:16, nrow = 4, byrow = TRUE, dimnames

= list(rname,cname)) vect5

vect5[vect5[,3] > 7,]

**Output :**

* rname = c("r1","r2","r3","r4")
* cname = c("c1","c2","c3","c4")
* vect5 = matrix(1:16, nrow = 4, byrow = TRUE, dimnames = list(rname,cname))
* vect5

c1 c2 c3 c4 r1 1 2 3 4

r2 5 6 7 8

r3 9 10 11 12

r4 13 14 15 16

>

> vect5[vect5[,3] > 7,] c1 c2 c3 c4

r3 9 10 11 12

r4 13 14 15 16

* 1. Write a R program to convert a given matrix to a list of column-vectors

**Program :**

x = matrix(1:12, ncol=3) x

l = split(x, rep(1:ncol(x), each = nrow(x))) l

**Output :**

* x = matrix(1:12, ncol=3)
* x

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

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | 1 | 5 | 9 |
| [2,] | 2 | 6 | 10 |
| [3,] | 3 | 7 | 11 |
| [4,] | 4 | 8 | 12 |

* l = split(x, rep(1:ncol(x), each = nrow(x)))
* l

$`1`

[1] 1 2 3 4

$`2`

[1] 5 6 7 8

$`3`

[1] 9 10 11 12

* 1. Write a R program to find row and column index of maximum and minimum value in a given matrix **Program :**

mat7 = matrix(1:20, nrow = 4) mat7

a = max(mat7) b = min(mat7)

which(mat7 == a,arr.ind = TRUE) which(mat7 == b, arr.ind = TRUE) **Output :**

* mat7 = matrix(1:20, nrow = 4)
* mat7

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

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

* a = max(mat7)
* b = min(mat7)
* which(mat7 == a,arr.ind = TRUE) row col

[1,] 4 5

* which(mat7 == b, arr.ind = TRUE) row col

[1,] 1 1

**Task – 02 :**

**Aim :** Access the Databases and Tables from mysql to R

**Program :** install.packages("RMySQL") library(RMySQL)

mydb = dbConnect(MySQL(), user='root', password='', dbname='student', host='localhost')

mydb

dbListTables(mydb) dbListFields(mydb, 'ddl\_student')

rs = dbSendQuery(mydb, "select \* from ddl\_student") rs

data = fetch(rs, n=-1) data

**Output :**

* mydb = dbConnect(MySQL(), user='root', password='', dbname='student', host='localhost')
* mydb

<MySQLConnection:0,0>

* dbListTables(mydb)

[1] "ddl\_student" "dml\_student" "sample" "student\_data\_type" "student\_info"

* dbListFields(mydb, 'ddl\_student')

[1] "S\_no" "section" "Roll\_no" "maths" "python" "college" "dept"

* rs = dbSendQuery(mydb, "select \* from ddl\_student")
* rs

<MySQLResult:2,0,2>

* data = fetch(rs, n=-1)
* data

S\_no section Roll\_no maths python college dept

|  |  |  |  |
| --- | --- | --- | --- |
| 1 NA | <NA> 208w1a12a0 98.50 | 99.0 | vrsec <NA> |
| 2 NA | <NA> 208w1a1299 92.40 | 96.0 | vrsec <NA> |
| 3 NA | <NA> 208w1a1291 100.98 | 99.9 | vrsec <NA> |

**Task – 03 :**

**Aim :** Load the CSV file into R and manipulate data inside csv.

**Program :**

getwd()

setwd("E:/venkat sai/rstudio\_language") # it will set the new directory location

getwd() # it will return the present location of working directory. cdata <- read.csv("company.csv")

cdata

# Analyzing the CSV file is.data.frame(cdata) ncol(cdata)

nrow(cdata)

# getting the maximum salary from the csv file sal <- max(cdata$salary)

sal

# getting the person details from the max salary psal <- subset(cdata, salary == max(salary)) psal

temp = subset(cdata, dept == "IT") temp

# Employess less than 600 in IT department lesit = subset(cdata, salary < 600 & dept == "IT") lesit

# Employess joined After 2014 Year

afyear = subset(cdata, as.Date(start\_date) > as.Date("2014-01- 01"))

afyear

# Writing into CSV file write.csv(afyear,"output.csv") read.csv("output.csv") **Output :**

* getwd()

[1] "C:/Users/SHREEE/OneDrive/Documents"

* setwd("E:/venkat sai/rstudio\_language") # it will set the new directory location

>

* getwd() # it will return the present location of working directory.

[1] "E:/venkat sai/rstudio\_language"

* cdata <- read.csv("company.csv")
* cdata

id name salary start\_date dept 1 1 Rick 623.30 2012-01-01 IT

2 2 Dan 515.20 2013-09-23 Operations

3 3 Michelle 611.00 2014-11-15 IT

|  |  |  |  |
| --- | --- | --- | --- |
| 4 4 | | Ryan 729.00 2014-05-11 | HR |
| 5 5 | | Gary 843.25 2015-03-27 | Finance |
| 6 6 | | Nina 578.00 2013-05-21 | IT |
| 7 | 7 | Simon 632.80 2013-07-30 Operations | |
| 8 | 8 | Guru 722.50 2014-06-17 Finance | |

* # Analyzing the CSV file
* is.data.frame(cdata)

[1] TRUE

* ncol(cdata)

[1] 5

* nrow(cdata)

[1] 8

* # getting the maximum salary from the csv file
* sal <- max(cdata$salary)
* sal

[1] 843.25

* # getting the person details from the max salary
* psal <- subset(cdata, salary == max(salary))
* psal

id name salary start\_date dept

5 5 Gary 843.25 2015-03-27 Finance

* temp = subset(cdata, dept == "IT")
* temp

id name salary start\_date dept 1 1 Rick 623.3 2012-01-01 IT

3 3 Michelle 611.0 2014-11-15 IT

6 6 Nina 578.0 2013-05-21 IT

* # Employess less than 600 in IT department
* lesit = subset(cdata, salary < 600 & dept == "IT")
* lesit

id name salary start\_date dept 6 6 Nina 578 2013-05-21 IT

* # Employess joined After 2014 Year
* afyear = subset(cdata, as.Date(start\_date) > as.Date("2014- 01-01"))
* afyear

id name salary start\_date dept

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 3 Michelle 611.00 2014-11-15 | | IT |
| 4 | 4 Ryan 729.00 2014-05-11 | | HR |
| 5 | 5 | Gary 843.25 2015-03-27 Finance | |
| 8 | 8 | Guru 722.50 2014-06-17 Finance | |

* # Writing into CSV file
* write.csv(afyear,"output.csv")
* read.csv("output.csv")

X id name salary start\_date dept

|  |  |  |  |
| --- | --- | --- | --- |
| 1 3 | 3 Michelle 611.00 2014-11-15 | | IT |
| 2 4 | 4 Ryan 729.00 2014-05-11 | | HR |
| 3 5 | 5 | Gary 843.25 2015-03-27 Finance | |
| 4 8 | 8 | Guru 722.50 2014-06-17 Finance | |

**Task – 04 :**

**Aim :** load Excel file into R and do changes in data

**Program :** install.packages("readxl") library("readxl")

getwd()

setwd("E:/venkat sai/rstudio\_language") getwd()

read\_excel("product\_list.xlsx") # writing into an excel file install.packages("writexl") library("writexl")

x = 10:1

y = -4:5

q = c("Hockey","Football","Baseball","Curling","Rugby","Lacrosse"," Basketball","Tennis","Cricket","Soccer")

theDF = data.frame(x,y,q) theDF write\_xlsx(theDF,"sports.xlsx") **Output :**

* getwd()

[1] "E:/venkat sai/rstudio\_language"

* setwd("E:/venkat sai/rstudio\_language")
* getwd()

[1] "E:/venkat sai/rstudio\_language"

* read\_excel("product\_list.xlsx") # A tibble: 4 x 2

Product Price

<chr> <dbl>

1. Refrigerator 1200
2. Oven 750
3. Dishwasher 900
4. Coffee Maker 300

* x = 10:1
* y = -4:5
* q = c("Hockey","Football","Baseball","Curling","Rugby","Lacrosse"," Basketball","Tennis","Cricket","Soccer")
* theDF = data.frame(x,y,q)
* theDF

x y q

1. 10 -4 Hockey
2. 9 -3 Football
3. 8 -2 Baseball
4. 7 -1 Curling
5. 6 0 Rugby
6. 5 1 Lacrosse
7. 4 2 Basketball
8. 3 3 Tennis
9. 2 4 Cricket
10. 1 5 Soccer

* write\_xlsx(theDF,"sports.xlsx")

**Result** : Successfully completed the 4 tasks

# R LAB – 06

**Task – 01 :**

**Aim :** using the apply , aggregate , data.table functionalites manipulate the any one random data..

### Program :

# apply function

theMatrix = matrix(1:9, nrow = 3) theMatrix

# 1 means sum of rows and 2 means sum of columns apply(theMatrix, 1, sum)

apply(theMatrix, 2, sum) theMatrix[2,3] <- NA apply(theMatrix, 1, sum)

apply(theMatrix, 1, sum, na.rm = TRUE) rowSums(theMatrix, na.rm = TRUE) colSums(theMatrix, na.rm = TRUE)

# lapply & sapply functions

theList <- list(A = matrix(1:9,3), B = 1:5, C = matrix(1:4,2), D = 2)

theList

lapply(theList, sum) # returns list sapply(theList, sum) # returns data frame

theNames <- c("Tyson", "Rizwanullah", "Dragon Emperor", "A.Charan")

lapply(theNames, nchar) sapply(theNames, nchar) # mapply function

f3 <- function(x,y)

{

NROW(x) + NCOL(y)

}

flist = list(A = matrix(1:16,4), B = matrix(1:16,2), C = 1:5) flist

slist = list(A = matrix(1:16,4), B = matrix(1:16,2), C = 15:1)

slist

mapply(identical, flist, slist) mapply(f3, flist, slist)

# using aggregate function eid = c(2501 : 2509)

eid

ename = c("Rizwan", "Ajay", "Mounav", "Charan",

"Srinivas", "Vamsi", "Deepak", "Abhi", "Pavan") ename

desig = c("sales", "accounts", "manager", "sales", "sales", "accounts", "accounts", "manager", "sales")

desig

dept\_id = c(10,10,10,10,20,20,20,30,30)

dept\_id

salary = c(23000, 35000, 40000, 80000, 230000, 98000,

50000, 85000, 130000)

salary

employee = data.frame(eid, ename, desig, dept\_id, salary)

employee

aggregate(salary~dept\_id, employee, mean) aggregate(salary~dept\_id, employee, max) aggregate(salary~dept\_id+desig, employee, mean) aggregate(salary~dept\_id+desig, employee, min) #data()

install.packages("plyr") require("plyr") #data(package = "plyr") # using plyr package

employee1 = data.frame(eid, ename, desig, dept\_id, salary)

employee1

employee1$eid[employee1$salary < 36000] <- 0 employee1

any(is.na(employee1$eid))

# using data table install.packages("data.table") require("data.table")

theDT = data.table(eid, ename, desig, dept\_id, salary) theDT

class(theDT$ename) class(employee1$ename) theDT[1:2,]

theDT[theDT$eid >= 2504] theDT[, list(eid,desig)] theDT[, ename]

theDT[, list(ename)]

# if yu use column name as character name then yu should use with attribute

theDT[, "ename", with = FALSE] theDT[, c("ename","eid"), with = FALSE] # get all the tables

tables()

# setting the key in the data table setkey(theDT, salary)

theDT key(theDT) tables()

theDT[theDT$salary > 50000]

setkey(theDT, desig, dept\_id, salary) tables()

theDT

# aggregate on data tables

theDT[, mean(salary), by = dept\_id]

theDT[, list(price = mean(salary)), by = dept\_id] theDT[, mean(salary), by = list(dept\_id,desig)]

theDT[, list(price = mean(salary), desig = mean(eid)), by

= dept\_id] #data()

### Output :

* theMatrix = matrix(1:9, nrow = 3)
* theMatrix [,1] [,2] [,3]

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | 1 | 4 | 7 |
| [2,] | 2 | 5 | 8 |
| [3,] | 3 | 6 | 9 |

* apply(theMatrix, 1, sum) [1] 12 15 18
* apply(theMatrix, 2, sum) [1] 6 15 24
* theMatrix[2,3] <- NA
* apply(theMatrix, 1, sum)

[1] 12 NA 18

* apply(theMatrix, 1, sum, na.rm = TRUE) [1] 12 7 18
* rowSums(theMatrix, na.rm = TRUE) [1] 12 7 18
* colSums(theMatrix, na.rm = TRUE) [1] 6 15 16
* theList <- list(A = matrix(1:9,3), B = 1:5, C = matrix(1:4,2), D = 2)
* theList

$A

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

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | 1 | 4 | 7 |
| [2,] | 2 | 5 | 8 |
| [3,] | 3 | 6 | 9 |
| $B |  |  |  |

[1] 1 2 3 4 5

$C

[,1] [,2]

[1,] 1 3

[2,] 2 4

$D

[1] 2

* lapply(theList, sum) # returns list

$A [1] 45

$B [1] 15

$C [1] 10

$D

[1] 2

* sapply(theList, sum) # returns data frame A B C D

45 15 10 2

* theNames <- c("Tyson", "Rizwanullah", "Dragon Emperor", "A.Charan")
* lapply(theNames, nchar) [[1]]

[1] 5 [[2]] [1] 11 [[3]] [1] 14 [[4]]

[1] 8

* sapply(theNames, nchar)

Tyson Rizwanullah Dragon Emperor A.Charan

5 11 14 8

* f3 <- function(x,y)

+ {

+ NROW(x) + NCOL(y)

+ }

* flist = list(A = matrix(1:16,4), B = matrix(1:16,2), C = 1:5)
* flist

$A

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [1,] | 1 | 5 | 9 | 13 |
| [2,] | 2 | 6 | 10 | 14 |
| [3,] | 3 | 7 | 11 | 15 |
| [4,] | 4 | 8 | 12 | 16 |
| $B |  |  |  |  |

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]

[1,] 1 3 5 7 9 11 13 15

[2,] 2 4 6 8 10 12 14 16

$C

[1] 1 2 3 4 5

* slist = list(A = matrix(1:16,4), B = matrix(1:16,2), C = 15:1)
* slist

$A

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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [1,] | 1 | 5 | 9 | 13 | | | | |
| [2,] | 2 | 6 | 10 | 14 | | | | |
| [3,] | 3 | 7 | 11 | 15 | | | | |
| [4,] | 4 | 8 | 12 | 16 | | | | |
| $B |  |  |  |  | | | | |
| [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] | | | | | | | | |
| [1,] | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 |
| [2,] | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| $C |  |  |  |  |  |  |  |  |
| [1] 15 14 13 12 11 10 | | | | | | 9 8 | 7 6 | 5 4 3 2 1 |

* mapply(identical, flist, slist) A B C

TRUE TRUE FALSE

* mapply(f3, flist, slist) A B C

8 10 6

* eid = c(2501 : 2509)
* eid

[1] 2501 2502 2503 2504 2505 2506 2507 2508 2509

* ename = c("Rizwan", "Ajay", "Mounav", "Charan",

"Srinivas", "Vamsi", "Deepak", "Abhi", "Pavan")

|  |  |  |
| --- | --- | --- |
| * ename |  | |
| [1] "Rizwan" "Ajay" "Vamsi" "Deepak" | "Mounav" "Charan" "Abhi" "Pavan" | "Srinivas" |

* desig = c("sales", "accounts", "manager", "sales", "sales", "accounts", "accounts", "manager", "sales")
* desig

[1] "sales" "accounts" "manager" "sales" "sales" "accounts" "accounts" "manager" "sales"

* dept\_id = c(10,10,10,10,20,20,20,30,30)
* dept\_id

[1] 10 10 10 10 20 20 20 30 30

* salary = c(23000, 35000, 40000, 80000, 230000,

98000, 50000, 85000, 130000)

* salary

[1] 23000 35000 40000 80000 230000 98000 50000

85000 130000

* employee = data.frame(eid, ename, desig, dept\_id, salary)
* employee

eid ename desig dept\_id salary 1 2501 Rizwan sales 10 23000

1. 2502 Ajay accounts 10 35000
2. 2503 Mounav manager 10 40000
3. 2504 Charan sales 10 80000
4. 2505 Srinivas sales 20 230000

|  |  |  |
| --- | --- | --- |
| 6 2506 | Vamsi accounts | 20 98000 |
| 7 2507 | Deepak accounts | 20 50000 |
| 8 2508 | Abhi manager | 30 85000 |
| 9 2509 | Pavan sales | 30 130000 |

* aggregate(salary~dept\_id, employee, mean) dept\_id salary

1 10 44500

2 20 126000

3 30 107500

* aggregate(salary~dept\_id, employee, max) dept\_id salary

1 10 80000

2 20 230000

3 30 130000

* aggregate(salary~dept\_id+desig, employee, mean) dept\_id desig salary

1. 10 accounts 35000
2. 20 accounts 74000 3 10 manager 40000 4 30 manager 85000 5 10 sales 51500

6 20 sales 230000

7 30 sales 130000

* aggregate(salary~dept\_id+desig, employee, min)

dept\_id desig salary

1. 10 accounts 35000
2. 20 accounts 50000 3 10 manager 40000 4 30 manager 85000 5 10 sales 23000

6 20 sales 230000

7 30 sales 130000

* employee1 = data.frame(eid, ename, desig, dept\_id, salary)
* employee1

eid ename desig dept\_id salary 1 2501 Rizwan sales 10 23000

1. 2502 Ajay accounts 10 35000
2. 2503 Mounav manager 10 40000
3. 2504 Charan sales 10 80000
4. 2505 Srinivas sales 20 230000
5. 2506 Vamsi accounts 20 98000
6. 2507 Deepak accounts 20 50000
7. 2508 Abhi manager 30 85000

9 2509 Pavan sales 30 130000

* employee1$eid[employee1$salary < 36000] <- 0
* employee1

eid ename desig dept\_id salary

1. 0 Rizwan sales 10 23000
2. 0 Ajay accounts 10 35000
3. 2503 Mounav manager 10 40000
4. 2504 Charan sales 10 80000
5. 2505 Srinivas sales 20 230000
6. 2506 Vamsi accounts 20 98000
7. 2507 Deepak accounts 20 50000
8. 2508 Abhi manager 30 85000

9 2509 Pavan sales 30 130000

* any(is.na(employee1$eid))

[1] FALSE

* theDT = data.table(eid, ename, desig, dept\_id, salary)
* theDT

eid ename desig dept\_id salary 1: 2501 Rizwan sales 10 23000

2: 2502 Ajay accounts 10 35000

3: 2503 Mounav manager 10 40000

4: 2504 Charan sales 10 80000

5: 2505 Srinivas sales 20 230000

6: 2506 Vamsi accounts 20 98000

7: 2507 Deepak accounts 20 50000

8: 2508 Abhi manager 30 85000

9: 2509 Pavan sales 30 130000

* class(theDT$ename)

[1] "character"

* class(employee1$ename)

[1] "character"

* theDT[1:2,]

eid ename desig dept\_id salary 1: 2501 Rizwan sales 10 23000

2: 2502 Ajay accounts 10 35000

* theDT[theDT$eid >= 2504]

eid ename desig dept\_id salary 1: 2504 Charan sales 10 80000

2: 2505 Srinivas sales 20 230000

3: 2506 Vamsi accounts 20 98000

4: 2507 Deepak accounts 20 50000

5: 2508 Abhi manager 30 85000

6: 2509 Pavan sales 30 130000

* theDT[, list(eid,desig)] eid desig

1: 2501 sales

2: 2502 accounts

3: 2503 manager

4: 2504 sales

5: 2505 sales

6: 2506 accounts

7: 2507 accounts

8: 2508 manager

9: 2509 sales

* theDT[, ename]

[1] "Rizwan" "Ajay" "Mounav" "Charan" "Srinivas" "Vamsi" "Deepak" "Abhi" "Pavan"

* theDT[, list(ename)] ename

1: Rizwan

2: Ajay

3: Mounav

4: Charan

5: Srinivas

6: Vamsi

7: Deepak

8: Abhi

9: Pavan

* theDT[, "ename", with = FALSE] ename

1: Rizwan

2: Ajay

3: Mounav

4: Charan

5: Srinivas

6: Vamsi

7: Deepak

8: Abhi

9: Pavan

* theDT[, c("ename","eid"), with = FALSE] ename eid

1: Rizwan 2501

2: Ajay 2502

3: Mounav 2503

4: Charan 2504

5: Srinivas 2505

6: Vamsi 2506

7: Deepak 2507

8: Abhi 2508

9: Pavan 2509

* tables()

NAME NROW NCOL MB COLS KEY

1: theDT 9 5 0 eid,ename,desig,dept\_id,salary Total: 0MB

* setkey(theDT, salary)
* theDT

eid ename desig dept\_id salary 1: 2501 Rizwan sales 10 23000

2: 2502 Ajay accounts 10 35000

3: 2503 Mounav manager 10 40000

4: 2507 Deepak accounts 20 50000

5: 2504 Charan sales 10 80000

6: 2508 Abhi manager 30 85000

7: 2506 Vamsi accounts 20 98000

8: 2509 Pavan sales 30 130000

9: 2505 Srinivas sales 20 230000

* key(theDT)

[1] "salary"

* tables()

NAME NROW NCOL MB COLS KEY

1: theDT 9 5 0 eid,ename,desig,dept\_id,salary salary

Total: 0MB

* theDT[theDT$salary > 50000]

eid ename desig dept\_id salary 1: 2504 Charan sales 10 80000

2: 2508 Abhi manager 30 85000

3: 2506 Vamsi accounts 20 98000

4: 2509 Pavan sales 30 130000

5: 2505 Srinivas sales 20 230000

* setkey(theDT, desig, dept\_id, salary)
* tables()

NAME NROW NCOL MB COLS KEY

1: theDT 9 5 0 eid,ename,desig,dept\_id,salary desig,dept\_id,salary

Total: 0MB

* theDT

eid ename desig dept\_id salary 1: 2502 Ajay accounts 10 35000

2: 2507 Deepak accounts 20 50000

3: 2506 Vamsi accounts 20 98000

4: 2503 Mounav manager 10 40000

|  |  |
| --- | --- |
| 5: 2508 Abhi manager | 30 85000 |
| 6: 2501 Rizwan sales | 10 23000 |
| 7: 2504 Charan sales | 10 80000 |
| 8: 2505 Srinivas sales | 20 230000 |
| 9: 2509 Pavan sales | 30 130000 |

* theDT[, mean(salary), by = dept\_id]

|  |  |  |
| --- | --- | --- |
| dept\_id | | V1 |
| 1: | 10 44500 | |
| 2: | 20 126000 | |
| 3: | 30 107500 | |

* theDT[, list(price = mean(salary)), by = dept\_id] dept\_id price

1: 10 44500

2: 20 126000

3: 30 107500

>

* theDT[, mean(salary), by = list(dept\_id,desig)] dept\_id desig V1

1: 10 accounts 35000

2: 20 accounts 74000

3: 10 manager 40000

4: 30 manager 85000

5: 10 sales 51500

6: 20 sales 230000

7: 30 sales 130000

* theDT[, list(price = mean(salary), desig = mean(eid)), by = dept\_id]

dept\_id price desig

|  |  |
| --- | --- |
| 1: | 10 44500 2502.5 |
| 2: | 20 126000 2506.0 |
| 3: | 30 107500 2508.5 |

**Result :** Successfully Completed the Aim

**R LAB – 07**

**R LAB – 07**

**Task – 01 :**

**Aim :** Apply cbind , rbind , cast , melt functions on any inbuilt dataset in R soft

- Ware .

**Program :**

# package installations require("plyr") library("plyr")

install.packages("data.table") require("data.table") library("data.table")

install.packages("reshape2") require("reshape2") library("reshape2")

sport <- c("Hockey", "Baseball", "Football") league <- c("NHL", "MLB", "NFL")

trophy <- c("Stanley Cup", "Commissioner’s Trophy", "Vince Lombardi Trophy")

trophies1 <- cbind(sport,league,trophy) # vectors will become as columns

trophies0 <- rbind(sport,league,trophy) # vectors become as rows

trophies0 trophies1 class(trophies1)

trophies2<- data.frame(sport=c("Basketball", "Golf"), league=c("NBA", "PGA"),

trophy=c("Larry Championship Trophy", "Wanamaker Trophy"), stringsAsFactors=FALSE)

trophies2

trophies <- rbind(trophies1, trophies2) trophies

# Doing the JOINS getwd()

setwd("C:/Users/itadmin/Documents/") getwd()

df1 = data.frame(StudentId = c(101:106), Product = c("Hindi", "English",

"Maths", "Science", "Political Science", "Physics"))

df1

df2 = data.frame(StudentId = c(102, 104, 106,

107, 108),

df2

State = c("Manglore", "Mysore", "Pune", "Dehradun", "Delhi"))

df = merge(x = df1, y = df2, by = "StudentId") df

# left outer

df3 = merge(x = df1, y = df2, by = "StudentId", all.x = TRUE)

df3

# right outer

df4 = merge(x = df1, y = df2, by = "StudentId", all.y = TRUE)

df4

# full join

df5 = merge(x = df1, y = df2, by = "StudentId", all = TRUE)

df5

# cross join

df6 = merge(x = df1, y = df2, by = NULL) df6

# semi join (doubt topic) install.packages("dplyr") library(dplyr)

df7 = df1 %>% semi\_join(df2, by = "StudentId") df7

# anti join (doubt topic)

df8 = df1 %>% anti\_join(df2, by = "StudentId") df8

# doing CAST and MELT

# 1. CAST : transforming rows into columns # 2. MELT : transforming columns into row

# ships data set is a default dat set in the r packages. install.packages("MASS") install.packages("reshape")

library("MASS") library("reshape") library("reshape2")

ships

sd = (head(ships, n = 10)) # taking as data frame class(sd)

# in id attribute putting the type and year column constant molten\_ships = melt(sd, id = c("type","year")) molten\_ships

rec <- reshape2::dcast(molten\_ships, type+year~variable,sum) rec

**Output :**

sport <- c("Hockey", "Baseball", "Football")

* league <- c("NHL", "MLB", "NFL")
* trophy <- c("Stanley Cup", "Commissioner’s Trophy", "Vince Lombardi Trophy")
* trophies1 <- cbind(sport,league,trophy) # vectors will become as columns
* trophies0 <- rbind(sport,league,trophy) # vectors become as rows
* trophies0

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

sport "Hockey" "Baseball" "Football" league "NHL" "MLB" "NFL"

trophy "Stanley Cup" "Commissioner’s Trophy" "Vince Lombardi Trophy"

* trophies1

sport league trophy

[1,] "Hockey" "NHL" "Stanley Cup"

[2,] "Baseball" "MLB" "Commissioner’s Trophy" [3,] "Football" "NFL" "Vince Lombardi Trophy"

* class(trophies1)

[1] "matrix" "array"

trophies2<- data.frame(sport=c("Basketball", "Golf"), league=c("NBA", "PGA"),

+ trophy=c("Larry Championship Trophy", "Wanamaker Trophy"), stringsAsFactors=FALSE)

* trophies2

sport league trophy

1. Basketball NBA Larry Championship Trophy
2. Golf PGA Wanamaker Trophy

>

* trophies <- rbind(trophies1, trophies2)
* trophies

sport league trophy

1. Hockey NHL Stanley Cup
2. Baseball MLB Commissioner’s Trophy
3. Football NFL Vince Lombardi Trophy
4. Basketball NBA Larry Championship Trophy
5. Golf PGA Wanamaker Trophy

getwd()

[1] "E:/venkat sai/rstudio\_language"

* setwd(choose.dir())
* getwd()

[1] "E:/venkat sai/rstudio\_language"

df1 = data.frame(StudentId = c(101:106),

+ Product = c("Hindi", "English",

+ "Maths", "Science",

+ "Political Science",

+ "Physics"))

* df1

StudentId Product

|  |  |  |
| --- | --- | --- |
| 1 | 101 | Hindi |
| 2 | 102 | English |
| 3 | 103 | Maths |
| 4 | 104 | Science |

1. 105 Political Science
2. 106 Physics

>

* df2 = data.frame(StudentId = c(102, 104, 106,

+ 107, 108),

+ State = c("Manglore", "Mysore",

+ "Pune", "Dehradun", "Delhi"))

* df2

StudentId State

1. 102 Manglore
2. 104 Mysore
3. 106 Pune
4. 107 Dehradun
5. 108 Delhi

df = merge(x = df1, y = df2, by = "StudentId")

* df

StudentId Product State

1. 102 English Manglore
2. 104 Science Mysore
3. 106 Physics Pune

>

* # left outer
* df3 = merge(x = df1, y = df2, by = "StudentId",

+ all.x = TRUE)

* df3

StudentId Product State

1 101 Hindi <NA>

|  |  |  |
| --- | --- | --- |
| 2 | 102 | English Manglore |
| 3 | 103 | Maths <NA> |
| 4 | 104 | Science Mysore |

1. 105 Political Science <NA>
2. 106 Physics Pune

>

* # right outer
* df4 = merge(x = df1, y = df2, by = "StudentId",

+ all.y = TRUE)

* df4

StudentId Product State

1. 102 English Manglore
2. 104 Science Mysore
3. 106 Physics Pune
4. 107 <NA> Dehradun
5. 108 <NA> Delhi

>

* # full join
* df5 = merge(x = df1, y = df2, by = "StudentId",

+ all = TRUE)

* df5

StudentId Product State

1 101 Hindi <NA>

|  |  |  |
| --- | --- | --- |
| 2 | 102 | English Manglore |
| 3 | 103 | Maths <NA> |
| 4 | 104 | Science Mysore |
| 5 | 105 Political Science <NA> | |
| 6 | 106 Physics Pune | |
| 7 | 107 <NA> Dehradun | |
| 8 | 108 <NA> Delhi | |
| > |  | |

* # cross join
* df6 = merge(x = df1, y = df2, by = NULL)
* df6

StudentId.x Product StudentId.y State

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 101 | Hindi | 102 Manglore |
| 2 | 102 | English | 102 Manglore |
| 3 | 103 | Maths | 102 Manglore |
| 4 | 104 | Science | 102 Manglore |
| 5  6 | 105 Poli  106 | tical Science Physics | 102 Manglor  102 Manglore |
| 7 | 101 | Hindi | 104 Mysore |
| 8 | 102 | English | 104 Mysore |
| 9 | 103 | Maths | 104 Mysore |
| 10 | 104 | Science | 104 Mysore |

e

11 105 Political Science 104 Mysore

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12 | 106 | Physics | 104 | Mysore | |
| 13 | 101 | Hindi | 106 | Pune | |
| 14 | 102 | English | 106 | Pune | |
| 15 | 103 | Maths | 106 | Pune | |
| 16 | 104 | Science | 106 | Pune | |
| 17 | 105 Political Science | | 106 | | Pune |
| 18 | 106 Physics | | 106 | | Pune |
| 19 | 101 | Hindi | 107 Dehradun | | |
| 20 | 102 | English | 107 Dehradun | | |
| 21 | 103 | Maths | 107 Dehradun | | |
| 22 | 104 | Science | 107 Dehradun | | |

1. 105 Political Science 107 Dehradun
2. 106 Physics 107 Dehradun

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 25 | 101 | Hindi | 108 | Delhi | |
| 26 | 102 | English | 108 | Delhi | |
| 27 | 103 | Maths | 108 | Delhi | |
| 28 | 104 | Science | 108 | Delhi | |
| 29 | 105 Political Science | | 108 | | Delhi |
| 30 | 106 Physics | | 108 | | Delhi |

df7 = df1 %>% semi\_join(df2, by = "StudentId")

* df7

StudentId Product

1. 102 English
2. 104 Science
3. 106 Physics

>

* # anti join (doubt topic)
* df8 = df1 %>% anti\_join(df2, by = "StudentId")
* df8

StudentId Product

|  |  |  |
| --- | --- | --- |
| 1 | 101 | Hindi |
| 2 | 103 | Maths |

3 105 Political Science

library("reshape2")

* ships

type year period service incidents 1 A 60 60 127 0

2 A 60 75 63 0

3 A 65 60 1095 3

4 A 65 75 1095 4

5 A 70 60 1512 6

6 A 70 75 3353 18

7 A 75 60 0 0

8 A 75 75 2244 11

9 B 60 60 44882 39

10 B 60 75 17176 29

11 B 65 60 28609 58

12 B 65 75 20370 53

13 B 70 60 7064 12

14 B 70 75 13099 44

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 B | 75 | 60 | 0 | 0 |
| 16 B | 75 | 75 | 7117 | 18 |
| 17 C | 60 | 60 | 1179 | 1 |
| 18 C | 60 | 75 | 552 | 1 |
| 19 C | 65 | 60 | 781 | 0 |
| 20 C | 65 | 75 | 676 | 1 |
| 21 C | 70 | 60 | 783 | 6 |
| 22 C | 70 | 75 | 1948 | 2 |
| 23 C | 75 | 60 | 0 | 0 |
| 24 C | 75 | 75 | 274 | 1 |
| 25 D | 60 | 60 | 251 | 0 |
| 26 D | 60 | 75 | 105 | 0 |
| 27 D | 65 | 60 | 288 | 0 |
| 28 D | 65 | 75 | 192 | 0 |
| 29 D | 70 | 60 | 349 | 2 |
| 30 D | 70 | 75 | 1208 | 11 |
| 31 D | 75 | 60 | 0 | 0 |
| 32 D | 75 | 75 | 2051 | 4 |
| 33 E | 60 | 60 | 45 | 0 |
| 34 E | 60 | 75 | 0 | 0 |
| 35 E | 65 | 60 | 789 | 7 |
| 36 E | 65 | 75 | 437 | 7 |
| 37 E | 70 | 60 | 1157 | 5 |
| 38 E | 70 | 75 | 2161 | 12 |
| 39 E | 75 | 60 | 0 | 0 |
| 40 E | 75 | 75 | 542 | 1 |

* sd = (head(ships, n = 10)) # taking as data frame
* class(sd)

[1] "data.frame"

molten\_ships = melt(sd, id = c("type","year"))

* molten\_ships

type year variable value

1. A 60 period 60
2. A 60 period 75
3. A 65 period 60
4. A 65 period 75
5. A 70 period 60
6. A 70 period 75
7. A 75 period 60
8. A 75 period 75
9. B 60 period 60
10. B 60 period 75
11. A 60 service 127
12. A 60 service 63
13. A 65 service 1095
14. A 65 service 1095
15. A 70 service 1512
16. A 70 service 3353
17. A 75 service 0
18. A 75 service 2244
19. B 60 service 44882

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | B | 60 | service 17176 | |
| 21 A | | 60 incidents | | 0 |
| 22 A | | 60 incidents | | 0 |
| 23 A | | 65 incidents | | 3 |
| 24 A | | 65 incidents | | 4 |
| 25 A | | 70 incidents | | 6 |
| 26 A | | 70 incidents | | 18 |
| 27 A | | 75 incidents | | 0 |
| 28 A | | 75 incidents | | 11 |
| 29 B | | 60 incidents | | 39 |
| 30 B | | 60 incidents | | 29 |

rec <- dcast(molten\_ships, type+year~variable,sum)

* rec

type year period service incidents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 A | 60 | 135 | 190 | 0 |
| 2 A | 65 | 135 | 2190 | 7 |
| 3 A | 70 | 135 | 4865 | 24 |
| 4 A | 75 | 135 | 2244 | 11 |
| 5 B | 60 | 135 | 62058 | 68 |

**Result :** Successfully executed all lines in the program…

**R LAB – 08**

**Task – 01 :**

**Aim :** Apply R math functions in a data or any dataset

### Program :

# Doing Math With Simulation R

# Computing the probabilites of the given vector exact\_one = function(p)

{

notp = 1 - p tot = 0.0

for(i in 1:length(p))

{

tot = tot + p[i] + prod(notp[-i])

}

return(tot)

}

v1 = c(1,2,3,4,5)

a1 = exact\_one(v1)

a1

# Cumulative Sum and Product Of an Vector cumsum(v1)

cumprod(v1)

# minima and maxima

mat1 <- matrix(c(1,5,6,2,3,2), nrow = 3) mat1

min(mat1[,1], mat1[,2])

pmin(mat1[,1], mat1[,2]) pmin(mat1[1,], mat1[2,], mat1[3,])

# Sorting in R

unsort = c(13,2,5,2,3) sort(unsort)

unsort

order(unsort) # it will get the indices of the sorted values in the original vector

# sorting the dataframe v2 = c('def', 'ab', 'zzzz') v3 = c(2,5,1)

y = data.frame(v1 = v2, v2 = v3) y

r <- order(y$v2) r

z <- y[r,] z

rank(v3) rank(v1)

# Linear ALgebra

crossprod(1:3,c(5,12,13)) # 1\*5 + 2\*12 + 3\*13 = 68

# Matrix Multiplication

a = matrix(1:4, ncol = 2, byrow = TRUE) a

b = matrix(c(1,0,-1,1), nrow = 2) b

mat\_mul = a %\*% b mat\_mul

# solving the equations

s1 = matrix(c(1,1,-1,1), nrow = 2, ncol = 2) s2 = c(2,4)

solve(s1,s2) solve(s1)

# matrix Transpose

q1 = matrix(1:9, nrow = 3, byrow = TRUE) q1

t(q1)

# determinent of matrix det(q1)

# finding the eigen values eigen(q1)

# Set operations in R p <- c(1,2,5)

q <- c(5,1,8,9)

union(p,q) intersect(p,q)

setdiff(p,q) setdiff(q,p)

setequal(p,q) setequal(p, c(1,2,5))

5 %in% p

10 %in% q choose(5,3)

# Finding Symmetric Difference symetric = function(a,b)

{

sx = setdiff(a,b) sy = setdiff(b,a)

result = union(sx,sy)

return(result)

}

f = c(1,2,5)

g = c(5,1,8,9)

ans = symetric(f,g) ans

### Output :

exact\_one = function(p)

+ {

+ notp = 1 - p

+ tot = 0.0

+

+ for(i in 1:length(p))

+ {

+ tot = tot + p[i] + prod(notp[-i])

+ }

+

+ return(tot)

+ }

>

* v1 = c(1,2,3,4,5)
* a1 = exact\_one(v1)
* a1 [1] 39
* cumsum(v1) [1] 1 3 6 10 15
* cumprod(v1)

[1] 1 2 6 24 120

* mat1 <- matrix(c(1,5,6,2,3,2), nrow = 3)
* mat1

[,1] [,2]

|  |  |  |
| --- | --- | --- |
| [1,] | 1 | 2 |
| [2,] | 5 | 3 |
| [3,] | 6 | 2 |

* min(mat1[,1], mat1[,2])

[1] 1

* pmin(mat1[,1], mat1[,2]) [1] 1 3 2
* pmin(mat1[1,], mat1[2,], mat1[3,]) [1] 1 2
* unsort = c(13,2,5,2,3)
* sort(unsort) [1] 2 2 3 5 13
* unsort

[1] 13 2 5 2 3

* order(unsort) [1] 2 4 5 3 1
* v2 = c('def', 'ab', 'zzzz')
* v3 = c(2,5,1)
* y = data.frame(v1 = v2, v2 = v3)
* y

v1 v2

1. def 2
2. ab 5 3 zzzz 1

* r <- order(y$v2)
* r

[1] 3 1 2

>

* z <- y[r,]
* z

v1 v2 3 zzzz 1

1. def 2
2. ab 5

* rank(v3) [1] 2 3 1
* rank(v1) [1] 1 2 3 4 5
* crossprod(1:3,c(5,12,13)) [,1]

[1,] 68

* a = matrix(1:4, ncol = 2, byrow = TRUE)
* a

[,1] [,2]

[1,] 1 2

[2,] 3 4

* b = matrix(c(1,0,-1,1), nrow = 2)
* b

[,1] [,2]

|  |  |  |
| --- | --- | --- |
| [1,] | 1 | -1 |
| [2,] | 0 | 1 |
| > |  |  |

* mat\_mul = a %\*% b
* mat\_mul [,1] [,2]

|  |  |  |
| --- | --- | --- |
| [1,] | 1 | 1 |
| [2,] | 3 | 1 |

* s1 = matrix(c(1,1,-1,1), nrow = 2, ncol = 2)
* s2 = c(2,4)
* solve(s1,s2) [1] 3 1
* solve(s1) [,1] [,2]

[1,] 0.5 0.5

[2,] -0.5 0.5

* q1 = matrix(1:9, nrow = 3, byrow = TRUE)
* q1

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

[1,] 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
| [2,] | 4 | 5 | 6 |
| [3,] | 7 | 8 | 9 |

* t(q1)

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

|  |  |  |  |
| --- | --- | --- | --- |
| [1,] | 1 | 4 | 7 |
| [2,] | 2 | 5 | 8 |
| [3,] | 3 | 6 | 9 |

* det(q1)

[1] 6.661338e-16

* eigen(q1)

eigen() decomposition

$values

[1] 1.611684e+01 -1.116844e+00 -1.303678e-15

$vectors

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

[1,] -0.2319707 -0.78583024 0.4082483

[2,] -0.5253221 -0.08675134 -0.8164966

[3,] -0.8186735 0.61232756 0.4082483

* p <- c(1,2,5)
* q <- c(5,1,8,9)
* union(p,q) [1] 1 2 5 8 9
* intersect(p,q) [1] 1 5

>

* setdiff(p,q)

[1] 2

* setdiff(q,p) [1] 8 9

>

* setequal(p,q)

[1] FALSE

* setequal(p, c(1,2,5))

[1] TRUE

>

* 5 %in% p

[1] TRUE

* 10 %in% q

[1] FALSE

>

* choose(5,3) [1] 10
* symetric = function(a,b)

+ {

+ sx = setdiff(a,b)

+ sy = setdiff(b,a)

+ result = union(sx,sy)

+

+ return(result)

+ }

* f = c(1,2,5)
* g = c(5,1,8,9)
* ans = symetric(f,g)
* ans [1] 2 8 9

**Result :** Successfully Executed the all lines in R