Practical No. 6 Date: / / 201

Aim: Program to perform matrix operations on Star War Box Office figures.

Objectives:

- To study R Matrices & its operations.
- Implement a program to perform operations on Star War Box Office figures.

Theory:

R Matrix

Matrices can be considered as natural extensions of vectors. Similar to vectors, contain elements of the same atomic types. But, elements are arranged in a two-dimensional (row & column) layout. A matrix may contain numerical, character, or logical values. Usually, matrices containing numeric elements are used in mathematical calculations.

Creating Matrix

Syntax

```
matrix(data, nrow, ncol, byrow, dimnames)
```

- data: input vector elements of matrix
- nrow: number of rows in matrix
- ncol: number of columns in matrix
- byrow: If TRUE, arranges input vector elements row-wise
- dimname: names of rows & columns

Examples

```
matrix(1:6, nrow = 2)
     [,1] [,2] [,3]
[1,]  1     3    5
[2,]  2     4    6
matrix(1:6, ncol = 3)
     [,1] [,2] [,3]
[1,]  1     3    5
[2,]  2     4    6
matrix(1:6, nrow = 2, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
```

Examples (Recycling)

```
matrix(1:3, nrow = 2, ncol= 3)
```

```
     [,1] [,2] [,3]
[1,]  1     3    2
[2,]  2     1    3
matrix(1:4, nrow = 2, ncol= 3)
     [,1] [,2] [,3]
[1,]  1     3    1
[2,]  2     4    2
Warning message:
In matrix(1:4, nrow = 2, ncol = 3):
data length [4] is not a sub-multiple or multiple of the number of columns [3]
```

Augmenting Matrix

By Adding New Rows

```
M <- matrix(1:6, nrow = 2, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
rbind(M, 7:9)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
[3,]  7     8    9
```

By Adding New Columns

```
M <- matrix(1:6, nrow = 2, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
cbind(M, c(7,8))
     [,1] [,2] [,3] [,4]
[1,]  1     2    3    7
[2,]  4     5    6    8
```

Naming Matrix

rownames() Function

```
M1 <- matrix(1:6, nrow = 2, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
```

```
rownames(M) <- c("row#1", "row#2")
     [,1] [,2] [,3]
row#1  1    2    3
row#2  4    5    6
```

colnames() Function

```
M1 <- matrix(1:6, nrow = 2, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
colnames(M) <- c("col#1", "col#2", "col#3")
    col#1 col#2 col#3
[1,]   1    2    3
[2,]   4    5    6
```

Using dimnames in matrix() Function

```
M1 <- matrix(1:6, nrow = 2, byrow = TRUE,
           dimnames = list(c("row#1", "row#2"),
                           c("col#1", "col#2", "col#3")))
    col#1 col#2 col#3
row#1  1    2    3
row#2  4    5    6
```

Accessing (Subsetting) Matrix

Matrices are extensions of vectors, So as methods to access elements are. Element(s) of a matrix can be accessed by using the column and row index of the element(s). Accessing Single Element or Row/Column returns vector. Accessing Multiple Elements or Rows/Columns can returns vector or sub-matrix. Element(s) can also be accessed by name(s) of Row(s)/Column(s). Accessing can also be done by using Logical Values. If required recycling is performed by R.

Single Element

```
M <- matrix(1:9, nrow = 3, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
[3,]  7     8    9


M[1,3]
[1]  3
M[3,2]
[1]  8
```

Single Row/Column

```
M <- matrix(1:9, nrow = 3, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
[3,]  7     8    9

M[3,]
[1]  7    8    9

M[,3]
[1]  3    6    9

M[3]
[1]  7    #Element at index 3 w.r.t. whole matrix
```

Multiple Elements or Rows/Columns

```
M <- matrix(1:9, nrow = 3, byrow = TRUE)
     [,1] [,2] [,3]
[1,]  1     2    3
[2,]  4     5    6
[3,]  7     8    9

M[2, c(2,3)]
[1]  5    6

M[c(1,2), c(2,3)]
     [,1] [,2]
[1,]  2     3
[2,]  5     6
```

Using Row-names & Column-names

```
M <- matrix(1:12, nrow = 3, byrow = TRUE)
rownames(M) <- c('R#1', 'R#2', 'R#3')
colnames(M) <- c('C#1', 'C#2', 'C#3', 'C#4')
      C#1  C#2  C#3 C#4
 R#1  1     2    3   4
 R#2  5     6    7   8
 R#3  9    10   11  12
```

```
M[“R#1”, “C#3”]
[1]  3

M[2, “C#4”]
[1]  8          #Combination of index & name is allowed
```

Using Logical Value

```
M <- matrix(1:12, nrow = 3, byrow = TRUE)
rownames(M) <- c(‘R#1’, ‘R#2’, ‘R#3’)
colnames(M) <- c(‘C#1’, ‘C#2’, ‘C#3’, ‘C#4’)
     C#1   C#2  C#3 C#4
 R#1  1     2    3   4
 R#2  5     6    7   8
 R#3  9    10   11  12

M[c(TRUE,FALSE,FALSE),c(FALSE,FALSE,TRUE,FALSE)]
[1]  3

M[c(F,T,F),c(F,F,T,F)]
[1]  7              #Using Logical Values requires c() function
Using Logical Values
M[c(F,F,T),c(F,T,F,T)]
M[c(F,F,T),c(F,T)]
“C#2”    “C#4”
 10       12   #c(F,T) is recycled to c(F,T,F,T)

M[c(T,F,T), c(T,F,T,T)]
      C#1  C#3  C#4
R#1   1     3    4
R#3   9    11   12
```

Matrix Arithmetic

Various mathematical operations (+, -, *, /, ^) can be performed on the matrices using the R arithmetic operators. Similar to vector in matrix arithmetic operations are done element-wise. Operations can be done with single value vector, multiple value vector or matrix. Result of an arithmetic operation is a matrix. Recycling is performed whenever necessary. In addition, two matrix specific functions are

- colSums(): calculate sum of each column element.
- rowSums(): calculate sum of each row element.

Both functions returns results as a vector.

With Single Value

```
M <- matrix(1:6, nrow = 3)
rownames(M) <- c("R#1", "R#2", "R#3")
colnames(M) <- c("C#1", "C#2")
     C#1   C#2
 R#1  1     4
 R#2  2     5
 R#3  3     6

M + 10
      C#1  C#2
 R#1  11    14
 R#2  12    15
 R#3  13    16
```

With Multiple Values (Recycling)

```
M <- matrix(1:6, nrow = 3)
rownames(M) <- c("R#1", "R#2", "R#3")
colnames(M) <- c("C#1", "C#2")
     C#1   C#2
 R#1  1     4
 R#2  2     5
 R#3  3     6

M + c(10,20,30)
      C#1  C#2
 R#1  11    14
 R#2  22    25
 R#3  33    36  #Vector is always recycled column-wise
```

rowSums() & colSums() Functions

```
M <- matrix(1:6, nrow = 3)
rownames(M) <- c("R#1", "R#2", "R#3")
colnames(M) <- c("C#1", "C#2")
     C#1   C#2
 R#1  1     4
 R#2  2     5
 R#3  3     6
```

```
colSums(M)
 C#1 C#2
  6  15
rowSums(M)
 R#1 R#2 R#3
  5   7   9
```

Algorithm

1. Start.
2. Create a matrix “Boxoffice”.
3. Name matrix rows & column appropriately.
4. Read choice for matrix operations from menu as
    a. Add Row
    b. Add Column
    c. Access Rows
    d. Access Columns
    e. Rowsum
    f. Colsum
    g. Display
5. As per choice perform matrix operations as
    a. If choice is “a”, add new row to existing matrix.
    b. If choice is “b”, add new column to existing matrix.
    c. If choice is “c”, access single/multiple rows from matrix.
    d. If choice is “d”, access single/multiple columns from matrix.
    e. If choice is “e”, add contents of any one row.
    f. If choice is “f”, add contents of any one column.
    g. If choice is “g”, display contents of matrix.
6. Stop.