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

The basic syntax for creating a matrix in R is −

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

Following is the description of the parameters used −

* **data** is the input vector which becomes the data elements of the matrix.
* **nrow** is the number of rows to be created.
* **ncol** is the number of columns to be created.
* **byrow** is a logical clue. If TRUE then the input vector elements are arranged by row.
* **dimname** is the names assigned to the rows and columns.

Example

Create a matrix taking a vector of numbers as input

# Elements are arranged sequentially by row.

M <- matrix(c(3:14), nrow = 4, byrow = TRUE)

print(M)

# Elements are arranged sequentially by column.

N <- matrix(c(3:14), nrow = 4, byrow = FALSE)

print(N)

# Define the column and row names.

rownames = c("row1", "row2", "row3", "row4")

colnames = c("col1", "col2", "col3")

P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))

print(P)

When we execute the above code, it produces the following result −

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

[1,] 3 4 5

[2,] 6 7 8

[3,] 9 10 11

[4,] 12 13 14

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

[1,] 3 7 11

[2,] 4 8 12

[3,] 5 9 13

[4,] 6 10 14

col1 col2 col3

row1 3 4 5

row2 6 7 8

row3 9 10 11

row4 12 13 14

Accessing Elements of a Matrix

Elements of a matrix can be accessed by using the column and row index of the element. We consider the matrix P above to find the specific elements below.

# Define the column and row names.

rownames = c("row1", "row2", "row3", "row4")

colnames = c("col1", "col2", "col3")

# Create the matrix.

P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))

# Access the element at 3rd column and 1st row.

print(P[1,3])

# Access the element at 2nd column and 4th row.

print(P[4,2])

# Access only the 2nd row.

print(P[2,])

# Access only the 3rd column.

print(P[,3])

When we execute the above code, it produces the following result −

[1] 5

[1] 13

col1 col2 col3

6 7 8

row1 row2 row3 row4

5 8 11 14

Matrix Computations

Various mathematical operations are performed on the matrices using the R operators. The result of the operation is also a matrix.

The dimensions (number of rows and columns) should be same for the matrices involved in the operation.

Matrix Addition & Subtraction

# Create two 2x3 matrices.

matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)

print(matrix1)

matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)

print(matrix2)

# Add the matrices.

result <- matrix1 + matrix2

cat("Result of addition","\n")

print(result)

# Subtract the matrices

result <- matrix1 - matrix2

cat("Result of subtraction","\n")

print(result)

When we execute the above code, it produces the following result −

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

[1,] 3 -1 2

[2,] 9 4 6

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

[1,] 5 0 3

[2,] 2 9 4

Result of addition

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

[1,] 8 -1 5

[2,] 11 13 10

Result of subtraction

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

[1,] -2 -1 -1

[2,] 7 -5 2

Matrix Multiplication & Division

# Create two 2x3 matrices.

matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)

print(matrix1)

matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)

print(matrix2)

# Multiply the matrices.

result <- matrix1 \* matrix2

cat("Result of multiplication","\n")

print(result)

# Divide the matrices

result <- matrix1 / matrix2

cat("Result of division","\n")

print(result)

When we execute the above code, it produces the following result −

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

[1,] 3 -1 2

[2,] 9 4 6

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

[1,] 5 0 3

[2,] 2 9 4

Result of multiplication

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

[1,] 15 0 6

[2,] 18 36 24

Result of division

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

[1,] 0.6 -Inf 0.6666667

[2,] 4.5 0.4444444 1.5000000

**Matrix Construction**

There are various ways to construct a matrix. When we construct a matrix directly with data elements, the matrix content is filled along the column orientation by default. For example, in the following code snippet, the content of B is filled along the columns consecutively.

> B = matrix(   
+   c(2, 4, 3, 1, 5, 7),   
+   nrow=3,   
+   ncol=2)   
   
> B             # B has 3 rows and 2 columns   
     [,1] [,2]   
[1,]    2    1   
[2,]    4    5   
[3,]    3    7

**Transpose**

We construct the **transpose** of a matrix by interchanging its columns and rows with the function t .

> t(B)          # transpose of B   
     [,1] [,2] [,3]   
[1,]    2    4    3   
[2,]    1    5    7

**Combining Matrices**

The columns of two matrices having the same number of rows can be combined into a larger matrix. For example, suppose we have another matrix C also with 3 rows.

> C = matrix(   
+   c(7, 4, 2),   
+   nrow=3,   
+   ncol=1)   
   
> C             # C has 3 rows   
     [,1]   
[1,]    7   
[2,]    4   
[3,]    2

Then we can combine the columns of B and C with cbind.

> cbind(B, C)   
     [,1] [,2] [,3]   
[1,]    2    1    7   
[2,]    4    5    4   
[3,]    3    7    2

Similarly, we can combine the rows of two matrices if they have the same number of columns with the rbind function.

> D = matrix(   
+   c(6, 2),   
+   nrow=1,   
+   ncol=2)   
   
> D             # D has 2 columns   
     [,1] [,2]   
[1,]    6    2   
   
> rbind(B, D)   
     [,1] [,2]   
[1,]    2    1   
[2,]    4    5   
[3,]    3    7   
[4,]    6    2

**Deconstruction**

We can deconstruct a matrix by applying the c function, which combines all column vectors into one.

> c(B)   
[1] 2 4 3 1 5 7