R Matrix: (Create and Modify Matrix, and Access Matrix Elements)

R Matrix (Create and Modify Matrix, and Access Matrix Elements)

In this article, you will learn to work with matrix in R. You will learn to create and modify matrix, and access matrix elements.

Matrix is a two dimensional data structure in R programming.

Matrix is similar to vector but additionally contains the dimension attribute. All attributes of an object can be checked with the `attributes()` function (dimension can be checked directly with the `dim()` function).

We can check if a variable is a matrix or not with the `class()` function.

```
> a
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> class(a)
[1] "matrix"
> attributes(a)
$dim
[1] 3 3
> dim(a)
[1] 3 3
```

How to create a matrix in R programming?

Matrix can be created using the `matrix()` function.

Dimension of the matrix can be defined by passing appropriate value for arguments `nrow` and `ncol`.

Providing value for both dimension is not necessary. If one of the dimension is provided, the other is inferred from length of the data.

```
> matrix(1:9, nrow = 3, ncol = 3)
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> # same result is obtained by providing only one dimension
> matrix(1:9, nrow = 3)
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

We can see that the matrix is filled column-wise. This can be reversed to row-wise filling by passing `TRUE` to the argument `byrow`.

```
> matrix(1:9, nrow=3, byrow=TRUE)    # fill matrix row-wise
[,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

In all cases, however, a matrix is stored in column-major order internally as we will see in the subsequent sections.

It is possible to name the rows and columns of matrix during creation by passing a 2 element list to the argument `dimnames`.

```
> x <- matrix(1:9, nrow = 3, dimnames = list(c("X","Y","Z"), c("A","B","C")))
> x
A B C
X 1 4 7
Y 2 5 8
Z 3 6 9
```

These names can be accessed or changed with two helpful functions `colnames()` and `rownames()`.

```
> colnames(x)
[1] "A" "B" "C"
> rownames(x)
[1] "X" "Y" "Z"
> # It is also possible to change names
> colnames(x) <- c("C1","C2","C3")
> rownames(x) <- c("R1","R2","R3")
> x
C1 C2 C3
R1  1  4  7
R2  2  5  8
R3  3  6  9
```

Another way of creating a matrix is by using functions `cbind()` and `rbind()` as in column bind and row bind.

```
> cbind(c(1,2,3),c(4,5,6))
[,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> rbind(c(1,2,3),c(4,5,6))
[,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Finally, you can also create a matrix from a vector by setting its dimension using `dim()`.

```
> x <- c(1,2,3,4,5,6)
> x
[1] 1 2 3 4 5 6
> class(x)
[1] "numeric"
> dim(x) <- c(2,3)
> x
[,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> class(x)
[1] "matrix"
```

How to access Elements of a matrix?

We can access elements of a matrix using the square bracket `[` indexing method. Elements can be accessed as `var[row, column]`. Here `rows` and `columns` are vectors.

Using integer vector as index

We specify the row numbers and column numbers as vectors and use it for indexing.

If any field inside the bracket is left blank, it selects all.

We can use negative integers to specify rows or columns to be excluded.

```
> x
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> x[c(1,2),c(2,3)]    # select rows 1 & 2 and columns 2 & 3
[,1] [,2]
[1,]    4    7
[2,]    5    8
> x[c(3,2),]    # leaving column field blank will select entire columns
[,1] [,2] [,3]
[1,]    3    6    9
[2,]    2    5    8
> x[,]    # leaving row as well as column field blank will select entire matrix
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> x[-1,]    # select all rows except first
[,1] [,2] [,3]
[1,]    2    5    8
[2,]    3    6    9
```

One thing to notice here is that, if the matrix returned after indexing is a row matrix or column matrix, the result is given as a vector.

```
> x[1,]
[1] 1 4 7
> class(x[1,])
[1] "integer"
```

This behavior can be avoided by using the argument `drop = FALSE` while indexing.

```
> x[1,,drop=FALSE]  # now the result is a 1X3 matrix rather than a vector
[,1] [,2] [,3]
[1,]    1    4    7
> class(x[1,,drop=FALSE])
[1] "matrix"
```

It is possible to index a matrix with a single vector.

While indexing in such a way, it acts like a vector formed by stacking columns of the matrix one after another. The result is returned as a vector.

```
> x
[,1] [,2] [,3]
[1,]    4    8    3
[2,]    6    0    7
[3,]    1    2    9
> x[1:4]
[1] 4 6 1 8
> x[c(3,5,7)]
[1] 1 0 3
```

Using logical vector as index

Two logical vectors can be used to index a matrix. In such situation, rows and columns where the value is `TRUE` is returned. These indexing vectors are recycled if necessary and can be mixed with integer vectors.

```
> x
[,1] [,2] [,3]
[1,]    4    8    3
[2,]    6    0    7
[3,]    1    2    9
> x[c(TRUE,FALSE,TRUE),c(TRUE,TRUE,FALSE)]
[,1] [,2]
[1,]    4    8
[2,]    1    2
> x[c(TRUE,FALSE),c(2,3)]    # the 2 element logical vector is recycled to 3 element vector
[,1] [,2]
[1,]    8    3
[2,]    2    9
```

It is also possible to index using a single logical vector where recycling takes place if necessary.

```
> x[c(TRUE, FALSE)]
[1] 4 1 0 3 9
```

In the above example, the matrix `x` is treated as vector formed by stacking columns of the matrix one after another, i.e., `(4,6,1,8,0,2,3,7,9)`.

The indexing logical vector is also recycled and thus alternating elements are selected. This property is utilized for filtering of matrix elements as shown below.

```
> x[x>5]    # select elements greater than 5
[1] 6 8 7 9
> x[x%%2 == 0]    # select even elements
[1] 4 6 8 0 2
```

Using character vector as index

Indexing with character vector is possible for matrix with named row or column. This can be mixed with integer or logical indexing.

```
> x
A B C
[1,] 4 8 3
[2,] 6 0 7
[3,] 1 2 9
> x[,"A"]
[1] 4 6 1
> x[TRUE,c("A","C")]
A C
[1,] 4 3
[2,] 6 7
[3,] 1 9
> x[2:3,c("A","C")]
A C
[1,] 6 7
[2,] 1 9
```

How to modify a matrix in R?

We can combine assignment operator with the above learned methods for accessing elements of a matrix to modify it.

```
> x
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> x[2,2] <- 10; x    # modify a single element
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2   10    8
[3,]    3    6    9
> x[x<5] <- 0; x    # modify elements less than 5
[,1] [,2] [,3]
[1,]    0    0    7
[2,]    0   10    8
[3,]    0    6    9
```

A common operation with matrix is to transpose it. This can be done with the function `t()`.

```
> t(x)    # transpose a matrix
[,1] [,2] [,3]
[1,]    0    0    0
[2,]    0   10    6
[3,]    7    8    9
```

We can add row or column using `rbind()` and `cbind()` function respectively. Similarly, it can be removed through reassignment.

```
> cbind(x, c(1, 2, 3))    # add column
[,1] [,2] [,3] [,4]
[1,]    0    0    7    1
[2,]    0   10    8    2
[3,]    0    6    9    3
> rbind(x,c(1,2,3))    # add row
[,1] [,2] [,3]
[1,]    0    0    7
[2,]    0   10    8
[3,]    0    6    9
[4,]    1    2    3
> x <- x[1:2,]; x    # remove last row
[,1] [,2] [,3]
[1,]    0    0    7
[2,]    0   10    8
```

Dimension of matrix can be modified as well, using the `dim()` function.

```
> x
[,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> dim(x) <- c(3,2); x    # change to 3X2 matrix
[,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> dim(x) <- c(1,6); x    # change to 1X6 matrix
[,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    4    5    6
```