R Arrays

In R, arrays are the data objects which allow us to store data in more than two dimensions. In R, an array is created with the help of the **array()** function. This array() function takes a vector as an input and to create an array it uses vectors values in the **dim** parameter.

**For example**- if we will create an array of dimension (2, 3, 4) then it will create 4 rectangular matrices of 2 row and 3 columns.

R Array Syntax

There is the following syntax of R arrays:

1. array\_name **<-** array(data, dim= (row\_size, column\_size, matrices, dim\_names))

**data**

The data is the first argument in the array() function. It is an input vector which is given to the array.

**matrices**

In R, the array consists of multi-dimensional matrices.

**row\_size**

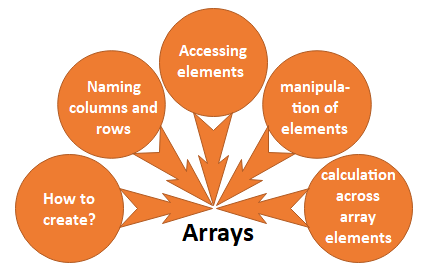
This parameter defines the number of row elements which an array can store.

**column\_size**

This parameter defines the number of columns elements which an array can store.

**dim\_names**

This parameter is used to change the default names of rows and columns.



How to create?

In R, array creation is quite simple. We can easily create an array using vector and array() function. In array, data is stored in the form of the matrix. There are only two steps to create a matrix which are as follows

1. In the first step, we will create two vectors of different lengths.
2. Once our vectors are created, we take these vectors as inputs to the array.

Let see an example to understand how we can implement an array with the help of the vectors and array() function.

**Example**

1. #Creating two vectors of different lengths
2. vec1 **<-c**(1,3,5)
3. vec2 **<-c**(10,11,12,13,14,15)
5. #Taking these vectors as input to the array
6. res **<-** array(c(vec1,vec2),dim=c(3,3,2))
7. print(res)

**Output**

, , 1

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

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

, , 2

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

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

Naming rows and columns

In R, we can give the names to the rows, columns, and matrices of the array. This is done with the help of the dim name parameter of the array() function.

It is not necessary to give the name to the rows and columns. It is only used to differentiate the row and column for better understanding.

Below is an example, in which we create two arrays and giving names to the rows, columns, and matrices.

**Example**

1. #Creating two vectors of different lengths
2. vec1 **<-c**(1,3,5)
3. vec2 **<-c**(10,11,12,13,14,15)
5. #Initializing names for rows, columns and matrices
6. col\_names **<-** c("Col1","Col2","Col3")
7. row\_names **<-** c("Row1","Row2","Row3")
8. matrix\_names **<-** c("Matrix1","Matrix2")
10. #Taking the vectors as input to the array
11. res **<-** array(c(vec1,vec2),dim=c(3,3,2),dimnames=list(row\_names,col\_names,matrix\_names))
12. print(res)

**Output**

, , Matrix1

Col1 Col2 Col3

Row1 1 10 13

Row2 3 11 14

Row3 5 12 15

, , Matrix2

Col1 Col2 Col3

Row1 1 10 13

Row2 3 11 14

Row3 5 12 15

Accessing array elements

Like C or C++, we can access the elements of the array. The elements are accessed with the help of the index. Simply, we can access the elements of the array with the help of the indexing method. Let see an example to understand how we can access the elements of the array using the indexing method.

**Example**

1. , , Matrix1
2. Col1 Col2 Col3
3. Row1    1   10   13
4. Row2    3   11   14
5. Row3    5   12   15
7. , , Matrix2
8. Col1 Col2 Col3
9. Row1    1   10   13
10. Row2    3   11   14
11. Row3    5   12   15
13. Col1 Col2 Col3
14. 5   12   15
16. [1] 13
18. Col1 Col2 Col3
19. Row1    1   10   13
20. Row2    3   11   14
21. Row3    5   12   15

Manipulation of elements

The array is made up matrices in multiple dimensions so that the operations on elements of an array are carried out by accessing elements of the matrices.

**Example**

1. #Creating two vectors of different lengths
2. vec1 **<-c**(1,3,5)
3. vec2 **<-c**(10,11,12,13,14,15)
5. #Taking the vectors as input to the array1
6. res1 **<-** array(c(vec1,vec2),dim=c(3,3,2))
7. print(res1)
9. #Creating two vectors of different lengths
10. vec1 **<-c**(8,4,7)
11. vec2 **<-c**(16,73,48,46,36,73)
13. #Taking the vectors as input to the array2
14. res2 **<-** array(c(vec1,vec2),dim=c(3,3,2))
15. print(res2)
17. #Creating matrices from these arrays
18. mat1 **<-** res1[,,2]
19. mat2 **<-** res2[,,2]
20. res3 **<-** mat1+mat2
21. print(res3)

**Output**

, , 1

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

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

, , 2

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

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

, , 1

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

[1,] 8 16 46

[2,] 4 73 36

[3,] 7 48 73

, , 2

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

[1,] 8 16 46

[2,] 4 73 36

[3,] 7 48 73

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

[1,] 9 26 59

[2,] 7 84 50

[3,] 12 60 88

Calculations across array elements

For calculation purpose, r provides **apply()** function. This apply function contains three parameters i.e., x, margin, and function.

This function takes the array on which we have to perform the calculations. The basic syntax of the apply() function is as follows:

1. apply(x, margin, fun)

Here, x is an array, and a margin is the name of the dataset which is used and fun is the function which is to be applied to the elements of the array.

**Example**

1. #Creating two vectors of different lengths
2. vec1 **<-c**(1,3,5)
3. vec2 **<-c**(10,11,12,13,14,15)
5. #Taking the vectors as input to the array1
6. res1 **<-** array(c(vec1,vec2),dim=c(3,3,2))
7. print(res1)
9. #using apply function
10. result **<-** apply(res1,c(1),sum)
11. print(result)

**Output**

, , 1

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

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

, , 2

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

[1,] 1 10 13

[2,] 3 11 14

[3,] 5 12 15

[1] 48 56 64