Arrays are essential data storage structures defined by a fixed number of dimensions. Arrays are used for the allocation of space at contiguous memory locations.

In R Programming Language Uni-dimensional arrays are called vectors with the length being their only dimension. Two-dimensional arrays are called matrices, consisting of fixed numbers of rows and columns. R Arrays consist of all elements of the same data type. Vectors are supplied as input to the function and then create an array based on the number of dimensions.

**Creating an Array**

An R array can be created with the use of **array()** the function. A list of elements is passed to the array() functions along with the dimensions as required.

**Syntax:**

*array(data, dim = (nrow, ncol, nmat), dimnames=names)*

***where***

*nrow: Number of rows*

*ncol : Number of columns*

*nmat: Number of matrices of dimensions nrow \* ncol*

*dimnames : Default value = NULL.*

Otherwise, a list has to be specified which has a name for each component of the dimension. Each component is either a null or a vector of length equal to the dim value of that corresponding dimension.

**Uni-Dimensional Array**

A vector is a uni-dimensional array, which is specified by a single dimension, length. A Vector can be created using ‘**c()**‘ function. A list of values is passed to the c() function to create a vector.

vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

print (vec1)

*# cat is used to concatenate*

*# strings and print it.*

cat ("Length of vector : ", length(vec1))

**Multi-Dimensional Array**

A two-dimensional matrix is an array specified by a fixed number of rows and columns, each containing the same data type. A matrix is created by using **array()** function to which the values and the dimensions are passed.

*# arranges data from 2 to 13*

*# in two matrices of dimensions 2x3*

arr = array(2:13, dim = c(2, 3, 2))

print(arr)

Vectors of different lengths can also be fed as input into the **array()** function. However, the total number of elements in all the vectors combined should be equal to the number of elements in the matrices. The elements are arranged in the order in which they are specified in the function.

vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

vec2 <- c(10, 11, 12)

*# elements are combined into a single vector,*

*# vec1 elements followed by vec2 elements.*

arr = array(c(vec1, vec2), dim = c(2, 3, 2))

print (arr)

**Dimension of the Array**

We will use dim function to find out the dimension of the R array.

*# for multi dimension array*

arr = array(2:13, dim = c(2, 3, 2))

dim(arr)

This specifies the dimensions of the R array. In this case, we are creating a 3D array with dimensions 2x3x2. The first dimension has size 2, the second dimension has size 3, and the third dimension has size 2.

**Naming of Arrays**

The row names, column names and matrices names are specified as a vector of the number of rows, number of columns and number of matrices respectively. By default, the rows, columns and matrices are named by their index values.

row\_names <- c("row1", "row2")

col\_names <- c("col1", "col2", "col3")

mat\_names <- c("Mat1", "Mat2")

*# the naming of the various elements*

*# is specified in a list and*

*# fed to the function*

arr = array(2:14, dim = c(2, 3, 2),

dimnames = list(row\_names,

col\_names, mat\_names))

print (arr)

**Accessing arrays**

The R arrays can be accessed by using indices for different dimensions separated by commas. Different components can be specified by any combination of elements’ names or positions.

**Accessing Uni-Dimensional Array**

The elements can be accessed by using indexes of the corresponding elements.

vec <- c(1:10)

*# accessing entire vector*

cat ("Vector is : ", vec)

*# accessing elements*

cat ("Third element of vector is : ", vec[3])

**Accessing entire matrices**

vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

vec2 <- c(10, 11, 12)

row\_names <- c("row1", "row2")

col\_names <- c("col1", "col2", "col3")

mat\_names <- c("Mat1", "Mat2")

arr = array(c(vec1, vec2), dim = c(2, 3, 2),

dimnames = list(row\_names,

col\_names, mat\_names))

arr

*# accessing matrix 1 by index value*

print ("Matrix 1")

print (arr[,,1])

**Accessing specific rows and columns of matrices**

Rows and columns can also be accessed by both names as well as indices.

vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

vec2 <- c(10, 11, 12)

row\_names <- c("row1", "row2")

col\_names <- c("col1", "col2", "col3")

mat\_names <- c("Mat1", "Mat2")

arr = array(c(vec1, vec2), dim = c(2, 3, 2),

dimnames = list(row\_names,

col\_names, mat\_names))

arr

*# accessing matrix 1 by index value*

print ("1st column of matrix 1")

print (arr[, 1, 1])

*# accessing matrix 2 by its name*

print ("2nd row of matrix 2")

print(arr["row2",,"Mat2"])

**Accessing elements individually**

Elements can be accessed by using both the row and column numbers or names.

vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

vec2 <- c(10, 11, 12)

row\_names <- c("row1", "row2")

col\_names <- c("col1", "col2", "col3")

mat\_names <- c("Mat1", "Mat2")

arr = array(c(vec1, vec2), dim = c(2, 3, 2),

dimnames = list(row\_names, col\_names, mat\_names))

*# accessing matrix 1 by index value*

print ("2nd row 3rd column matrix 1 element")

print (arr[2, "col3", 1])

*# accessing matrix 2 by its name*

print ("2nd row 1st column element of matrix 2")

print(arr["row2", "col1", "Mat2"])

**Accessing subset of array elements**

A smaller subset of the array elements can be accessed by defining a range of row or column limits.

row\_names <- c("row1", "row2")

col\_names <- c("col1", "col2", "col3", "col4")

mat\_names <- c("Mat1", "Mat2")

arr = array(1:15, dim = c(2, 4, 2),

dimnames = list(row\_names, col\_names, mat\_names))

arr

*# print elements of both the rows and columns 2 and 3 of matrix 1*

print (arr[, c(2, 3), 1])

**Adding elements to array**

Elements can be appended at the different positions in the array. The sequence of elements is retained in order of their addition to the array. The time complexity required to add new elements is O(n) where n is the length of the array. The length of the array increases by the number of element additions. There are various in-built functions available in R to add new values:

* **c(vector, values):** c() function allows us to append values to the end of the array. Multiple values can also be added together.
* **append(vector, values):** This method allows the values to be appended at any position in the vector. By default, this function adds the element at end. **append(vector, values, after=length(vector))** adds new values after specified length of the array specified in the last argument of the function.
* **Using the length function of the array:** Elements can be added at length+x indices where x>0.

*# creating a uni-dimensional array*

x <- c(1, 2, 3, 4, 5)

*# addition of element using c() function*

x <- c(x, 6)

print ("Array after 1st modification ")

print (x)

*# addition of element using append function*

x <- append(x, 7)

print ("Array after 2nd modification ")

print (x)

*# adding elements after computing the length*

len <- length(x)

x[len + 1] <- 8

print ("Array after 3rd modification ")

print (x)

*# adding on length + 3 index*

x[len + 3]<-9

print ("Array after 4th modification ")

print (x)

*# append a vector of values to the array after length + 3 of array*

print ("Array after 5th modification")

x <- append(x, c(10, 11, 12), after = length(x)+3)

print (x)

*# adds new elements after 3rd index*

print ("Array after 6th modification")

x <- append(x, c(-1, -1), after = 3)

print (x)