Arrays

In this lesson, we will learn what arrays are in R and how to create them.

We'll cover the following

In the last two lessons, we learned about vectors and lists. Both of these data structures are *one dimensional* meaning they have only **length** and no other dimension.

On the plus side, arrays in R language can be **n-dimensional**. However, like **vectors**, arrays can store only values having the same data type.

Creating Arrays

In R, arrays can be created using the `array()` keyword. Inputs to `array()` include **concatenated vectors** and the `dim` argument for creating an array.

Syntax for Creating Arrays

```
array(<dataVectors>, dim = <vectorOfDimensions>)
```

Optionally, we can also provide names for each dimension!

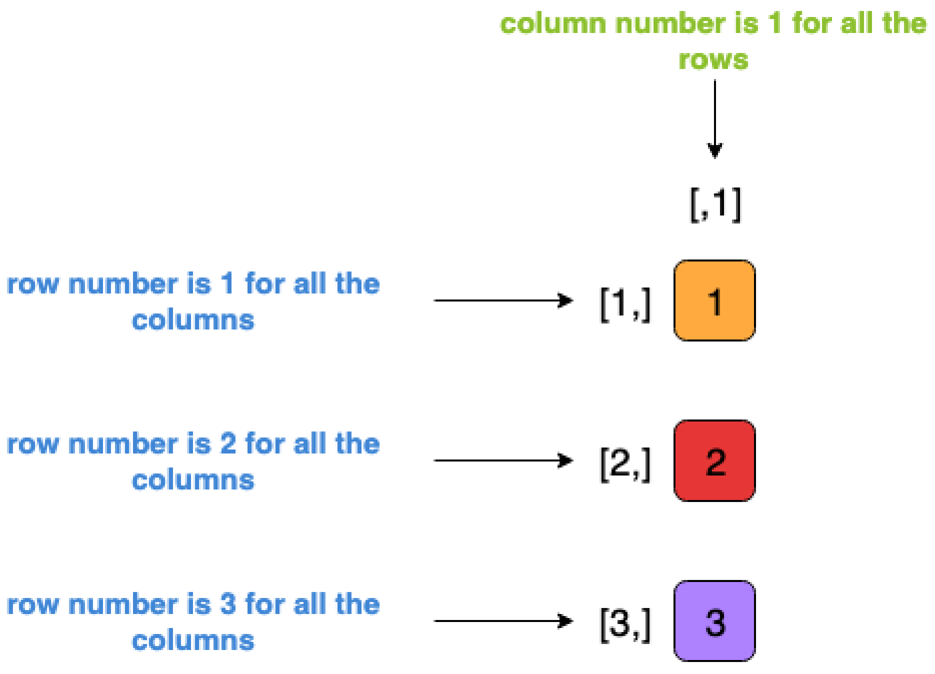
Let's begin with making a simple two-dimensional array with one vector.

```
myVector <- c (1, 2, 3) # create a vector
myArray <- array (myVector, dim = c(3, 1))
print (myArray)
```

m x 1 array

The dimensions of this array are $3x1$ where the number of rows is 3 and the number of columns is 1. This is also called a **single column matrix**.

Notice the row and column labels that are displayed on the console by using `print()` with an array.


column number is 1 for all the rows
[,1]
row number is 1 for all the columns
[1,]
1
row number is 2 for all the columns
[2,]
2
row number is 3 for all the columns
[3,]
3

Default row and column labels in an array

The illustration above shows the meaning of the row and column labels displayed when an array is printed.

Now let's code an array that has only one row but multiple columns:

```
myVector <- c (1, 2, 3) # create a vector
myArray <- array (myVector, dim = c(1,3))
print (myArray)
```

1xm array

The dimensions of this array are $1x3$ where the number of rows is 1 and the number of columns is 3. This is also called a **single row matrix**.

Now let's create an array of dimensions $2x2$:

```
myVector <- c (1, 2, 3, 4) # create a vector
```

```
myArray <- array (myVector, dim = c(2, 2))
print(myArray)
```

n x m array

Now let’s see how to create an array of **two** $3x3$ matrices each with 3 rows and 3 columns.

```
myVector1 <- c (1, 2, 3)
myVector2 <- c (4, 5, 6)
myVector3 <- c (7, 8, 9)

myArray <- array (c(myVector1, myVector2, myVector3), dim = c(3, 3, 2))
print (myArray)
```

Two matrices of dimensions 3x3

Here, in **line number 5**

```
dim = c(3, 3, 2)
# First argument is the number of rows, the second argument is the number of c
olumns and the third argument is the number of matrices.
```

There are **two** matrices (notice the last argument) of size $3x3$.

Now, let’s try making an array using vectors of different sizes.

```
myVector1 <- c (1, 2, 3)
myVector2 <- c (4, 5, 6, 7, 8, 9)

myArray <- array (c(myVector1, myVector2), dim = c(3, 3, 2))
print (myArray)
```

Creating array with vectors of different length

Accessing and Modifying Arrays

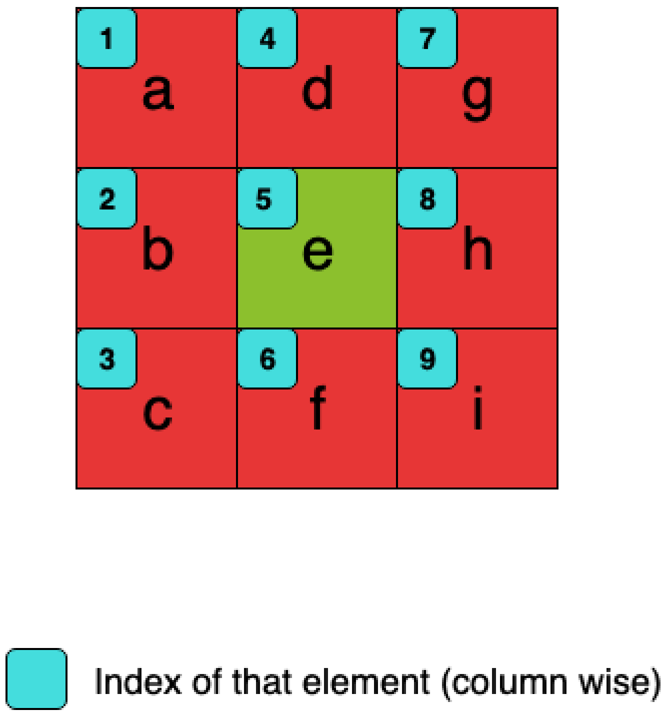
We can access a specific element of the array using the square brackets `[]`. Below is an example code to demonstrate how to access arrays.

```
myVector1 <- c (1, 2, 3)
myVector2 <- c (4, 5, 6, 7, 8, 9)

myArray <- array (c(myVector1, myVector2), dim = c(3, 3, 2))

# Accessing the element located at row 1, column 1 in the first matrix
print (myArray[1, 1, 1])

# Accessing the element located at row 2, column 3 in the second matrix
print (myArray[2, 3, 2])
```

Accessing elements in an array

We can access an element based on a single index as well. R language uses **column major** order which means if only a single index is used to access an array, then the first n indexes will belong to the first n rows of the first column.

Let’s have a look at an illustration:



Column major indexes

Let’s have a look at the code: In the illustration above `"e"` has index 5, and our code prints `"e"` when index 5 is accessed.

```
myVector <- c ("a", "b", "c", "d", "e", "f", "g", "h", "i")
```

```
myArray <- array (myVector, dim = c(3, 3, 1))
print(myArray[5]) # Uses column major for accessing this index
```

Modifying element at 1x1x1

We can also **modify** an element in an array using the two methods:

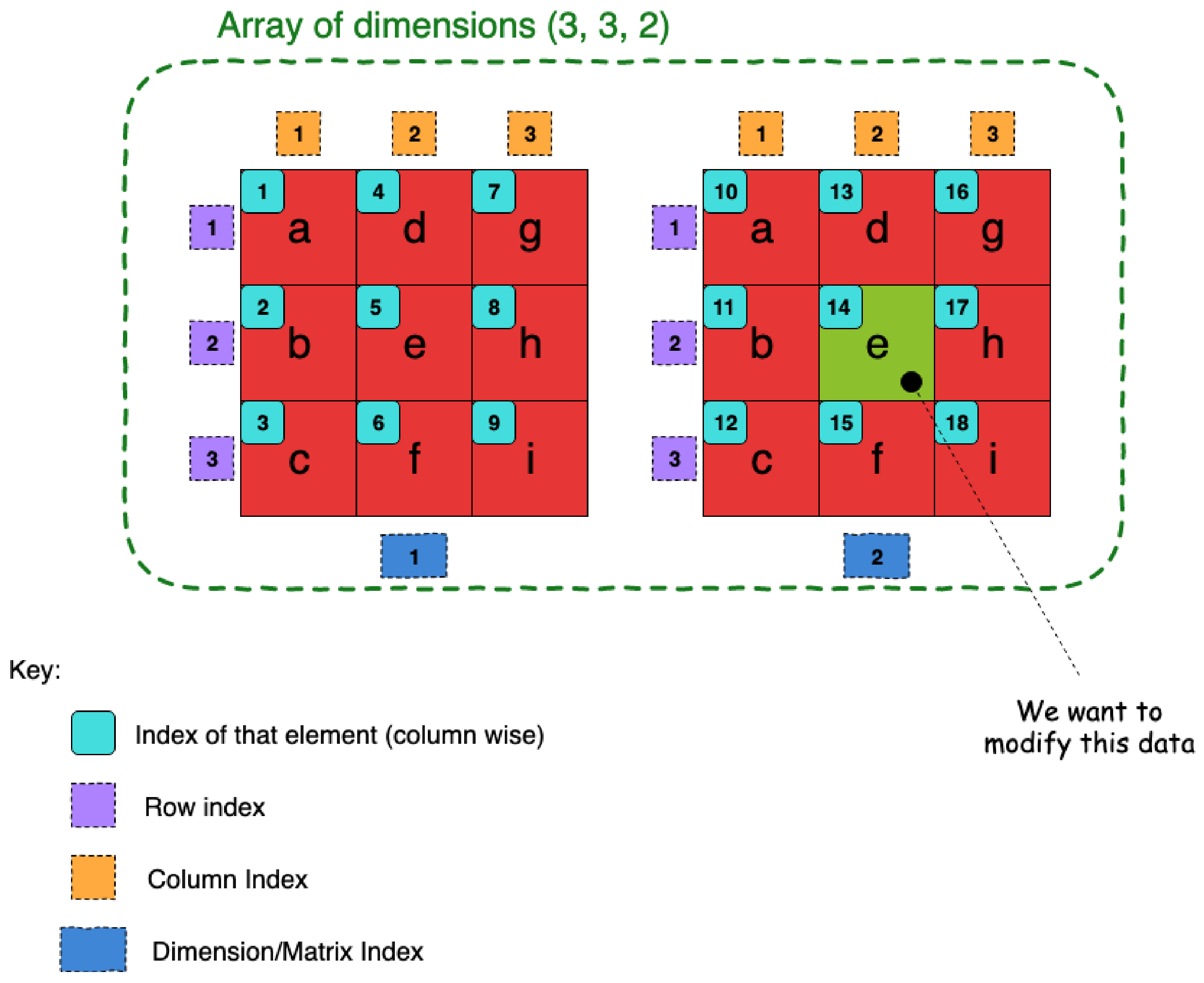


Illustration representing the index that needs to be modified

Using_Specific_Indexes | Using_Single_Index

```
myVector <- c ("a", "b", "c", "d", "e", "f", "g", "h", "i")
myArray <- array (myVector, dim = c(3, 3, 2))

print("Original Array:")
print(myArray)

myArray[2, 2, 2] = "E" # Modifying the element located at row 2, column 2 in the second matrix
print("Modified Array:")
print (myArray)
```

Modifying based on row and column indexes

In the next lesson, we will be learning about the R object: **matrix**.