Vectors

Here, we will learn what vectors are in R and how to create them.

We'll cover the following

- Creating Vectors
  - Creating Vectors by Concatenation
  - Converting Vectors to Strings
- Inserting Elements in a Vector
- Accessing and Modifying Vectors

A **Vector** is a basic data structure in R. It contains elements of the same type at each index. The data types can be

- Logical
- Integer
- Numeric
- Character
- Complex

Creating Vectors

The keyword `vector()` is used to create a vector of a **fixed type** and **fixed length**.

```
vector ("numeric", 5) # numeric vector with 0 at every index

vector ("complex", 5) # complex vector with 0+0i at every index

vector ("logical", 5) # logical vector with FALSE at every index

vector ("character", 5) # character vector with "" at every index
```

Vector of fixed type and fixed length.

> Using this method every index will now have a value of **zero**, **FALSE**, **null string** or something equivalent to *nothing* for the specified data type.

A vector’s type can be checked with `typeof()`, and the number of elements in the vector can be checked with `length()`.

We have already discussed these but let’s see their usage with vectors again.

typeof() length()

```
myNumericVector <- vector ("numeric", 5)
typeof(myNumericVector)

myComplexVector <- vector ("complex", 5)
typeof(myComplexVector)
```

Example of typeof() function with vectors

Now, what if we do not want our vector to be initialized by *nothing* but want to make specific initializations?

Creating Vectors by Concatenation

The `c()` function can be used to create vectors of objects by concatenating things together. By using this function, we can directly specify the content of the vector.

```
myRealNumericVector <- c(1, 2, 3, 4)            # numeric

myDecimalNumericVector <- c(0.1, 0.2, 0.3, 0.4)   # numeric

myLogiacalVector <- c(TRUE, FALSE)              # logical

myCharacterVector <- c("a", "b", "c")           # character

myIntegerVector <- 1:10                         # integer

myComplexVector <- c(1+1i, 2+2i)                # complex
```

Different types of vectors.

> You can also make a vector with just one value. R saves single values as a

vector of length 1.

```
myVector <- 5
is.vector(myVector)
length(myVector)
```

Vector with just one element.

In the above code snippet, the function `is.vector()` returns `true` if the variable is a vector and `false` otherwise.

> Here `is.vector()` is a built-in R function. We will be covering built-in functions in a later chapter.

Converting Vectors to Strings

Suppose we want to convert a vector of strings into a single string. We can do that by using `paste()`. We can use the argument `collapse` with `paste()` to concatenate strings in a vector and removing the quotation marks between them.

> The `collapse` parameter specifies the character to be used between individual vector elements.

collapse="." collapse=""

```
myVector <- c("learning", "is", "fun")
paste(myVector, collapse = ".")
```

paste() with collapse = "." argument

Inserting Elements in a Vector

We can add elements in a vector using the same `c()` method.

By using c() we can concatenate two vectors, in addition to inserting a number into a vector.

```
myVector <- c(1, 2, 3, 4)
cat("Original Vector: ")
print(myVector)

myVector <- c(0, myVector)
cat("Appending 0 at the start of the vector: ")
print(myVector)

myVector <- c(myVector, 5)
cat("Appending 5 at the end of the vector: ")
print(myVector)

tempVector <- c(6, 7, 8)
myVector <- c(myVector, tempVector)
cat("Appending another vector at the end of the original vector: ")
print(myVector)
```

Inserting element at the start and end of a vector.

Accessing and Modifying Vectors

We can fetch an element at a specific index in a vector by using the vector's name with square brackets `[]` around the specified index.

```
vectorName[<index>]
```

> Indexing starts at 1, which means that the first element of the vector is at index 1.

```
myVector <- c(0, 1, 2, 3)
print(myVector[1])
```

Fetching element at index 1.

We can also modify the value at a specific index of the vector.

```
myVector <- c(0, 1, 2, 3)
myVector[1] <- 5
print(myVector[1])
```

```
Modifying element at index 1.
```

Now that we have looked at **vectors**, let’s move on to **lists** in the next lesson.