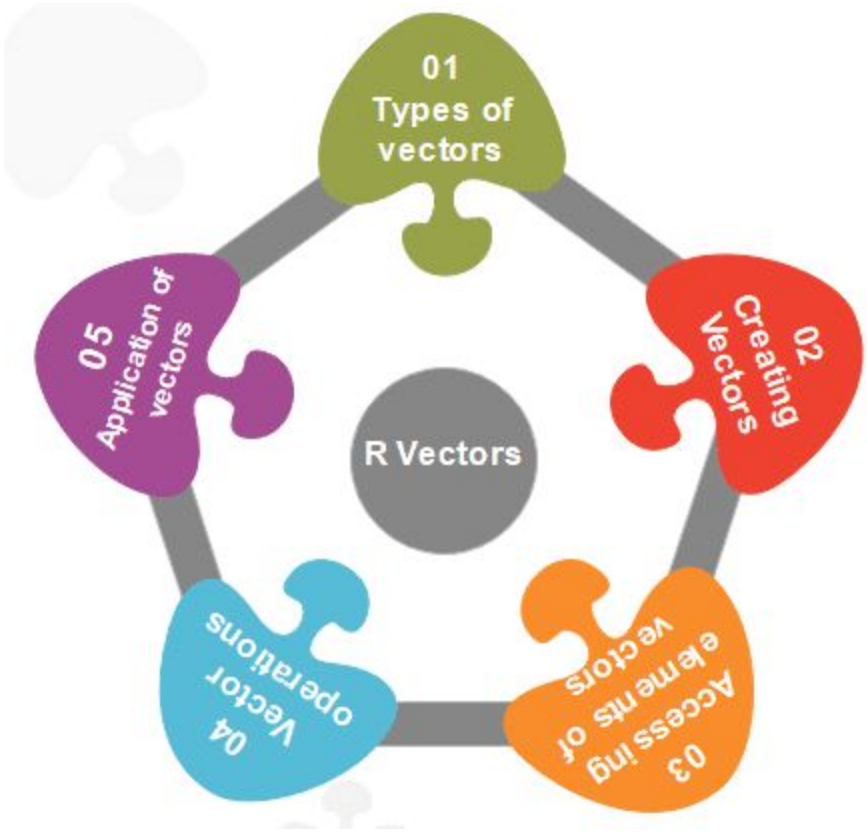
R Vector

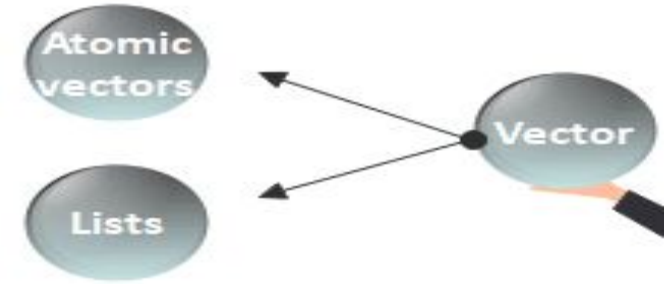
A **vector** is a basic data structure which plays an important role in R programming.

In R, a sequence of elements which share the same data type is known as vector. A vector supports logical, integer, double, character, complex, or raw data type. The elements which are contained in vector known as **components** of the vector. We can check the type of vector with the help of the **typeof()** function.



The length is an important property of a vector. A vector length is basically the number of elements in the vector, and it is calculated with the help of the length() function.

Vector is classified into two parts, i.e., **Atomic vectors** and **Lists**. They have three common properties, i.e., **function type, function length**, and **attribute function**.



There is only one difference between atomic vectors and lists. In an atomic vector, all the elements are of the same type, but in the list, the elements are of different data types.

How to create a vector in R?

In R, we use c() function to create a vector. This function returns a one-dimensional array or simply vector. The c() function is a generic function which combines its argument. All arguments are restricted with a common data type which is the type of the returned value. There are various other ways

to create a vector in R, which are as follows:

1) Using the colon(:) operator

We can create a vector with the help of the colon operator. There is the following syntax to use colon operator:

1. z**<-x:y**

This operator creates a vector with elements from x to y and assigns it to z.

Example:

```
1. a<-4:-10
2. a
```

Output

```
[1]  4  3  2  1  0  -1  -2  -3  -4  -5  -6  -7  -8  -9  -10
```

2) Using the seq() function

In R, we can create a vector with the help of the seq() function. A sequence function creates a sequence of elements as a vector. The seq() function is used in two ways, i.e., by setting step size with ?by' parameter or specifying the length of the vector with the 'length.out' feature.

Example:

```
1. seq_vec<-seq(1,4,by=0.5)
2. seq_vec
3. class(seq_vec)
```

Output

```
[1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0
```

Example:

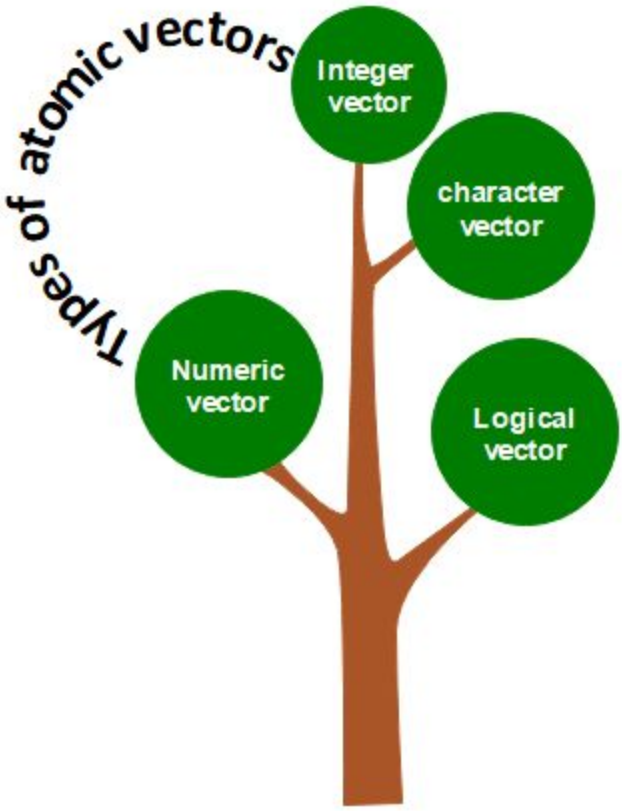
```
1. seq_vec<-seq(1,4,length.out=6)
2. seq_vec
3. class(seq_vec)
```

Output

```
[1]  1.0  1.6  2.2  2.8  3.4  4.0
[1]  "numeric"
```

Atomic vectors in R

In R, there are four types of atomic vectors. Atomic vectors play an important role in Data Science. Atomic vectors are created with the help of **c()** function. These atomic vectors are as follows:



Numeric vector

The decimal values are known as numeric data types in R. If we assign a decimal value to any variable d, then this d variable will become a numeric type. A vector which contains numeric elements is known as a numeric vector.

Example:

```
1. d<-45.5
2. num_vec<-c(10.1, 10.2, 33.2)
3. d
4. num_vec
5. class(d)
6. class(num_vec)
```

Output

```
[1]  45.5
[1]  10.1  10.2  33.2
[1]  "numeric"
[1]  "numeric"
```

Integer vector

A non-fraction numeric value is known as integer data. This integer data is represented by "Int." The Int size is 2 bytes and long Int size of 4 bytes. There is two way to assign an integer value to a variable, i.e., by using as.integer() function and appending of L to the value.

A vector which contains integer elements is known as an integer vector.

Example:

```
1. d<-as.integer(5)
2. e<-5L
3. int_vec<-c(1,2,3,4,5)
4. int_vec<-as.integer(int_vec)
5. int_vec1<-c(1L,2L,3L,4L,5L)
6. class(d)
7. class(e)
8. class(int_vec)
9. class(int_vec1)
```

Output

```
[1]  "integer"
[1]  "integer"
[1]  "integer"
[1]  "integer"
```

Character vector

A character is held as a one-byte integer in memory. In R, there are two different ways to create a character data type value, i.e., using as.character() function and by typing string between double quotes("") or single quotes('').

A vector which contains character elements is known as an integer vector.

Example:

```
1. d<-'shubham'
2. e<-"Arpita"
3. f<-65
4. f<-as.character(f)
5. d
6. e
7. f
8. char_vec<-c(1,2,3,4,5)
9. char_vec<-as.character(char_vec)
10. char_vec1<-c("shubham","arpita","nishka","vaishali")
11. char_vec
12. class(d)
13. class(e)
14. class(f)
15. class(char_vec)
16. class(char_vec1)
```

Output

```
[1]  "shubham"
[1]  "Arpita"
[1]  "65"
[1]  "1"   "2"   "3"   "4"   "5"
```

```
[1]  "shubham"   "arpita"   "nishka"   "vaishali"
[1]  "character"
[1]  "character"
[1]  "character"
[1]  "character"
[1]  "character"
```

Logical vector

The logical data types have only two values i.e., True or False. These values are based on which condition is satisfied. A vector which contains Boolean values is known as the logical vector.

Example:

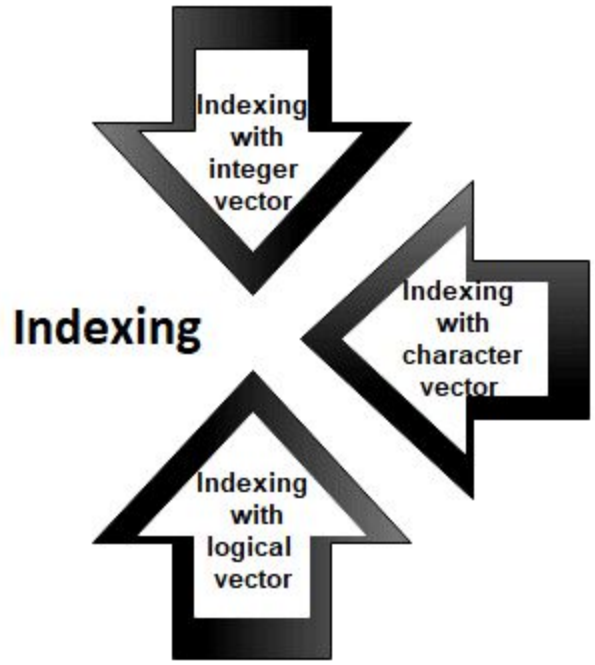
```
1. d<-as.integer(5)
2. e<-as.integer(6)
3. f<-as.integer(7)
4. g<-d>e
5. h<-e<f
6. g
7. h
8. log_vec<-c(d<e, d<f, e<d,e<f,f<d,f<e)
9. log_vec
10. class(g)
11. class(h)
12. class(log_vec)
```

Output

```
[1]  FALSE
[1]  TRUE
[1]  TRUE   TRUE   FALSE   TRUE   FALSE   FALSE
[1]  "logical"
[1]  "logical"
[1]  "logical"
```

Accessing elements of vectors

We can access the elements of a vector with the help of vector indexing. Indexing denotes the position where the value in a vector is stored. Indexing will be performed with the help of integer, character, or logic.



1) Indexing with integer vector

On integer vector, indexing is performed in the same way as we have applied in C, C++, and java. There is only one difference, i.e., in C, C++, and java the indexing starts from 0, but in R, the indexing starts from 1. Like other programming languages, we perform indexing by specifying an integer value in square braces [] next to our vector.

Example:

```
1. seq_vec<-seq(1,4,length.out=6)
2. seq_vec
3. seq_vec[2]
```

Output

```
[1] 1.0 1.6 2.2 2.8 3.4 4.0
[1] 1.6
```

2) Indexing with a character vector

In character vector indexing, we assign a unique key to each element of the vector. These keys are uniquely defined as each element and can be accessed very easily. Let's see an example to understand how it is performed.

Example:

```
1. char_vec<-c("shubham"=22,"arpita"=23,"vaishali"=25)
2. char_vec
3. char_vec["arpita"]
```

Output

```
shubham   arpita vaishali
  22      23     25
arpita
     23
```

3) Indexing with a logical vector

In logical indexing, it returns the values of those positions whose corresponding position has a logical vector TRUE. Let see an example to understand how it is performed on vectors.

Example:

```
1. a<-c(1,2,3,4,5,6)
2. a[c(TRUE,FALSE,TRUE,TRUE,FALSE,TRUE)]
```

Output

```
[1] 1 3 4 6
```

Vector Operation

In R, there are various operation which is performed on the vector. We can add, subtract, multiply or divide two or more vectors from each other. In data science, R plays an important role, and operations are required for data manipulation. There are the following types of operation which are performed on the vector.

1) Combining vectors

The c() function is not only used to create a vector, but also it is also used to combine two vectors. By combining one or more vectors, it forms a new vector which contains all the elements of each vector. Let see an example to see how c() function combines the vectors.

Example:

```
1. p<-c(1,2,4,5,7,8)
2. q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
3. r<-c(p,q)
```

Output

```
[1] "1"       "2"       "4"       "5"       "7"       "8"
[7] "shubham" "arpita"  "nishka"  "gunjan"  "vaishali" "sumit"
```

2) Arithmetic operations

We can perform all the arithmetic operation on vectors. The arithmetic operations are performed member-by-member on vectors. We can add, subtract, multiply, or divide two

vectors. Let see an example to understand how arithmetic operations are performed on vectors.

Example:

```
1. a<-c(1,3,5,7)
2. b<-c(2,4,6,8)
3. a+b
4. a-b
5. a/b
6. a%%b
```

Output

```
[1]  3  7 11 15
[1] -1 -1 -1 -1
[1]  2 12 30 56
[1] 0.5000000 0.7500000 0.8333333 0.8750000
[1] 1 3 5 7
```

3) Logical Index vector

With the help of the logical index vector in R, we can form a new vector from a given vector. This vector has the same length as the original vector. The vector members are TRUE only when the corresponding members of the original vector are included in the slice; otherwise, it will be false. Let see an example to understand how a new vector is formed with the help of logical index vector.

Example:

```
1. a<-c("Shubham","Arpita","Nishka","Vaishali","Sumit","Gunjan")
2. b<-c(TRUE,FALSE,TRUE,TRUE,FALSE,FALSE)
3. a[b]
```

Output

```
[1] "Shubham"   "Nishka"    "Vaishali"
```

4) Numeric Index

In R, we specify the index between square braces [ ] for indexing a numerical value. If our index is negative, it will return us all the values except for the index which we have specified. For example, specifying [-3] will prompt R to convert -3 into its absolute value and then search for the value which occupies that index.

Example:

```
1. q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
2. q[2]
3. q[-4]  excluding 4th position
4. q[15]
```

Output

```
[1] "arpita"
[1] "shubham"  "arpita"   "nishka"   "vaishali" "sumit"
[1] NA
```

5) Duplicate Index

An index vector allows duplicate values which means we can access one element twice in one operation. Let see an example to understand how duplicate index works.

Example:

```
1. q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
2. q[c(2,4,4,3)]
```

Output

```
[1] "arpita" "gunjan" "gunjan" "nishka"
```

6) Range Indexes

Range index is used to slice our vector to form a new vector. For slicing, we used colon(:) operator. Range indexes are very helpful for the situation involving a large operator. Let see an example to understand how slicing is done with the help of the colon operator to form a new vector.

Example:

```
1. q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
```

2. b<-q[2:5]
3. b

Output

[1] "arpita" "nishka" "gunjan" "vaishali"

7) Out-of-order Indexes

In R, the index vector can be out-of-order. Below is an example in which a vector slice with the order of first and second values reversed.

Example:

1. q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")b<-q[2:5]
2. q[c(2,1,3,4,5,6)]

Output

[1] "arpita" "shubham" "nishka" "gunjan" "vaishali" "sumit"

8) Named vectors members

We first create our vector of characters as:

1. z=c("TensorFlow","PyTorch")
2. z

Output

[1] "TensorFlow" "PyTorch"

Once our vector of characters is created, we name the first vector member as "Start" and the second member as "End" as:

1. names(z)=c("Start","End")
2. z

Output

Start End
"TensorFlow" "PyTorch"

We retrieve the first member by its name as follows:

```
1. z["Start"]
```

Output

```
Start
"TensorFlow"
```

We can reverse the order with the help of the character string index vector.

```
1. z[c("Second","First")]
```

Output

```
   Second        First
"PyTorch"    "TensorFlow"
```

Applications of vectors

1. In machine learning for principal component analysis vectors are used. They are extended to eigenvalues and eigenvector and then used for performing decomposition in vector spaces.
2. The inputs which are provided to the deep learning model are in the form of vectors. These vectors consist of standardized data which is supplied to the input layer of the neural network.
3. In the development of support vector machine algorithms, vectors are used.
4. Vector operations are utilized in neural networks for various operations like image recognition and text processing.