**What are Operators in R?**

**Operators** are the symbols directing the compiler to perform various kinds of operations between the operands. Operators simulate the various mathematical, logical, and decision operations performed on a set of Complex Numbers, Integers, and Numericals as input operands.

**R Operators**

R supports majorly four kinds of binary operators between a set of operands. In this article, we will see various types of **operators in R Programming language**and their usage.

**Types of the operator in R language**

* Arithmetic Operators
* Logical Operators
* Relational Operators
* Assignment Operators
* Miscellaneous Operators

## ****Arithmetic Operators****

Arithmetic Operators modulo using the specified operator between operands, which may be either scalar values, complex numbers, or vectors. The R operators are performed element-wise at the corresponding positions of the vectors.

### ****Addition operator (+)****

The values at the corresponding positions of both operands are added. Consider the following R operator snippet to add two vectors:

a <- c (1, 0.1)

b <- c (2.33, 4)

print (a+b)

**Subtraction Operator (-)**

The second operand values are subtracted from the first. Consider the following R operator snippet to subtract two variables:

a <- 6

b <- 8.4

print (a-b)

**Multiplication Operator (\*)**

The multiplication of corresponding elements of vectors and Integers are multiplied with the use of the ‘\*’ operator.

B= c(4,4)

C= c(5,5)

print (B\*C)

**Division Operator (/)**

The first operand is divided by the second operand with the use of the ‘/’ operator.

a <- 10

b <- 5

print (a/b)

**Power Operator (^)**

The first operand is raised to the power of the second operand.

a <- 4

b <- 5

print(a^b)

**Modulo Operator (%%)**

The remainder of the first operand divided by the second operand is returned.

a <- 4

b <- 5

print(a%%b)

The following R code illustrates the usage of all Arithmetic R operators.

*# R program to illustrate*

*# the use of Arithmetic operators*

vec1 <- c(0, 2)

vec2 <- c(2, 3)

*# Performing operations on Operands*

cat ("Addition of vectors :", vec1 + vec2, "\n")

cat ("Subtraction of vectors :", vec1 - vec2, "\n")

cat ("Multiplication of vectors :", vec1 \* vec2, "\n")

cat ("Division of vectors :", vec1 / vec2, "\n")

cat ("Modulo of vectors :", vec1 %% vec2, "\n")

cat ("Power operator :", vec1 ^ vec2)

**Logical Operators**

Logical Operators in R simulate element-wise decision operations, based on the specified operator between the operands, which are then evaluated to either a True or False boolean value. Any non-zero integer value is considered as a TRUE value, be it a complex or real number.

**Element-wise Logical AND operator (&)**

Returns True if both the operands are True.

list1 <- c(**TRUE**, 0.1)

list2 <- c(0,4+3i)

print(list1 & list2)

Any non zero integer value is considered as a TRUE value, be it complex or real number.

**Element-wise Logical OR operator (|)**

Returns True if either of the operands is True.

list1 <- c(**TRUE**, 0.1)

list2 <- c(0,4+3i)

print(list1|list2)

**NOT operator (!)**

A unary operator that negates the status of the elements of the operand.

list1 <- c(0,**FALSE**)

print(!list1)

**Logical AND operator (&&)**

Returns True if both the first elements of the operands are True.

list1 <- c(**TRUE**, 0.1)

list2 <- c(0,4+3i)

print(list1[1] && list2[1])

Compares just the first elements of both the lists.

**Logical OR operator (||)**

Returns True if either of the first elements of the operands is True.

list1 <- c(**TRUE**, 0.1)

list2 <- c(0,4+3i)

print(list1[1]||list2[1])

The following R code illustrates the usage of all Logical Operators in R:

*# R program to illustrate*

*# the use of Logical operators*

vec1 <- c(0,2)

vec2 <- c(**TRUE**,**FALSE**)

*# Performing operations on Operands*

cat ("Element wise AND :", vec1 & vec2, "\n")

cat ("Element wise OR :", vec1 | vec2, "\n")

cat ("Logical AND :", vec1[1] && vec2[1], "\n")

cat ("Logical OR :", vec1[1] || vec2[1], "\n")

cat ("Negation :", !vec1)

**Relational Operators**

The Relational Operators in R carry out comparison operations between the corresponding elements of the operands. Returns a boolean TRUE value if the first operand satisfies the relation compared to the second. A TRUE value is always considered to be greater than the FALSE.

**Less than (<)**

Returns TRUE if the corresponding element of the first operand is less than that of the second operand. Else returns FALSE.

list1 <- c(**TRUE**, 0.1,"apple")

list2 <- c(0,0.1,"bat")

print(list1<list2)

**Less than equal to (<=)**

Returns TRUE if the corresponding element of the first operand is less than or equal to that of the second operand. Else returns FALSE.

list1 <- c(**TRUE**, 0.1, "apple")

list2 <- c(**TRUE**, 0.1, "bat")

*# Convert lists to character strings*

list1\_char <- as.character(list1)

list2\_char <- as.character(list2)

*# Compare character strings*

print(list1\_char <= list2\_char)

**Greater than (>)**

Returns TRUE if the corresponding element of the first operand is greater than that of the second operand. Else returns FALSE.

list1 <- c(**TRUE**, 0.1, "apple")

list2 <- c(**TRUE**, 0.1, "bat")

print(list1\_char > list2\_char)

**Greater than equal to (>=)**

Returns TRUE if the corresponding element of the first operand is greater or equal to that of the second operand. Else returns FALSE.

list1 <- c(**TRUE**, 0.1, "apple")

list2 <- c(**TRUE**, 0.1, "bat")

print(list1\_char >= list2\_char)

**Not equal to (!=)**

Returns TRUE if the corresponding element of the first operand is not equal to the second operand. Else returns FALSE.

list1 <- c(**TRUE**, 0.1,'apple')

list2 <- c(0,0.1,"bat")

print(list1!=list2)

The following R code illustrates the usage of all Relational Operators in R:

*# R program to illustrate*

*# the use of Relational operators*

vec1 <- c(0, 2)

vec2 <- c(2, 3)

*# Performing operations on Operands*

cat ("Vector1 less than Vector2 :", vec1 < vec2, "\n")

cat ("Vector1 less than equal to Vector2 :", vec1 <= vec2, "\n")

cat ("Vector1 greater than Vector2 :", vec1 > vec2, "\n")

cat ("Vector1 greater than equal to Vector2 :", vec1 >= vec2, "\n")

cat ("Vector1 not equal to Vector2 :", vec1 != vec2, "\n")

**Assignment Operators**

Assignment Operators in R are used to assigning values to various data objects in R. The objects may be integers, vectors, or functions. These values are then stored by the assigned variable names. There are two kinds of assignment operators: Left and Right

**Left Assignment (<- or <<- or =)**

Assigns a value to a vector.

vec1 = c("ab", **TRUE**)

print (vec1)

**Right Assignment (-> or ->>)**

Assigns value to a vector.

c("ab", **TRUE**) ->> vec1

print (vec1)

The following R code illustrates the usage of all Relational Operators in R:

*# R program to illustrate*

*# the use of Assignment operators*

vec1 <- c(2:5)

c(2:5) ->> vec2

vec3 <<- c(2:5)

vec4 = c(2:5)

c(2:5) -> vec5

*# Performing operations on Operands*

cat ("vector 1 :", vec1, "\n")

cat("vector 2 :", vec2, "\n")

cat ("vector 3 :", vec3, "\n")

cat("vector 4 :", vec4, "\n")

cat("vector 5 :", vec5)

**Miscellaneous Operators**

Miscellaneous Operator are the mixed operators in R that simulate the printing of sequences and assignment of vectors, either left or right-handed.

**%in% Operator**

Checks if an element belongs to a list and returns a boolean value TRUE if the value is present  else FALSE.

val <- 0.1

list1 <- c(**TRUE**, 0.1,"apple")

print (val %in% list1)

Checks for the value 0.1 in the specified list. It exists, therefore, prints TRUE.

**%\*% Operator**

This operator is used to multiply a matrix with its transpose. Transpose of the matrix is obtained by interchanging the rows to columns and columns to rows. The number of columns of the first matrix must be equal to the number of rows of the second matrix. Multiplication of the matrix A with its transpose, B, produces a square matrix.   
Ar∗cxBc∗r−>Pr∗r  *Ar*∗*c*​*xBc*​∗*r*−>*Pr*∗*r*​

mat = matrix(c(1,2,3,4,5,6),nrow=2,ncol=3)

print (mat)

print( t(mat))

pro = mat %\*% t(mat)

print(pro)

The following R code illustrates the usage of all Miscellaneous Operators in R:

R

*# R program to illustrate*

*# the use of Miscellaneous operators*

mat <- matrix (1:4, nrow = 1, ncol = 4)

print("Matrix elements using : ")

print(mat)

product = mat %\*% t(mat)

print("Product of matrices")

print(product,)

cat ("does 1 exist in prod matrix :", "1" %in% product)