

BASUDEV GODABARI DEGREE COLLEGE, KESAIBAHAL

Department of Computer Science

“Self Study Module”

Module Details:

- **Class – 6th Semester**
- **Subject Name: COMPUTER SCIENCE**
- **Paper Name : Data Science**

UNIT-2: STRUCTURE

R Programming Basics: Overview of R, R data types and objects, reading and writing data, Control structures, functions, scoping rules, dates and times, Loop functions, debugging tools, Simulation, code profiling.

Learning Objective

OBJECTIVES: To learn emerging issues related to various fields of data science. • To understand the underlying principles of data science, exploring data analysis. •

You Can use the Following Learning Video link related to above topic:

https://youtu.be/_V8eKsto3Ug

You Can also use the following Books

Text Books

1. Rachel Schutt, Cathy O'Neil, "Doing Data Science: Straight Talk from the Frontline" by Schroff/O'Reilly, 2013.

Reference Books

1. Foster Provost, Tom Fawcett, "Data Science for Business" What You Need to Know About Data Mining and Data-Analytic Thinking by O'Reilly, 2013.
2. John W. Foreman, "Data Smart: Using data Science to Transform Information into Insight" by John Wiley & Sons, 2013.
3. Eric Segel, "Predictive Analytics: The Power to Predict who Will Click, Buy, Lie, or Die", 1st Edition, by Wiley, 2013.

<https://www.pdfdrive.com/>

R Programming Tutorial



R Programming Tutorial is designed for both beginners and professionals. Our tutorial provides all the basic and advanced concepts of data analysis and visualization.

R is a software environment which is used to analyze statistical information and graphical representation. R allows us to do modular programming using functions.

Our R tutorial includes all topics of R such as introduction, features, installation, rstudio ide, variables, datatypes, operators, if statement, vector, data handling, graphics, statistical modelling, etc. This programming language was named R, based on the first name letter of the two authors (Robert Gentleman and Ross Ihaka).

What is R Programming

"R is an interpreted computer programming language which was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand." The **R Development Core Team** currently develops R. It is also a software environment used to analyze **statistical information, graphical representation, reporting, and data modeling**. R is the implementation of the **S programming** language, which is combined with **lexical scoping semantics**.

R not only allows us to do branching and looping but also allows to do modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python, and FORTRAN languages to improve efficiency.

In the present era, R is one of the most important tool which is used by researchers, data analyst, statisticians, and marketers for retrieving, cleaning, analyzing, visualizing, and presenting data.

Features of R programming

R is a domain-specific programming language which aims to do data analysis. It has some unique features which make it very powerful. The most important arguably being the notation of vectors. These vectors allow us to perform a complex operation on a set of values in a single command. There are the following features of R programming:

1. It is a simple and effective programming language which has been well developed.
2. It is data analysis software.
3. It is a well-designed, easy, and effective language which has the concepts of user-defined, looping, conditional, and various I/O facilities.
4. It has a consistent and incorporated set of tools which are used for data analysis.
5. For different types of calculation on arrays, lists and vectors, R contains a suite of operators.
6. It provides effective data handling and storage facility.
7. It is an open-source, powerful, and highly extensible software.
8. It provides highly extensible graphical techniques.
9. It allows us to perform multiple calculations using vectors.
10. R is an interpreted language.

Why use R Programming?

There are several tools available in the market to perform data analysis. Learning new languages is time taken. The data scientist can use two excellent tools, i.e., R and Python. We may not have time to learn them both at the time when we get started to learn data science. Learning statistical modeling and algorithm is more important

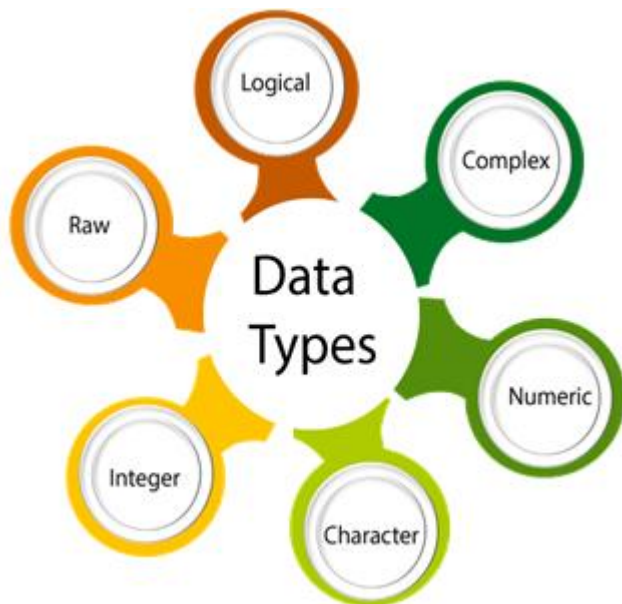
than to learn a programming language. A programming language is used to compute and communicate our discovery.

Data Types in R Programming

In programming languages, we need to use various variables to store various information. Variables are the reserved memory location to store values. As we create a variable in our program, some space is reserved in memory.

In R, there are several data types such as integer, string, etc. The operating system allocates memory based on the data type of the variable and decides what can be stored in the reserved memory.

There are the following data types which are used in R programming:



| Data type | Example | Description |
|------------------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| Logical | True, False | It is a special data type for data with only two possible values which can be construed as true/false. |
| Numeric | 12,32,112,5432 | Decimal value is called numeric in R, and it is the default computational data type. |
| Integer | 3L, 66L, 2346L | Here, L tells R to store the value as an integer, |
| Complex | Z=1+2i, t=7+3i | A complex value in R is defined as the pure imaginary value i. |
| Character | 'a', '"good"', 'TRUE', '35.4' | In R programming, a character is used to represent string values. We convert objects into character values with the help of as.character() function. |
| Raw | | A raw data type is used to holds raw bytes. |

1. **#Logical Data type**
2. `variable_logical<- TRUE`
3. `cat(variable_logical,"\\n")`
4. `cat("The data type of variable_logical is ",class(variable_logical),"\\n\\n")`
- 5.
6. **#Numeric Data type**
7. `variable_numeric<- 3532`

```

8. cat(variable_numeric,"\n")
9. cat("The data type of variable_numeric is ",class(variable_numeric),"\n\n")
10.
11. #Integer Data type
12. variable_integer<- 133L
13. cat(variable_integer,"\n")
14. cat("The data type of variable_integer is ",class(variable_integer),"\n\n")
15.
16. #Complex Data type
17. variable_complex<- 3+2i
18. cat(variable_complex,"\n")
19. cat("The data type of variable_complex is ",class(variable_complex),"\n\n")
20.
21. #Character Data type
22. variable_char<- "Learning r programming"
23. cat(variable_char,"\n")
24. cat("The data type of variable_char is ",class(variable_char),"\n\n")
25.
26. #Raw Data type
27. variable_raw<- charToRaw("Learning r programming")
28. cat(variable_raw,"\n")
29. cat("The data type of variable_char is ",class(variable_raw),"\n\n")

```

OUT PUT

Command Prompt

```

C:\Users\ajeet\R>Rscript datatype.R
TRUE
The data type of variable_logical is  logical

3532
The data type of variable_numeric is  numeric

133
The data type of variable_integer is  integer

3+2i
The data type of variable_complex is  complex

Learning r programming
The data type of variable_char is  character

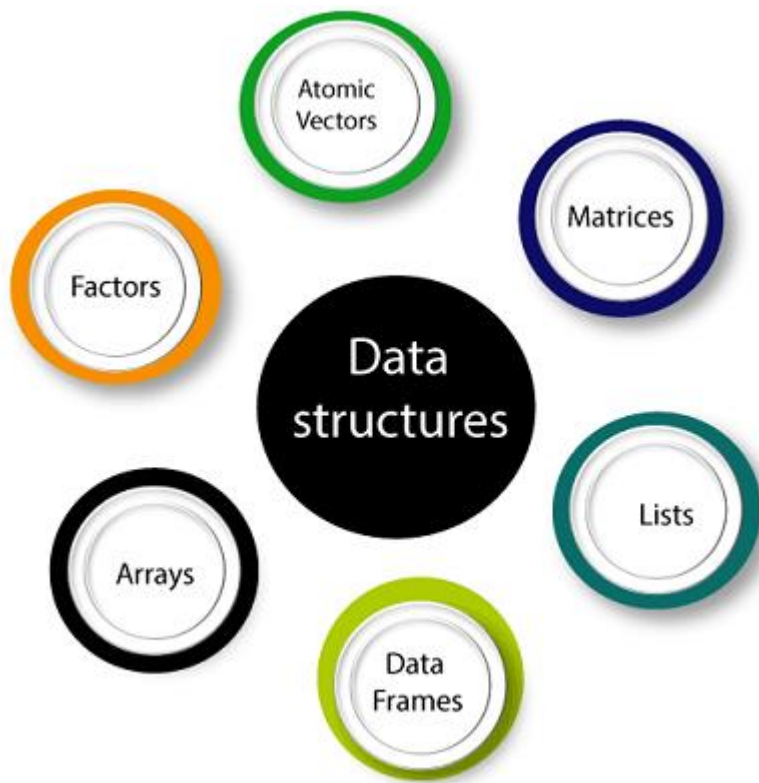
4c 65 61 72 6e 69 6e 67 20 72 20 70 72 6f 67 72 61 6d 6d 69 6e 67
The data type of variable_char is  raw

```

Data Structures in R Programming

Data structures are very important to understand. Data structure are the objects which we will manipulate in our day-to-day basis in R. Dealing with object conversions is the most common sources of despairs for beginners. We can say that everything in R is an object.

R has many data structures, which include:



1. Atomic vector
2. List
3. Array
4. Matrices
5. Data Frame
6. Factors

Vectors

A vector is the basic data structure in R, or we can say vectors are the most basic R data objects. There are six types of atomic vectors such as logical, integer, character, double, and raw. **"A vector is a collection of elements which is most commonly of mode character, integer, logical or numeric"** A vector can be one of the following two types:

1. Atomic vector
2. Lists

List

In R, **the list** is the container. Unlike an atomic vector, the list is not restricted to be a single mode. A list contains a mixture of data types. The list is also known as generic vectors because the element of the list can be of any type of R object. **"A list is a special type of vector in which each element can be a different type."**

We can create a list with the help of `list()` or `as.list()`. We can use `vector()` to create a required length empty list.

Arrays

There is another type of data objects which can store data in more than two dimensions known as arrays. **"An array is a collection of a similar data type with contiguous memory allocation."** Suppose, if we create an array of dimension (2, 3, 4) then it creates four rectangular matrices of two rows and three columns.

In R, an array is created with the help of **array()** function. This function takes a vector as an input and uses the value in the dim parameter to create an array.

Matrices

A matrix is an R object in which the elements are arranged in a two-dimensional rectangular layout. In the matrix, elements of the same atomic types are contained. For mathematical calculation, this can use a matrix containing the numeric element. A matrix is created with the help of the matrix() function in R.

Syntax

The basic syntax of creating a matrix is as follows:

1. matrix(data, no_row, no_col, by_row, dim_name)

Data Frames

A **data frame** is a two-dimensional array-like structure, or we can say it is a table in which each column contains the value of one variable, and row contains the set of value from each column.

There are the following characteristics of a data frame:

1. The column name will be non-empty.
2. The row names will be unique.
3. A data frame stored numeric, factor or character type data.
4. Each column will contain same number of data items.

Factors

Factors are also data objects that are used to categorize the data and store it as levels. Factors can store both strings and integers. Columns have a limited number of unique values so that factors are very useful in columns. It is very useful in data analysis for statistical modeling.

Factors are created with the help of **factor()** function by taking a vector as an input parameter.

Variables in R Programming

Variables are used to store the information to be manipulated and referenced in the R program. The R variable can store an atomic vector, a group of atomic vectors, or a combination of many R objects.

Language like C++ is statically typed, but R is a dynamically typed, means it check the type of data type when the statement is run. A valid variable name contains letter, numbers, dot and underlines characters. A variable name should start with a letter or the dot not followed by a number.

| Name of variable | Validity | Reason for valid and invalid |
|---------------------------|----------|------------------------------------------------------------------------------------------------------------------------|
| _var_name | Invalid | Variable name can't start with an underscore(_). |
| var_name, var.name | Valid | Variable can start with a dot, but dot should not be followed by a number. In this case, the variable will be invalid. |
| var_name% | Invalid | In R, we can't use any special character in the variable name except dot and underscore. |

| | | |
|-------------------|---------|-------------------------------------------------------------------------------|
| 2var_name | Invalid | Variable name cant starts with a numeric digit. |
| .2var_name | Invalid | A variable name cannot start with a dot which is followed by a digit. |
| var_name2 | Valid | The variable contains letter, number and underscore and starts with a letter. |

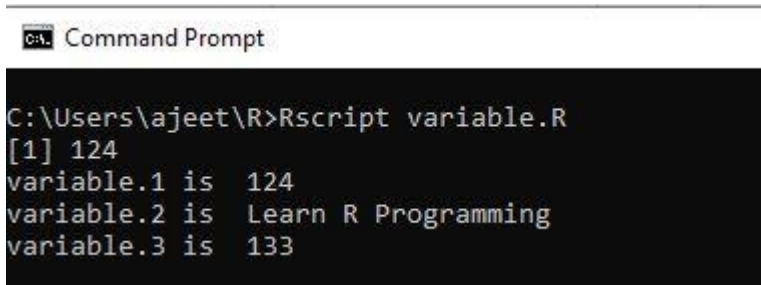
Assignment of variable

In R programming, there are three operators which we can use to assign the values to the variable. We can use leftward, rightward, and equal_to operator for this purpose.

There are two functions which are used to print the value of the variable i.e., print() and cat(). The cat() function combines multiples values into a continuous print output.

1. **# Assignment using equal operator.**
2. variable.1 = 124
- 3.
4. **# Assignment using leftward operator.**
5. variable.2 <- "Learn R Programming"
- 6.
7. **# Assignment using rightward operator.**
8. 133L -> variable.3
- 9.
10. **print**(variable.1)
11. cat ("variable.1 is ", variable.1 ,"\n")
12. cat ("variable.2 is ", variable.2 ,"\n")
13. cat ("variable.3 is ", variable.3 ,"\n")

When we execute the above code in our R command prompt, it will give us the following output:



```

C:\Users\ajeet\R>Rscript variable.R
[1] 124
variable.1 is 124
variable.2 is Learn R Programming
variable.3 is 133

```

Data types of variable

R programming is a dynamically typed language, which means that we can change the data type of the same variable again and again in our program. Because of its dynamic nature, a variable is not declared of any data type. It gets the data type from the R-object, which is to be assigned to the variable.

We can check the data type of the variable with the help of the class() function. Let's see an example:

1. variable_y<- 124
2. cat("The data type of variable_y is ",**class**(variable_y),"\n")
- 3.
4. variable_y<- "Learn R Programming"
5. cat(" Now the data type of variable_y is ",**class**(variable_y),"\n")

- 6.
7. `variable_y<- 133L`
8. `cat(" Next the data type of variable_y becomes ",class(variable_y),"\\n")`

output:

```

C:\Users\ajeet\R>Rscript datatype.R
The data type of variable_y is  numeric
Now the data type of variable_y is  character
Next the data type of variable_y becomes  integer
  
```

Keywords in R Programming

In programming, a keyword is a word which is reserved by a program because it has a special meaning. A keyword can be a command or a parameter. Like in C, C++, Java, there is also a set of keywords in R. A keyword can't be used as a variable name. Keywords are also called as "reserved names."

There are the following keywords as per **?reserved** or **help(reserved)** command:



| | | |
|-------|----------|-------------|
| if | else | repeat |
| while | function | for |
| next | break | TRUE |
| FALSE | NULL | Inf |
| NaN | NA | NA_integer_ |

| | | |
|----------|-------------|---------------|
| NA_real_ | NA_complex_ | NA_character_ |
| | | |

1) if

The if statement consists of a Boolean expression which is followed by one or more statements. In R, if statement is the simplest conditional statement which is used to decide whether a block of the statement will be executed or not.

Example:

1. `a<-11`
2. `if(a<15)`
3. `+ print("I am lesser than 15")`

Output:

```
[1] "I am lesser than 15"
>
```

2) else

The R else statement is associated with if statement. When the if statement's condition is false only then else block will be executed. Let see an example to make it clear:

Example:

1. `a<-22`
2. `if(a<20){`
3. `cat("I am lesser than 20")`
4. `}else{`
5. `cat("I am larger than 20")`
6. `}`

Output:

```
cmd Command Prompt
C:\Users\ajeet\R>Rscript Keyword.R
I am larger than 20
```

3) repeat

The repeat keyword is used to iterate over a block of code multiple numbers of times. In R, repeat is a loop, and in this loop statement, there is no condition to exit from the loop. For exiting the loop, we will use the break statement.

Example:

1. `x <- 1`
2. `repeat {`
3. `cat(x)`
4. `x = x+1`

```
5.  if (x == 6){
6.      break
7.  }
8.  }
```

Output:

```
C:\> Command Prompt

C:\Users\ajeet\R>Rscript Keyword.R
12345
```

4) while

A while keyword is used as a loop. The while loop is executed until the given condition is true. This is also used to make an infinite loop.

Example:

```
1.  a <- 20
2.  while(a!=0){
3.      cat(a)
4.      a = a-2
5.  }
```

Output:

```
C:\> Command Prompt

C:\Users\ajeet\R>Rscript Keyword.R
2018161412108642
```

5) function

A function is an object in R programming. The keyword function is used to create a user-define function in R. R has some pre-defined functions also, such as seq, mean, and sum.

Example:

```
1.  new.function <- function(n) {
2.      for(i in 1:n) {
3.          a <- i^2
4.          print(a)
5.      }
6.  }
7.  new.function(6)
```

Output:

Command Prompt

```
C:\Users\ajeet\R>Rscript Keyword.R
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
```

6) for

The **for** is a keyword which is used for looping or iterating over a sequence (dictionary, string, list, set or tuple).

We can execute a set of a statement once for each item in the iterator (list, set, tuple, etc.) with the help of for loop.

Example:

```
1. v <- LETTERS[1:4]
2. for ( i in v) {
3.   print(i)
4. }
```

Output:

Command Prompt

```
C:\Users\ajeet\R>Rscript Keyword.R
[1] "A"
[1] "B"
[1] "C"
[1] "D"
```

7) next

The next keyword skips the current iteration of a loop without terminating it. When R parser found next, it skips further evaluation and starts the new iteration of the loop.

Example:

```
1. v <- LETTERS[1:6]
2. for ( i in v) {
3.   if (i == "D") {
4.     next
5.   }
6.   print(i)
7. }
```

Output:

Command Prompt

```
C:\Users\ajeet\R>Rscript Keyword.R
[1] "A"
[1] "B"
[1] "C"
[1] "E"
[1] "F"
```

8) break

The **break** keyword is used to terminate the loop if the condition is true. The control of the program firstly passes to the outer statement then passes to the body of the break statement.

Example:

```
1. n<-1
2. while(n<10){
3.   if(n==3)
4.     break
5.   n=n+1
6.   cat(n,"\n")
7. }
8. cat("End of the program")
```

Output:

Command Prompt

```
C:\Users\ajeet\R>Rscript Keyword.R
2
3
End of the program
```

9) TRUE/FALSE

The TRUE and FALSE keywords are used to represent a Boolean true and Boolean false. If the given statement is true, then the interpreter returns true else the interpreter returns false.

Command Prompt

```
C:\Users\ajeet\R>Rscript Keyword.R
[1] TRUE
[1] FALSE
```

10) NULL

In R, NULL represents the null object. NULL is used to represent missing and undefined values. NULL is the logical representation of a statement which is neither TRUE nor FALSE.

Example:

```
1. as.null(list(a = 1, b = "c"))
```

Output:

```
Command Prompt
C:\Users\ajeeet\R>Rscript Keyword.R
NULL
```

11) Inf and NaN

The `is.finite` and `is.infinite` function returns a vector of the same length indicating which elements are finite or infinite.

`Inf` and `-Inf` are positive and negative infinity. `NaN` stands for 'Not a Number.' `NaN` applies on numeric values and real and imaginary parts of complex values, but it will not apply to the values of integer vectors.

Usage

1. `is.finite(x)`
2. `is.infinite(x)`
3. `is.nan(x)`
- 4.
5. `Inf`
6. `NaN`

12) NA

`NA` is a logical constant of length 1 that contains a missing value indicator. It can be coerced to any other vector type except `raw`. There are other types of constant also, such as `NA_Integer_`, `NA_real_`, `NA_complex_`, and `NA_character_`. These constants are of the other atomic vector type which supports missing values.

Usage

1. `NA`
2. `is.na(x)`
3. `anyNA(x, recursive = FALSE)`
- 4.
5. `## S3 method for class 'data.frame'`
6. `is.na(x)`
- 7.
8. `is.na(x) <- value`

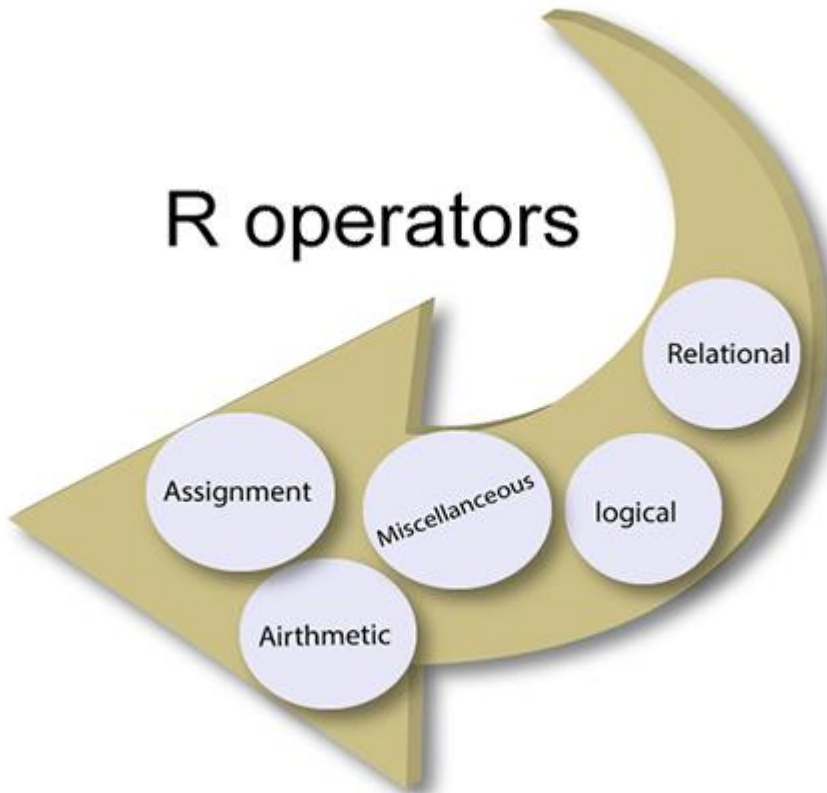
Operators in R

In **computer programming**, an operator is a symbol which represents an action. An operator is a symbol which tells the compiler to perform specific **logical** or **mathematical** manipulations. R programming is very rich in built-in operators.

In **R programming**, there are different types of operator, and each operator performs a different task. For data manipulation, There are some advance operators also such as model formula and list indexing.

There are the following types of operators used in R:

R operators



1. [Arithmetic Operators](#)
2. [Relational Operators](#)
3. [Logical Operators](#)
4. [Assignment Operators](#)
5. [Miscellaneous Operators](#)

Arithmetic Operators

Arithmetic operators are the symbols which are used to represent arithmetic math operations. The operators act on each and every element of the vector. There are various arithmetic operators which are supported by R.

| S. No | Operator | Description | Example |
|-------|----------|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| 1. | + | This operator is used to add two vectors in R. <code>a <- c(2, 3.3, 4)</code> | <pre>b <- c(11, 5, 3) print(a+b)</pre> <p>It will give us the following output:</p> <pre>[1] 13.0 8.3 5.0</pre> |
| 2. | - | This operator is used to divide a vector from another one. <code>a <- c(2, 3.3, 4)</code> | <pre>b <- c(11, 5, 3) print(a-b)</pre> <p>It will give us the following output:</p> <pre>[1] -9.0 -1.7 3.0</pre> |

| | | | |
|----|------|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 3. | * | This operator is used to multiply two vectors with each other. a <- c(2, 3.3, 4) | <pre>b <- c(11, 5, 3) print(a*b)</pre> <p>It will give us the following output:</p> <pre>[1] 22.0 16.5 4.0</pre> |
| 4. | / | This operator divides the vector from another one. a <- c(2, 3.3, 4) | <pre>b <- c(11, 5, 3) print(a/b)</pre> <p>It will give us the following output:</p> <pre>[1] 0.1818182 0.6600000 4.0000000</pre> |
| 5. | %% | This operator is used to find the remainder of the first vector with the second vector. a <- c(2, 3.3, 4) | <pre>b <- c(11, 5, 3) print(a%%b)</pre> <p>It will give us the following output:</p> <pre>[1] 2.0 3.3 0</pre> |
| 6. | %%/% | This operator is used to find the division of the first vector with the second(quotient). | <pre>a <- c(2, 3.3, 4) b <- c(11, 5, 3) print(a%%/b)</pre> <p>It will give us the following output:</p> <pre>[1] 0 0 4</pre> |
| 7. | ^ | This operator raised the first vector to the exponent of the second vector. a <- c(2, 3.3, 4) | <pre>b <- c(11, 5, 3) print(a^b)</pre> <p>It will give us the following output:</p> <pre>[1] 0248.0000 391.3539 4.0000</pre> |

Relational Operators

A relational operator is a symbol which defines some kind of relation between two entities. These include numerical equalities and inequalities. A relational operator compares each element of the first vector with the corresponding element of the second vector. The result of the comparison will be a Boolean value. There are the following relational operators which are supported by R:

| S. No | Operator | Description | Example |
|-------|----------|---------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | > | This operator will return TRUE when every element in the first vector is greater than the corresponding element of the second vector. | <pre>a <- c(1, 3, 5) b <- c(2, 4, 6) print(a>b)</pre> <p>It will give us the following output:</p> <pre>[1] FALSE FALSE FALSE</pre> |

| | | | |
|----|----|------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 2. | < | This operator will return TRUE when every element in the first vector is less than the corresponding element of the second vector. | <pre>a <- c(1, 9, 5) b <- c(2, 4, 6) print(a<b)</pre> <p>It will give us the following output:</p> <pre>[1] FALSE TRUE FALSE</pre> |
| 3. | <= | This operator will return TRUE when every element in the first vector is less than or equal to the corresponding element of another vector. | <pre>a <- c(1, 3, 5) b <- c(2, 3, 6) print(a<=b)</pre> <p>It will give us the following output:</p> <pre>[1] TRUE TRUE TRUE</pre> |
| 4. | >= | This operator will return TRUE when every element in the first vector is greater than or equal to the corresponding element of another vector. | <pre>a <- c(1, 3, 5) b <- c(2, 3, 6) print(a>=b)</pre> <p>It will give us the following output:</p> <pre>[1] FALSE TRUE FALSE</pre> |
| 5. | == | This operator will return TRUE when every element in the first vector is equal to the corresponding element of the second vector. | <pre>a <- c(1, 3, 5) b <- c(2, 3, 6) print(a==b)</pre> <p>It will give us the following output:</p> <pre>[1] FALSE TRUE FALSE</pre> |
| 6. | != | This operator will return TRUE when every element in the first vector is not equal to the corresponding element of the second vector. | <pre>a <- c(1, 3, 5) b <- c(2, 3, 6) print(a!=b)</pre> <p>It will give us the following output:</p> <pre>[1] TRUE FALSE TRUE</pre> |

Logical Operators

The logical operators allow a program to make a decision on the basis of multiple conditions. In the program, each operand is considered as a condition which can be evaluated to a false or true value. The value of the conditions is used to determine the overall value of the op1 **operator** op2. Logical operators are applicable to those vectors whose type is logical, numeric, or complex.

The logical operator compares each element of the first vector with the corresponding element of the second vector.

There are the following types of operators which are supported by R:

| S. No | Operator | Description | Example |
|-------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | & | This operator is known as the Logical AND operator. This operator takes the first element of both the vector and returns TRUE if both the elements are TRUE. | <pre>a <- c(3, 0, TRUE, 2+2i) b <- c(2, 4, TRUE, 2+3i) print(a&b)</pre> <p>It will give us the following output:</p> <pre>[1] TRUE FALSE TRUE TRUE</pre> |
| 2. | | This operator is called the Logical OR operator. This operator takes the first element of both the vector and returns TRUE if one of them is TRUE. | <pre>a <- c(3, 0, TRUE, 2+2i) b <- c(2, 4, TRUE, 2+3i) print(a b)</pre> <p>It will give us the following output:</p> <pre>[1] TRUE TRUE TRUE TRUE</pre> |
| 3. | ! | This operator is known as Logical NOT operator. This operator takes the first element of the vector and gives the opposite logical value as a result. | <pre>a <- c(3, 0, TRUE, 2+2i) print(!a)</pre> <p>It will give us the following output:</p> <pre>[1] FALSE TRUE FALSE FALSE</pre> |
| 4. | && | This operator takes the first element of both the vector and gives TRUE as a result, only if both are TRUE. | <pre>a <- c(3, 0, TRUE, 2+2i) b <- c(2, 4, TRUE, 2+3i) print(a&&b)</pre> <p>It will give us the following output:</p> <pre>[1] TRUE</pre> |
| 5. | | This operator takes the first element of both the vector and gives the result TRUE, if one of them is true. | <pre>a <- c(3, 0, TRUE, 2+2i) b <- c(2, 4, TRUE, 2+3i) print(a b)</pre> <p>It will give us the following output:</p> <pre>[1] TRUE</pre> |

Assignment Operators

An assignment operator is used to assign a new value to a variable. In R, these operators are used to assign values to vectors. There are the following types of assignment

| S. No | Operator | Description | Example |
|-------|----------------|----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <- or = or <<- | These operators are known as left assignment operators. | <pre>a <- c(3, 0, TRUE, 2+2i) b <<- c(2, 4, TRUE, 2+3i) d = c(1, 2, TRUE, 2+3i) print(a) print(b) print(d)</pre> <p>It will give us the following output:</p> <pre>[1] 3+0i 0+0i 1+0i 2+2i [1] 2+0i 4+0i 1+0i 2+3i [1] 1+0i 2+0i 1+0i 2+3i</pre> |
| 2. | -> or ->> | These operators are known as right assignment operators. | <pre>c(3, 0, TRUE, 2+2i) -> a c(2, 4, TRUE, 2+3i) ->> b print(a) print(b)</pre> <p>It will give us the following output:</p> <pre>[1] 3+0i 0+0i 1+0i 2+2i [1] 2+0i 4+0i 1+0i 2+3i</pre> |

operators which are supported by R:

Miscellaneous Operators

Miscellaneous operators are used for a special and specific purpose. These operators are not used for general mathematical or logical computation. There are the following miscellaneous operators which are supported in R

| S. No | Operator | Description | Example |
|-------|----------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | : | The colon operator is used to create the series of numbers in sequence for a vector. | <pre>v <- 1:8 print(v)</pre> <p>It will give us the following output:</p> <pre>[1] 1 2 3 4 5 6 7 8</pre> |
| 2. | %in% | This is used when we want to identify if an element belongs to a vector. | <pre>a1 <- 8 a2 <- 12 d <- 1:10 print(a1%in%) print(a2%in%)</pre> <p>It will give us the following output:</p> <pre>[1] FALSE [1] FALSE</pre> |

| | | | |
|----|----|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3. | %% | It is used to multiply a matrix with its transpose. | <pre>M=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=TRUE) T=m%%T(m) print(T)</pre> <p>It will give us the following output:</p> <pre>14 32 32 77</pre> |
|----|----|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|

R if Statement

The if statement consists of the Boolean expressions followed by one or more statements. The if statement is the simplest decision-making statement which helps us to take a decision on the basis of the condition.

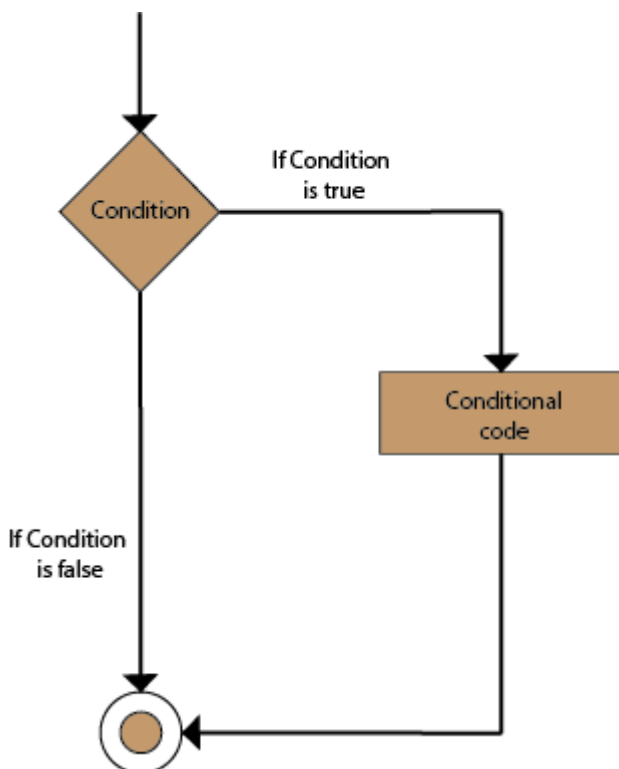
The if statement is a conditional programming statement which performs the function and displays the information if it is proved true.

The block of code inside the if statement will be executed only when the boolean expression evaluates to be true. If the statement evaluates false, then the code which is mentioned after the condition will run.

The syntax of if statement in R is as follows:

1. **if**(boolean_expression) {
2. // If the boolean expression is true, then statement(s) will be executed.
3. }

Flow Chart

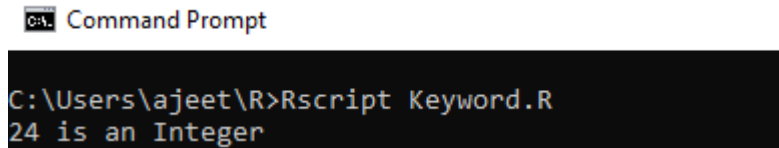


Let see some examples to understand how if statements work and perform a certain task in R.

Example 1

```
1. x <-24L
2. y <- "shubham"
3. if(is.integer(x))
4. {
5.   print("x is an Integer")
6. }
```

Output:

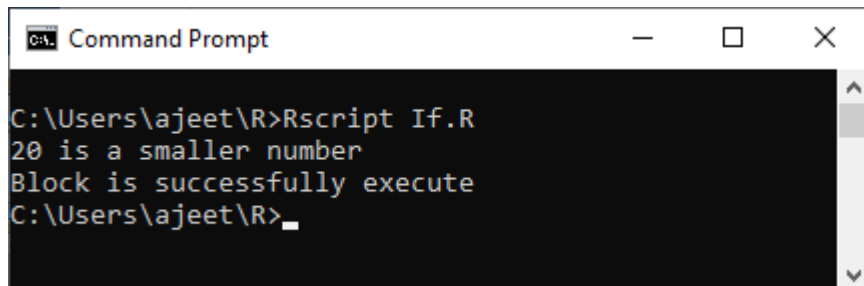


```
C:\Users\ajeet\R>Rscript Keyword.R
24 is an Integer
```

Example 2

```
1. x <-20
2. y<-24
3. count=0
4. if(x<y)
5. {
6.   cat(x,"is a smaller number\n")
7.   count=1
8. }
9. if(count==1){
10.  cat("Block is successfully execute")
11. }
```

Output:



```
C:\Users\ajeet\R>Rscript If.R
20 is a smaller number
Block is successfully execute
C:\Users\ajeet\R>
```

Example 3

```
1. x <-1
2. y<-24
3. count=0
4. while(x<y){
5.   cat(x,"is a smaller number\n")
6.   x=x+2
7.   if(x==15)
8.     break
9. }
```

Output:

```
C:\Users\ajeet\R>Rscript If.R
1 is a smaller number
3 is a smaller number
5 is a smaller number
7 is a smaller number
9 is a smaller number
11 is a smaller number
13 is a smaller number

C:\Users\ajeet\R>
```

Example 4

1. `x <- -24`
2. `if(x%%2==0){`
3. `cat(x," is an even number")`
4. `}`
5. `if(x%%2!=0){`
6. `cat(x," is an odd number")`
7. `}`

Output:

```
C:\Users\ajeet\R>Rscript If.R
24 is an even number
C:\Users\ajeet\R>_
```

Example 5

1. `year`
2. `1 = 2011`
3. `if(year1 %% 4 == 0) {`
4. `if(year1 %% 100 == 0) {`
5. `if(year1 %% 400 == 0) {`
6. `cat(year,"is a leap year")`
7. `} else {`
8. `cat(year,"is not a leap year")`
9. `}`
10. `} else {`
11. `cat(year,"is a leap year")`
12. `}`
13. `} else {`
14. `cat(year,"is not a leap year")`
15. `}`

Output:

```
Command Prompt

C:\Users\ajeet\R>Rscript If.R
[1] "2011 is not a leap year"

C:\Users\ajeet\R>
```

```
Command Prompt

Microsoft Windows [Version 10.0.18362.239]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ajeet>cd R

C:\Users\ajeet\R>Rscript if-else.R
character is a vowel
character is = u
C:\Users\ajeet\R>
```

```
Command Prompt

C:\Users\ajeet\R>Rscript switch.R
Division= 5NULL

C:\Users\ajeet\R>
```