**R - Introduction**

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac.

R is free software distributed under a GNU-style copy left, and an official part of the GNU project called **GNU S**.

**Features of R :**

* R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
* R has an effective data handling and storage facility,
* R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
* R provides a large, coherent and integrated collection of tools for data analysis.
* R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

Local Environment Setup

You can download the Windows installer version of R from [R-3.2.2 for Windows (32/64 bit)](https://cran.r-project.org/bin/windows/base/) and save it in a local directory.

As it is a Windows installer (.exe) with a name "R-version-win.exe". You can just double click and run the installer accepting the default settings. If your Windows is 32-bit version, it installs the 32-bit version. But if your windows is 64-bit, then it installs both the 32-bit and 64-bit versions

## R Command Prompt

Once you have R environment setup, then it’s easy to start your R command prompt by just typing the following command at your command prompt –

This will launch R interpreter and you will get a prompt > where you can start typing your program as follows –

> myString <- "Hello, World!"

> print ( myString)

[1] "Hello, World!"

Here first statement defines a string variable myString, where we assign a string "Hello, World!" and then next statement print() is being used to print the value stored in variable myString.

R Script File

Usually, you will do your programming by writing your programs in script files and then you execute those scripts at your command prompt with the help of R interpreter called **Rscript**. So let's start with writing following code in a text file called test.R as under –

# My first program in R Programming

myString <- "Hello, World!"

print ( myString)

Save the above code in a file test.R and execute it at Linux command prompt as given below. Even if you are using Windows or other system, syntax will remain same.

# **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.

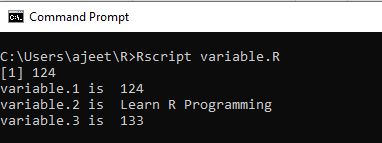
There are two functions which are used to print the value of the variable i.e., print() and cat(). The cat() function combines multiples values into a continuous print output.

16M

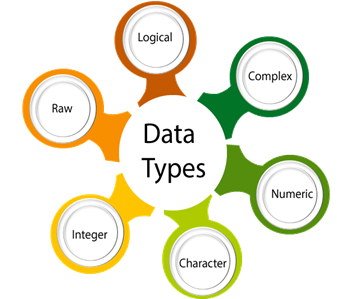
251

Triggers in SQL (Hindi)

1. # Assignment using equal operator.
2. variable.1 = 124
4. # Assignment using leftward operator.
5. variable.2 <- "Learn R Programming"
7. # Assignment using rightward operator.
8. 133L -> variable.3
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")

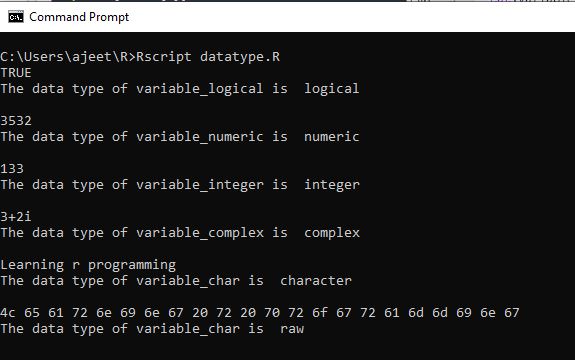


There are the following data types which are used in R programming:

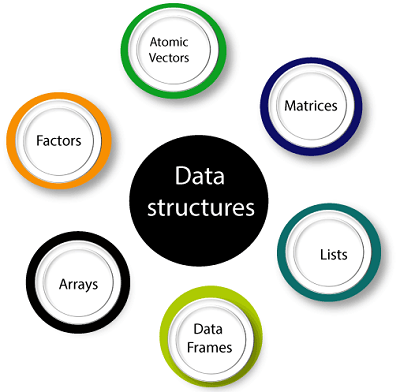


|  |  |  |
| --- | --- | --- |
| **Data type** | **Example** | **Description** |
| **Logical** | True, False | It is a special data type for data with only two possible values which can be construed as true/false. |
| **Numeric** | 12,32,112,5432 | Decimal value is called numeric in R, and it is the 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 ofas.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")
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")
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")
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")
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")
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")



# **Data Structures in R Programming**



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."**

Skip Ad

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.

## Hello World Program

* [Find the Factorial of a Number](https://www.datamentor.io/r-programming/examples/factorial)
* # take input from the user
* num = as.integer(readline(prompt="Enter a number: "))
* factorial = 1
* # check is the number is negative, positive or zero
* if(num < 0) {
* print("Sorry, factorial does not exist for negative numbers")
* } else if(num == 0) {
* print("The factorial of 0 is 1")
* } else {
* for(i in 1:num) {
* factorial = factorial \* i
* }
* print(paste("The factorial of", num ,"is",factorial))
* }
* Take inputs from user :

my.name <- readline(prompt="Enter name: ")

my.age <- readline(prompt="Enter age: ")

# convert character into integer

my.age <- as.integer(my.age)

print(paste("Hi,", my.name, "next year you will be", my.age+1, "years old."))

Enter name: Mary

Enter age: 17

[1] "Hi, Mary next year you will be 18 years old."

|  |
| --- |
| [R Program to Add Two Vectors](https://www.datamentor.io/r-programming/examples/add-two-vector) |
| [Find Sum, Mean and Product of Vector in R Programming](https://www.datamentor.io/r-programming/examples/sum-mean-product-vector) |
| [R Program to Take Input From User](https://www.datamentor.io/r-programming/examples/user-input) |
| [R Program to Generate Random Number from Standard Distributions](https://www.datamentor.io/r-programming/examples/random-number) |
| [R Program to Sample from a Population](https://www.datamentor.io/r-programming/examples/sample-population) |
| [R Program to Find Minimum and Maximum](https://www.datamentor.io/r-programming/examples/minimum-maximum) |
| [R Program to Sort a Vector](https://www.datamentor.io/r-programming/examples/sort-vector) |
| [R Program to Find the Factorial of a Number](https://www.datamentor.io/r-programming/examples/factorial) |
| [R Multiplication Table](https://www.datamentor.io/r-programming/examples/multiplication-table) |
| [R Program to Check Prime Number](https://www.datamentor.io/r-programming/examples/prime-number) |
| [R Program to check Armstrong Number](https://www.datamentor.io/r-programming/examples/armstrong-number) |
| [R Program to Print the Fibonacci Sequence](https://www.datamentor.io/r-programming/examples/fibonacci-sequence) |
| [R Program to Check for Leap Year](https://www.datamentor.io/r-programming/examples/leap-year) |
| [Check if a Number is Odd or Even in R Programming](https://www.datamentor.io/r-programming/examples/odd-even) |
| [R Program to Check if a Number is Positive, Negative or Zero](https://www.datamentor.io/r-programming/examples/positive-negative-zero) |
| [R Program to Find the Sum of Natural Numbers](https://www.datamentor.io/r-programming/examples/sum-natural-number) |
| [Convert Decimal into Binary using Recursion in R](https://www.datamentor.io/r-programming/examples/decimal-to-binary) |
| [R program to Find the Factorial of a Number Using Recursion](https://www.datamentor.io/r-programming/examples/factorial-recursion) |
| [R Program to Find the Factors of a Number](https://www.datamentor.io/r-programming/examples/factors-of-number) |
| [Fibonacci Sequence Using Recursion in R](https://www.datamentor.io/r-programming/examples/fibonacci-recursion) |
| [R Program to Find H.C.F. or G.C.D.](https://www.datamentor.io/r-programming/examples/gcd-hcf) |
| [R Program to Find L.C.M.](https://www.datamentor.io/r-programming/examples/least-common-multiple) |
| [R Program to Make a Simple Calculator](https://www.datamentor.io/r-programming/examples/simple-calculator) |
| [Sum of Natural Numbers Using Recursion](https://www.datamentor.io/r-programming/examples/sum-natural-number-recursion) |