UNIT II

Data Types in R Programming Language

Each variable in R has an associated data type. Each R-Data Type requires different amounts of memory and has some specific operations which can be performed over it.

Data Types in R are:

1. numeric – (3,6.7,121)
2. Integer – (2L, 42L; where 'L' declares this as an integer)
3. logical – ('True')
4. complex – (7 + 5i; where 'i' is imaginary number)
5. character – ("a", "B", "c is third", "69")
6. raw – (as.raw(55); raw creates a raw vector of the specified length)

R Programming language has the following basic R-data types and the following table shows the data type and the values that each data type can take.

Basic Data Types	Values	Examples
Numeric	Set of all real numbers	"numeric_value <- 3.14"
Integer	Set of all integers, Z	"integer_value <- 42L"
Logical	TRUE and FALSE	"logical_value <- TRUE"
Complex	Set of complex numbers	"complex_value <- 1 + 2i"
Character	"a", "b", "c", …, "@", "#", "$", …., "1", "2", …etc	"character_value <- "Hello Geeks"
raw	as.raw()	"single_raw <- as.raw(255)"

1. Numeric Data type in R

Decimal values are called numeric in R. It is the default R data type for numbers in R. If you assign a decimal value to a variable x as follows, x will be of numeric type. Real numbers with a decimal point are represented using this data type in R. It uses a format for double-precision floating-point numbers to represent numerical values.

- R

```
# A simple R program
# to illustrate Numeric data type

# Assign a decimal value to x
x = 5.6

# print the class name of variable
print(class(x))

# print the type of variable
print(typeof(x))
```

Output

```
[1] "numeric"

[1] "double"
```

Even if an integer is assigned to a variable y, it is still saved as a numeric value.

- R

```
# A simple R program
# to illustrate Numeric data type

# Assign an integer value to y
y = 5

# print the class name of variable
print(class(y))

# print the type of variable
print(typeof(y))
```

Output

```
[1] "numeric"

[1] "double"
```

When R stores a number in a variable, it converts the number into a “double” value or a decimal type with at least two decimal places.
This means that a value such as “5” here, is stored as 5.00 with a type of double and a class of numeric. And also y is not an integer here can be confirmed with the **is.integer()** function.

- R

```
# A simple R program
# to illustrate Numeric data type

# Assign a integer value to y
y = 5

# is y an integer?
print(is.integer(y))
```

Output

```
[1] FALSE
```

2. Integer Data type in R

R supports integer data types which are the set of all integers.
You can create as well as convert a value into an integer type using the **as.integer()** function.
You can also use the capital 'L' notation as a suffix to denote that a particular value is of the integer R data type.

- R

```
# A simple R program
# to illustrate integer data type

# Create an integer value
x = as.integer(5)

# print the class name of x
print(class(x))

# print the type of x
print(typeof(x))

# Declare an integer by appending an L suffix.
y = 5L

# print the class name of y
print(class(y))

# print the type of y
print(typeof(y))
```

Output

```
[1] "integer"

[1] "integer"

[1] "integer"
```

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

3. Logical Data type in R

R has logical data types that take either a value of **true** or **false**.
A logical value is often created via a comparison between variables.
Boolean values, which have two possible values, are represented by this R data type:
FALSE or TRUE

- R

```
# A simple R program
# to illustrate logical data type

# Sample values
x = 4
y = 3

# Comparing two values
z = x > y

# print the logical value
print(z)

# print the class name of z
print(class(z))

# print the type of z
print(typeof(z))
```

Output

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

4. Complex Data type in R

R supports complex data types that are set of all the complex numbers. The complex data type is to store numbers with an imaginary component.

- R

```
# A simple R program
# to illustrate complex data type

# Assign a complex value to x
x = 4 + 3i
```

```
# print the class name of x
print(class(x))

# print the type of x
print(typeof(x))
```

Output

```
[1] "complex"

[1] "complex"
```

5. Character Data type in R

R supports character data types where you have all the alphabets and special characters. It stores character values or strings. Strings in R can contain alphabets, numbers, and symbols.

The easiest way to denote that a value is of character type in R data type is to wrap the value inside single or double inverted commas.

- R

```
# A simple R program
# to illustrate character data type

# Assign a character value to char
char = "Geeksforgeeks"

# print the class name of char
print(class(char))

# print the type of char
print(typeof(char))
```

Output

```
[1] "character"

[1] "character"
```

There are several tasks that can be done using R data types. Let's understand each task with its action and the syntax for doing the task along with an R code to illustrate the task.

6. Raw data type in R

To save and work with data at the byte level in R, use the raw data type. By displaying a series of unprocessed bytes, it enables low-level operations on binary data. Here are some speculative data on R's raw data types:

- R

```
# Create a raw vector
x <- as.raw(c(0x1, 0x2, 0x3, 0x4, 0x5))
```

```
print(x)
```

Output

```
[1] 01 02 03 04 05
```

Five elements make up this raw vector x, each of which represents a raw byte value.

Find Data Type of an Object in R

To find the data type of an object you have to use **class()** function. The syntax for doing that is you need to pass the object as an argument to the function **class()** to find the data type of an object.

Syntax

```
class(object)
```

Example

- R

```
# A simple R program
# to find data type of an object

# Logical
print(class(TRUE))

# Integer
print(class(3L))

# Numeric
print(class(10.5))

# Complex
print(class(1+2i))

# Character
print(class("12-04-2020"))
```

Output

```
[1] "logical"

[1] "integer"

[1] "numeric"

[1] "complex"

[1] "character"
```

Type verification

You can verify the data type of an object, if you doubt about it's data type.

To do that, you need to use the prefix "is." before the data type as a command.

Syntax:

```
is.data_type(object)
```

Example

- R

```
# A simple R program
# Verify if an object is of a certain datatype

# Logical
print(is.logical(TRUE))

# Integer
print(is.integer(3L))

# Numeric
print(is.numeric(10.5))

# Complex
print(is.complex(1+2i))

# Character
print(is.character("12-04-2020"))

print(is.integer("a"))

print(is.numeric(2+3i))
```

Output

```
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] FALSE
[1] FALSE
```