**What are Variables in R**

A variable is a memory allocated for the storage of specific data and the name associated with the variable is used to work around this reserved block.

The name given to a variable is known as its variable name. Usually a single variable stores only the data belonging to a certain data type.

The name is so given to them because when the program executes there is subject to change hence it varies from time to time.

Variables are used for storing data where that value can be altered based to your need. Unique name has to be given to variable (also for functions and objects) is identifier.

**Rules for writing Identifiers in R**

* A valid variable name consists of a combination of alphabets, numbers, dot(.), and underscore(\_) characters. Example: var.1\_ is valid
* Apart from the dot and underscore operators, no other special character is allowed. Example: var$1 or var#1 both are invalid
* Variables can start with alphabets or dot characters. Example: .var or var is valid
* The variable should not start with numbers or underscore. Example: 2var or \_var is invalid.
* If a variable starts with a dot the next thing after the dot cannot be a number. Example: .3var is invalid
* The variable name should not be a reserved keyword in R. Example: TRUE, FALSE,etc.

**Best Practices for Writing Identifiers**

Former versions of R used underscore to assign values. So, the period (.) operator was used broadly in variable names that have multiple words. Present versions of R support underscore (\_) as valid identifier. But it is considered to be not a good practice to use period for word separators. Here’s an example, a.variable.name is preferred over a\_variable\_name.

**Creating Variables in R Language**

Let’s look at ways of declaring and initializing variables in R language:

R supports three ways of variable assignment:

* Using equal operator- operators use an arrow or an equal sign to assign values to variables.
* Using the leftward operator- data is copied from right to left.
* Using the rightward operator- data is copied from left to right.

Syntax for creating R Variables

Types of Variable Creation in R:

* ***Using equal to operators******variable\_name = value***
* ***using leftward operator******variable\_name <- value***
* ***using rightward operator******value -> variable\_name***

Note: ->> and <<- can also be used.

**Important Methods for R Variables**

R provides some useful methods to perform operations on variables. These methods are used to determine the data type of the variable, finding a variable, deleting a variable, etc.

1. class(): It can be described as the 'container' of an object, an attribute that makes this object behave externally with functions that take this particular 'type' of object as argument. It is a kind of label that is ascribed to describe an object.
2. typeof(): It is what this object is made of inside the container, the 'type' of underlying values. eveals the most fundamental datatype for R objects, going all the way down to the underlying C language datatypes in which R is implemented. So, it indicates how the variable is stored in the computer memory in terms of the lowest-level datatypes available in R. An object can only have one typeof type. The different possibilities can be found from help(typeof).

A key difference between the two is that typeof is read-only whereas class is read-write. That is, we cannot directly change an object's typeof type; we must convert the underlying object into an object of a different type; indeed, there is no assignment allowed for typeof. In contrast, class allows assignment, which, for basic datatypes, converts the variable from one datatype to another. So, a variable's typeof can be changed by indirectly converting it by changing its class:

some\_numbers <- c(1L, 2L, 3L) # 'L' added to create integers, not doubles

some\_numbers

#> [1] 1 2 3

typeof(some\_numbers)

#> [1] "integer"

class(some\_numbers)

#> [1] "integer"

typeof(some\_numbers) <- 'double'

#> Error in typeof(some\_numbers) <- "double": could not find function "typeof<-"

class(some\_numbers) <- 'character'

some\_numbers

#> [1] "1" "2" "3"

typeof(some\_numbers)

#> [1] "character"

class(some\_numbers)

#> [1] "character"

1. ls(): This built-in function is used to know all the present variables in the workspace. This is generally helpful when dealing with a large number of variables at once and helps prevents overwriting any of them.

ls()

1. rm(): This is again a built-in function used to delete an unwanted variable within your workspace. This helps clear the memory space allocated to certain variables that are not in use thereby creating more space for others. The name of the variable to be deleted is passed as an argument to it

rm(variable)

**What are Constants in R**

Constants are entities within a program whose value can’t be changed. There are 2 basic types of constant. These are numeric constants and character constants.

**Numeric Constants**

All the numbers you will be using within a program fall under this category. There are sub types like integer, double or complex, which is checked usng typeof() function. Example:

> typeof (6)

[1] "double"

> typeof (4L)

[1] "integer"

**Character Constants**

These can be signified by means of either single quotes (‘) or using double quotes (“) as delimiters.

> 'ray'

[1] "ray"

> typeof("karlos")

[1] "character"