SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

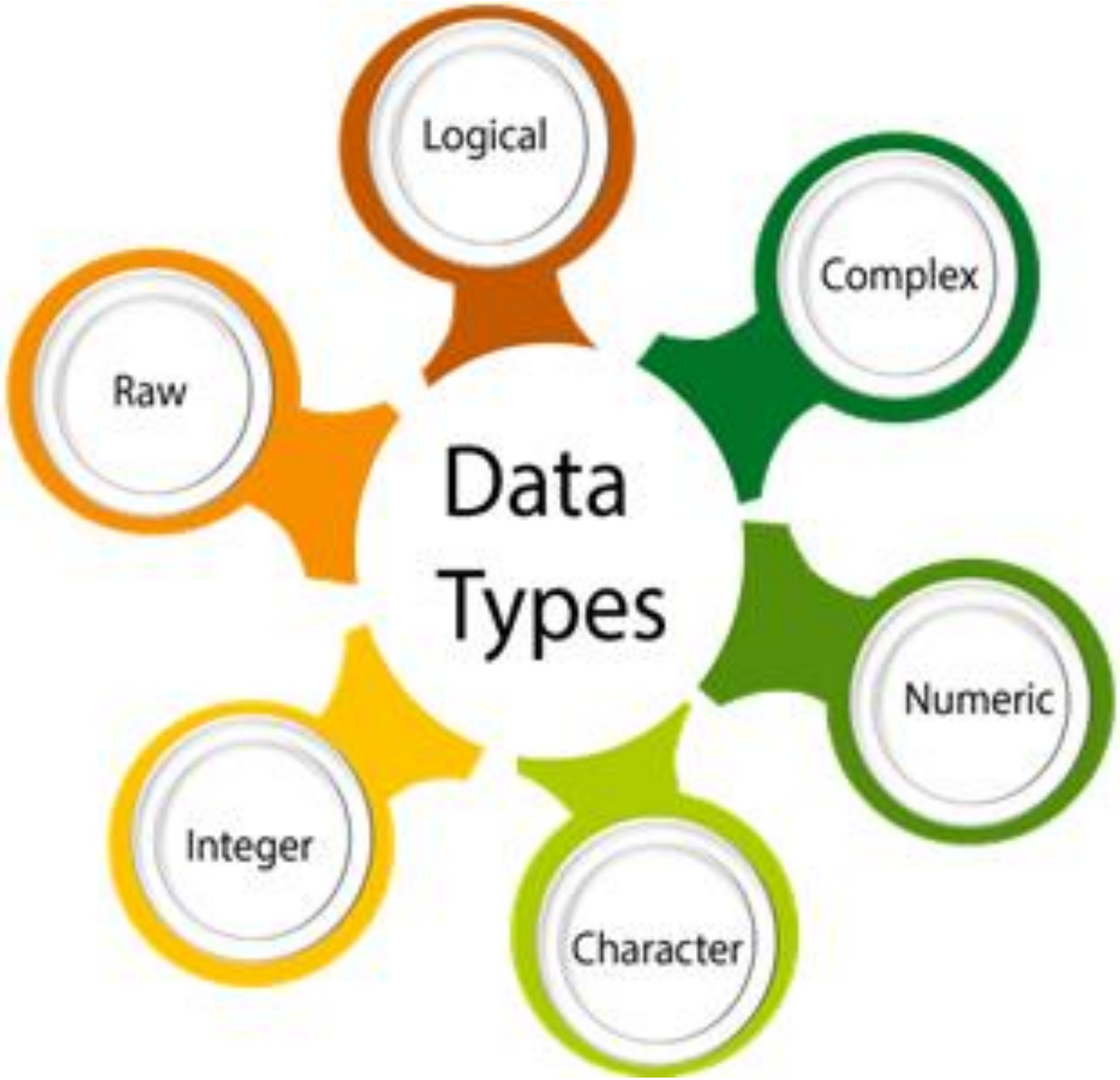
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

COURSE NAME :19CS407 DATA ANALYTICS WITH R

II YEAR /IV SEMESTER

Unit 4- R PROGRAMMING BASICS

Topic : Variables and Data Types


Logical
Complex
Raw
Data
Types
Numeric
Integer
Character

Variables in R Programming

- A variable is a name given to a memory location, which is used to store values in a computer program.
- Variables in R programming can be used to store numbers (real and complex), words, matrices, and even tables.
- R is a dynamically programmed language which means that unlike other programming languages, we do not have to declare the data type of a variable before we can use it in our program

Variables in R Programming

- ✓ It should contain letters, numbers, and only dot or underscore characters.
- ✓ It should not start with a number (eg:- 2iota)
- ✓ It should not start with a dot followed by a number (eg:- .2iota)
- ✓ It should not start with an underscore (eg:- _iota)
- ✓ It should not be a reserved keyword.

Reserved Keywords in R

Following are the reserved keywords in R				
for	in	repeat	while	function
if	else	next	break	TRUE
FALSE	NULL	Inf	NaN	NA
NA_integer_	NA_real_	NA_complex_	NA_character_	...

Reserved Keywords in R

- NA:– Not Available is used to represent missing values.
- NULL:- It represents a missing or an undefined value.
- NaN:– It is a short form for Not a Number(eg:- 0/0).
- TRUE/FALSE: – These are used to represent Logical values.
- Inf :– It denotes Infinity(eg:- 1/0).
- If else, repeat, while, function, for, in, next, and break:– These are Used as looping statements, conditional statements, and functions.
- … :- It is used to pass argument settings from one function to another.

Reserved Keywords in R

- ✓ NA_integer_, NA_real_, NA_complex_, and NA_ character _:- These represent missing values of other atomic types.
- ✓ For example:

x = 15 implicitly assigns a numeric data type to the variable ‘x’.

mystring = “Hello, World!”

Constants in R

- ✓ The entities whose values are fixed are called constants.
- ✓ There are two types of constants in R:

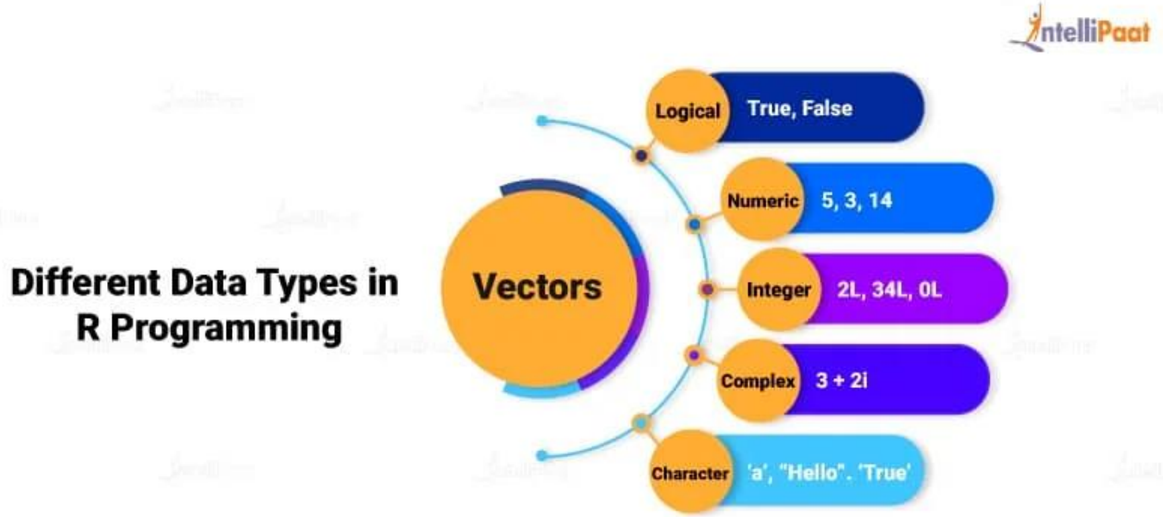
- ✓ Numeric Constants: All numeric values such as integer, double, or complex fall under this category. Numeric constants followed by ‘L’ and ‘i’ are considered as integer and complex respectively. And, numeric constants preceded by 0x/0X are treated as hexadecimal numbers.

- ✓ Character Constants: These constants are represented by single (‘) or double (“) quotes called delimiters.

Data Types in R Programming

- The fundamental or atomic data types in R Programming are as follows:

- Numeric
- Integer
- Complex
- Character
- Logical



Numeric Data Type

- In R, if we assign any decimal value to a variable it becomes a variable of a numeric data type.
- For example, the statement below assigns a numeric data type to the variable “x”.

x = 45.6

- And, the following statement is used to print the data type of the variable “x”:

class(x)

- Output:- [1] "numeric"

Integer Data Type

- ✓ To create an integer variable in R, we need to call the (as.Integer) function while assigning value to a variable.
- ✓ For example:-

e = as.integer(3)

class(e)

- ✓ Output: [1] "integer"
- ✓ Another way of creating an integer variable is by using the L keyword as follows:

x = 5L

class(x)

- ✓ Output: [1] "integer"

Complex Data Type

- ✓ The values containing the imaginary number ‘i’ (iota) are called complex values.
- ✓ The following code gives an error when run:

sqrt(−1)

Output:[1] NaN

Warning message:

In sqrt(−1) : NaNs produced

- ✓ To overcome this error, we coerce the value (−1) into a complex value and denote it as ‘i’.

sqrt(as.complex(−1)

Output:[1] 0+1i

- ✓ For example:

z = 2 + 3i

Character Data Type

- ✓ This data type is used to represent strings.
- ✓ For example:

str1 = "Sam"

class(str1)

Output: [1] "character"

- ✓ We can also use the as.character() function to convert objects into character values.
- ✓ For example:

x = as.character(55.7)

print(x)

Output:[1] "55.7"

class(x)

Output:[1] "character"

Logical Data Type

- ✓ A logical data type stores either of the two values: TRUE -/FALSE. A logical value is often generated when two values are compared.
- ✓ For example:-

x = 3

y = 5

a = x > y

a

Output:

FALSE

Logical Data Type

- ✓ Three standard logical operations., AND(&), OR(|), and NOT(!) yield a variable of the logical data type.
- ✓ For example:-

x= TRUE; y = FALSE

x & y

Output:

[1] FALSE

x | y

Output:

[1] TRUE

!x

Output:

[1] FALSE

R-object

- ✓ Vectors
- ✓ Lists
- ✓ Matrices
- ✓ Arrays
- ✓ Factors
- ✓ Data Frames

Vectors

- When you want to create vector with more than one element, you should use c() function which means to combine the elements into a vector.
- **Live Demo**
- **# Create a vector.**
- **apple <- c('red','green',"yellow")**
- **print(apple)**

- **# Get the class of the vector.**
- **print(class(apple))**
- When we execute the above code, it produces the following result –

- **[1] "red" "green" "yellow"**
- **[1] "character"**

Lists

- A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

- **Live Demo**
- **# Create a list.**
- **list1 <- list(c(2,5,3),21.3,sin)**

- **# Print the list.**
- **print(list1)**
- When we execute the above code, it produces the following result –

- **[[1]]**
- **[1] 2 5 3**

- **[[2]]**
- **[1] 21.3**

- **[[3]]**
- **function (x) .Primitive("sin")**

Matrices

- A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

- **Live Demo**
- **# Create a matrix.**
- **M = matrix(c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)**
- **print(M)**
- When we execute the above code, it produces the following result –

- **[,1] [,2] [,3]**
- **[1,] "a" "a" "b"**
- **[2,] "c" "b" "a"**

Arrays

- While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimension. In the below example we create an array with two elements which are 3x3 matrices each.

- Live Demo
- # Create an array.
- a <- array(c('green','yellow'),dim = c(3,3,2))
- print(a)
- When we execute the above code, it produces the following result –

- , , 1

Arrays

- When we execute the above code, it produces the following result –

- , , 1

- [,1] [,2] [,3]
- [1,] "green" "yellow" "green"
- [2,] "yellow" "green" "yellow"
- [3,] "green" "yellow" "green"

- , , 2

- [,1] [,2] [,3]
- [1,] "yellow" "green" "yellow"
- [2,] "green" "yellow" "green"
- [3,] "yellow" "green" "yellow"

Factors

- Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

- Factors are created using the factor() function. The nlevels functions gives the count of levels.

Factors

- **# Create a vector.**
- **apple_colors <- c('green','green','yellow','red','red','red','green')**

- **# Create a factor object.**
- **factor_apple <- factor(apple_colors)**

- **# Print the factor.**
- **print(factor_apple)**
- **print(nlevels(factor_apple))**
- When we execute the above code, it produces the following result –

- **[1] green green yellow red red red green**
- **Levels: green red yellow**
- **[1] 3**

Data Frames

- Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

- Data Frames are created using the data.frame() function.

Data Frames

```
# Create the data frame.
BMI <-  data.frame(
   gender = c("Male", "Male","Female"),
   height = c(152, 171.5, 165),
   weight = c(81,93, 78),
   Age = c(42,38,26)
)
print(BMI)
```

When we execute the above code, it produces the following result –

```
  gender height weight Age
1   Male  152.0     81  42
2   Male  171.5     93  38
3 Female  165.0     78  26
```

Assessment 1

References

1. João Moreira, Andre Carvalho, Tomás Horvath – “A General Introduction to Data Analytics” – Wiley -2018

Thank You