Data Types, Vector, Matrices and Operators

Course Objectives

GKTCS INNOVATIONS

- Gain a foundational understanding of Business Analytics
- Install R, R-studio, and workspace setup, and learn about the various R packages
- Master R programming and understand how various statements are executed in R

- **Gain an in-depth understanding of data structure used in R and learn to import export data in R**

- **Define, understand and use the various apply functions and DPLYR functions**

- **Shiny Apps and Dashboard**

- **Text Mining and Open NLP Introduction**

R Data Types

GKTCS INNOVATIONS

01 Vectors

02 Lists

03 Matrices

04 DataFrame

05 Factors

One of the key features of R is that it can handle complex statistical operations in an easy and optimised way.

R handles complex computations using:

- **Vector – A basic data structure of R containing the same type of data**
- **Matrices – A matrix is a rectangular array of numbers or other mathematical objects. We can do operations such as addition and multiplication on Matrix in R.**

- **Lists** – Lists store collections of objects when vectors are of same type and length in a matrix.

- **Data Frames** – Generated by combining together multiple vectors such that each vector becomes a separate column.

Vectors in R

GKTCS INNOVATIONS

In R programming, the very basic data types are the R-objects called vectors which hold elements of different classes.

c is function which means to combine the elements into a vector.

```
# Create a vector
apple <- c('red','green',"yellow")
print(apple)

# Get the class of the vector.
print(class(apple))
```

Vectors in R

GKTCS INNOVATIONS

- These data types in R can be logical, integer, double, character complex or raw

- In R using the function, typeof() one can check the data type of vector

- One more significant property of R vector is its length. The function length() determines the number of elements in the vector

```
>c(2, 3, 5) [1] 2 3 5
[1] 2 3 5
>length(c("aa", "bb", "cc", "dd", "ee"))
[1] 5
```

Vectors in R

GKTCS INNOVATIONS

Data Type	Example	Verify

Logical	TRUE, FALSE	`v <- TRUE`
		`print(class(v))`
		
		it produces the following result –
		
		`[1] "logical"`

Numeric	12.3, 5, 999	`v <- 23.5`
		`print(class(v))`
		
		it produces the following result –
		
		`[1] "numeric"`

Vectors in R

GKTCS INNOVATIONS

Data Type	Example	Verify

Integer	2L, 34L, 0L	`v <- 2L`
		`print(class(v))`
		
		it produces the following result –
		
		`[1] "integer"`

Complex	3 + 2i	`v <- 2+5i`
		`print(class(v))`
		
		it produces the following result –
		
		`[1] "complex"`

Vectors in R

Data Type	Example	Verify

Character	'a' , '"good", "TRUE", '23.4'	`v <- "TRUE"`
		`print(class(v))`
		
		it produces the following result –
		
		`[1] "character"`

Raw	"Hello" is stored as 48 65 6c 6c 6f	`v <- charToRaw("Hello")`
		`print(class(v))`
		
		it produces the following result –
		
		`[1] "raw"`

List in R

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.

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

# Print the list.
print(list1)
```

```r
# 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 in R

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

```
# 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 in R

- 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.

```
# Create an array.
a <- array(c('green','yellow'),dim = c(3,3,2))
print(a)
```

Arrays in R

GKTCS INNOVATIONS

```
# 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
[,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 in R

GKTCS INNOVATIONS

- **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 n levels functions gives the count of levels.**

Factors in R

GKTCS INNOVATIONS

```
# 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))
```

o/p

```
[1] green green  yellow red red red green
Levels: green red yellow
# applying the n levels function we can know the number
of distinct values
[1] 3
```

Data Frames in R

GKTCS INNOVATIONS

- **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 in R

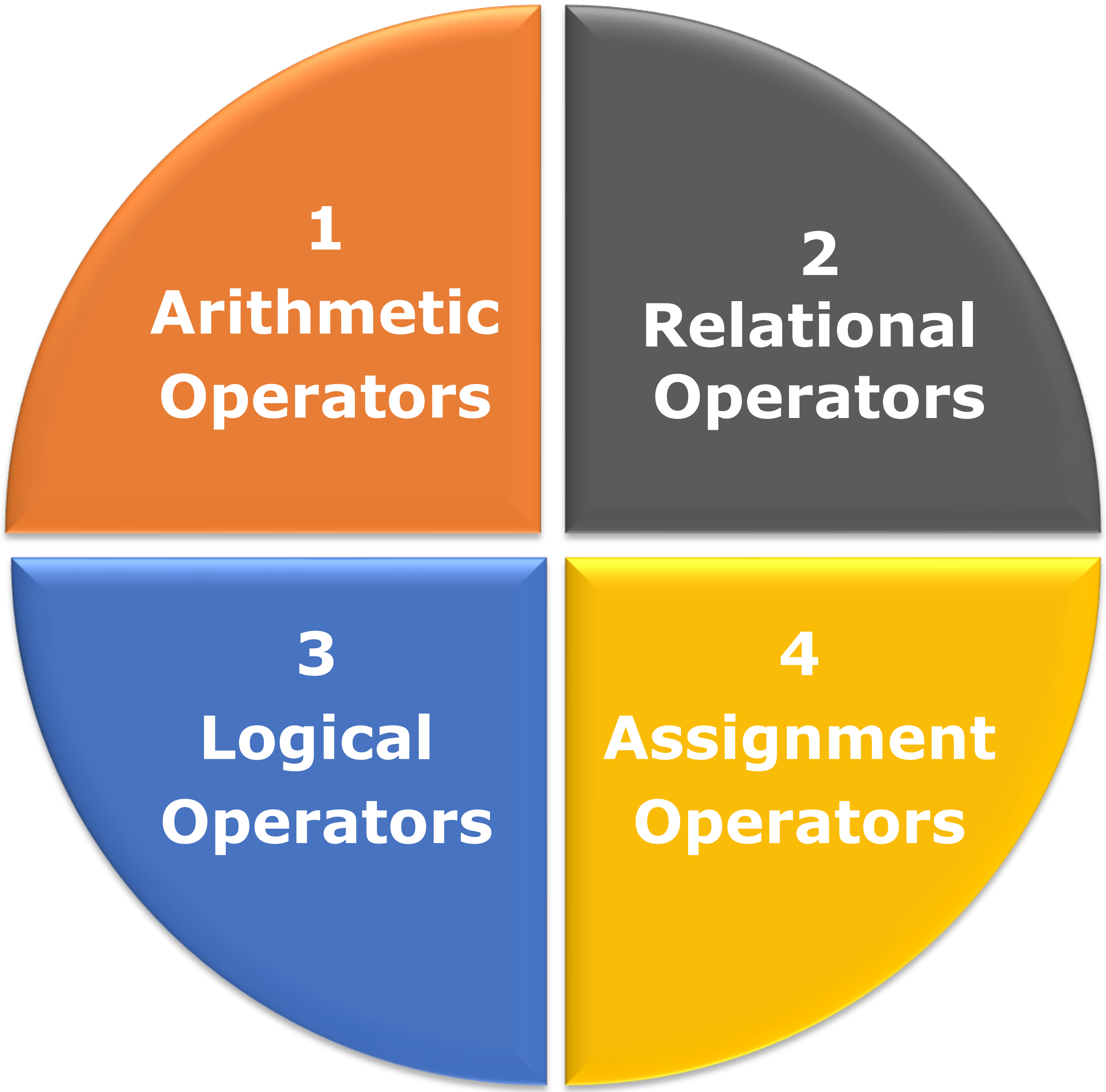
GKTCS INNOVATIONS

```
# 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
```

Operators in R



Arithmetic Operators

GKTCS INNOVATIONS

These operators are used to carry out mathematical operations like addition and multiplication. Here is a list of arithmetic operators available in R.

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Examples

```
>x <- 5
>y <- 16
>x+y
>[1] 21
>x-y
>[1] -11
>x*y
>[1] 80
>y/x
>[1] 3.2
>y%/%x
>[1] 3
>y%%x
>[1] 1
>y^x
>[1] 1048576
```

Relational Operators

GKTCS INNOVATIONS

Relational operators are used to compare between values. Here is a list of relational operators available in R.

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Examples

GKTCS INNOVATIONS

```
>x <- 5
>y <- 16
>x<y
>[1] TRUE
>x>y
>[1] FALSE
>x<=5
>[1] TRUE
> y>=20
>[1] FALSE
>y == 16
>[1] TRUE
>x != 5
>[1] FALSE
```

Operation on Vectors

GKTCS INNOVATIONS

We can use the function c() (as in concatenate) to make vectors in R. All operations are carried out in element-wise fashion. Here is an example.

```
>x <- c(2,8,3)
>y <- c(6,4,1)
>x+y
>[1]  8      12     4
>x>y
>[1]  FALSE   TRUE   TRUE
```

When there is a mismatch in length (number of elements) of operand vectors, the elements in shorter one is recycled in a cyclic manner to match the length of the longer one.

R will issue a warning if the length of the longer vector is not an integral multiple of the shorter vector.

```
>x <- c(2,1,8,3)
>y <- c(9,4)
>x+y # Element of y is recycled to 9,4,9,4
>[1] 11  5  17   7
>x-1 # Scalar 1 is recycled to 1,1,1,1
>[1] 1   0    7    2
>x+c(1,2,3)
>[1] 3   3   11   4
```

Warning message:

In x + c(1, 2, 3) :

longer object length is not a multiple of shorter object length

Logical Operators

GKTCS INNOVATIONS

Logical operators are used to carry out Boolean operations like AND, OR etc.

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

- Operators **&** and **|** perform element-wise operation producing result having length of the longer operand.

- But **&&** and **||** examines only the first element of the operands resulting into a single length logical vector.

- **Zero** is considered FALSE and **non-zero** numbers are taken as TRUE.

```
>x <- c(TRUE,FALSE,0,6)
>y <- c(FALSE,TRUE,FALSE,TRUE)
>!x
>[1] FALSE TRUE TRUE FALSE
>x&y
[1] FALSE FALSE FALSE TRUE
>x&&y
[1] FALSE
>x|y
[1]    TRUE TRUE FALSE TRUE
>x||y
[1] TRUE
```

Assignment Operators

GKTCS INNOVATIONS

- **These operators are used to assign values to variables.**
- **The operators <- and = can be used, almost interchangeably, to assign to variable in the same environment.**
- **The << operator is used for assigning to variables in the parent environments (more like global assignments). The rightward assignments, although available are rarely used.**

Operator	Description
<-, <<-, =	Leftwards assignment
->, ->>	Rightwards assignment

Examples

```
> x <- 5
> x
[1] 5
> x = 9
> x
[1] 9
> 10 -> x
>x [1]
 10
```

GKTCS INNOVATIONS

Examples

```
>Console
# An example
>x <- c(1:10)
>x[(x>8) | (x<5)]
# yields 1 2 3 4 9 10
# How it works
>x <-
>xc(1:10)
1 2 3 4 5 6 7 8 9 10
>x > 8
F F F F F F F F T T
```

Examples

GKTCS INNOVATIONS

```
>x < 5
T T T T F F F F FF
>x > 8 | x < 5
T T T T F F F F T T
>x[c(T,T,T,T,F,F,F,F,T,T)]
1 2 3 4 9 10
```