# Advance Programming Lab(CS-3095)

# KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

## School of Computer Engineering



*2 Credit*

R Programming

# Protocol



- Students should be regular and come prepared for the lab practice.
- In case a student misses a class, it is his/her responsibility to complete the missed experiment(s).
- Students should bring the observation book, prescribed textbook and class note for the reference.
- Once the experiment(s) get executed, they should show the results to the instructors and copy the same in their observation book.
- The experiments have to be implemented using R-Studio.
- Bi-weekly Evaluation of Lab Record
- Written Quiz and Written Viva should be conducted at any point of time and without prior notice.

# Evaluation



## *Evaluation Procedure*

| Type | Mark |
|---|---|
| Continuous Evaluation | 60 |
| End Semester Lab Examination | 40 |

## *Scheme for Continuous Evaluation*

| Type | Mark |
|---|---|
| Record | 20 |
| Written Quiz | 20 |
| Written Viva | 10 |
| Attendance | 10 |

## *Scheme for Continuous Evaluation*

| Type | Mark |
|---|---|
| Program Development | 15 |
| Program Execution | 15 |
| Written Viva | 10 |

# Lab Objectives



- Explore and understand how to use the R documentation
- Expand R by installing R packages
- Read Structured Data into R from various sources
- Understand the different data types and structures in R
- Using R for mathematical operations
- Use of vectorized and matrix calculations
- Write user-defined and looping constructs R functions
- Reshape data to support different analyses

# Lab Outcome



- Able to install and configure R
- Able to install different packages
- Able to import and export files
- Able to perform mathematical operations
- Able to perform operations for matrix and vectors
- Able to perform analysis of data in different perspective.

# Lab Contents



| Sr # | Major and Detailed Coverage Area | Lab# |
|---|---|---|
| 1 | ❑ R-Overview<br>❑ Environment Setup<br>❑ Data Types<br>❑ Variables<br>❑ Operators<br>❑ Basic Syntax | 1 |

# R Overview

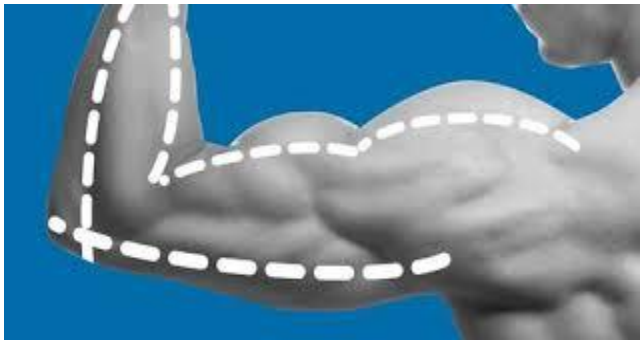


- R is a programming language and software environment for data analysis, statistical analysis, graphics representation and reporting.
- R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand in the early 1990, and is currently developed by the R Development Core Team.
- R is a high-level interpreted computer language (sometimes called scripting language). Everything is pitched toward helping you analyze data.
- It is a GNU project, which means that it is free, open source software.
- R supports a mixture of programming paradigms. At its core, it is an imperative language (you write a script that does one calculation after another), but it also supports object oriented programming (data and functions are combined inside classes) and functional programming (functions are first-class objects; you treat them like any other variable, and you can call them recursively).

# Congratulation !



You've just begun your quest to become an R programmer. So you don't pull any mental muscles, this starts you off gently with a nice warm-up.



# Installation



**https://www.rstudio.com/products/rstudio/download/**

- It is an R-specific IDE. That means that you lose the ability to code (easily) in multiple languages, but you do get some features especially for R.
- Use it if you mainly write R code, don't need advanced editor features, and want a shallow learning curve or the ability to run remotely.

# Your first few Programs



```
# Print Hello World.
print("Hello World")

# Add two numbers.
print(23.9+11.6)

# Mean of 1 to 5
print(mean(1:5))
```

Well done! You've calculated a statistic using R.

# Get Help in R



Before you get started writing R code, the most important thing to know is how to get help. There are lots of ways to do this. Firstly, if you want help on a function or a dataset that you know the name of, type ? followed by the name of the function. To find functions, type two question marks (??) followed by a keyword related to the problem to search. Special characters, reserved words, and multiword search terms need enclosing in double or single quotes.

**For example:**

?mean #opens the help page for the mean function

?"+" #opens the help page for addition

?"if" #opens the help page for if, used for branching code

??plotting #searches for topics containing words like "plotting"

??"regression model" #searches for topics containing phrases like this

# Get Help in R cont'd



The functions help and help.search do the same things as ? and ??, respectively, but with these you always need to enclose your arguments in quotes. The following commands are equivalent to the previous lot:

```
help("mean")

help("+")

help("if")

help.search("plotting")

help.search("regression model")
```

# R Nuts and Bolts



**Entering Input:** At the R prompt we type expressions. The **<-** symbol is the assignment operator.

```
x <- 1

print(x)
```

**Evaluation:** When a complete expression is entered, it is evaluated and the result of the evaluated expression is returned. The result may be auto-printed.

```
x <- 5 ## nothing printed

x ## auto-printing occurs

print(x) ## explicit printing
```

[1] 5 => The [1] shown in the output indicates that x is a vector and 5 is its first element.

# R- Data Types



Generally, while doing programming in any programming language, you need to use various variables to store various information. Variables are nothing but reserved memory locations to store values. This means that, when you create a variable you reserve some space in memory.

You may like to store information of various data types like character, integer, floating point, double floating point, boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. In contrast to other programming languages like C and java in R, the variables are not declared as some data type. **The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable.** There are many types of R-objects. The frequently used ones are:

**Vectors, Lists, Matrices, Arrays, Factors, Data Frames**

# Vector Object



The simplest of these objects is the vector object and there are six data types of these atomic vectors, also termed as six classes of vectors.

| Data Type | Example | Verify | Output |
|---|---|---|---|
| Logical | TRUE, FALSE | v <- TRUE<br>print(class(v)) | [1] "logical" |
| Numeric | 12.3, 5, 999 | v <- 23.5<br>print(class(v)) | [1] "numeric" |
| Integer | 2L, 34L, 0L | v <- 2L<br>print(class(v)) | [1] "integer" |
| Complex | 3 + 2i | v <- 5+10i<br>print(class(v)) | [1] “complex" |
| Character | 'a' , "good", "TRUE", '23.4' | v <- “TRUE”<br>print(class(v)) | [1] “character" |
| Raw | "Hello" is stored as 48 65 6c 6c 6f | v <- charToRaw (“TRUE”)<br>print(class(v)) | [1] “raw" |

# Vector Object cont'd



When the vector to be created with more than one element, c() function should be used 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))
```

# List Object



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)
```

**Output:-** 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 Object



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)
```

**Output:-** When we execute the above code, it produces the following result:

```
     [,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
```

# Arrays Object



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)
```

**Output:-** 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 Object



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.

```
# 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))
```

# Data Frames Object



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.

```
# 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)
```

# R-Variables



A variable provides named storage that the programs can manipulate. A variable in R can store an atomic vector, group of atomic vectors or a combination of many R - objects. A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

## *Variable Assignment*

The variables can be assigned values using leftward, rightward and equal to operator.

```
# Assignment using equal operator.
var.1 = c(0,1,2,3)

# Assignment using leftward operator.
var.2<-c("learn","R")

# Assignment using rightward operator.
c(TRUE,1)->var.3
```

The values of the variables can be printed using print() or cat() function. The cat() function combines multiple items into a continuous print output.

```
print(var.1)
cat("var.1 is ",var.1,"\n")
```

# R-Variables cont'd



## Data Type of a Variable

In R, a variable itself is not declared of any data type, rather it gets the data type of the R - object assigned to it. So R is called a **dynamically typed language**, which means that we can change a variable's data type of the same variable again and again when using it in a program.

```
var_x <-"Hello"
cat("The class of var_x is ",class(var_x),"\n")

var_x <-34.5
cat("Now the class of var_x is ",class(var_x),"\n")

var_x <-27L
cat("   Next the class of var_x becomes ",class(var_x),"\n")
```

## Finding Variables

To know all the variables currently available in the workspace we use the ls() function. E.g. print(ls())

## Deleting Variables

Variables can be deleted by using the rm() function. E.g. rm(var.3)

# R-Operators



An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. R language is rich in built-in operators and provides following types of operators.

*Types of Operators*

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Miscellaneous Operators

# R-Operators : Arithmetic Operators



```
v <-c(2,5.5,6)

t <-c(8,3,4)

print(v+t)

print(v-t)

print(v*t)

print(v/t)

print(v%%t)  # Give the remainder of the first  vector  with the second

print(v%/%t)  #  Give the result  of division of first vector with second (quotient)

print(v^t) # The first vector raised to the exponent of second vector
```

# R-Operators cont'd



*Relational Operators*

```
v <-c(2,5.5,6)
t <-c(8,3,4)
print(v>t)
print(v<t)
print(v==t)
print(v<=t)
print(v>=t)
print(v!=t)
```

*Logical Operators*

```
v <-c(2,TRUE,6)
t <-c(8,FALSE,4)
print(v&t) # Element - wise Logical AND operator
print(v|t) # Element - wise Logical OR operator
print(!v)
print(v&&t) # Logical AND operator
print(v||t) # Logical OR operator
```

# R-Operators cont'd



### Left Assignment Operator

```
v <-c(2,5.5,6)

t <<-c(8,3,4)

u = c(18,13,14)

print(v)

print(t)

print(u)
```

### Right Assignment Operator

```
c(2,5.5,6) -> v

c(8,3,4) ->> t

print(v)

print(t)
```

### Miscellaneous Operator

```
v <-2:8
print(v)
```

Colon operator: It creates the series of numbers in sequence for a vector

```
v1 <-8
v2 <-12
t <-1:10
print(v1 %in% t)
print(v2 %in% t)
```

%in% - This operator is used to identify if an element belongs to a vector.

# Thank You
# End of Lab 1

# Assignment



1. Initialize some variables in the R workspace. Now analyze and display what are variables are created, then delete the single variable as well as all the created variables.
2. Initialize some variables with different types of value. Now analyze what is the type of those variables.
3. Write an R-script to initialize your roll no., name and branch then display all the details.
4. Write an R-script to initialize two variables, then find out the sum, multiplication, subtraction and division of them.
5. Write an R-script to enter a 3-digits number from the keyboard, then find out sum of all the 3-digits.
6. Write an R-script to enter the radius of a circle, then calculate the area and circumference of the circle.
7. Write an R-script to calculate the compound interest of the given P, T, R.
8. Write an R-script to enter two numbers from the keyboard, then swap them without using 3rd variable.

# Assignment



9. Write an R-script to enter two numbers and implement all the relational operators on that two numbers.
10. Write an R-script to convert given paisa into its equivalent rupee and paisa as per the following format. 550 paisa = 5 Rupee and 50 paisa.
11. Write an R-script to convert given second into its equivalent hour, minute and second as per the following format. Example. 7560 second = 2 Hour, 27 Minute and 40 Second.
12. Write an R-script to convert a quantity in meter entered through keyboard into its equivalent kilometer & meter as per the following format. Example - 2430 meter = 2 Km and 430 meter.
13. A cashier has currency notes of denominations 10, 50 and 100. If the amount to be withdrawn is input through the keyboard in hundreds, write an R-script to find the total number of currency notes of each denomination the cashier will have to give to the withdrawer.
14. Ramesh's basic salary is input through the keyboard. His dearness allowance is 40% of basic salary, and house rent allowance is 20% of basic salary. Write an R-script to calculate his gross salary.