

Programming to



School of Computer Engineering,
KIIT University

26.6.07

Lab Objectives

4

- ☐ 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

Evaluation Procedure

Type	Mark
Continuous Evaluation	60
End Semester Lab Examination	40

Internal	Before Mid Sem	After Mid Sem
1	Program Execution (10)	Program Execution (10)
2	Record (5)	Record (5)
3	Quiz (20)	
4	Viva (10)	

Lab Objectives

4

- ☐ 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

5

- ☐ 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



6

Sr #	Major and Detailed Coverage Area	Lab#
1	<ul style="list-style-type: none"><input type="checkbox"/> R-Overview<input type="checkbox"/> Environment Setup<input type="checkbox"/> Data Types<input type="checkbox"/> Variables<input type="checkbox"/> Operators<input type="checkbox"/> Basic Syntax	1

R Overview



7

- ❑ 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.



What R is and what it is not

■ R is

- ☐ a programming language
- ☐ a statistical package
- ☐ an interpreter
- ☐ Open Source

■ R is not

- ☐ a database
- ☐ a collection of “black boxes”
- ☐ a spreadsheet software package
- ☐ commercially supported

Installation



9

<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.

Using RStudio



The screenshot shows the RStudio desktop environment with a red border. It is divided into four main panes:

- Script editor:** The top-left pane containing R code. A callout bubble points to it with the text "Script editor".
- Workspace and History:** The top-right pane showing variables in the workspace and a history of executed commands. A callout bubble points to it with the text "View variables in workspace and history file".
- Console:** The bottom-left pane showing the output of the R code. A callout bubble points to it with the text "View help, plots & files; manage packages".
- R Documentation:** The bottom-right pane showing the documentation for the 'complex' package. A callout bubble points to it with the text "R console".

The code in the script editor is as follows:

```
1 x=matrix(1:12, 3, 4)
2 # plot(x)
3 plot(x)
4 # apply
5 # sum
6 # })
7 y=matrix(1:12, 4, 3)
8
9 z=as.data.frame(x)
10 ?mean
11 ?sqrt
12 ?factor
13 mean(5,6)
14 sqrt(c(9,16))
15 length(paste("hello","you"))
16 length(c("hello","you"))
17 b=1:6
```

The console output is as follows:

```
> length(c("hello","you"))
[1] 3
> length(paste("hello","you"))
[1] 1
> paste("hello","you")
[1] "hello you"
> length(c("hello","you"))
[1] 2
> length(c("hello","you"))
[1] 2
> b=1:6
fix(b)
fix(b)
> fix(b)
> View(y)
```

The R Documentation pane shows the following content:

R: Complex Vectors

complex (base)

Complex Vectors

Description

Basic functions which support complex arithmetic in R.

Usage

```
complex(length.out = 0, real = numeric(), imaginary = numeric(),
        modulus = 1, argument = 0)
as.complex(x, ...)
is.complex(x)
```

Re(z)

Im(z)

Mod(z)

Arg(z)

Arguments

length.out numeric. Desired length of the output vector, inputs being recycled as needed.

real numeric vector.

Naming Convention

- must start with a letter (A-Z or a-z)
- can contain letters, digits (0-9), and/or periods “.”
- case-sensitive
 - `mydata` different from `MyData`
- do not use underscore “_”
- To quit R, use `>q()`

Examples

```
# Print Hello World.
```

```
print("Hello World")
```

```
# Add two numbers.
```

```
print(23.9+11.6)
```

```
# Mean of 1 to 5
```

```
print(mean(1:5))
```

Get Help in R



11

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



12

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



13

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



14

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



15

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	<pre>v <- TRUE print(class(v))</pre>	[1] "logical"
Numeric	12.3, 5, 999	<pre>v <- 23.5 print(class(v))</pre>	[1] "numeric"
Integer	2L, 34L, 0L	<pre>v <- 2L print(class(v))</pre>	[1] "integer"
Complex	3 + 2i	<pre>v <- 5+10i print(class(v))</pre>	[1] "complex"
Character	'a' , "good", "TRUE", '23.4'	<pre>v <- "TRUE" print(class(v))</pre>	[1] "character"
Raw	"Hello" is stored as 48 65 6c 6c 6f	<pre>v <- charToRaw ("TRUE") print(class(v))</pre>	[1] "raw"

Vector Object cont'd



16

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))
```

R-Variables



22

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



23

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



24

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



25

```
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

26

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&&v) # Logical AND operator
```

```
print(v||v) # Logical OR operator
```

R-Operators cont'd



27

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.
```




List Object



17

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.
```

```
# Print the list.
```

```
list1 <-list(c(2,5,3),21.3,sin)    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")
```



Assignment

- "<-" used to indicate assignment

```
x<-c(1,2,3,4,5,6,7)
```

```
x<-c(1:7)
```

```
x<-1:4
```

- *note: as of version 1.4 "=" is also a valid assignment operator*

R as a calculator

```
> 5 + (6 + 7) * pi^2  
[1] 133.3049
```

```
> log(exp(1))  
[1] 1
```

```
> log(1000, 10)  
[1] 3
```

```
> sin(pi/3)^2 + cos(pi/3)^2  
[1] 1
```

```
> Sin(pi/3)^2 + cos(pi/3)^2  
Error: couldn't find function "Sin"
```

R as a calculator

```
> log2(32)
```

```
[1] 5
```

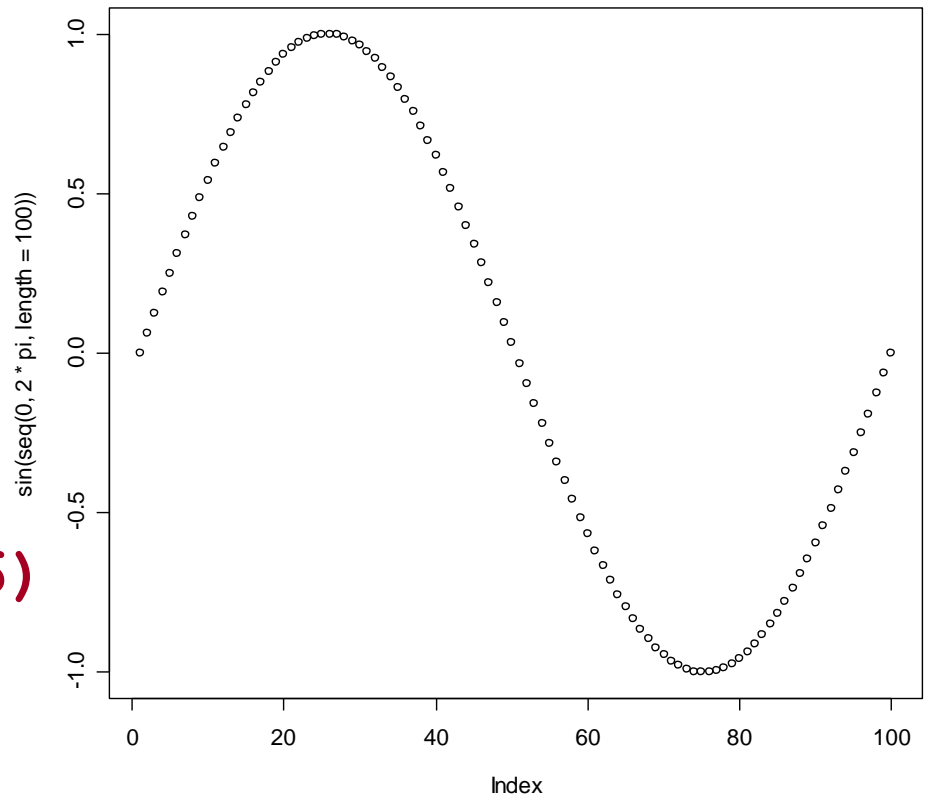
```
> sqrt(2)
```

```
[1] 1.414214
```

```
> seq(0, 5, length=6)
```

```
[1] 0 1 2 3 4 5
```

```
> plot(sin(seq(0, 2*pi, length=100)))
```



R-Variables



22

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")
```



Variables

- A variable is a symbolic name given to stored information
- Variables are assigned using either "=" or "<-"

```
> x<-12.6
```

```
> x
```

```
[1] 12.6
```


Missing values

- R is designed to handle statistical data and therefore predestined to deal with missing values

- Numbers that are "not available"

```
> x <- c(1, 2, 3, NA)
```

```
> x + 3
```

```
[1] 4 5 6 NA
```

- "Not a number"

```
> log(c(0, 1, 2))
```

```
[1] -Inf 0.0000000 0.6931472
```

```
> 0/0
```

```
[1] NaN
```



Data Types

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

Basic (atomic) data types

■ Logical

```
> x <- T;    y <- F
```

```
> x; y
```

```
[1] TRUE
```

```
[1] FALSE
```

■ Numerical

```
> a <- 5; b <-  
sqrt(2)
```

```
> a; b
```

```
[1] 5
```

```
[1] 1.414214
```

■ Character

```
> a <- "1"; b <- 1
```

```
> a; b
```

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

```
[1] 1
```

```
> a <- "character"
```

```
> b <- "a"; c <- a
```

```
> a; b; c
```

```
[1] "character"
```

```
[1] "a"
```

```
[1] "character"
```

R-Operators : Arithmetic Operators



25

```
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

26

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&&v) # Logical AND operator
```

```
print(v||v) # Logical OR operator
```

R-Operators cont'd



27

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.

R Program to Take Input From User

- **readline()** function to take input from the user (terminal).
- This function will return a single element character vector.

Example

```
my.name <- readline(prompt="Enter name: ")
```

```
my.age <- readline(prompt="Enter age: ")
```

```
# convert character into integer
```

```
my.age <- as.integer(my.age)
```


```
print(paste("Hi,", my.name, "next year you will be",  
my.age+1, "years old."))
```


character vector into integer using the function as.integer().

prompt argument is printed in front of the user input. It usually ends on ": ".

Bar plot

```
marks = c(70, 95, 80, 74)
barplot(marks, main = "Comparing marks of 5
  subjects",
xlab = "Marks",
ylab = "Subject",
names.arg = c("English", "Science", "Math.", "Hist."),
col = "darkred", horiz = FALSE)
```


- 
1. Write a R program to take input from the user (name and age) and display the values.
 2. Write an R-script to initialize your rollno., name and branch then display all the details.
 3. Write an R-script to initialize two variables, then find out the sum, multiplication, subtraction and division of them.
 4. Write an R-script to enter a 3-digits number from the keyboard, then find out sum of all the 3-digits.
 5. Write an R-script to enter the radius of a circle, then calculate the area and circumference of the circle.

- 
6. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.
 7. Write a R program to create a vector which contains 10 random integer values between -50 and +50.
 8. Write a R program to find the maximum and the minimum value of a given vector
 9. Write a R program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type.
 10. Write a R program to compute sum, mean and product of a given vector elements.
 11. 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.