Intoduction to R

Course Name & Code: MAT 2001- Statistics for Engineers (Lab)

Objectives:

This course provides the knowledge to Install and use R for simple programming tasks, extended R libraries and packages. Which helps to Develop R Programs using Looping Constructs and R mathematical functions that can be used for data exploration in R.

Course Outcomes:

At the end of this course student will be able to

CO1: Master the use of the R interactive environment.
CO2: Expand R by installing R packages.
CO3: Develop Loop constructs in R.
CO4: Use R for descriptive statistics.
CO5: Use R for inferential statistics

Reference Books

1. R Cookbook, Paulteetor, orielly.
2. R in action,Rob Kabacoff,Manning.

About R

R is a free, open-source software and programming language developed in 1995 at the University of Auckland as an environment for statistical computing and graphics (Ikaha and Gentleman, 1996). Since then R has become one of the dominant software environments for data analysis and is used by a variety of scientific disiplines, including soil science, ecology, and geoinformatics (Envirometrics CRAN Task View; Spatial CRAN Task View). R is particularly popular for its graphical capabilities, but it is also prized for it's GIS capabilities which make it relatively easy to generate raster-based models. More recently, R has also gained several packages which are designed specifically for analyzing soil data.

1. a software environment:

- statistics
- graphics
- programming
- calculator
- GIS
- etc…

2. a language to explore, summarize, and model data

- functions = verbs
- objects = nouns

To Install R and R Packages

1. Open an internet browser and go to www.r-project.org.
2. Click the "download R" link in the middle of the page under "Getting Started."
3. Select a CRAN location (a mirror site) and click the corresponding link.
4. Click on the "Download R for WINDOWS" link at the top of the page.
5. Click on the file containing the latest version of R under "Files."
6. Save the .pkg file, double-click it to open, and follow the installation instructions.
7. Now that R is installed, you need to download and install RStudio.

To Install RStudio

1. Go to www.rstudio.com and click on the "Download RStudio" button.
2. Click on "Download RStudio Desktop."
3. Click on the version recommended for your system, or the latest Mac version, save the .dmg file on your computer, double-click it to open, and then drag and drop it to your applications folder.

DATA TYPES

You may like to store information of various data types like character, wide 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.

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:** A basic data structure of R containing the same type of data.
- **Matrices:** A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.
- **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 modelling.
- **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.

- **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.

Modes

All objects have a certain *mode*. Some objects can only deal with one *mode* at a time, others can store elements of multiple *modes*. R distinguishes the following modes:

1. **integer**: integers (e.g. 1, 2 or -69)
2. **numeric**: real numbers (e.g 2.336, -0.35)
3. **complex**: complex or imaginary numbers
4. **character**: elements made up of text-strings (e.g. "text", "Hello World!", or "123")
5. **logical**: data containing logical constants (i.e. TRUE and FALSE)

By *atomic*, we mean the vector only holds data of a single type.

R PROGRAMMING

- **character**: "a", "swc"
- **numeric**: 2, 15.5
- **integer**: 2L (the L tells R to store this as an integer)
- **logical**: TRUE, FALSE
- **complex**: 1+4i (complex numbers with real and imaginary parts)

R provides many functions to examine features of vectors and other objects, for example

- class( ) - what kind of object is it (high-level)?
- typeof( ) - what is the object's data type (low-level)?
- length( ) - how long is it? What about two dimensional objects?

1. Use R to calculate the following:

   I. 31 * 78

      Sol: > 31*78

      [1] 2418

   II. 697 / 41

      Sol: > 697 / 41

      [1] 17

2. Assign the value of 39 to x

   Sol: > x<-39

   > x

   [1] 39

3. Assign the value of 22 to y

   Sol: > y<-22

   > y

   [1] 22

4. Make z the value of x - y

   Sol: > z<- x - y

5. Display the value of z in the console

   Sol: > z

   [1] 17

6. Calculate the square root of 2345, and perform a log2 transformation on the result.

   Sol : > log2(sqrt(2345))

   [1] 5.597686

7. Type the following code, which assigns numbers to objects x and y.

   x <- 10 y <- 20

   I. Calculate the product of x and y.

      Sol: > x<-10

      > y<-20

      > x*y

      [1] 200

   II. Store the result in a new object called z.

      Sol: > z<-x*y

      > z

      [1] 200

Numeric Functions

Function	Description
abs(x)	absolute value
sqrt(x)	square root
ceiling(x)	ceiling(3.475) is 4
floor(x)	floor(3.475) is 3
trunc(x)	trunc(5.99) is 5
round(x, digits=n)	round(3.475, digits=2) is 3.48
signif(x, digits=n)	signif(3.475, digits=2) is 3.5
cos(x), sin(x), tan(x)	also acos(x), cosh(x), acosh(x), etc.
log(x)	natural logarithm
log10(x)	common logarithm
exp(x)	e^x

Vectors: A basic data structure of R containing the same type of data.

Creating Vector

Vectors are generally created using the c() function.

Since, a vector must have elements of the same type, this function will try and coerce elements to the same type, if they are different.

Coercion is from lower to higher types from logical to integer to double to character.

```
> x <- c(1, 5, 4, 9, 0)

> typeof(x)

[1] "double"

> length(x)

[1] 5

> x <- c(1, 5.4, TRUE, "hello")

> x

[1] "1"     "5.4"   "TRUE"  "hello"
```

```
> typeof(x)

[1] "character"
```

If we want to create a vector of consecutive numbers, the : operator is very helpful.

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

Creating Matrices

To create matrices we will use the matrix() function. The matrix() function takes the following arguments:

• data an R object (this could be a vector).
• nrow the desired number of rows.
• ncol the desired number of columns.
• byrow a logical statement to populate the matrix by either row or by column.

Creation of matrix

a) matrix1 <- matrix (data = 1, nrow = 3, ncol = 3)
Sol:

```
> matrix1 <- matrix ( data = 1, nrow = 3, ncol = 3)
> matrix1
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
```

c) v1<- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)
Sol:

```
> v1<- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)
```

Manipulation of Matrix

f) matrix1

Sol:

```
> matrix1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

g) matrix1[1,3]

Sol:

```
> matrix1[1, 3]
[1] 7


[4,]    4    8
```

e) matrix1 = matrix(1:9, nrow = 3) matrix1 + 2

Sol:

```
> matrix1 = matrix(1:9, nrow = 3)
> matrix1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> matrix1+2
     [,1] [,2] [,3]
[1,]    3    6    9
[2,]    4    7   10
[3,]    5    8   11
```

Manipulation of Matrix

f) matrix1

Sol:

```
> matrix1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

g) matrix1[1,3]

Sol:

```
> matrix1[1, 3]
[1] 7
```

h) matrix1[2,]

Sol:

```
> matrix1[ 2, ]
[1] 2 5 8
```

i) matrix1[,-2]

Sol:

```
> matrix1[,-2]
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
```

j) matrix1[1,1] = 15

Sol:

```
> matrix1[1, 1] = 15
> matrix1
     [,1] [,2] [,3]
[1,]   15    4    7
[2,]    2    5    8
[3,]    3    6    9
```

k) matrix1[,2] = 1

Sol:

```
> matrix1
     [,1] [,2] [,3]
[1,]   15    1    7
[2,]    2    1    8
[3,]    3    1    9
```

Mathematical Operations

R can do matrix arithmetic. Below is a list of some basic operations we can do.

- + - * / standard scalar or by element operations
- %*% matrix multiplication
- t() transpose
- solve() inverse
- det() determinant
- chol() cholesky decomposition
- eigen() eigenvalues and eigenvectors
- crossprod() cross product.

Exercise:

Construct the Matrix

(a)

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{pmatrix}$$

SOL:

```
> m<-matrix(nrow=2,ncol=4,data=c(1,3,5,7,2,4,6,8) , byrow=TRUE)
> m
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

b) Calculate Transpose.

Sol:

```
> t(m)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
```

c) Calculate Inverse.

Sol:

```
> solve(m)
Error in solve.default(m) : 'a' (2 x 4) must be square

> m<-matrix(nrow=3,ncol=3,data=c(1,3,5,7,2,4,6,8,9) , byrow=TRUE)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    7    2    4
[3,]    6    8    9
```

d) Calculate the determinant

sol:

```
> det(m)
[1] 89
```

e) Calculate the Multiplication of the matrix.
Sol:

```
> m1<-m%*%m
> m1
     [,1] [,2] [,3]
[1,]   52   49   62
[2,]   45   57   79
[3,]  116  106  143
>
```

f) Construct a matrix with 10 columns and 10 rows, all filled with random numbers between 0 and 100.
Sol:

```
m <- matrix(runif(100), ncol=10)
```

g) Calculate the row means of this matrix (*Hint:* use rowMeans). Also calculate the standard deviation across the row means (now also use sd().
Sol:

```
> m1<-rowMeans(m)
> m1
 [1] 0.3885344 0.6758386 0.4342555 0.5735385 0.5112892
0.4370579 0.4852983
 [8] 0.6234814 0.6275129 0.7056754
> sd(m1)
[1] 0.1104536
```

Factors:

Factor is a data structure used for fields that takes only predefined, finite number of values (categorical data).

For example, a data field such as marital status may contain only values from single, married, separated, divorced, or widowed.

In such case, we know the possible values beforehand and these predefined, distinct values are called levels. Following is an example of factor in R.

```
> x

[1] single  married married single

Levels: married single
```

Here, we can see that factor x has four elements and two levels. We can check if a variable is a factor or not using class() function.

Similarly, levels of a factor can be checked using the levels() function.

```
> class(x)

[1] "factor"

> levels(x)

[1] "married" "single"
```

Creating factor in R?

We can create a factor using the function factor(). Levels of a factor are inferred from the data if not provided.

```
> x <- factor(c("single", "married", "married", "single"));
```

```
> x

[1] single  married married single

Levels: married single
```

Q.1 If x = c(1, 2, 3, 3, 5, 3, 2, 4, NA), what are the levels of factor(x)?
Sol:

```
> x = c(1, 2, 3, 3, 5, 3, 2, 4, NA)

> levels(factor(x))

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

Q.2 Let x <- c(11, 22, 47, 47, 11, 47, 11). If an R expression factor(x, levels=c(11, 22, 47), ordered=TRUE) is executed, what will be the 4th element in the output?

Sol:

```
> x <- c(11, 22, 47, 47, 11, 47, 11)
> factor(x, levels=c(11, 22, 47), ordered=TRUE)
[1] 11 22 47 47 11 47 11
Levels: 11 < 22 < 47
```

Q.3 If z <- c("p", "a" , "g", "t", "b"), then What is the R expression will replace the third element in z with "b".

Sol:

```
> z <- c("p", "a" , "g", "t", "b")
> z[3] <- "b"
> z
[1] "p" "a" "b" "t" "b"
```

Q.4 If z <- factor(c("p", "q", "p", "r", "q")) and levels of z are "p", "q" ,"r", write an R expression that will change the level "p" to "w" so that z is equal to: "w", "q" , "w", "r" , "q".

Sol:

```
> z <- factor(c("p", "q", "p", "r", "q"))
> levels(z)[1] <- "w"
> z
[1] w q w r q
Levels: w q r
```

Q.5 If: s1 <- factor(sample(letters, size=5, replace=TRUE)) and s2 <- factor(sample(letters, size=5, replace=TRUE)), write an R expression that will concatenate s1 and s2 in a single factor with 10 elements.

Sol:

```
> s1 <- factor(sample(letters, size=5, replace=TRUE))
> s2 <- factor(sample(letters, size=5, replace=TRUE))
```

```
> factor(c(levels(s1)[s1], levels(s2)[s2]))
 [1] c e q l t v b k t c
Levels: b c e k l q t v
```

Q.6 Consider the factor responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "Agree")), with the following output:

Sol:

```
> responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "Agr
ee"))
> responses
[1] Agree   Agree    Strongly Agree Disagree      Agree
Levels: Agree Disagree Strongly Agree
```

Q.7 If x <- factor(c("high", "low", "medium", "high", "high", "low", "medium")), write an R expression that will provide unique numeric values for various levels of x with the following output:

Sol:

```
> x <- factor(c("high", "low", "medium", "high", "high", "low", "medium"))
> data.frame(levels = unique(x), value = as.numeric(unique(x)))
  levels value
1   high     1
2    low     2
3 medium     3
```

Data Frames

Data frame is a two dimensional data structure in R. It is a special case of a list which has each component of equal length.

Each component form the column and contents of the component form the rows.

Creating Data Frame in R

We can create a data frame using the data.frame() function.

For example, the above shown data frame can be created as follows.

```
> x <- data.frame("SN" = 1:2, "Age" = c(21,15), "Name" = c("John","Dora"))

> str(x)   # structure of x

'data.frame':  2 obs. of  3 variables:

$ SN  : int  1 2

$ Age : num  21 15

$ Name: Factor w/ 2 levels "Dora","John": 2 1
```

Notice above that the third column, Name is of type factor, instead of a character vector.

By default, data.frame() function converts character vector into factor.

```
$ SN  : int  1 2

$ Age : num  21 15

$ Name: chr  "John" "Dora"
```

Many data input functions of R like, read.table(), read.csv(), read.delim(), read.fwf() also read data into a data frame.

LISTS

List is a data structure having components of mixed data types.

A vector having all elements of the same type is called atomic vector but a vector having elements of different type is called list.

We can check if it's a list with typeof() function and find its length using length().

Creating a list

List can be created using the list() function.

```
> x <- list("a" = 2.5, "b" = TRUE, "c" = 1:3)
```

Here, we create a list x, of three components with data types double, logical and integer vector respectively.

Its structure can be examined with the str() function.

```
> str(x)
```

We can create the same list without the tags as follows. In such scenario, numeric indices are used by default.

```
> x <- list(2.5,TRUE,1:3)

> x
```

Q.1 If: p <- c(2,7,8), q <- c("A", "B", "C") and x <- list(p, q), then what is the value of x[2]?

Sol:

```
p <- c(2,7,8)

q <- c("A", "B", "C")
```

```
x <- list(p, q)
x[2]
 [[1]]
[1] "A" "B" "C"
```

Q.2 If: w <- c(2, 7, 8) v <- c("A", "B", "C") x <- list(w,v), then which R statement will replace "A" in x with "K".

Sol:

```
w <- c(2, 7, 8)
v <- c("A", "B", "C")
x <- list(w, v)
x[[2]][1] <- "K"
x

[[1]]
[1] 2 7 8

[[2]]
[1] "K" "B" "C"
```

Q.3 If a <- list ("x"=5, "y"=10, "z"=15), which R statement will give the sum of all elements in a?

Sol:

```
a <- list ("x"=5, "y"=10, "z"=15)
sum(unlist(a))

[1] 30
```

Viva Voice Questions and Answers- Cycle-I

1. **What is R?**

   R is a programming language which is used for developing statistical software and data analysis. It is being increasingly deployed for machine learning applications as well.

2. **How R commands are written?**

   By using # at the starting of the line of code like #division commands are written.

3. **What is t-tests() in R?**

   It is used to determine that the means of two groups are equal or not by using t.test() function.

4. **What are the disadvantages of R Programming?**

   The disadvantages are:-

   - Lack of standard GUI
   - Not good for big data.
   - Does not provide spreadsheet view of data.

5. **What is the use of With () and By () function in R?**

   with() function applies an expression to a dataset.

   #with (data, expression)

   By() function applies a function t each level of a factors.

   #by (data, factor list, function)