Introduction to R

Deepak R. Bharti
Ph.D. Bioinformatics
SCIS, JNU
New Delhi

Session - I

1. **What is R ?**
2. **Interfaces to R, GUIs**
3. **Rstudio- An IDE for R**
4. **R-cran packages**

What is R ?

- Free and Open Source Software
- Based On S and developed by Ross Ihaka and Robert Gentleman, University of Auckland
- Software for Statistical Data Analysis
- Interpreted Language
- Data Storage, analysis and visualization
- Available on all platforms
- Supports command line as well as GUI

Interfaces to R

- The most commonly used graphical IDE for R is **Rstudio**.
- Rattle
- Rcommander
- RKWard
- R-Jupyter
- Other includes Netbeans, Eclipse etc.

Rstudio IDE

- IDE (integrated development environment) provides comprehensive facilities to computer programmers for software development.
- An IDE generally consists of a source code editor, build automation tools, and a debugger.
- Some advance IDE also contains compiler Interprester as well. eg. Netbeans, Eclipse Etc.

- Installation
  - A. Installation of R-core and R-base
    - Windows : https://cran.r-project.org/bin/windows/base/
    - Mac : http://cran.us.r-project.org/
    - Linux:
      - Debian/Ubuntu :
        - Sudo deb http://ftp.iitm.ac.in/ubuntu/ YOUR_UBUNTU_VERSION_HERE main
        - sudo apt-get install r-base
      - Redhat/centOS :
        - sudo yum install epel-release
        - Sudo yum update && sudo yum install r-base
      - Source code
        - *.tar.gz or *.zip file : make

- B. Installtion of R-studio
  - https://www.rstudio.com/products/rstudio/download/
  - Windows, Mac, Fedora, Ubuntu

R-cran packages

- Installation
  - Using R
    - install.packages(<PACKAGE>)
  - Command Line (UNIX based system)
    - R CMD INSTALL <PACKAGE.tar.gz>
  - Rstudio Utility
  - Rtools (windows user)
    - http://jtleek.com/modules/01_DataScientistToolbox/02_10_rtools/#1

Basics in R

- Functional Programming
  - Everything done through functions
  - Strict named arguments
  - Abbreviations supports(eg. T for TRUE, F for False)

- Object Oriented
  - Everything is an object
  - “<-” is an assignment operator

- Getting Help

  - Using “?”
  - help(“name/keyword”)
  - help.start()
  - example(“keyword”)
  - Stack-Exchange
  - Active mailing list
  - Archives

  etc .....

Few common commands

- For finding current directory : getwd()
- For setting of directory : setwd()
- List files present in directory : dir()
- List obejects in workshape : ls()
- Remove objects : rm(OBJ)
- Read delimated file : read.table(),read.csv()
- Read data from url : read.data.url()
- See type/class of object : typeof()
- List data types of data matrix/ data frame : str(OBJ)
- Check dimension of data : dim(OBJ)
- Length of object : length(OBJ)

Session - II

1. **Vectors in R**
2. **Matrices in R**

Data types in R

- R Supports virtually any type of data
- Numbers, characters, logicals (TRUE/ FALSE)
- Arrays of virtually unlimited sizes
- Vectors (numerical, character, logical) and Matrices
- Lists: Can Contain mixed type variables
- Data Frame: Rectangular Data Set
- Factors : variable which are nominal

	Linear	Rectangular
All elements are of similiar type	Vectors	Matrix
Mixed	List	Dataframe

Vectors

- All elements are of similar type (numerical, character, logical)

  - a <- c(4,2,8,4.3,1,6,-21,8,0) # numeric vector
  - b <- c("one","two","three") # character vector
  - c <-c(TRUE,TRUE,TRUE,FALSE) #logical vector

- Exrercise :
  - See what happens when you mixed data types in vectors. Does data types changes ?
  - If yes who it will effect on your analysis ?
  - eg.
    - a <- c(1,2,3,4)
    - typeof(a)
    - a <- c(1,2,3,4,”hello”)
    - typeof(a)

Vector operations :

- Slicing
  - A <- seq(1,15,by=2)
  - A[1:5]
  - A[c(2,5,8)]
- Basic operations (elemnt by element)
  - Addition
  - Substration
  - Multiplication
  - Division
  - Exponent
  - Repeat
  - Concatenate (two vectors)

Characters

Numeric data

Matrices

- All columns in a matrix must have the same mode(numeric, character, etc.) and the same length.

- Syntax
  my_matrix <- matrix(vector, nrow=r, ncol=c, byrow=FALSE, dimnames=list(char_vector_rownames, char_vector_colnames))

  - Example

    - mat_y <- matrix(1:20, nrow=5,ncol=4)
    - mat_A <- matrix(seq(1:20),nrow=4,ncol=5,byrow= T,dimnames=list(c("row1","row2","row3","row4"),c("col1","col2","col3","col4", "col5")))

Matrix operations :

- Slicing
  - mat_A[2,]
  - mat_A[1:3,]
  - mat_A[,4]
  - mat_A[1:3,2:4
  - mat_A[2,3]
- Basic operations(element by element)
  - Addition
  - Substration
  - Multiplication
  - Division
  - Exponent

Exrercise :

Q:1 See what happens when you multiply or add vector with matrix ?

Example :

```
mat_vec <- mat_A * a
```

Q2 See what happens when you use “rep” with matrix.

Example :

```
rep( mat_A,3)
```

Q:3 Test the function “rbind” and “cbind” over given matrix. Why both functions are useful ?

Session - III

1. Data Frames in R

2. Looking at Data in R

3. Merging Files

Dataframes :

A data frame is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.).

```
d <- c(1,2,3,4)
e <- c("red", "white", "red", NA)
f <- c(TRUE,TRUE,TRUE,FALSE)
mydata <- data.frame(d,e,f)
names(mydata) <- c("ID","Color","Passed") # variable names
mydata
```

Basic operations :

- Inbuilt data

  Many inbuilt data are provided with R. With package “datasets”

  - data()

- Subsetting
  - By row
  - By column
  - By logicals

- Creating Data frame
  - *data.frame(obj)* : to create data frame from list or matrix

    Example :

```
dd <- data.frame(mat_A)
class(mat_A)
```

Looking at data :

- Check datatypes of columns
  - str()

    Example:

    ```
    data <- airquality
    str(data)
    ```
- Few useful functions
  - nrow()
  - ncol()
  - dim()
  - summary()
  - colnames()
  - Names()
  - head()
  - tail()

Merging Files:

- Merges two dataframes using identical column names (the addition is horizontal)

  Example :

  ```
  d.frame1 <-  data.frame(CustomerId = c(1:6), Product = c(rep("TV", 3), rep("Radio", 3)))

  d.frame2 <-  data.frame(CustomerId = c(2, 4, 6), State = c(rep("Goa", 2), rep("Delhi", 1)))
  ```

  - Inner join : merge(d.frame1,d.frame2)
  - Outer join: merge(x = d.frame1, y = d.frame2, by = "CustomerId", all = TRUE)
  - Left outer: merge(x = d.frame1, y = d.frame2, by = "CustomerId", all.x = TRUE)
  - Right outer: merge(x = d.frame1, y = d.frame2, by = "CustomerId", all.y = TRUE)
  - Cross join: merge(x = d.frame1, y = d.frame2, by = NULL)

Session - IV

1. Logical Statements in R

2. Conditional selection using "and", "or" and "ifelse" operator

3. Using apply, sapply, lapply in R

4. Data Manipulation

Logical statements in R

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
isTRUE(x)	test if X is TRUE
is.na(x)	check if X is NA
is.nan(x)	check if X is NaN
is.inf(x)	check whether X is inf
.	
.	
Etc..	

Conditionals:

- & (and)

```
x <- 1
y <- 1
(x ==1) & (y == 1)
```

- && (double and)

```
s <- 1:6
(s > 2 ) & (s < 5)
(s > 2) && (s <5)
```

See the difference

- | (or)

```
x <- 1
y <- 1
(x ==1) | (y == 1)
```

- || (double or)

```
s <- 1:6
(s > 2 ) | (s < 5)
(s > 2) || (s <5)
```

See the difference

- If-else

  if (cond) expr

  if (cond) expr1 else expr2

  - Example 1

    x <- 5

    if (x < 5) {print("x is less than 5")} else {"x is greater or equal to 5"}
  - Example 2

    a = c(5,7,2,9)

    ifelse(a %% 2 == 0,"even","odd")
  - Example 3

    data <- data.frame(a=c(0,0,2,3),b=c(0,5,0,8))

    *transform(data,mulm=ifelse(a> 0 & b>0,log(a*b), NA))*

apply() :

Returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Syntax : apply(x,margin,function)

Data <- mtcars
Col.mean <- apply(Data,2,mean)
Row.mean <- apply(Data,1,mean)

Define own function

check_length <- apply(Data, 2, function(x) length(x[x>20]))

sapply() :

- For vector and list modification / transformation. It applies function over list and vector and returns vector

sapply(1:3, function(x) x^2)
sapply(Data,mean)
l = (a=1:10,b=11:20) # mean of values using sapply
sapply(l, mean)

lapply() :

- lapply function is applied for operations on list objects and returns a list object of same length of original set.

lapply(l,mean)

Data Manipulation in R

- R package : dplyr
  - filter – It filters the data based on a condition

    data(“mtcars”)

    mydata <- mtcars

    filter(mydata, cyl > 4 & gear > 4)
  - select – It is used to select columns of interest from a data set

    select(mydata,cyl,mpg,hp)
  - arrange – It is used to arrange data set values on ascending or descending order

    mydata%>% select(cyl, wt, gear)%>% arrange(wt)
  - mutate – It is used to create new variables from existing variables

    *mydata%>% select(cyl, wt, mpg,gear)%>% mutate(newvariable = mpg*cyl)*

- summarise (with group_by) – It is used to perform analysis by commonly used operations such as min, max, mean count

```
data(iris)
iris_data <- iris
iris_data%>%
    group_by(Species)%>%
    summarise(Average = mean(Sepal.Length, na.rm = TRUE))
```

- *For more examples visit here*

Session - V

Visulaization of data

- Pie chart
  - Simple pie chart

```
slices <- c(25, 24, 22, 16,13 )
lbls <- c("wheat", "rice", "chickpea", "soyabean", "pea")
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices,labels = lbls, col=rainbow(length(lbls)),main="Crops production – 2016")
```

  - 3D Pie chart

```
library(plotrix)
pie3D(slices,labels=lbls,explode=0.1,main="Crop production – 2016")
```

- *Exrercise :*

Create a pie chart for iris data species type. (bonus marks : with percent)

Bar Plots:

barplot(table(iris_data$Species))

Stacked bar plots

```
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears and VS",  xlab="Number of Gears",
col=c("darkblue","red"),      legend = rownames(counts))
```

Grouped bar plots

```
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears and VS",
  xlab="Number of Gears", col=c("darkblue","red"),
       legend = rownames(counts),beside=T)
```

Box plots:

Boxplots can be created for individual variables or for variables by group. It helps to identify outliers in data.

boxplot(mpg~cyl,data=mtcars, main="Car Milage Data", xlab="Number of Cylinders", ylab="Miles Per Gallon")

Notched Box plot **

*boxplot(len~supp*dose, data=ToothGrowth, notch=TRUE, col=(c("gold","darkgreen")), main="Tooth Growth", xlab="Suppliment and Dose*

Note : if two boxes' notches do not overlap this is ‘strong evidence’ their medians differ

Outlier Detection

outlier_values <- boxplot.stats(mtcars$wt)$out
boxplot(mtcars$wt)
mtext(paste("Outliers: ", paste(outlier_values, collapse=", ")), cex=0.6)

** https://www.statmethods.net/graphs/boxplot.html

Histogram

Frequency distribution of quantitative variable

Sepal.L <- iris$Sepal.Length
hist(Sepal.L)
table(Sepal.L)
hist(Sepal.L,breaks=c(4:8))

Line charts

created with the function lines(x, y, type=) where x and y are numeric vectors. type= can take the following values:

Type	description
p	points
l	lines
o	overplotted points and lines
b, c	points (empty if "c") joined by lines
s, S	stair steps
h	histogram-like vertical lines
n	does not produce any points or lines

Example

```
x <- c(1:5)
y <- x * 2 # create some data
par(pch=23, col="green4") # plotting symbol and color
par(mfrow=c(2,4)) # all plots on one page
opts = c("p","l","o","b","c","s","S","h")
for(i in 1:length(opts)){
  heading = paste("type=",opts[i])
  plot(x, y, type="n", main=heading)
  lines(x, y, type=opts[i])
}
```

Scatter Plot

A scatter plot pairs up values of two quantitative variables in a data set and display them in cartesian system.

- 2D scatter plot

plot(wt, mpg, main="Scatterplot ", xlab="Car Weight ", ylab="Miles Per Gallon ", pch=22)

 # fit a line

abline(lm(mpg~wt), col="red") # regression line (y~x)

 # scatter matrix

pairs(~mpg+disp+drat+wt,data=mtcars, main="Scatterplot Matrix")

 # specialized package

library(car)

scatterplot.matrix(~mpg+disp+drat+wt|cyl, data=mtcars,

main="Three Cylinder Options")

- 3D scatter plot

  library(scatterplot3d)

  attach(iris)

  scatterplot3d(Sepal.Length,Sepal.Width,Petal.Length, main="3D Scatterplot")

Contact :

Email : deepak.bharti03@gmail.com