Introduction to R-Language

Dr. K.R. Chowdhary, Professor &
Campus Director, JIETCOE

JIET College of Engineering
Email: kr.chowdhary@jietjodhpur.ac.in
Web-Page: http://www.krchowdhary.com

July 9, 2016

Motivation

- S language: developed at Bell Labs for statistics, simulation, graphics
- S-PLUS: for commercial implementation
- R: Implementation under GPL (GNU General Public License), open source
- interpreted program code, object orientation
- easily extensible by self-written routines, packages, DLLs
- many types of graphics (mainly static)
- standardized, simple-to-used data format (data.frame)
- well developed format for fitting (regression) models
- -ve: no "standard" GUI yet
- Most widely used language in bioinformatics
- Standard for data mining and biostatistical analysis
- Technical advantages: free, open-source, available for all OSs

R-language

1. Language essentials: Objects; functions, vectors, missing values, matrices and arrays, factors, lists, data frames. Indexing, sorting and implicit loops. Logical operators. Packages and libraries.
2. Flow control: for, while, if/else, repeat, break.
3. Probability distributions: Built-in distributions in R; densities, cumulatives, quantiles, random numbers.
4. Statistical graphics: Graphical devices. High level plots. Low level graphics functions.
5. Statistical functions: One and two-sample inference, regression and correlation, tabular data, power, sample size calculations.

R is:

1. a suite of software facilities for:
    - reading and manipulating data
    - computation
    - conducting statistical analyses
    - displaying the results
2. a programming environment for big-data analysis and graphics
3. a platform for development and implementation of new algorithms
4. Software and packages can be downloaded from www.cran.r-project.org

How to start?

```
$ R          # to start R language command prompt: >
> 1+1
[1] 2 # The digit 1 within brackets indicates that the display
starts at the first element
> 1+2*3^4
[1] 163
> x=1; y=2
> x+y
[1] 3
> x=seq(-pi, pi, by=0.1)
> plot(x, sin(x), col="red", main="Sine-curve")
> help(function-name); e.g. help(sin), or ?sin
> x=rnorm(100)    # vector of 100 N(0,1) random variables
> hist(x, col="orange")          # histogram
```

Useful tools

- Install R-language on Linux by:

  $ sudo apt-get install r-base

  Some of the common R tools are:

```
> ls()        # list all R objects
> x = 1:3
> x           # show object (vector: x)
> print(x)    # show object (vector: x), also within
              # R scripts and functions
> fun = function(x) sin(x)
> fun         # show object (function: fun)
> rm(x)       # delete object x
> q()         # quit R
```

Data types

```
> x=1            # int
> y = pi         # float
> x= "a"         # char
> y= "my text"   # text
> x = TRUE       # logical
> y = 1 > 2
> x = c(1,2,3)   # vectors
> x = 1:3
> y = rep(2, 10)
> x=1:20
> x= matrix(x, 5, 4)    # matrix (x, row=5, col =4)
```

Normalization, distribution, plotting

```
> x=rnorm(100)
> mean(x)
> sd(x)
> plot(rnorm(10000), rnorm(10000))
> x=seq(-5, 5, by=0.1)
> plot(x, dnorm(x), col="black", lwd=2)
> x=seq(0,1, length=20)
> plot(sin(2*pi*x))                  # points
> plot(sin(2*pi*x), type="l")        # lines
> plot(sin(2*pi*x), type="p")        # points
> plot(sin(2*pi*x), type="b")        # points & lines
```

Running saved program & saving plots

- Running saved program

  > cat plotprog.r

  x=seq(0,1, length=20)

  plot(sin(2*pi*x), type="b")

  > source("plotprog.r")

  (Graph is created on screen)

- Saving a plot

  > jpeg('rplot.jpg')

  > (commands to plot graph or run a plotting program)

  > source("plotprog.r")

  > dev.off()

- other approach

  > dev.copy(png, 'myplot.png') # give this when plot is displayed

  > dev.off()

```
? volcano
data(volcano)
x = 10*(1:nrow(volcano))
y = 10*(1:ncol(volcano))
# Creates a 2-D image of x and y co-ordinates.
image(x, y, volcano, col = terrain.colors(100),
axes = FALSE)
# Adds contour lines to the current plot.
contour(x, y, volcano, levels = seq(90, 200, by=5),
add = TRUE, col = "peru")
# Adds x and y axes to the plot.
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
# Draws a box around the plot.
box()
# Adds a title.
title(main = "Maunga Whau Volcano", font.main = 4)
```

Arrays and Matrices

- An array can be considered as a multiply subscripted collection of data entries.

  x=array(data-vector, dim-vector)

  > x=array(1:20, dim=c(4,5)) # generate a 4xy 5 array

  > x

  > z = array(0, c(3,4,2)) # z is all zeros

  > x=array(1:9, dim(3,3))

  > x*x # element by element mult.

  > x%*%y # is mat. mult. #provided they are compatible to mul.
- read.table() function: reads the entire data frame directly

Handling tabular objects

> houseprice=read.table("houses.data")

> houseprice

- Editing data: When invoked on a data frame or matrix, edit brings up a separate spreadsheet-like environment for editing. This is useful for making small changes once a data set has been read. The command

  > hnew = edit(houseprice)

  edits the data of houseprice and assigns to hnew. To edit the same we use xold=edit(xold).
- Saving data: The function write.table writes in to a file an object, typically a data frame, but this can be any kind of object (vector, matrix, ...).

  > write.table(hnew, file = "hnew.data", append = FALSE, quote = TRUE, sep = " ", eol = "\n", row.names = TRUE, col.names = TRUE)

u1 = rnorm(30) # create a vector filled with random normal values print("This loop calculates the square of the first 10 elements of vector u1")

usq = 0

for(i in 1:10)

{

usq[i]=u1[i]*u1[i] # i-th element of u1 squared into i-th position of usq

print(usq[i])

}

print(i)

Program is in file "usq.r"

Control-flow in R

```
# nested for: multiplication table
mymat = matrix(nrow=30, ncol=30) # create a 30 x 30
matrix (of 30 rows and 30 columns)
for(i in 1:dim(mymat)[1]) # for each row
{
for(j in 1:dim(mymat)[2]) # for each column
{
mymat[i,j] = i*j # assign values based on position: product
of two indexes
}
}
```

Saved as nestfor.r