Basic R Programming

Silvia (Shuchang Liu)

Feb 3rd, 2014

1 Why R?

- R is high-level programming language and popular among biostatistician
- R is free open source and everyone can contribute to R (e.g. publish paper to develop new R packages)
- R can install many ready-made packages

2 Resources and installation

1. R and RStudio installation
   - R can be installed on Windows, Linux and Mac (`http://www.r-project.org/`)
   - RStudio is a nice IDE for R with GUI interface like MATLAB (`http://www.rstudio.com/`)
2. Package installation
   - To install general R packages
     To install a package in the first time

     ```
     > install.packages("packageName")
     ```

     Load the package before using it

     ```
     > library(packageName)
     ```
   - To install Bioconductor packages
     Bioconductor is an open source software for bioinformatics

     ```
     > source("http://bioconductor.org/biocLite.R")
     > biocLite("packageName")
     > library(packageName)
     ```

3 Basic R file formats

file description	R file	MATLAB file
script file	fileName.r	fileName.m
workspace file	fileName.rdata	fileName.mat

4 Some examples

```
> # hi, I am the comment
> print("Hello world!")

[1] "Hello world!"

> a = 1          # semicolon is optional here
> b <- 2.3       # var <- value, to assign the value to var
> c <- 3e2
> d <- a + b +c  # no output
> a + b + c      # show out the output

[1] 303.3

> a == b         # a equals to b?

[1] FALSE

> a != b         # a not equals to b?

[1] TRUE

> 5/4

[1] 1.25

> 1/0

[1] Inf

> cos(pi)

[1] -1

> max(4,5)

[1] 5

> log(9, base=3)  # parameter for the function

[1] 2

> log(9)          # o.w. by default, base=exp(1)

[1] 2.197225

> ls()            # list all the variables

[1] "a" "b" "c" "d"

> exists("b")      # Does "b" exist?

[1] TRUE

> rm(list=ls())   # remove all the variables in the workspace
```

5 Data Structures

5.1 Vectors

Creat a vector

```
> v1 <- -2:5
> v1

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

> v2 <- c(3, 2, 7.2, 0.9, 100)
> v2

[1]   3.0   2.0   7.2   0.9 100.0

> v3 <- c("aa","B","c2")
> v3

[1] "aa" "B"  "c2"

> seq(from=4, to=7, by=0.5)

[1] 4.0 4.5 5.0 5.5 6.0 6.5 7.0

> rep(v2, each=2)

 [1]   3.0   3.0   2.0   2.0   7.2   7.2   0.9   0.9 100.0 100.0

> rep(v2, times=2)

 [1]   3.0   2.0   7.2   0.9 100.0   3.0   2.0   7.2   0.9 100.0

> rep(v2, times=1:5)

 [1]   3.0   2.0   2.0   7.2   7.2   7.2   0.9   0.9   0.9   0.9 100.0 100.0
[13] 100.0 100.0 100.0

> sample(1:10)

 [1] 10  8  3  7  2  9  4  6  5  1
```

Reference elements

```
> v2

[1]   3.0   2.0   7.2   0.9 100.0

> v2[3]

[1] 7.2

> v2[c(2,4)]
```

```
[1] 2.0 0.9
> v2[-c(2,4)]
[1]   3.0   7.2 100.0
> v2[v2>5]
[1]   7.2 100.0
> v2[2]=88
```

Vector operations

```
> v2
[1]   3.0  88.0   7.2   0.9 100.0
> which(v2>=3)
[1] 1 2 3 5
> which.min(v2)
[1] 4
> length(v2)
[1] 5
> v2*10+1
[1]   31  881   73   10 1001
> (v2*10+1)[2:4][2]
[1] 73
```

5.2 Matrices

```
> d <- sample(1:20)
> d
 [1]  3 17  2 16 14 12  4  8 15 20 13  7  6 19  1 10  9 11 18  5
> # creat a matrix
> m1 <- matrix(data=d,nrow=4,ncol=5,byrow=TRUE,
+           dimnames=list(rows=c("cat","dog","rat","fish"),
+                         cols=c("a","b","c","d","e")))
> m1
```

```
      cols
rows    a  b  c  d  e
  cat   3 17  2 16 14
  dog  12  4  8 15 20
  rat  13  7  6 19  1
  fish 10  9 11 18  5

> # reference matrix
> m1[2,4]

[1] 15

> m1["dog","d"]    # reference by name

[1] 15

> m1[c(2,4),3]

 dog fish
   8   11

> m1[c("dog","fish"),"c"]

 dog fish
   8   11

> m1[3,]

 a  b  c  d  e
13  7  6 19  1

> m1["rat",]

 a  b  c  d  e
13  7  6 19  1

> # matrix dimension
> dim(m1)         # matrix dimension

[1] 4 5

> nrow(m1)        # number of rows

[1] 4

> ncol(m1)        # number of column

[1] 5

> length(m1)      # element length

[1] 20
```

```
> colnames(m1)    # column name

[1] "a" "b" "c" "d" "e"

> rownames(m1)[4] <- "elephant"     # change row names
> m2 <- matrix(c(111,222,333,444),2,2)
> m2

     [,1] [,2]
[1,]  111  333
[2,]  222  444

> m1[2:3,c(3,5)] <- m2
> m1

          cols
rows        a  b   c  d   e
  cat       3 17   2 16  14
  dog      12  4 111 15 333
  rat      13  7 222 19 444
  elephant 10  9  11 18   5

> # matrix operation
> m3 <- matrix(1:6,2,3)
> m3

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> m4 <- matrix(11:16,2,3,byrow=TRUE)
> m4

     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   14   15   16

> m5 <- matrix(21:26,3,2)
> m5

     [,1] [,2]
[1,]   21   24
[2,]   22   25
[3,]   23   26

> m3*m4    # element-wise manipulation

     [,1] [,2] [,3]
[1,]   11   36   65
[2,]   28   60   96
```

```
> m3%*%m5  # matrix manipulation, if m3%*%m4, it will throw out error message

     [,1] [,2]
[1,]  202  229
[2,]  268  304

> # several useful functions
> t(m1)    # transpose

    rows
cols cat dog rat elephant
   a   3  12  13       10
   b  17   4   7        9
   c   2 111 222       11
   d  16  15  19       18
   e  14 333 444        5

> apply(m1,1,sum)   # sum of each row

     cat      dog      rat elephant
      52      475      705       53

> apply(m1,1,sqrt)  # pay attention to the output dimension

   rows
         cat       dog       rat elephant
  a 1.732051  3.464102  3.605551 3.162278
  b 4.123106  2.000000  2.645751 3.000000
  c 1.414214 10.535654 14.899664 3.316625
  d 4.000000  3.872983  4.358899 4.242641
  e 3.741657 18.248288 21.071308 2.236068

> apply(m1,2, function(x) return(sum(x^2+1)))

     a      b      c      d      e
   426    439  61734   1170 308250

> rbind(m3,m4)   # row combination

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
[3,]   11   12   13
[4,]   14   15   16

> cbind(m3,m4)   # column combination

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    5   11   12   13
[2,]    2    4    6   14   15   16

> as.vector(m3)

[1] 1 2 3 4 5 6
```

5.3 Arrays

An array is a muti-dimensional matrix.

```
> # generate an array var
> a <- array(1:24,dim=c(4,3,2),
+            dimnames=list(c("a","b","c","d"),c("x","y","z"),c("old","new") ))
> a

, , old

  x y  z
a 1 5  9
b 2 6 10
c 3 7 11
d 4 8 12

, , new

   x  y  z
a 13 17 21
b 14 18 22
c 15 19 23
d 16 20 24

> # reference elements
> a[2,1,"new"]   # pay attention to the dimension change

[1] 14

> a[-2,"x",]

  old new
a   1  13
c   3  15
d   4  16

> # operation
> dim(a)

[1] 4 3 2

> apply(a,3,mean)

 old  new
 6.5 18.5
```

5.4 Lists

```
> # generate a list var
> l1 <- list(name=c("Peter","Lily","Emma"),c("yes","no"),
+           age=c(20,40,33,rep(18,times=3)),
+           value=matrix(1:6,2,3))
> l1
```

```
$name
[1] "Peter" "Lily"  "Emma"

[[2]]
[1] "yes" "no"

$age
[1] 20 40 33 18 18 18

$value
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> # reference elements
> l1[[2]]

[1] "yes" "no"

> l1[["age"]]

[1] 20 40 33 18 18 18

> l1$name

[1] "Peter" "Lily"  "Emma"

> l1[["value"]][2,]

[1] 2 4 6

> # list operation
> names(l1)

[1] "name"  ""      "age"   "value"

> length(l1)

[1] 4

> unlist(l1)   #produce a vector of all the elements

  name1   name2   name3                    age1    age2    age3    age4    age5
"Peter"  "Lily"  "Emma"   "yes"    "no"    "20"    "40"    "33"    "18"    "18"
   age6  value1  value2  value3  value4  value5  value6
   "18"     "1"     "2"     "3"     "4"     "5"     "6"

> l2 <- list(matrix(2,4,5),matrix(1:10,2,5),diag(3))
> l2
```

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

[[2]]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10

[[3]]
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

> lapply(l2,sum)  # return a list

[[1]]
[1] 40

[[2]]
[1] 55

[[3]]
[1] 3

> sapply(l2,dim) # return a vector or matrix

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

5.5 Dataframes

```
> # generate a dataframe
> ID <- 100:103
> name <- c("Peter","Lily","Emma","Joe")
> sex <- c("M","F","F","M")
> age <- c(22,30,16,44)
> married <- c(F,T,F,T)
> d1 <- data.frame(ID,name,sex,age,married)
> rownames(d1) <-name
> d1

       ID  name sex age married
Peter 100 Peter   M  22   FALSE
```

```
Lily  101  Lily   F  30    TRUE
Emma  102  Emma   F  16   FALSE
Joe   103   Joe   M  44    TRUE
```

```
> # reference elements, just like a matrix
> d1["Emma","age"]
```

```
[1] 16
```

```
> d1[d1$ID==103,]
```

```
     ID name sex age married
Joe 103  Joe   M  44    TRUE
```

```
> # operation
> d2 <- d1[order(d1[,"age"]),]
> d2
```

```
       ID  name sex age married
Emma  102  Emma   F  16   FALSE
Peter 100 Peter   M  22   FALSE
Lily  101  Lily   F  30    TRUE
Joe   103   Joe   M  44    TRUE
```

6 Control Structures

6.1 if()...else

```
if(condition){
expression 1, if TRUE
}else{
expression 2, if FALSE
}
```

```
> passExam <- TRUE
> if(passExam==TRUE){
+   print("Congratulations!")
+ }else{
+   print("Try it again!")
+ }
```

```
[1] "Congratulations!"
```

6.2 ifelse()

```
ifelse(condition, TRUE expression, FALSE expression)
```

```
> x <- matrix(sample(1:6),2,3)
> x
```

```
     [,1] [,2] [,3]
[1,]    2    6    1
[2,]    5    3    4
```

```
> y <- ifelse(x>3,1,0)
> y
```

```
     [,1] [,2] [,3]
[1,]    0    1    0
[2,]    1    0    1
```

6.3 switch()

```
switch (statement, list)
```

```
> grade="D"
> switch(grade, A="Great job!", B="Not bad", C="So-so", D="Donnot cry! I trust you!")
```

```
[1] "Donnot cry! I trust you!"
```

6.4 for loops

```
for (name in vector){
  statement
}
```

```
> sum(1:10)
```

```
[1] 55
```

```
> a <- 0
> for (i in 1:10){
+   a <- a+i
+ }
> a
```

```
[1] 55
```

6.5 repeat loops

```
repeat{
  expression
  if(condition) break
}
```

```
> a <- 0
> i <- 0
> repeat{
+   i <- i+1
+   a <- a+i
+   if (i>=10) break
+ }
> a
```

```
[1] 55
```

6.6 while loops

```
while(condition){
  expression
}
```

```
> a <- 0
> i <- 0
> while(i<10){
+   i <- i+1
+   a <- a+i
+ }
> a
```

```
[1] 55
```

7 Functions

```
functionName <- function(arguments){
  body
}
```

```
> # example 1:
> my.func <- function(grade){
+   switch(grade, A="Great job!", B="Not bad", C="So-so", D="Fail again?... OK... Just cry!")
+ }
> my.func("D")
```

```
[1] "Fail again?... OK... Just cry!"
```

```
> # example 2: a function to calculate the factorial
> my.fac <- function(x){
+   if(x==1){
+     return(x)
+   }else{
+     return(x*my.fac(x-1))
+   }
+ }
> my.fac(5)
```

```
[1] 120
```

```
> factorial(5)
```

```
[1] 120
```

```
> # example 3: a function with mutiple arguments
> my.func2 <- function(x,y){
+   len=length(y)
+   return(x[1:len])
+ }
> my.func2(c(4,3,2,1),c(1,2))

[1] 4 3
```

8 Read and write files

8.1 Data output

```
> d1

       ID  name sex age married
Peter 100 Peter   M  22   FALSE
Lily  101  Lily   F  30    TRUE
Emma  102  Emma   F  16   FALSE
Joe   103   Joe   M  44    TRUE

> # output the file after converting it to a data frame
> write.table(d1,file="myFile.txt",append=FALSE,quote=FALSE,sep="\t",row.names=TRUE,col.names=TRUE)
> # save current work space
> save(d1,file="work.rdata")
```

8.2 Data input

```
> # input the data as a data frame format
> rm(list=ls())
> data=read.table(file="myFile.txt",header=TRUE,sep="\t",skip=0)
> data

       ID  name sex age married
Peter 100 Peter   M  22   FALSE
Lily  101  Lily   F  30    TRUE
Emma  102  Emma   F  16   FALSE
Joe   103   Joe   M  44    TRUE

> # load workspace file
> load("work.rdata")
> ls()

[1] "d1"   "data"
```

9 Graph

```
> # set the figure contain four sub graphs
> par(mfrow=c(2,2))   # 2-by-2 sub graph
```

```
> # graph 1: plot
> x <- 1:10
> y <- seq(0.1,1,by=0.1)              # theoretical value
> z <- y + rnorm(10,mean=0,sd=0.1)  # actual value
> plot(x, z, type="p",col="red",main="Plot", xlab="x label", ylab="value")
> lines(x, y)
> # graph 2: histogram
> age <- rnorm(1000,mean=20,sd=3)
> hist(age,main="Histogram",xlab="age",ylab="counts")
> # graph 3: boxplot
> boxplot(age,main="boxplot",ylab="age")
> # graph 4: qq plot
> qqnorm(age,main="qq plot")
```

10 At the end

1. R is similar to MATLAB, but has differences in many details
2. R programming courses
   UPitt: BIOST 2094 – STATISTICAL COMPUTING AND DATA ANALYSIS USING R
3. Some useful references
   `http://cran.r-project.org/doc/manuals/r-release/R-lang.html`
   `http://www.cyclismo.org/tutorial/R/`
   `http://ww2.coastal.edu/kingw/statistics/R-tutorials/`
   OR, just google it!

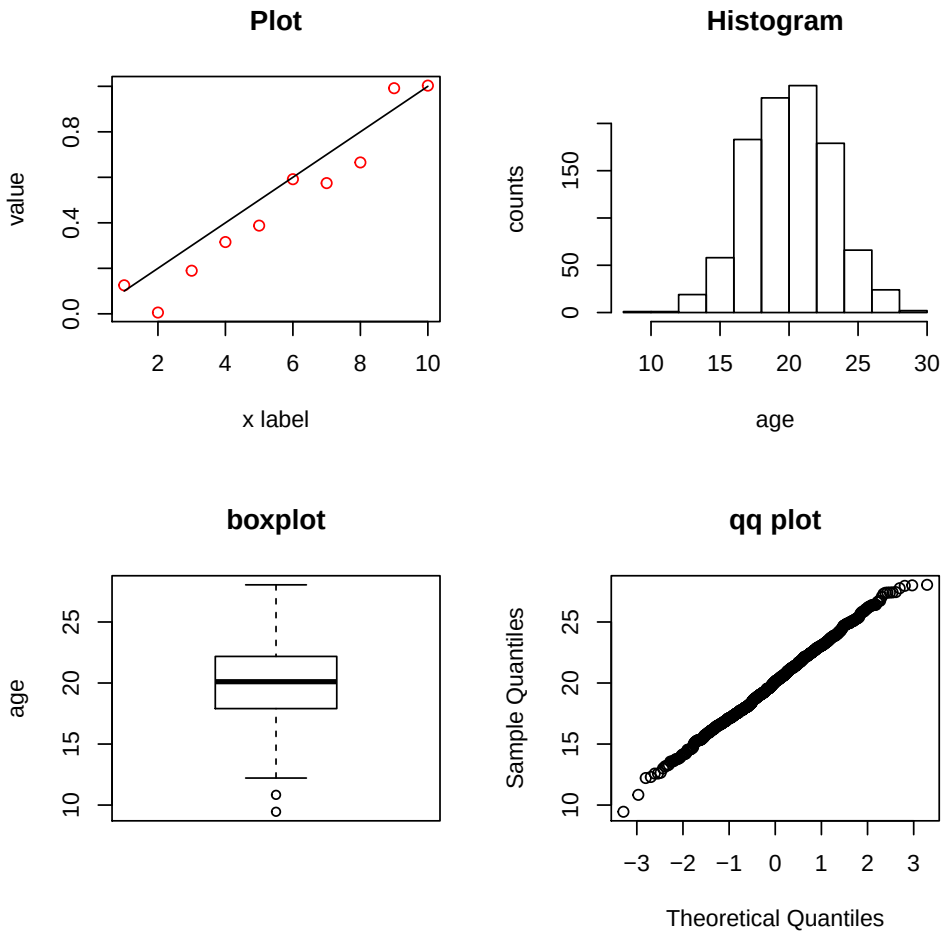

Figure 1: Figure examples