More topics in R

I. 10 Basic R functions you should know

To invoke the function you must include an argument list in (), even if the list is empty.
q Quit the session
help/? Get help on a function or object
help.start Allow the use of a web browser for reading help
data List the available packages or attach a package
library List available packages or attach a package
objects/ls List the objects in the workspace
rm Remove objects from the workspace
summary Summarize an object
str Show the low-level structure of an object

II. Graphics control

A. Writing graphs to files without copying and pasting

```
pdf(file="/temp/mypic.pdf")
graphic commands
dev.off()
```

Homework Q1: Find the command to produce *.png, *.tiff, and *.jpeg files.

B. The par() command

This command controls many graphic options that are available. You will most likely be interested in changing the number of graphs per graphics page (i.e. you can produce one metafile that contains 4 graphs.)

```
y<-rnorm(100)
par(mfrow=c(1,3))  ### create three plots in one row
##### default: adj=0.5
hist(y, xlab="Standard normal distribution",
breaks=seq(from=-4, to=4, by=1), main="Centered Title")
par(adj=0)
```

```
hist(y, xlab="Standard normal distribution",
breaks=seq(from=-4, to=4, by=1), main="Left Justified Title")
par(adj=1)
hist(y, xlab="Standard normal distribution",
breaks=seq(from=-4, to=4, by=1), main="Right Justified
Title")
par(adj=0.5)
```

To see the default settings for graphics, type: par()
To see more options, type: help("par")

III. Programming

A. If statements

The **If** statement allows you to perform a (set of) statements if a logical expression is evaluated to True.

if(condition) statement 1 else statemnet2

where *condition* is a logical expression
statement1 is an action for R to perform if the logical expression is TRUE, you can list a sequence of statements by enclosing the in { }.
statement2 is an action for R to perform if the logical expression is FALSE, you can list a sequence of statements by enclosing the in { }.

Examples:

```
standard<-TRUE
m=4
v=8
if (standard) {
        y<-rnorm(100)
        hist(y)
        m1<-mean(y)
        v1<-var(y)
} else {
        y<-rnorm(100, mean=m, sd=sqrt(v))
        hist(y)
        m1<-mean(y)
        v1<-var(y)
}
```

B. Ifelse statement

The *ifelse* statement is the vector version of the if statement.

ifelse(test, true.value, false.value)

where *test* is a logical expression for values in a vector
true.value is the value to be assigned if test is evaluated to TRUE
false.value is the value to be assigned if test is evaluated to FALSE

Examples:

```
y<-runif(100)
trt<-ifelse(y<0.5, 1, 0)
trt
table(trt)
```

C. For Loops

A *for* loop allows a statement to be iterated as a variable assumes values in a specified sequence.

for(variable in sequence) statement(s)

where *variable* is usually just a dummy counter variable for the loop
sequence is the range for the counter variable
statement(s) is a statement to perform or list of statements to perform if you are using multiple statements, enclose the in { }.

Examples:

```
for(I in 1:5) print(i)

y<-rep(0, 50)
for(j in 1:length(y)){
    y[j] <-mean(rnorm(10, mean=3, sd=2))
    hist(y, breaks=seq(from=1, to=5, by=0.5))
}
```

D. Advanced Topics: apply and sweep

```
my.matrix <- matrix(seq(1,12,1), nrow=4)
apply.matrix <- t(apply(X = my.matrix, MARGIN = 1, FUN = function(x) x/sum(x)))
sweep.matrix <- sweep(x = my.matrix, MARGIN = 1, STATS = rowSums(my.matrix), FUN="/")
```

IV. Running R source code

```
getwd()
setwd(“/Users/yen-yiho/Desktop/Stat704/Notes/Lecture1-IntroductionToR”)
source(“MoreCode.R”)
```

V. Bootstrap Method

```
varbs<-function(x, B=200){
   bsmean<-rep(0, B)
   len<-length(x)
   for(i in 1:B){
             bsmean[i]<-mean(sample(x, len, replace=T))
   }
   genmean<-mean(bsmean)
   bsvar<-sum((bsmean-genmean)^2/(B-1))
   return(bsvar)
}

x<-rnorm(100)
varbs(x)
```