STA 250/MTH 342 Intro to Mathematical Statistics
Lab Session 2 / Jan 19, 2015 / Handout

In this session we will briefly introduce how to program **R**. Since the main topic of the lab sessions is the statistical applications, we have to treat the programming aspect briefly, though it absolutely deserves a separated course. We will go through the materials quickly, and many contents serve just as a reference for your later use.

See: `https://stat.duke.edu/courses/Spring15/sta250/labs/` for links to source code and data. Submit lab solutions via email to: `sta250@stat.duke.edu`. Any plots should be included in postscript form as attachments. The email subject must be "STA250 . . . " with ". . . " replaced by your name.

1: The `if...else` statement. The document page shows

```
  if(cond) expr
  if(cond) cons.expr  else  alt.expr
```

Let's try the following commands.

```
> myUniversity <- "Duke"
> if(myUniversity == "Duke") print("great!")
[1] "great!"
> if(myUniversity == "Duke")
+ print("great~~")
[1] "great~~"
> if(myUniversity=="Duke"){print("great!");a <- 300; print(a)}
[1] "great!"
[1] 300
> if(myUniversity=="Duke"){
+ print("great!")
+ a <- 300
+ print(a)
+ }
[1] "great!"
[1] 300
> if(177+68==235)print("good")else print("why wrong?")
[1] "why wrong?"
> if(a == 299){
+ print(100)
+ }else print(200)
[1] 200
```

2: Scripts in Files. Let's create a new file "`a.R`", open it, write the following scripts, save and close it.

```
a <- 0
for(i in 1:100){
  a <- a + i
}
print(a)
```

One may run the script on command-line prompt as "`Rscript a.R`", or call the script in **R**,

```
> source("a.R")
[1] 5050
> a
[1] 5050
```

3: Loops. R document page shows

```
for(var in seq) expr
while(cond) expr
repeat expr
break
next
```

Now let's try the following scripts.

```
> for(i in 4:6){
+ print(i)
+ print(i^2)
+ }
[1] 4
[1] 16
[1] 5
[1] 25
[1] 6
[1] 36
> while(i>4){
+ print(-i)
+ i <- i-1
+ }
[1] -6
[1] -5
> repeat{
+ i <- i+1
+ if(i > 7) break
+ print(i)
+ }
[1] 5
[1] 6
[1] 7
> while(i >0){
+ i <- i-1
+ if(i > 3) next
+ print(i)
+ }
[1] 3
[1] 2
[1] 1
[1] 0
```

TASK 1 Please try the above commands of this section yourself. Copy and paste your types and what you have got on the screen to the email. It is fine not to delete the mistakes and errors. □

4: Functions.

TASK 2 Open the documentation page of "`function`", read it, copy about 30 lines to the email, and quit it. □

Let's generate a file "`b.R`" with the following scripts.

```
fac <- function(n){
  if(n == 1){
    return(1)
  }else{
    return(n * fac(n-1))
  }
}
```

Then we call from **R** the function we created in the file.

```
xin@bug:~/work/ta_20140110/lab2$ R -q
> source("b.R")
> ls()
[1] "fac"
> fac
function(n){
  if(n == 1){
    return(1)
  }else{
    return(n * fac(n-1))
  }
}
> fac(1)
[1] 1
> fac(5)
[1] 120
> fac("Duke University")
Error in n - 1 : non-numeric argument
to binary operator
> fac(0)
Error: evaluation nested too deeply:
infinite recursion / options(expressions=)?
> # Questions for geeks:
> # What happens?
> # How to make this code robust?
```

TASK 3 Recall the **Fibonacci numbers** $1, 1, 2, 3, 5, 8, \cdots$. They are defined by the recurrence relation

$$F_1 = 1, \quad F_2 = 1, \quad F_n = F_{n-1} + F_{n-2} \text{ for } n = 3, 4, \cdots .$$

Please design a function to compute the Fibonacci numbers, store it into a file "`fib.R`", and call it in **R** to compute F_{20}. Please copy the command-line operations and output into the email, and attach the file "`fib.R`". □

5: Debugging.

```
> a <- function(n){repeat{i <- runif(1);browser();print(n)}}
> a
function(n){repeat{i <- runif(1);browser();print(n)}}
> a(0)
Called from: a(0)
Browse[1]> i
[1] 0.5230853
Browse[1]> c # to continue
[1] 0
Called from: a(0)
Browse[1]> Q # to quit debuggin
```

6: List. In **R**, list is a container of all the data types.

```
> a <- list()
> a[[1]] <- "Duke University"
> a[[2]] <- 1:5
> a[[3]] <- as.list(1:3)
> a[[4]] <- diag(4)
> a[[5]]<-function(n)
+ ifelse(n<=1,1,a[[5]](n-1)*n)
> a
[[1]]
[1] "Duke University"

[[2]]
[1] 1 2 3 4 5

[[3]]
[[3]][[1]]
[1] 1
[[3]][[2]]
[1] 2

[[3]][[3]]
[1] 3


[[4]]
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1

[[5]]
function (n)
ifthen(n <= 1, 1, a[[5]](n - 1) * n)
```

```
> a[[5]](10)
[1] 3628800
> a[1:3]
[[1]]
[1] "Duke University"

[[2]]
[1] 1 2 3 4 5

[[3]]
[[3]][[1]]
[1] 1

[[3]][[2]]
[1] 2

[[3]][[3]]
[1] 3



> a[1]
[[1]]
[1] "Duke University"

> a[[1]]
[1] "Duke University"
> typeof(a[1])
[1] "list"
> typeof(a[[1]])
[1] "character"
> lapply(a,function(x)typeof(x))
[[1]]
[1] "character"

[[2]]
[1] "integer"

[[3]]
[1] "list"

[[4]]
[1] "double"

[[5]]
[1] "closure"

> unlist(lapply(a,function(x)typeof(x)))
[1] "character" "integer"   "list"
"double"    "closure"
> unlist(lapply(1:7,a[[5]]))
[1]    1    2    6   24  120  720 5040
> x <- list(1:5,2:6,3:7)
> x
[[1]]
[1] 1 2 3 4 5

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

[[3]]
[1] 3 4 5 6 7
> cbind(x[[1]],x[[2]],x[[3]])
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    5
[4,]    4    5    6
[5,]    5    6    7
> do.call(rbind,x)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    2    3    4    5    6
[3,]    3    4    5    6    7
> do.call(cbind,x)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    5
[4,]    4    5    6
[5,]    5    6    7
```

7: Saving/loading Data.

```
> a <- matrix(runif(15),ncol=3)
> b <- 1:5
> a
          [,1]      [,2]       [,3]
[1,] 0.6586850 0.5608741 0.83961529
[2,] 0.7749791 0.6909845 0.82593192
[3,] 0.3917981 0.6763472 0.87556824
[4,] 0.9284825 0.4331762 0.03126614
[5,] 0.5587880 0.8050524 0.18024710
> save(a,b,file="data.Rdata")
> q()
Save workspace image? [y/n/c]: n
xin@bug:~/work/ta_20140110/lab2$ R -q
> ls()
```

```
character(0)
> load("data.Rdata")
> ls()
[1] "a" "b"
> a
          [,1]      [,2]       [,3]
[1,] 0.6586850 0.5608741 0.83961529
[2,] 0.7749791 0.6909845 0.82593192
[3,] 0.3917981 0.6763472 0.87556824
[4,] 0.9284825 0.4331762 0.03126614
[5,] 0.5587880 0.8050524 0.18024710
> b
[1] 1 2 3 4 5
```

At this stage, be careful when storing functions.

8: String Manipulating.

```
> a <- c("Duke", "Yale", "Oxford", "Peking", "Princeton")
> a
[1] "Duke"      "Yale"      "Oxford"    "Peking"    "Princeton"
> sort(a) # sort the string vector according to dictionary order
[1] "Duke"      "Oxford"    "Peking"    "Princeton" "Yale"
> order(a)
[1] 1 3 4 5 2
> b <- paste(a, "University") # the paste will generate a space
> b
[1] "Duke University"      "Yale University"      "Oxford University"
[4] "Peking University"    "Princeton University"
> paste(a, collapse = ", ") # paste the whole vector into a single string
[1] "Duke, Yale, Oxford, Peking, Princeton"
> grepl("Duke", b) # check if the string contains "Duke"
[1]  TRUE FALSE FALSE FALSE FALSE
> grep("Duke", b) # same function, just give index
[1] 1
> gsub("University","People", b) # replace "University" with "people"
[1] "Duke People"      "Yale People"      "Oxford People"    "Peking People"
[5] "Princeton People"
> strsplit(b[1:3], split = " ") # split strings
[[1]]
[1] "Duke"       "University"

[[2]]
[1] "Yale"       "University"

[[3]]
[1] "Oxford"     "University"
```

9: Communicating Tables. When communicating data with other softwares, one usually need text-format tables.

```
> load("data.Rdata")
> a
          [,1]      [,2]       [,3]
[1,] 0.6586850 0.5608741 0.83961529
[2,] 0.7749791 0.6909845 0.82593192
[3,] 0.3917981 0.6763472 0.87556824
[4,] 0.9284825 0.4331762 0.03126614
[5,] 0.5587880 0.8050524 0.18024710
> colnames(a) <- c("one", "two", "last")
> a
           one       two       last
[1,] 0.6586850 0.5608741 0.83961529
[2,] 0.7749791 0.6909845 0.82593192
[3,] 0.3917981 0.6763472 0.87556824
```

```
[4,] 0.9284825 0.4331762 0.03126614
[5,] 0.5587880 0.8050524 0.18024710
> rownames(a) <- c("red", "green",
+ "black", "white","yellow")
> a
             one       two       last
red    0.6586850 0.5608741 0.83961529
green  0.7749791 0.6909845 0.82593192
black  0.3917981 0.6763472 0.87556824
white  0.9284825 0.4331762 0.03126614
yellow 0.5587880 0.8050524 0.18024710
> colnames(a)
[1] "one"  "two"  "last"
> rownames(a)
[1] "red"    "green"  "black"  "white"  "yellow"
> write.table(a, quote=F,file="data.txt")
> read.table("data.txt")
             one       two       last
red    0.6586850 0.5608741 0.83961529
green  0.7749791 0.6909845 0.82593192
black  0.3917981 0.6763472 0.87556824
white  0.9284825 0.4331762 0.03126614
yellow 0.5587880 0.8050524 0.18024710
> q()
Save workspace image? [y/n/c]: n
xin@bug:~/work/ta_20140110/lab2$ R -q
> ls()
character(0)
> a <- as.matrix(read.table("data.txt"))
> a
             one       two       last
red    0.6586850 0.5608741 0.83961529
green  0.7749791 0.6909845 0.82593192
black  0.3917981 0.6763472 0.87556824
white  0.9284825 0.4331762 0.03126614
yellow 0.5587880 0.8050524 0.18024710
```

10: Other Functions.

```
> set.seed(12345)
> a <- runif(10)
> a
 [1] 0.7209039 0.8757732 0.7609823 0.8861246 0.4564810 0.1663718 0.3250954
 [8] 0.5092243 0.7277053 0.9897369
> runif(10)
 [1] 0.034535435 0.152373490 0.735684952 0.001136587 0.391203335 0.462494654
 [7] 0.388143982 0.402485142 0.178963585 0.951658754
> runif(10)
 [1] 0.4537281 0.3267524 0.9654153 0.7074819 0.6445426 0.3898285 0.6985436
 [8] 0.5440579 0.2264672 0.4845578
> set.seed(12345)
> runif(10)
 [1] 0.7209039 0.8757732 0.7609823 0.8861246 0.4564810 0.1663718 0.3250954
 [8] 0.5092243 0.7277053 0.9897369
> names(a) <- paste("bird", 1:10, sep = "-")
> a
   bird-1    bird-2    bird-3    bird-4    bird-5    bird-6    bird-7    bird-8
0.7209039 0.8757732 0.7609823 0.8861246 0.4564810 0.1663718 0.3250954 0.5092243
   bird-9   bird-10
0.7277053 0.9897369
> range(a)
[1] 0.1663718 0.9897369
> max(a)
[1] 0.9897369
> min(a)
[1] 0.1663718
> a==max(a)
 bird-1  bird-2  bird-3  bird-4  bird-5  bird-6  bird-7  bird-8  bird-9 bird-10
  FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE   FALSE    TRUE
```

```
> which(a==max(a))
bird-10
     10
> summary(a)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1664  0.4697  0.7243  0.6418  0.8471  0.9897
> sample(a,size=7,replace=F)
   bird-5    bird-3    bird-8   bird-10    bird-4    bird-2    bird-9
0.4564810 0.7609823 0.5092243 0.9897369 0.8861246 0.8757732 0.7277053
```

~~END~~