Exploratory Data Analysis

Chapter 3

Instructor: Seokho Lee

Hankuk University of Foreign Studies

3. R Programming

3.1. Control flow

Example (`if()`)

```
> x<-1:5
> x
[1] 1 2 3 4 5
> y<- -2:2
> y
[1] -2 -1  0  1  2
> if(any(x<0)) print(x)     # print() is not executed
> if(any(y<0)) print(abs(y))     # print() is executed
[1] 2 1 0 1 2
> if(any(y<0)){     # make a block using {}
+  print(abs(y))
+  print('y contains negative values')
+ }
[1] 2 1 0 1 2
[1] "y contains negative values"
```

- `if`(*cond*) *expr*

  if *cond* is `TRUE`, execute *expr*. Otherwise, *expr* is not executed.

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

  if *cond* is `TRUE`, execute *cond.expr*. Otherwise, *alt.expr* is executed.

Example (if())

```
> if(pi>3) print('expr if statement') else print('expr else statement')
[1] "expr if statement"
> if(pi<3) print('expr if statement') else print('expr else statement')
[1] "expr else statement"
> if(length(x)==5){     # multiple conditions
+  if(sum(x)==15) print('Vector x length=5, sum=15')
+ } else {
+  print('Vector x length!=5')
+ }
[1] "Vector x length=5, sum=15"
> if(length(x)==5 && sum(x)==15){     # alternative form of multiple conditions
+  print('Vector x length=5, sum=15')
+ } else {
+  print('Vector x length!=5')
+ }
[1] "Vector x length=5, sum=15"
```

Example (`ifelse()`)

```
> y<- -2:2
> ifelse(y>=0,y,-y)
[1] 2 1 0 1 2
> abs(y)
[1] 2 1 0 1 2
> tmp<-3
> ifelse(tmp>0, 'positive', ifelse(tmp<0,'negative','zero'))
[1] "positive"
```

- `ifelse`(*test*,*yes*,*no*)

  yes is returned if *test* is `TRUE`, and *no* is returned if *test* is `FALSE`.

Example (switch())

```
> x<-1:10
> switch(EXPR=1,mean(x),median(x),sd(x),range(x))
[1] 5.5
> switch(EXPR=2,mean(x),median(x),sd(x),range(x))
[1] 5.5
> switch(EXPR=3,mean(x),median(x),sd(x),range(x))
[1] 3.02765
> switch(EXPR=4,mean(x),median(x),sd(x),range(x))
[1]  1 10
> switch(EXPR=5,mean(x),median(x),sd(x),range(x))
NULL
```

- switch(EXPR, value1, value2, . . .)

  If EXPR=1, then return value1. If EXPR=2, return value2. etc.

Example (switch())

```
> x<-1:10
> type<-'mean'
> switch(EXPR=type,
+        MEAN=, mean=mean(x),
+        MEDIAN=, median=median(x),
+        SD=, sd=sd(x),
+        range(x))
[1] 5.5
> type<-'SD'
> switch(EXPR=type,
+        MEAN=, mean=mean(x),
+        MEDIAN=, median=median(x),
+        SD=, sd=sd(x),
+        range(x))
[1] 3.02765
> type<-'Sd'
> switch(EXPR=type,
+        MEAN=, mean=mean(x),
+        MEDIAN=, median=median(x),
+        SD=, sd=sd(x),
+        range(x))
[1]  1 10
```

- `for(var in seq) expr`
- `while(cond) expr`
- `repeat expr`

Example (loop statements)

```
> for(i in 5:1) print(rep(i,i))
[1] 5 5 5 5 5
[1] 4 4 4 4
[1] 3 3 3
[1] 2 2
[1] 1
> sum.x<-0
> for(i in 1:10)    # sum from 1 to 10
+ sum.x<-sum.x+i
> sum.x
[1] 55
> for(i in 2:9)     # form a multiplication table
+ for(j in 1:9)
+ cat(i,'*',j,'=',i*j,'\n')
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
...
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

- `break`
  stop the loop.
- `next`
  stop the current step and move to the next in the loop.

Example (`break` and `next`)

```
> sum.x<-0
> for(i in 1:10){
+  sum.x <- sum.x + i
+  if(sum.x>20) break
+  cat(i, sum.x, '\n')
+ }
1 1
2 3
3 6
4 10
5 15
```

Example (break and next)

```
> for(i in 2:9){
+     if(i>5) break
+     for(j in 1:9){
+         if(j>1) break
+         cat(i, '*', j, '=', i*j, '\n')
+    }
+ }
2 * 1 = 2
3 * 1 = 3
4 * 1 = 4
5 * 1 = 5
>
> x<-0
> sum.x<-0
> while(x<10){
+    x <- x+1
+    if(x<9) next
+    print(x)
+    sum.x <- sum.x+x
+ }
[1] 9
[1] 10
> sum.x
[1] 19
```

3.2. Function

Definition (function)

function_name <- function (arguments) body

Example (function definition)

```
> my.mean<-function(data) sum(data)/length(data)
> ls()
[1] "my.mean"
> my.mean
function(data) sum(data)/length(data)
> my.mean(1:10)
[1] 5.5
> mean(1:10)
[1] 5.5
```

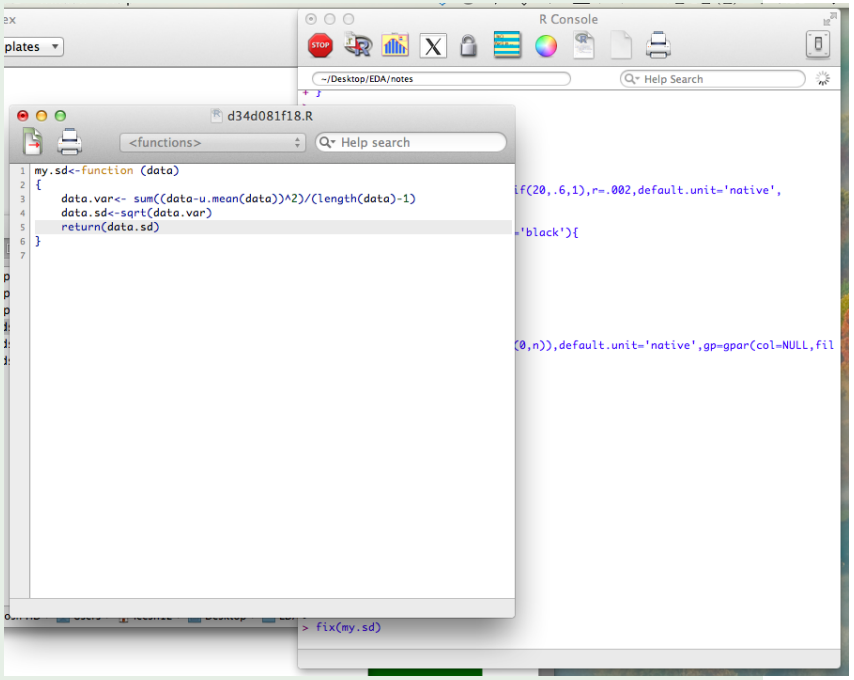
- Editor program is useful to write a long program including R function. R program provides an editor but there are several editors freely available for R programming.

Example

```
# try this in the R console
> fix(my.sd)
```

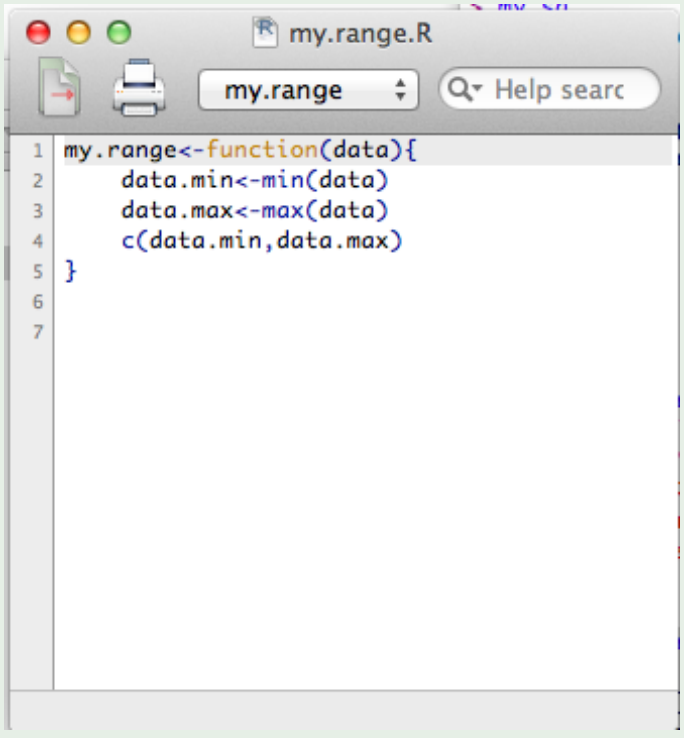
Example (R editor)

```
> my.sd
function (data)
{
data.var<- sum((data-my.mean(data))^2)
/(length(data)-1)
data.sd<-sqrt(data.var)
return(data.sd)
}
> my.sd(1:10)
[1] 3.02765
> sd(1:10)
[1] 3.02765
```



Example (source())

```
> source('my.range.R')
> my.range
function(data){
data.min<-min(data)
data.max<-max(data)
c(data.min,data.max)
}
> my.range(1:10)
[1]  1 10
> range(1:10)
[1]  1 10
```



Example (function structure)

```
> my.range2<-function(data,opt='r'){
+  data.min<-min(data)
+  data.max<-max(data)
+
+  switch(EXPR=opt, diff=data.max-data.min, r=c(data.min,data.max), NA)
+ }
>
> my.range2(data=1:10)
[1]  1 10
> my.range2(datas=1:10)
다음에 오류my.range2(datas = 1:10) : 사용되지 않은 인수 (datas = 1:10)
> my.range2(1:10)
[1]  1 10
> my.range2(data=1:10,opt='diff')
[1] 9
> my.range2(1:10,'diff')
[1] 9
> my.range2(1:10,'x')
[1] NA
> my.range2(1:10,'r')
[1]  1 10
> my.range2('diff',1:10)
다음에 오류switch(EXPR = opt, diff = data.max - data.min, r = c(data.min,  :
  EXPR는 길이 1 인 벡터여야 합니다
> my.range2(opt='diff',data=1:10)
[1] 9
```

Example (function structure)

```
> my.range3<-function(data,...){
+  data.min<-min(data)
+  data.max<-max(data)
+
+  args<-list(...)
+  print(args)
+
+  opt<-ifelse(!is.null(args$opt),args$opt,'')
+
+  switch(EXPR=opt, diff=data.max-data.min, r=c(data.min,data.max), NA)
+ }
>
> my.range3(1:10)
list()
[1] NA
> my.range3(1:10,opt='r')
$opt
[1] "r"
[1]  1 10
> my.range3(1:10,opt='r',arg1='a',arg2=1:10)
$opt
[1] "r"
$arg1
[1] "a"
$arg2
 [1]  1  2  3  4  5  6  7  8  9 10
[1]  1 10
```

Example (function structure)

```
> my.mean
function(data) sum(data)/length(data)
> my.sd
function(data){
data.var<-sum((data-my.mean(data))^2)/(length(data)-1)
data.sd<-sqrt(data.var)
return(data.sd)
}
> data.sd
에러: 개체 'data.sd'이 없습니다
```

Example (function structure)

```
> rng<-my.range3(1:10,opt='r')
$opt
[1] "r"

> rng
[1]  1 10
> diff(rng)
[1] 9
```

Example

```
> is.function(my.sd)
[1] TRUE
> args(my.sd)
function (data)
NULL
> attributes(my.sd)
$srcref
function(data){
data.var<-sum((data-my.mean(data))^2)/(length(data)-1)
data.sd<-sqrt(data.var)
return(data.sd)
}
```