Advanced Scientific Computing with R

3. Conditions, loops, apply and functions

Michael Hahsler

Southern Methodist University

September 15, 2015

These slides are largely based on "An Introduction to R"
`http://CRAN.R-Project.org/`

Table of Contents

if

```
R> x <- 12
R> if(x>10) {          # result of condition needs length 1
+      cat("x is >10")
+ } else {
+      cat("x is <=10")
+ }
x is >10
R> x <- c(12, 16, 3)
R> if(all(x>10)) cat("All values in x are >10")
R> if(any(x>10)) cat("There is at least one value >10")
There is at least one value >10
R> c(FALSE,TRUE,TRUE) | c(FALSE,TRUE,FALSE)
[1] FALSE  TRUE  TRUE
R> c(FALSE,TRUE,TRUE) || c(FALSE,TRUE,FALSE)
[1] FALSE
```

Table of Contents

for

```
R> x <- 0
R> for(i in 1:5) {
+     x <- x+i
+ }
R> x
[1] 15
R> sum(1:5)
[1] 15
R> l <- list(a=2, b=1:2, c=4)
R> x <- 0
R> for(i in l) { x <- x+i }
R> x
[1] 7 8
```

while

```
R> x <- 0
R> i <- 1
R> while(i <=5) { x <- x+i; i<-i+1 }
R> x
[1] 15
```

`break` and `next` work as expected.
Note: Loops are not very frequently used in R since most problems can be solved more efficiently using functions and vectorization.

Table of Contents

Functions

R is a functional programming language. Functions are objects of mode “function”.

```
R> inc <- function(x) { x+1 }
R> inc
function(x) { x+1 }
R> mode(inc)
[1] "function"
R> inc(5)
[1] 6
R> inc(1:10)
 [1]  2  3  4  5  6  7  8  9 10 11
```

Since functions are regular (first class) objects they can be passed on as arguments and returned by functions.

Named arguments and defaults

```
R> inc <- function(x, by = 1) { x + by }
R> inc(5)
[1] 6
R> inc(1:5, 10)
[1] 11 12 13 14 15
R> inc(1:5, by=10)
[1] 11 12 13 14 15
R> inc(by=10, x=1:5)
[1] 11 12 13 14 15
R> inc(matrix(1:4, nrow=2), 10)
     [,1] [,2]
[1,]   11   13
[2,]   12   14
```

Functions return the value of the last expression (or use `return(val)`).

Table of Contents

lapply/sapply – apply functions to each element in a lists

```
R> l <- list(1:3, 6, 7:3)
R> lapply(l, FUN=function(x) { rev(x) })
[[1]]
[1] 3 2 1

[[2]]
[1] 6

[[3]]
[1] 3 4 5 6 7
R> sapply(l, length)
[1] 3 1 5
```

apply – apply functions to a matrix

```
R> m <- matrix(1:9, nrow=3)
R> apply(m, MARGIN=1, sum)
[1] 12 15 18
R> apply(m, MARGIN=2, sum)
[1]  6 15 24
R> rowSums(m)
[1] 12 15 18
R> colSums(m)
[1]  6 15 24
```

Table of Contents

Exercises

1. Create x by `x <- runif(100)`. Write a function with the name `avg_gt` with two formal arguments: a vector x and a value gt. The functions computes the average of the values greater than gt in x. Write a version with a loop and if and one version without loops and if statements.
2. Create a list with 5 numeric vectors (lengths and values of your choice). Sort all vectors in the list. Hint: see `sort()`.
3. Write a function which computes the smallest value in each column of a given matrix. Create a random 5×5 matrix to test the function.