MSMS 105 - Computing with R

Ananda Biswas

Contents

1 Implicit Looping

1.1 Vectorized Operations

```
x <- 1:10
x + 14

##  [1] 15 16 17 18 19 20 21 22 23 24

x^2

##  [1]   1   4   9  16  25  36  49  64  81 100
```

1.2 *apply* family of functions

```
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

1.2.1 *apply*

In *apply*, the first argument is data; second argument is 1 or 2, 1 indicating rows and 2 indicating columns; the third argument is the function to be applied.

```
apply(iris[, 1:4], 1, sum)

##   [1] 10.2  9.5  9.4  9.4 10.2 11.4  9.7 10.1  8.9  9.6 10.8 10.0  9.3  8.5 11.2
##  [16] 12.0 11.0 10.3 11.5 10.7 10.7 10.7  9.4 10.6 10.3  9.8 10.4 10.4 10.2  9.7
##  [31]  9.7 10.7 10.9 11.3  9.7  9.6 10.5 10.0  8.9 10.2 10.1  8.4  9.1 10.7 11.2
##  [46]  9.5 10.7  9.4 10.7  9.9 16.3 15.6 16.4 13.1 15.4 14.3 15.9 11.6 15.4 13.2
##  [61] 11.5 14.6 13.2 15.1 13.4 15.6 14.6 13.6 14.4 13.1 15.7 14.2 15.2 14.8 14.9
##  [76] 15.4 15.8 16.4 14.9 12.8 12.8 12.6 13.6 15.4 14.4 15.5 16.0 14.3 14.0 13.3
##  [91] 13.7 15.1 13.6 11.6 13.8 14.1 14.1 14.7 11.7 13.9 18.1 15.5 18.1 16.6 17.5
## [106] 19.3 13.6 18.3 16.8 19.4 16.8 16.3 17.4 15.2 16.1 17.2 16.8 20.4 19.5 14.7
## [121] 18.1 15.3 19.2 15.7 17.8 18.2 15.6 15.8 16.9 17.6 18.2 20.1 17.0 15.7 15.7
## [136] 19.1 17.7 16.8 15.6 17.5 17.8 17.4 15.5 18.2 18.2 17.2 15.7 16.7 17.3 15.8
```

```
apply(iris[, 1:4], 2, sum)

## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##        876.5        458.6        563.7        179.9
```

```
apply(iris[, 1:4], 2, FUN = function(a) {
    sum(a^2)
})  # produces sum of square of all elements in a column

## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##      5223.85      1430.40      2582.71       302.33
```

1.2.2 *lapply*

In *lapply*, the first argument is a list (say X) and the second argument is a function. It returns a list of the same length of X, each element of which is the result of applying the function to the corresponding element of X.

lapply stands for "list apply".

```
mat1 <- matrix(data = rbinom(14, 25, prob = 0.5))
mat2 <- matrix(data = rbinom(24, 25, prob = 0.5))
mat3 <- matrix(data = rbinom(34, 25, prob = 0.5))
my_list <- list(mat1, mat2, mat3)

lapply(my_list, FUN = sum)

## [[1]]
## [1] 178
##
## [[2]]
## [1] 305
##
## [[3]]
## [1] 398
```

```
lapply(my_list, FUN = mean)

## [[1]]
## [1] 12.71429
##
## [[2]]
## [1] 12.70833
##
## [[3]]
## [1] 11.70588

unlist(lapply(my_list, FUN = mean))

## [1] 12.71429 12.70833 11.70588
```

2 *break* statement

```
for (i in 1:10) {
    ifelse(i == 6, break, print(i^2))
    # exits the loop when i is equal to 6
}

## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

3 *next* statement

```
for (i in 1:10) {
    ifelse(i == 6, next, print(i^2))
    # skips the printing job when i is equal to 6
}

## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
## [1] 49
## [1] 64
## [1] 81
## [1] 100
```