LAB 03: INTRODUCTION TO R – DATA TYPES & BASIC OPERATIONS

LISTS

Lists are a special type of vector that can contain elements of different classes. Lists are very important data type in R and you should get to know them well.

```
> x <- list(1,"a",TRUE,1+4i)
> x
[[1]]
[1]  1

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```

FACTORS

Factors are used to represent categorical data. Factors can be unordered or ordered. One can think of a factor as an integer vector where each integer has a label.

- Factors are treated specially by modelling functions like lm() and glm()
- Using factors with labels is better than using integers because factors are self-describing; having a variable that has values "Male" and "Female" is better than a variable that has values 1 and 2.

```
> x <- factor(c("yes","yes","no","yes","no"))
> x
[1] yes yes no yes no
Levels: no yes

>table(x)
x

no yes
2  3

>unclass(x)
[1] 2 2 1 2 1
attr(,"levels")
[1] "no" "yes"
```

The order of the levels can be set using the levels argument to factor(). This can be important in linear modelling because the first level is used as baseline level.

```
> x <- factor(c("yes","yes","no","yes","no"), levels = c("yes","no"))
> x
[1] yes yes no yes no
Levels: yes no
```

DATA FRAMES

Data frames are used to store tabular data

- They are represented as a special type of list where every element of the list must have same length.
- Each element of the list can be thought of as a column and the length of each element of the list is number of rows.
- Unlike matrices, data frames can store different classes of objects in each column (just like lists), matrices must have every element be the same class.
- Data frames also have a special attribute called row.names
- Data frames are usually created by calling read.table() or read.csv()
- Can be converted to a matrix by calling data.matrix()

```
> x <- data.frame(foo = 1:4, bar = c(T,T,F,F))
> x

 Foo bar
1 1  True
2 2  True
3 3  False
4 4  False

• nrow(x)
[1] 4

>ncol(x)
[1] 2
```

CONTROL STRUCTURES

Control structures in R allow you to control the flow of exceution of the program depending on runtime conditions. Common structures are:

- if,else: testing the condition
- for: exceute a loop for a fixed number of time
- while: execute a loop while a condition is true

- repeat: execute an infinite loop
- break: break the execution of loop
- next: skip an iteration of a loop
- return: exit a function

```
This is a valid if structure
> if(x>3)
{
  y<-10
}
else
{
  y<-0
}

So is this one.
>if(x>3)
{
   10
}
Else
{
   0
}

For loop structure

for(i in 1:10)
{
print(i)
}

These three loops have the same behaviour
x<-c("a","b","c","d")

for(i in 1:4)
{
  Print(x[i])
}

for(i in seq_along(x))
{
  Print(x[i])
}

for(letter in x)
{
   Print(letter)
}

For(i in 1:4) print(x[i])

Nested for loops

x<-matrix(1:6,2,3)
```

```
for(i in seq_len(nrow(x)))
{
  For(j in seq_len(ncol(x)))
{
     Print(x[i,j])
}
}
```

While loop

```
count<-0

while(count<10){
   print(count)
   count<-count+1
}
```

Repeat

```
x0<-1
tol<-1e-8

repeat{
     x1<-ComputeEstimate()
     if(abs(x1-x0) < tol) {
         break
    } else {
        x0 <- x1
}
}
```

Next,return

```
For(i in 1:100){
     If(i<=20){
        ##skip first 20 iterations
        Next
}
##do something here
}

Return signals that a function should exit and return a value
```