**Week 3 – Loops & If-else statements**

This week we will focus on the following:

1. Loops
2. Apply
3. if-else

**If – Else**

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.

When using if, else if, else statements there are few points to keep in mind.

1. An if can have zero or one else and it must come after any else if's.
2. An if can have zero to many else if's and they must come before the else.
3. Once an else if succeeds, none of the remaining else if's or else's will be tested.

Base Example:

> y1 <- c('Apple',"Orange","Bananna")

>

> ### raw if-else example

> if("Mango" %in% y1) {

+ print("Mango is found")

+ } else {

+ print("Mango is not found")

+ }

[1] "Mango is not found"

>

> if("Mango" %in% y1) {

+ print("Mango is found")

+ } else if('Bananna' %in% y1){

+ print("Bananna is found")

+ } else {

+ print('Mango and Bananna is not found')

+ }

[1] "Bananna is found"

Functional Example (what we will use most often):

> ### functional if-else (we will use this for dataframes next week)

> ifelse('Mango' %in% y1 # condition

+ ,"Mango is found" # value if true

+ ,"Mango is not found" # value if false

+ )

[1] "Mango is not found"

>

> ifelse('Mango' %in% y1 # condition

+ ,"Mango is found" # value if true

+ ,ifelse('Bananna' %in% y1 # else if condition (if 1st is false)

+ ,'Bananna is found' # 2nd value if true

+ ,'Mango and Bananna is not found' # value if both are false

+ )

+ )

[1] "Bananna is found"

**Loops**

In computer Programming, a Loop is used to execute a group of instructions or a block of code multiple times, without writing it repeatedly. The block of code is executed based on a certain condition. Loops are the control structures of a program. Using Loops in computer programs simplifies rather optimizes the process of coding. There are three kinds of loops in R, for loops, while loops, and repeat loops.

**For Loops**

The for loop in R, also known as for cycle, is a repetitive iteration in loop of any code, where at each iteration some code is evaluated through the elements of a list or vector. This is less like the for keyword in other programming languages and works more like an iterator method as found in other object-orientated programming languages. With the for loop, we can execute a set of statements, once for each item in a vector, array, list, etc.

Examples:

**Basic syntax**

> for(i in 1:10) { # Head of for-loop

+

+ x1 <- i^2 # Code block

+ print(x1) # Print results

+ }

[1] 1

[1] 4

[1] 9

[1] 16

[1] 25

[1] 36

[1] 49

[1] 64

[1] 81

[1] 100

**Appending to a vector**

> # appending to a vector

> x3 <- numeric()

> for(i in 1:10) { # Head of for-loop

+ x3 <- c(x3, i^2) # Code block

+ }

> print(x3)

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

**Nested for loop – the wrong way and the right way (hashing)**

> # nested for loop (bad bad bad

> # --> need to using hashing to eliminate complexity)

> x4 <- character() # Create empty data object

> loop\_work <- 0

> for(i in 1:5) { # Head of first for-loop

+ for(j in 1:5) { # Head of nested for-loop

+ loop\_work <- loop\_work + 1

+ x4 <- c(x4, paste(LETTERS[i], letters[j], sep = "\_")) # Code block

+ }

+ }

> ### A better way using hashing --> this is on almost every tech interview

> hash\_table <- list()

> hash\_work <- 0

> x5 <- c()

> for( i in 1:5 ) {

+ hash\_work <- hash\_work + 1

+ hash\_table[[LETTERS[i]]] <- letters[1:5]

+ }

> for(j in 1:length(hash\_table)){

+ hash\_work <- hash\_work + 1

+ x5 <- c(x5,paste(names(hash\_table[j]),hash\_table[[j]],sep='\_'))

+ }

> print(hash\_work)

[1] 10

> print(loop\_work)

[1] 25

> #make sure we just made the same two vectors

> all(x4 == x5)

[1] TRUE

**Break statements**

> # for loop with break statement

> for(i in 1:10) { # Head of for-loop

+ x6 <- i^2 # Code block

+ print(x6) # Print results

+ if(i >= 5) { # Conditionally stop for-loop

+ break # Using break-statement

+ }

+ }

[1] 1

[1] 4

[1] 9

[1] 16

[1] 25

**Next statements**

> # for loop with next statement (skip)

> for(i in 1:10) { # Head of for-loop

+ if(i %in% c(1, 3, 5, 7, 9)) { # Conditionally skip iteration

+ next # Using next-statement

+ }

+ x7 <- i^2 # Code block

+ print(x7) # Print results

+ }

[1] 4

[1] 16

[1] 36

[1] 64

[1] 100

**Iterating over a data frame**

> # iterating over a dataframe

> iris\_new1 <- iris

> for(i in 1:ncol(iris\_new1)) { # Head of for-loop

+ if(grepl("Width", colnames(iris\_new1)[i])) { # Logical condition

+ iris\_new1[ , i] <- iris\_new1[ , i] + 1000 # Code block

+ }

+ }

> head(iris\_new1)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species

1 5.1 1003.5 1.4 1000.2 setosa

2 4.9 1003.0 1.4 1000.2 setosa

3 4.7 1003.2 1.3 1000.2 setosa

4 4.6 1003.1 1.5 1000.2 setosa

5 5.0 1003.6 1.4 1000.2 setosa

6 5.4 1003.9 1.7 1000.4 setosa