R Loops

Basic Loops in R programming

Repeat Loop

The Repeat loop executes the same code again and again until specifically taken out by the programmer.

NOTE If no specific *break* statement is written, then this will be infinite loop.

```
v = c("Inside", "loop")
cnt = 1
repeat{
  print(v)
  cnt = cnt+1
  if(cnt > 5){
    break
  }
}
```

```
## [1] "Inside" "loop"
## [1] "Inside" "loop"
## [1] "Inside" "loop"
## [1] "Inside" "loop"
## [1] "Inside" "loop"
```

While Loop

Loop runs till the test condition is TRUE. When test condition is FALSE then loop stops.

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

```
cnt = 1
while (cnt <= 4)
{
  print(cnt)
  cnt = cnt + 1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

For Loop

It works on every element of the given vector or array

One element from the given vector is taken in every iteration

```
v1 = c(10,20,30)
for( i in v1){
  print(i)
}
```

```
## [1] 10
## [1] 20
## [1] 30
```

```
v <- c("IACSD","is", "in","Pune")
for ( i in v) {
  print(i)
}
```

```
## [1] "IACSD"
## [1] "is"
## [1] "in"
## [1] "Pune"
```

Break statement

break statement will break the flow of the loop which is immediate.

NOTE If there is loop within loop then control will be taken out of only one loop.

```
for ( i in 1:4){
  print(i)
  if(i == 2)
    break
}
```

```
## [1] 1
## [1] 2
```

```
for ( i in 1:4){
  for (j in 10:25){
    print(j)
    if(j == 12)
      break
  }
}
```

```
## [1] 10
## [1] 11
## [1] 12
## [1] 10
## [1] 11
## [1] 12
## [1] 10
## [1] 11
## [1] 12
## [1] 10
## [1] 11
## [1] 12
```

Next Statement

The next statement in R programming language is useful when we want to skip the current iteration of a loop without terminating it.

```
v = LETTERS[1:6]
for ( i in v){
  if (i == "D"){
    next
  }
  print(i)
}
```

```
## [1] "A"
## [1] "B"
## [1] "C"
## [1] "E"
## [1] "F"
```

Making Loops Faster

lapply function

It is used to execute a given function on every element of input vector or array or list

Example:: In following code there is list with two sequences 1 to 5 and 5 to 10

Now lapply() will apply mean function on both elements from the list

```
x <- list(a = 1:5, b = 5:10)
x$a
```

```
## [1] 1 2 3 4 5
```

```
x$b
```

```
## [1]  5  6  7  8  9 10
```

```
lapply(x, mean)# find mean of each element in x
```

```
## $a
## [1] 3
## 
## $b
## [1] 7.5
```

lapply with multiple arguments

lapply can apply a function taking multiple arguments to a given vector or array or list

In following code runif function with two arguments min and max is applied on every element of a given vector or array or list

```
x <- 1:4
lapply(x, runif, min = 10, max = 80)
```

```
## [[1]]
## [1] 74.18367
## 
## [[2]]
## [1] 64.15528 67.47851
## 
## [[3]]
## [1] 25.40679 58.09359 52.95168
## 
## [[4]]
## [1] 62.13321 39.83950 54.79650 53.17861
```

sapply (simplify lapply)

sapply() calls lapply() on its input and then applies the following algorithm:

If the result is a list where every element is length 1, then a vector is returned

If the result is a list where every element is a vector of the same length (> 1), a matrix is returned.

If it can’t figure things out, a list is returned

Example

```
x <- list(a = 1:5, b = 11:15)
sapply(x, mean)
```

```
##  a  b 
##  3 13
```