Looping in R

tailwind

13 June 2021

## Looping in R

In this markdown, I am going to prepare notes for understanding looping in R.

Looping is useful when a task is to be run multiple times. Remember Python or CPP's 'for' and 'while' loops? Like that! This task could be a particular calculation or a particular function.

We can specify the values of the variable over which the loop should run, say for values of variable 'i' going from 1 to 10. In a loop we can also specify the tasks that the code should perform each time around the variable changes and takes on a new value.

Loops help us in reducing the manual effort and do the job at hand more efficiently.

There 3 loops in R: 1. For (for each..) Loop, 2. While loop (used when we are not aware as to how many times the loop will be run), this loop checks the condition specified by the user and while the condition is true, the algorithm performs the task specified by the user. 3. Repeat loop (equivalent to 'Do Until' loop in other programming languages). Here we first specify the statement and then specify the condition(s).In repeat loop, first the task is performed and then the condition is checked.

## 1. FOR LOOP (FOR EACH...)

x = c(2,5,1,7,9,3,6)  
#we want to know how many elements are greater than 4.  
#let's make the first count variable of 0  
n = 0  
#for loop, in which y is the temporary element which runs for each val in x  
for(y in x){  
 if(y>=4)   
 n=n+1  
 };   
print(n)

## [1] 4

#after the loop condition and run, we have used semicolon to perform next task

Let's look at one more example. Say a vector contains both positive and negative elements, we want to add up only the positive ones.

#vector and sum initialised  
pn = c(-4,5,6,2,5,-3,-8,-9,10,9,12,-20)  
summ = 0  
for (j in pn){  
 if(j>0) {  
 summ = summ+j  
 }  
}  
print(summ)

## [1] 49

Bonus example. Sum of all the even numbers in range 1 to 1000

nums = 1:1000  
evSum = 0  
#in R to find modulo you use '%%' instead of "%" in Python or CPP  
for(i in nums){  
 if(i%%2==0){  
 evSum = evSum+i  
 }  
}  
evSum

## [1] 250500

Dropping some values from a range and printing only a subset:

#we use the "{next}" condition in the for loop and a semicolon for "else"  
vs = 1:20  
for(b in vs){if(b<10) {next}; print(b)}

## [1] 10  
## [1] 11  
## [1] 12  
## [1] 13  
## [1] 14  
## [1] 15  
## [1] 16  
## [1] 17  
## [1] 18  
## [1] 19  
## [1] 20

#see how we have given the print command inside the curly brackets

Similar to "next" statement, there is a "break" statement in R (like Py).

for(a in vs){   
 if(a>=10){break};  
 print(a)  
 }

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5  
## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9

## While Loop

An individual has 5,000 right now. He wants his retirement corpus to be 5,00,000 and he won't touch the money in the intervening time period. How many years would it take for him to accumulate the sum? You can assume that he can earn a 10% p.a growth rate on his money.

Though we have the mathematical foundation for calculating the number of years, that is beside the point. We want to sit free and ask the computer to do the work. We want to tell the computer that while the sum is less than the target, keep on reinvesting (i.e keep on traversing through the loop)

$Sytax while(condition to be tested){statement to be executed}

amt = 5000 #initialised amount  
rate = 0.10 #rate of growth assumed  
n = 0  
while(amt<500000){amt = amt\*(1+rate);n=n+1}; print(c(n,amt))

## [1] 49.0 533594.8

## Repeat loop

Syntax is easy to understand. Be aware, though, that the loop is initialised with curly braces.

m = 1  
repeat{print(m); m=m+1; if(m==10){break}}

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5  
## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9