Loops and Loop alternatives in R

Nadia Aquil

10/31/2017

Introduction

Loops in R are useful for repeating the same action multiple times. Loops are basic building blocks in many programming languages, and it is important to understand when to use them in R and when not to. R loops are not always the most efficient option; I will also discuss an R features known as vectorization and the apply() functions.

Loops in R

The main loop constructs in R are `for`, `while`, and `repeat`, with additional clauses `break` and `next`.

While loops are used to repeat a section of code an unknown number of times until a certain condition is met. The control flow of a `while` loop is as follows:

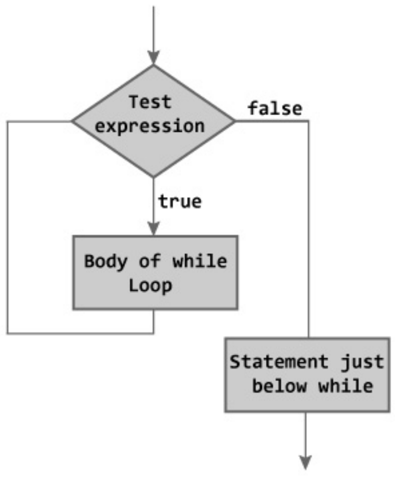
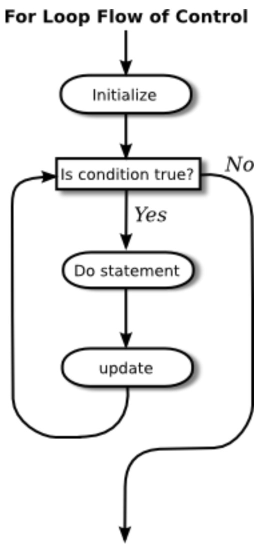
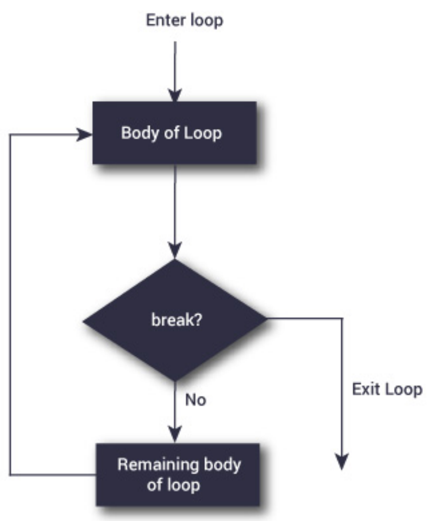


Figure: Flowchart of while Loop

For loops are used to repeat a section of code a specific number of times.The control flow of a `for` loop is as follows:



Repeat loops repeat forever and the programmer must place the condition explcitly inside the body of the loops. The `break` statement must be used to exit the loop; otherwise the loop will continue infinitely. The control flow of a `repeat` loop is as follows:



The command `break` ends the loop. The command `next` forces the loop to jump immediately to the next iteration. ## Loop examples Here are some examples of while loops.

print numbers 0 through 4: while loop:

```
#initialize variable
x <-0

while (x < 5) {
  print(x)
  #increment variable
  x <- x + 1
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

For loop:

```
for (i in 0:4){
  print(i)
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

Repeat loop:

```
x <-0
repeat {
  print(x)
  x = x + 1
  if (x == 5){
    #end loop
    break
  }
}
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

Loop Alternatives

Vectorized computations

Regular loops in R are known to be slow, because of the way R handles the boxing and unboxing of data objects. Some ways to improve speed when using loops include initializing new objects to full length before the loop (instead of increasing their size within the loop) and not doing things in the loop that can be done outside it. For example, if you need to expand an object in your loop, such as adding to an array, the object will have to be copied during every iteration of the loop, which can be costly. R is unlike most other languages in the it offers the option of using vectorized functions. Programmers used to other languages often have difficulty with this concept at first. A vectorized function is different from a loop in that it works on an entire vector of values at the same time, rather than on a single value at a time.

One example of a vectorized function is `sum` which adds up the values of a vector. The same result could be achieved by using a loop that iterates through each value and adds it to a total, but the vectorized function is cleaner and easier to read.

```r
vector <- c(1, 2, 3, 4)
sum(vector)
```

```
## [1] 10
```

Apply() functions

The apply() family is a package of functions that manipulate data in repetive ways without explicit use of loop constructs. The main function is `apply()`, which operates on arrays. It is called as follows: `apply(X, Margin, Func, ...)`, where `X` is an array, `Margin` is a varaible defining how the fucntion is applied. When `Margin = 1` it applies to rows, and when `Margin = 2` it applies to columns. When `MArgin = c(1, 2)`, it applies to both rows and columns. `Func` is the function you want to apply to the data. This can be any R function, including ones defined by the user. The following is an example of summing the values of each column in a matrix.

```r
#construct matrix
X <- matrix(c(1:20), nrow=5, ncol=5)
#sum values of each column
apply(X, 2, sum)
```

```
## [1] 15 40 65 90 15
```

There are variations of `apply()` that do different things. the `lapply()` function, for example, outputs a list.

Conclusion

Loops in R are a good tool for those new to programming, but as one gets more advanced, one should learn how to use vectorized functions and tools like `apply()` in order to more efficiently create repetitive code.

References

https://www.programiz.com/r-programming/repeat-loop

https://www.datacamp.com/community/tutorials/tutorial-on-loops-in-r

https://www.r-bloggers.com/how-to-write-the-first-for-loop-in-r/

http://www.dummies.com/programming/r/how-to-vectorize-your-functions-in-r/

https://stackoverflow.com/questions/7142767/why-are-loops-slow-in-r

https://www.datacamp.com/community/tutorials/r-tutorial-apply-family

http://adv-r.had.co.nz/Functionals.html