post01-lindsay-dahlen

Lindsay Dahlen

October 27, 2017

Loops in R

1) Introduction

In R, loops are used to iterate over a vector. Loops allow us to execute a block of code several times, thus avoiding repetitive code, which is a common rookie move. The purpose of this post is to familiarize the reader with basic loop components, summarize how to create loops, and identify cases when to best use them.

---

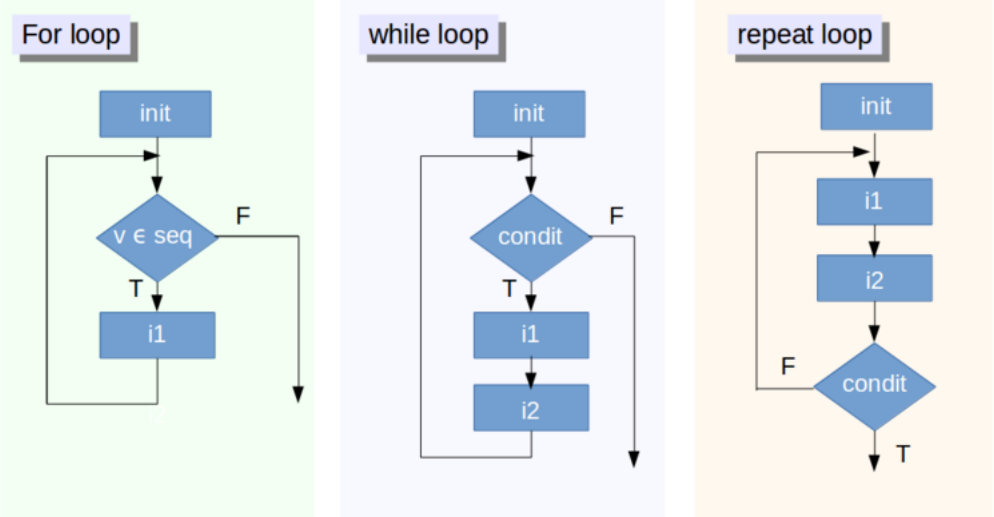
2) Motivation

My main motivation for investigating loops in R is because I particularly struggled on this subject in lab. I didn't understand how to write them or what they meant. I hope that by dedicating this post to loops I can better understand the theory behind them and be able to utilize them correctly. Especially since we seem to use loops in subsequent labs. In addition, I hope to be able to more effectively analyze data in my own research, and understanding the nuances of loops will help me achieve this goal.

---

3) About loops

There are three different types of loops supported by R. These are *for*, *while*, and *repeat*.



R Loop Structures

The most commonly used loop is *for*. A *for* loop is used to apply the same function calls to a collection of objects. This type of loop has three components: the output, the sequence, and the body. *For* loops are used when you know exactly how many times you want to execute the loop. *For* loops are used when you know the number of times you want to execute the loop. If the condition for the loop is false, the loop is not executed.

For loops are only useful if you know how long the input sequence should run for. If however you are running a coin flipping simulation and you want to loop until you get five heads in a row, you should use a *while* loop. This type of loop has two components: the condition and the body. A *while* loop will check the inputs first and then execute the loop as long as the condition is true.

Repeat loops are similar to *while* loops, but the inputted instructions are executed at least once, regardless if of the output of the loop. In this way it is the opposite of a *while* loop. It repeats the same code until a stop condition is met. Beware of creating an infinite loop!

In addition to these types of loops, there are a couple of loop control statements. These are *break* and *next*. *Break* will stop the loop, and is primarily used in *repeat* loops. *Next* will skip an iteration, jumping to the evaluation of the condition holding the current loop.

---

4) Example: *for* loop

Let's look at a situation where we want to print successive sample ids. Instead of typing "of sample A-1", "of sample A-2" we could use a *for* loop.

```
for (i in 1:15){
  print(paste("of sample A -",i))
}
```

```
## [1] "of sample A - 1"
## [1] "of sample A - 2"
## [1] "of sample A - 3"
## [1] "of sample A - 4"
## [1] "of sample A - 5"
## [1] "of sample A - 6"
## [1] "of sample A - 7"
## [1] "of sample A - 8"
## [1] "of sample A - 9"
## [1] "of sample A - 10"
## [1] "of sample A - 11"
## [1] "of sample A - 12"
## [1] "of sample A - 13"
## [1] "of sample A - 14"
## [1] "of sample A - 15"
```

A more complicated *for* loop could be used to estimate the average of squaring the result of a roll of a die.

```
nsides = 6
ntrials = 1000
trials = rep(0,ntrials)

for (j in 1:ntrials) # In a nested loop, "j" represents columns. "i" would represent lines
{
  trials[j] = sample(1:nsides,1)  # Get one sample at a time
}
mean(trials^2)
```

```
## [1] 14.806
```

5) Example: *while* loop

Let's look at a simulation where we flip a coin. To find how many tries it takes to get five heads in a row, we would use a *while* loop.

```
flip <- function() sample(c("T", "H"), 1) # Our coin flipping function

flips <- 0
nheads <- 0

# Our while function
while (nheads < 5) {
  if (flip() == "H") {
    nheads <- nheads + 1
  } else {
    nheads <- 0
  }
  flips <- flips + 1
}

flips
```

```
## [1] 196
```

6) Example: *repeat* loop

Let's say that you want to find a sequence of squares starting from 2 that is under 100. You can use a *repeat* loop to execute this, Note that you need *break* in order to stop the loop!

```
value <- 2

repeat {
  value <- value ^ 2
  print(value)
  if (value >= 100) break # don't forget to include "break"!
}
```

```
## [1] 4
## [1] 16
## [1] 256
```

7) Conclusion

As it has hopefully been made clear, using loops can help you effectively write code and analyze data. Fortunately, R allows for a few different ways to set up loops which allows the user the flexibility to write loops that complements their situation. It should be noted that loops are not the most efficient way to deal with data in R, as vectorizations can actually allow for faster calculations. The *apply* function family also serves as a

sort of “loop”, allowing you to manipulate slices of data in the form of matrices or arrays in a repetitive way. In fact, most people don’t use loops when using R. It is nonetheless an important skill to learn, as evidenced by the fact that we had an entire lab devoted to it.

While I am somewhat dismayed that loops are not commonly used in coding in R, I am grateful that researching the materials to make this post has helped me better understand the theory behind loops. I am now also aware of more efficient alternatives to loops. I hope that reading it has also helped you better understand how and when to use loops.

---

8) References

- Programiz. “R for Loop”. https://www.programiz.com/r-programming/for-loop
- Programiz. “R while Loop”. https://www.programiz.com/r-programming/while-loop
- R for Data Science. “Iteration”. http://r4ds.had.co.nz/iteration.html
- Sanchez, Gaston. “Introduction to Loops”. October 17, 2017. https://github.com/ucb-stat133/stat133-fall-2017/blob/master/tutorials/08-intro-to-loops.md
- Software Carpentry Foundation. “Loops in R”. 2016-2017. https://swcarpentry.github.io/r-novice-inflammation/15-supp-loops-in-depth/
- Tutorials Point. “R - Loops”. https://www.tutorialspoint.com/r/r_loops.htm
- Theuwissen, Martijn. “How to write the first for loop in R”. December 2, 2015. https://www.r-bloggers.com/how-to-write-the-first-for-loop-in-r/
- “18.05 R Tutorial: For Loops”. https://ocw.mit.edu/ans7870/18/18.05/s14/html/r-tut-forloop.html