POL 345 Precept Week 8: For loop

Jing Qian

2020/03/31

Welcome! This document contains materials that would be helpful for week 8 of POL 345. Specifically, we will cover two important control structures in programming: *for loop* and *if statement*, as well as the *K-Means* algorithm.

In this document, we will focus on the *for loop*.

1. For loop[1]

1.1 Basic Idea of for loop

A lot of times, you might want to repeat a specific block of code for each element in a vector, or each row/column in a dataframe/matrix, etc. The *for loop* is a widely-used control structure to complete this kind of task without copy-and-paste the same chunk of code again and again.

The basic syntax of *for loop* is as the following:

```
for (val in sequence){
  statement
}
```

In the chunk above:

- `sequence` is a vector contains elements to be iterated in the loop.
- `val` is a "placeholder", which will take on the value of each element in `seuqnce`, and would be used in the block of code in `statement`.
- `statement` is the block of code to be executed repeatedly during the loop.

Below is an example of a basic for loop:

```
x = 1:10
for (i in x){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

As you can see, in the example above, `sequence` is now the object `x`, which is a numerical vector contains ten integers: $\{1, 2, \ldots, 10\}$. `i` is the "placeholder" (`val`). And the `statement` is simply `print(i)`. In this loop, `i` takes each value in the vector `x`, then the code `print(i)` is executed.

The basic structure of the for loop can be illustrated in the following figure:

[1] Some materials from https://www.datamentor.io/r-programming/for-loop/

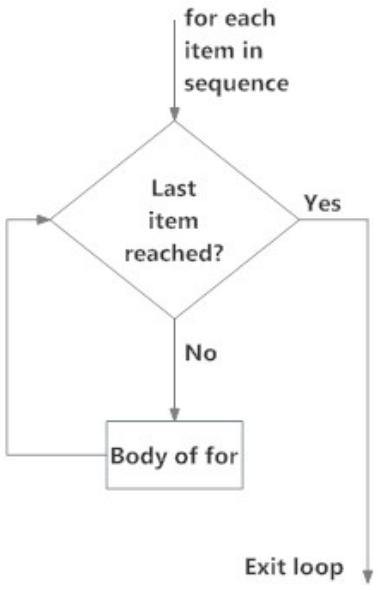

Fig: operation of for loop

1.2 Examples of for loop in different cases

In the example above, we loop over each element in `sequence` and also do some operations with that element. A common situation involving for loop is to loop over a vector of *indices*, and use that to iteratively extract each element from another object. For example:

```
y = c("A", "B", "C", "D")
for (i in 1:length(y)){
  print(y[i])
}
```

```
## [1] "A"
## [1] "B"
## [1] "C"
## [1] "D"
```

Here, the vector to be looped over is `1:length(y)` which is a numerical vector of four elements: `c(1, 2, 3, 4)`. Instead of doing any operation with these numbers, we loop over this vector to extract the corresponding element from the vector `y`, which contains four characters: `c("A", "B", "C", "D")`, and use square brackets to extract the 1th, 2nd, ... element from the vector `y`.

Similarly, we can also loop over each row or column from a dataframe/matrix. Again, by using square brackets:

```
congress = read.csv("data/congress.csv",
                    stringsAsFactors = F)
congress_head = congress[1:10, ] #The first 10 rows from the congress dataset

#Loop over each row
for (i in 1:nrow(congress_head)){
  print(congress_head[i, ])
}
```

```
##   congress district state    party   name dwnom1 dwnom2
## 1       80        0   USA Democrat TRUMAN -0.276  0.016
```

```
##   congress district   state    party       name dwnom1 dwnom2
## 2       80        1 ALABAMA Democrat BOYKIN  F. -0.026  0.796
##   congress district   state    party      name dwnom1 dwnom2
## 3       80        2 ALABAMA Democrat GRANT  G. -0.042  0.999
##   congress district   state    party        name dwnom1 dwnom2
## 4       80        3 ALABAMA Democrat ANDREWS  G. -0.008  1.005
##   congress district   state    party      name dwnom1 dwnom2
## 5       80        4 ALABAMA Democrat HOBBS  S. -0.082  1.066
##   congress district   state    party      name dwnom1 dwnom2
## 6       80        5 ALABAMA Democrat RAINS  A.  -0.17   0.87
##   congress district   state    party       name dwnom1 dwnom2
## 7       80        6 ALABAMA Democrat JARMAN  P. -0.124   0.99
##   congress district   state    party        name dwnom1 dwnom2
## 8       80        7 ALABAMA Democrat MANASCO  C. -0.031  0.892
##   congress district   state    party      name dwnom1 dwnom2
## 9       80        8 ALABAMA Democrat JONES  R. -0.225  0.888
##    congress district   state    party       name dwnom1 dwnom2
## 10       80        9 ALABAMA Democrat BATTLE  L. -0.084  0.842
```

```r
#Loop over each column
for (i in 1:ncol(congress_head)){
  print(congress_head[, i])
}
```

```
##  [1] 80 80 80 80 80 80 80 80 80 80
##  [1] 0 1 2 3 4 5 6 7 8 9
##  [1] "USA"     "ALABAMA" "ALABAMA" "ALABAMA" "ALABAMA" "ALABAMA" "ALABAMA"
##  [8] "ALABAMA" "ALABAMA" "ALABAMA"
##  [1] "Democrat" "Democrat" "Democrat" "Democrat" "Democrat" "Democrat"
##  [7] "Democrat" "Democrat" "Democrat" "Democrat"
##  [1] "TRUMAN"      "BOYKIN  F."  "GRANT  G."   "ANDREWS  G." "HOBBS  S."
##  [6] "RAINS  A."   "JARMAN  P."  "MANASCO  C." "JONES  R."   "BATTLE  L."
##  [1] -0.276 -0.026 -0.042 -0.008 -0.082 -0.170 -0.124 -0.031 -0.225 -0.084
##  [1] 0.016 0.796 0.999 1.005 1.066 0.870 0.990 0.892 0.888 0.842
```

Of course, in very rare cases would we want to loop over a certain object and print out each of the element. The block of code (`sequence`) could be any operation that you want to do.

1.3 Store results from for loop

A lot of times, we want to *store* the results from a for loop. In order to do so, we often create a *container*, which is an empty object, and put the result from each iteration of the for loop inside the corresponding element of that container. For example:

```r
#Say we calculate the area of circles with different semidiameters

r = c(1, 3, 4, 5, 6) #a vector of the semidiameters of five circles
area = c() #The "container" (an empty vector) to store the calculated areas

#Let's write a for loop to calculate the area of each circle:
for (i in 1:length(r)){
  area[i] = pi * r[i]^2
}

print(area)
```

```
## [1]   3.141593  28.274334  50.265482  78.539816 113.097336
```

In the example above, we calculated the area of five different circles, with their semidiameters stored in the vector `r`. In order to do so, we loop over each element in `r`, and store the calculated area in the vector `area`. It's worth mentioning that, although we could directly loop over the vector `r` by using `for (i in r)`, this would be problematic when we want to store the results in the vecttor `area`. Therefore, loop over a vector of indices and extract the value from the `r` with indexing could help us to conveiently store the results.