Loops

Kira Lai

2025-01-17

for Loops

A for loop repeats a chunk of code many times, once for each element in a set of input. for loops provide a way to tell R, “Do this for every value of that.”

```
for (value in c(1,2,3)) {
   print("one run")
}
```

```
## [1] "one run"
## [1] "one run"
## [1] "one run"
```

What values will the for loop assign to value?

It will use the elements in the set that you run the loop on. `for` starts with the first element and then assigns a different element to value on each run of the for loop, until all of the elements have been assigned to value. For example, the for loop below will run `print(value)`

```
for (value in c(1,2,3)) {
  print(value)
}
## [1] 1
## [1] 2
## [1] 3
```

If you look at value after the loop runs, you will see that it still contains the value of the last element in the set:

```
value
## [1] 3
```

Example of for loops

```
chars <- vector(length = 4)
chars
## [1] FALSE FALSE FALSE FALSE
words <- c("My", "fourth", "for", "loop")

for (i in 1:4) {
  chars[i] <- words[i]
}

chars
## [1] "My"     "fourth" "for"    "loop"
## "My"     "fourth" "for"    "loop"
```

Expand.grid to create dataset

The `expand.grid` function in R provides a quick way to write out every combination of the elements in n vectors.

```
wheel <- c("DD", "7", "BBB", "BB", "B", "C", "0")
wheel
## [1] "DD"  "7"   "BBB" "BB"  "B"   "C"   "0"
combos <- expand.grid(wheel, wheel, wheel,
stringsAsFactors = FALSE)
head(combos,3)
##   Var1 Var2 Var3
## 1   DD   DD   DD
## 2    7   DD   DD
## 3  BBB   DD   DD
```

Using for loop to calculate the prize of each rows in a combo

Let’s use a for loop to calculate the prize for each row in combos

```
combos$prize <- NA
head(combos,4)
```

```
##   Var1 Var2 Var3 prob1 prob2 prob3 prize
## 1   DD   DD   DD  0.03  0.03  0.03    NA
## 2    7   DD   DD  0.03  0.03  0.03    NA
## 3  BBB   DD   DD  0.06  0.03  0.03    NA
## 4   BB   DD   DD  0.10  0.03  0.03    NA
```

Example cont

Construct a for loop that will run score on all 343 rows of combos. The loop should run score on the first three entries of the _i_th row of combos and should store the results in the _i_th entry of combos$prize

```
for (i in 1:nrow(combos)) {
  symbols <- c(combos[i, 1], combos[i, 2], combos[i, 3])
  combos$prize[i] <- score(symbols)
}
head(combos,3)
##    Var1 Var2 Var3 prob1 prob2 prob3 prize
## 1    DD   DD   DD  0.03  0.03  0.03   800
## 2     7   DD   DD  0.03  0.03  0.03     0
## 3   BBB   DD   DD  0.06  0.03  0.03     0
```

While loop

A while loop reruns a chunk while a certain condition remains TRUE. To create a while loop, follow while by a condition and a chunk of code, like this:

```
while (condition) {
  code
}
plays_till_broke <- function(start_with) {
  cash <- start_with
  n <- 0
  while (cash > 0) {
    cash <- cash - 2
    n <- n + 1
  }
  n
}

plays_till_broke(6)
## [1] 3
```

repeat Loops

`repeat` loops are even more basic than `while` loops. They will repeat a chunk of code until you tell them to stop (by hitting Escape) or until they encounter the command break, which will stop the loop.

You can use a `repeat` loop to recreate `plays_till_broke`, it simulates how long it takes to lose money while playing slots:

```
plays_till_broke <- function(start_with) {
  cash <- start_with
  n <- 0
  repeat {
    cash <- cash - 2
    n <- n + 1
    if (cash <= 0) {
      break
    }
  }
  n
}
plays_till_broke(100)
## [1] 50
```

Summary

- You can repeat tasks in R with `for`, `while`, and `repeat` loops. To use `for`, give it a chunk of code to run and a set of objects to loop through. `for` will run the code chunk once for each object.
- Repetition plays an important role in data science. It is the basis for simulation, as well as for estimates of variance and probability
- Loops are not the only way to create repetition in R (consider `replicate` for example), but they are one of the most popular ways.