Loops

Austin Mercado

2023-04-06

For loops are another way that we can tell a omputer to repeat tasks for us. They are versatile and very explicit, so that means that we are contorolling everything that is run on each iteration of the loop (mostly).As opposed to apply functions, where the iterations happen kind of under the hood, and the apply functions can only be used to loop over (iterate) on one function.

Loops can let us iterate over multiple functions and whole blocks of code.

THe general structure of a for loop

```
for (variable_used_inside_the_loop in object_with_values {
  do something with(value)
}
```

```
## Error: <text>:1:58: unexpected '{'
## 1: for (variable_used_inside_the_loop in object_with_values {
##                                                              ^
```

```
lengths <- c(13.3, 15, 100)

for (value in lengths) {
  mass <- 0.73 * value^2
  print(mass)
}
```

```
## [1] 129.1297
## [1] 164.25
## [1] 7300
```

```
numbers <- c(1, 2, 3, 4, 5)
for (number in numbers){
  number <- number * 3
  print(number)
}
```

```
## [1] 3
## [1] 6
## [1] 9
## [1] 12
## [1] 15
```

```
mass_lbs <- c(2.2, 3.5, 9.6, 1.2)
for (mass_lbs in mass_lbs){
  mass_kg <- mass_lbs * 2.2
  print(mass_kg)
}
```

```
## [1] 4.84
## [1] 7.7
## [1] 21.12
## [1] 2.64
```

Looping over an index

What is an index in R? It is the numeric position of values inside any data structure in R. For example in the following vector

```
flowers <- c("lilacs", "daisies", "jasmins")
str(flowers)
```

```
##  chr [1:3] "lilacs" "daisies" "jasmins"
```

```
# To access the second element in the vector, I need to use the index number 2 inside the square bracke
flowers[2]
```

```
## [1] "daisies"
```

We can use numbers as indices to loop over values inside a vector.

```
for (i in 1:3) {
  print(flowers[i])
}
```

```
## [1] "lilacs"
## [1] "daisies"
## [1] "jasmins"
```

```
birds = c('robin', 'woodpecker', 'blue jay', 'sparrow')
for (i in 1:length(birds)){
  print(birds[i])
}
```

```
## [1] "robin"
## [1] "woodpecker"
## [1] "blue jay"
## [1] "sparrow"
```

Storing results from a far loop using indices

So far we have just printed some values and results from equations

Usually what we need is to save the results of running a for loop, so that we can use them later

When we are using a function what do we do to store the results of the function? We assign to a variable name

```
my_results <- 0.73 * lengths^2

# but in for loops we do not have that option
```

```
for (i in 1:length(flowers)) {
  upper <- toupper(flowers[i])
  print(upper)
}
```

```
## [1] "LILACS"
## [1] "DAISIES"
## [1] "JASMINS"
```

```
# To create an empty vector, we use the function vector()

my_results <- vector(mode = "logical", length = length(flowers))
my_results
```

```
## [1] FALSE FALSE FALSE
```

```
for (i in 1:length(flowers)) {
  upper <- toupper(flowers[i])
  my_results[i] <- upper
}
my_results
```

```
## [1] "LILACS"  "DAISIES" "JASMINS"
```

```
radius <- c(1.3, 2.1, 3.5)
areas <- vector(mode = "numeric", length = length(radius))
for (i in 1:length(areas)){
  areas[i] <- pi * radius[i] ^ 2
}
areas
```

```
## [1]  5.309292 13.854424 38.484510
```

Looping over multiple obect with indices

We have 3 vectors

```r
dino_names <- c("T-rex", "Ankylosaur", "Triceratops")
# We have different a values for each of these dino species
a_values <- c(0.73, 5.4, 100)
b_values <- c(2, 0.5, 1.2)
dino_lengths <- c(15, 3, 20)
dino_masses <- vector(mode = "numeric", length = length(dino_names))
dino_masses
```

```
## [1] 0 0 0
```

We can iterate through these evalues within a loop

```r
for (i in 1:length(dino_names)) {
  print(dino_names[i])
  mass <- a_values[i] * dino_lengths[i]^b_values[i]
  dino_masses[i] <- mass
}
```

```
## [1] "T-rex"
## [1] "Ankylosaur"
## [1] "Triceratops"
```

```r
dino_masses
```

```
## [1]  164.250000    9.353074 3641.128406
```

```r
lengths = c(1.1, 2.2, 1.6)
widths = c(3.5, 2.4, 2.8)
areas <- vector(mode = "numeric", length = 3)
for (i in 1:length(lengths)) {
  areas[i] <- lengths[i] * widths[i]
}
areas
```

```
## [1] 3.85 5.28 4.48
```

Excercise

```r
list.files("../data-raw/individual_collar_data/")
```

```
##  [1] "collar-data-A1-2016-02-26.txt"  "collar-data-B2-2016-02-26.txt"
##  [3] "collar-data-C3-2016-02-26.txt"  "collar-data-D4-2016-02-26.txt"
##  [5] "collar-data-E5-2016-02-26.txt"  "collar-data-F6-2016-02-26.txt"
##  [7] "collar-data-G7-2016-02-26.txt"  "collar-data-H8-2016-02-26.txt"
##  [9] "collar-data-I9-2016-02-26.txt"  "collar-data-J10-2016-02-26.txt"
```

```r
individuals = paste(c('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'), c(1:10), sep = "")
for (individual in individuals) {
    lat = vector("numeric", 24)
    long = vector("numeric", 24)
    lat[1] = rnorm(1, mean = 26, sd = 2)
    long[1] = rnorm(1, mean = -35, sd = 3)
    for (i in 2:24) {
        lat[i] = lat[i - 1] + rnorm(1, mean = 0, sd = 1)
        long[i] = long[i - 1] + rnorm(1, mean = 0, sd = 1)
    }
    times = seq(from=as.POSIXct("2016-02-26 00:00", tz="UTC"),
                to=as.POSIXct("2016-02-26 23:00", tz="UTC"),
                by="hour")
    df = data.frame(date = "2016-02-26",
                    collar = individual,
                    time = times,
                    lat = lat,
                    long = long)
    write.csv(df, paste("collar-data-", individual, "-2016-02-26.txt", sep = ""))
}
zip("../data-raw/individual_collar_data.zip", list.files(pattern = "collar-data-[A-Z][0-9]+-.*"))
```