Loops

Anita Dhillon

2023-04-06

Loops iterate over multiple functions and whole blocks of code

General Structure of a loop

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

Exercise 1

The code below prints the numbers 1 through 5 one line at a time. Modify it to print each of these numbers multiplied by 3.

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

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

Exercise 2

Write a for loop that loops over the following vector and prints out the mass in kilograms (mass_kg = 2.2 * mass_lb)

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

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

Looping over using an index

What is an index in R?

- It is the numeric position of values inside any data structure in R For example n of the following vector

```
flowers <- c("Liliacs" , "daisies" , "jasmines")
str (flowers)
```

```
##  chr [1:3] "Liliacs" "daisies" "jasmines"
```

we can use numbers as indicators to loop over values within a vector.

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

```
## [1] 1
## [1] "Liliacs"
## [1] 2
## [1] "daisies"
## [1] 3
## [1] "jasmines"
```

Exercise 3

Complete the code below so that it prints out the name of each bird one line at a time.

```
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

- So far we have just printed some values and results from some equations.
- We need to save the results of running a for a loop , to use for later

When we are using a function, what do we use to store the results of the function? - we assign the result to a variable name

“{r} my_results <- 0.73 * lengths ˆ2

‘‘{r, eval=FALSE}
my_result <- for (variable in vector) {

}

The only way to save results from each iteration of the loop is to to save them into an empty object.

Run the following loop.

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

```
## [1] "LILIACS"
## [1] "DAISIES"
## [1] "JASMINES"
```

To store the output of running the function ‘toupper()’. we need to create an empty vector. To create this empty vector, we use the function ‘vector()’

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

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

We are now able to use this empty vector and indicies inside a loop to store results:

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

```
## [1] "LILIACS"  "DAISIES"  "JASMINES"
```

Exercise 4

Complete the code below so that it stores one area for each radius.

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

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

Looping over multiple objects with indices

we have 3 vectors

```
dino_names <- c("T-rex", "Ankylosaur" , "Triceratops")
# We have different a and db 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 values within a loop.

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

```
## [1] "T-rex"
## [1] 164.25
## [1] "Ankylosaur"
## [1] 9.353074
## [1] "Triceratops"
## [1] 3641.128
```

```
dino_masses
```

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

```
dino_masses[4] <- NA
dino_masses[100] <- NA
head(dino_masses)
```

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

Exercise 5

```
lengths = c(1.1,2.2,1.6)
widths = c(3.5,2.4,2.8)
areas <-vector(mode = "numeric", length = 3)
areas <- vector(mode="numeric",length=length(widths))
areas
```

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

```
for (i in 1: length(lengths)) {
  areas[i] <- lengths[i] * widths[i]
}
areas
```

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

Loops part 2

```
dino_data <-read.csv(file = "../Data-raw/dinosaur_lengths.csv")
head(dino_lengths)
```

```
## [1] 15  3 20
```

Size Estimates By Name Loop (Exercise 6)

Write a function mass_from_length() that uses the equation mass <- a * length^b to estimate the size of a dinosaur from its length. This function should take two arguments, length and species. For each of the following inputs for species, use the given values of a and b for the calculation from Seebacher 2001:

```
mass_from_length <- function(dino_lengths, dino_species){
  if (dino_species == "Stegosauria"){
    a <- 10.95
    b <- 2.64
  } else if (dino_species == "Theropoda"){
    a <- 0.73
    b <- 3.63
  } else if ( dino_species == "Sauropoda"){
    a <- 214.44
    b <- 1.46
  } else {
    a <- 25.37
    b <- 2.49
}
mass <- a * dino_lengths^b
return(mass)
}
```

Use this function and a for loop to calculate the estimated mass for each dinosaur, store the masses in a vector, and after all of the calculations are complete show the first few items in the vector using head().

```
my_results <- vector(mode = "numeric", length = nrow(dino_data))
for (i in 1: nrow(dino_data)) {
  mass_i <- mass_from_length(dino_length =
dino_data$lengths[i], dino_species = dino_data$species[i])
  print(mass_i)
  my_results[i] <- (mass_i)
 head(mass_i)
}
```

```
## [1] 24341.68
## [1] 27017.9
## [1] 67453.38
## [1] 22114.19
## [1] 53884.76
## [1] 52026.34
## [1] 57349.47
## [1] 14160.49
## [1] 49677.75
## [1] 42105.92
## [1] 10221.75
## [1] 15339.99
## [1] 70624.1
## [1] 23883.82
## [1] 28552.86
## [1] 18801.37
## [1] 19438.67
## [1] 49355.62
## [1] 19607.97
## [1] 16032.84
## [1] 49000.67
## [1] 50350.11
## [1] 15969.08
## [1] 29582.85
## [1] 15201.46
## [1] 12980.54
## [1] 9937.867
## [1] 9599.415
## [1] 49245.96
## [1] 23846.75
## [1] 53805.66
## [1] 53326.47
## [1] 44468.21
## [1] 15554.98
## [1] 18544.12
## [1] 53797.12
## [1] 24006.41
## [1] 82492.32
## [1] 17909.04
## [1] 38694.5
## [1] 80303.18
## [1] 19592.8
## [1] 10614.79
## [1] 29560.81
## [1] 71658.48
## [1] 67080.58
## [1] 83961.66
## [1] 48056.24
## [1] 26284.04
## [1] 21766
## [1] 63571.87
## [1] 5480.255
## [1] 33917.31
## [1] 22778.03
```

```
## [1] 13819.17
## [1] 21154.15
## [1] 17635.1
## [1] 14577.59
## [1] 24547.41
## [1] 14032.34
## [1] 30231.69
## [1] 39387.23
## [1] 11293.89
## [1] 72743.8
## [1] 23679.9
## [1] 64258.57
## [1] 14931.09
## [1] 16323.82
## [1] 9763.574
## [1] 49650.74
## [1] 51311.26
## [1] 7599.703
## [1] 42515.79
## [1] 34097.02
## [1] 23645.77
## [1] 29052.22
## [1] 46920.03
## [1] 70529.03
## [1] 9484.528
## [1] 44918.76
## [1] 68340.49
## [1] 44959.63
## [1] 66211.48
## [1] 48249.49
## [1] 11730.17
## [1] 53383.31
## [1] 52295.18
## [1] 30410.78
## [1] 32692.59
## [1] 35709.06
## [1] 40358.29
## [1] 38891.14
## [1] 30878.44
## [1] 19125.43
## [1] 41568.77
## [1] 45255.19
## [1] 8697.216
## [1] 19627.36
## [1] 40089.48
## [1] 73091.69
## [1] 13411.39
## [1] 33157.5
## [1] 10874.73
## [1] 24554.93
## [1] 16819.49
## [1] 18421.45
## [1] 47331.73
## [1] 19645.72
```

```
## [1] 38206.24
## [1] 53196.02
## [1] 22346.11
## [1] 63627.46
## [1] 22685.1
## [1] 33116.12
## [1] 13613.98
## [1] 34685.79
## [1] 41788.93
## [1] 18654.52
## [1] 43941.4
## [1] 101482.4
## [1] 89149.26
## [1] 36031.23
## [1] 20820.84
## [1] 48261.69
## [1] 22232.85
## [1] 59702.6
## [1] 88372.18
## [1] 16321.77
## [1] 22748.88
## [1] 88206.72
## [1] 22037.15
## [1] 11308.08
## [1] 57205.92
## [1] 25987.77
## [1] 49818.25
## [1] 13106.77
## [1] 53208.68
## [1] 32112.44
## [1] 55193.14
## [1] 16984.46
## [1] 10859.93
## [1] 93973.02
## [1] 52342.26
## [1] 19151.79
## [1] 93683.15
## [1] 13954.19
## [1] 45383.03
## [1] 15021.82
## [1] 35933.33
## [1] 140435.6
## [1] 20467.33
## [1] 23869.64
## [1] 29763.86
## [1] 49939.78
## [1] 15211.98
## [1] 57098.94
## [1] 23588.7
## [1] 27381.01
## [1] 85932.51
## [1] 71891.31
## [1] 9331.295
## [1] 14232.18
```

```
## [1] 32154.79
## [1] 32005.5
## [1] 16613.44
## [1] 7904.857
## [1] 56630.98
## [1] 26352.26
## [1] 19880.48
## [1] 15543.68
## [1] 15493.65
## [1] 13546.03
## [1] 44830.07
## [1] 36095.08
## [1] 42437.61
## [1] 48670.47
## [1] 19236.8
## [1] 51637.91
## [1] 59681.73
## [1] 44120.18
## [1] 9535.583
## [1] 59840.35
## [1] 47309.3
## [1] 36074.02
## [1] 54121.9
## [1] 44822.18
## [1] 14232.68
## [1] 34751.5
## [1] 11292.44
## [1] 57791.77
## [1] 49994.24
## [1] 43597.72
## [1] 22002.08
## [1] 19554.17
## [1] 13223.77
## [1] 15403.3
## [1] 49751.29
## [1] 68935.5
## [1] 9172.206
## [1] 90096.48
## [1] 25796.76
## [1] 50594.43
## [1] 61952.97
## [1] 20132.53
## [1] 34580.53
## [1] 13979.44
## [1] 15481.07
## [1] 12104
## [1] 21789.44
## [1] 54009.09
## [1] 13812.36
## [1] 8071.939
## [1] 21144.51
## [1] 44097.85
## [1] 16250.3
## [1] 70066
```

```
## [1] 11170.35
## [1] 22826.56
## [1] 40885.09
## [1] 17292.04
## [1] 18394.39
## [1] 50267.63
## [1] 70791.03
## [1] 28464.28
## [1] 41431.35
## [1] 38283.88
## [1] 14242.92
## [1] 36183.35
## [1] 97264.77
## [1] 52014.37
## [1] 32865.06
## [1] 21084.07
## [1] 11906.15
## [1] 17964.36
## [1] 14844.5
## [1] 13079.84
## [1] 76048.11
## [1] 18843.87
## [1] 64162.54
## [1] 30737.51
## [1] 37983.03
## [1] 18711.96
## [1] 22636.97
## [1] 29868.75
## [1] 42799.61
## [1] 41040.81
## [1] 43632.46
## [1] 103600.9
## [1] 31137.2
## [1] 40914.37
## [1] 10330.76
## [1] 23659.8
## [1] 19126.02
## [1] 17175.85
## [1] 28017.23
## [1] 54437.04
## [1] 44031.57
## [1] 20657.06
## [1] 13275.05
## [1] 98755.59
## [1] 8222.362
## [1] 45088.78
## [1] 108964.1
## [1] 56081.86
## [1] 5845.741
## [1] 26356.59
## [1] 72062.04
## [1] 59636.24
## [1] 14857.58
## [1] 45043.7
```

```
## [1] 47427.02
## [1] 34572.84
## [1] 68032.86
## [1] 11807.18
## [1] 27575.71
## [1] 18177.37
## [1] 60256.86
## [1] 22108.65
## [1] 33908.94
## [1] 37164.56
## [1] 26198.9
## [1] 39573.24
## [1] 21150.93
## [1] 45862.94
## [1] 23366.24
## [1] 16165.69
## [1] 10263.47
## [1] 17895.16
## [1] 24026.93
## [1] 33497.65
## [1] 35268.94
## [1] 15770.11
## [1] 48190.12
## [1] 33107.4
## [1] 20523.44
## [1] 21387.73
## [1] 15771.71
## [1] 12632.94
## [1] 28352.2
## [1] 10401.65
## [1] 41162.37
## [1] 16740.47
## [1] 29576.59
## [1] 28831.91
## [1] 21622.91
## [1] 50282.12
## [1] 26736.71
## [1] 18663.88
## [1] 10872.69
## [1] 13072.22
## [1] 35308.68
## [1] 17145.7
## [1] 19620.53
## [1] 1550.37
## [1] 74993.8
## [1] 11509.2
## [1] 16574.36
## [1] 94984.15
## [1] 9448.048
## [1] 56370.43
## [1] 32075.71
## [1] 47899.08
## [1] 27521.46
## [1] 24907.23
```

```
## [1] 12800.02
## [1] 34456.9
## [1] 39803.11
## [1] 19137.79
## [1] 9084.302
## [1] 61601.96
## [1] 20396.02
## [1] 7636.822
## [1] 15452.48
## [1] 53609.16
## [1] 11482.58
## [1] 75963.09
## [1] 21323.04
## [1] 17062.97
## [1] 24482.02
## [1] 19394.53
## [1] 61929.26
## [1] 54626.89
## [1] 29113.2
## [1] 53044.43
## [1] 17891.22
## [1] 21665.73
## [1] 21611.86
## [1] 13917.62
## [1] 21715
## [1] 44285
## [1] 10525.6
## [1] 31777.55
## [1] 45932.5
## [1] 16396.8
## [1] 78952.15
## [1] 21020.83
## [1] 9499.589
## [1] 39867.94
## [1] 11886.27
## [1] 13597.17
## [1] 110345.2
## [1] 32610.06
## [1] 50496.5
## [1] 23180.86
## [1] 20838.97
## [1] 27426.14
## [1] 51655.5
## [1] 52241.02
## [1] 27527.98
## [1] 40947.43
## [1] 26691.61
## [1] 23152.57
## [1] 43419.74
## [1] 44236.59
## [1] 60396.6
## [1] 15878.96
## [1] 70561.7
## [1] 17374.23
```

```
## [1] 10332.36
## [1] 34844.88
## [1] 59170.5
## [1] 43839.49
## [1] 28327.57
## [1] 10259.93
## [1] 24344.12
## [1] 80738.44
## [1] 23490.64
## [1] 15151.29
## [1] 40052.67
## [1] 31011.45
## [1] 22601.61
## [1] 36300.59
## [1] 28716.67
## [1] 21434.73
## [1] 35806.67
## [1] 27977.29
## [1] 13912.49
## [1] 50491.72
## [1] 69180.55
## [1] 45387.39
## [1] 21638.87
## [1] 12782.32
## [1] 52271.29
## [1] 41075.97
## [1] 19955.44
## [1] 74279.38
## [1] 19250.19
## [1] 19647.87
## [1] 39022.27
## [1] 53729.75
## [1] 35271.74
## [1] 9446.876
## [1] 33097.29
## [1] 70044.17
## [1] 23694.39
## [1] 15501.03
## [1] 13490.36
## [1] 7311.07
## [1] 63156.4
## [1] 40543.55
## [1] 19942.98
## [1] 30055.31
## [1] 26008.03
## [1] 26888.99
## [1] 62185.58
## [1] 18102.81
## [1] 125939.1
## [1] 17460.23
## [1] 31886.39
## [1] 14393.86
## [1] 83699.9
## [1] 62045.51
```

```
## [1] 60194.05
## [1] 36753.96
## [1] 80988.92
## [1] 39064.25
## [1] 32061.54
## [1] 51247.47
## [1] 67466.67
## [1] 17627.75
## [1] 24171.68
## [1] 25917.75
## [1] 67098.9
## [1] 57633.18
## [1] 17699.3
## [1] 18903.75
## [1] 13127.74
## [1] 17295.45
## [1] 42209.93
## [1] 23426.67
## [1] 118938
## [1] 50685.79
## [1] 18165.83
## [1] 41005.54
## [1] 46816.66
## [1] 31808.09
## [1] 53237.91
## [1] 23121.37
## [1] 25937.75
## [1] 37975.73
## [1] 47637.07
## [1] 42096.21
## [1] 127540.6
## [1] 94693.88
## [1] 12313.1
## [1] 24276.52
## [1] 15500.68
## [1] 16109.79
## [1] 15965.47
## [1] 54296.49
## [1] 44227.75
## [1] 47230.76
## [1] 14365.98
## [1] 153749.9
## [1] 59143.02
## [1] 18524.3
## [1] 6227.675
## [1] 13606.98
## [1] 47194.01
## [1] 19408.71
## [1] 49147
## [1] 103896.5
## [1] 38059.73
## [1] 41076.72
## [1] 46707
## [1] 30013.15
```

```
## [1] 41805.51
## [1] 20113.28
## [1] 24071.44
## [1] 34038.28
## [1] 39871.05
## [1] 8489.727
## [1] 24349.18
## [1] 47621.35
## [1] 33751.53
## [1] 44921.37
## [1] 26262.99
## [1] 16883.38
## [1] 14444.69
## [1] 13358.07
```

Add the results in the vector back to the original data frame. Show the first few rows of the data frame using head().

```
dino_data$masses <- my_results
head(dino_data)
```

```
##        species  lengths   masses
## 1  Stegosauria 18.52588 24341.68
## 2 Ankylosauria 16.43598 27017.90
## 3 Ankylosauria 23.73421 67453.38
## 4    Sauropoda 23.93411 22114.19
## 5 Ankylosauria 21.68718 53884.76
## 6 Ankylosauria 21.38363 52026.34
```

Calculate the mean mass for each species using dplyr.

```
library (dplyr)
dino_data %>%
  group_by(species) %>%
  summarise(mean_mass = mean(masses))
```

```
## # A tibble: 4 x 2
##   species      mean_mass
##   <chr>            <dbl>
## 1 Ankylosauria    46819.
## 2 Sauropoda       16104.
## 3 Stegosauria     31924.
## 4 Theropoda       45572.
```

Multi-File Analysis

If individual_collar_data.zip is not already in your working directory download the zip file using download.file()

```
download.file(url = "http://www.datacarpentry.org/semester-biology/data/individual_collar_data.zip", de
```

Unzip it using unzip()

```r
unzip("../data-raw/individual_collar_data.zip",exdir = "../data-raw/")
```

Obtain a list of all of the files with file names matching the pattern "collar-data-.*.txt" (using list.files())

```r
list.files(path = "../data-raw", pattern = "collar-data-.*.txt")
```

```
##  [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"
```

Use a loop to load each of these files into R and make a line plot (using geom_path()) for each file with long on the x axis and lat on the y axis. Graphs, like other types of output, won't display inside a loop unless you explicitly display them, so you need put your ggplot() command inside a print() statement. Include the name of the file in the graph as the graph title using labs().

Add code to the loop to calculate the minimum and maximum latitude in the file, and store these values, along with the name of the file, in a data frame. Show the data frame as output.

```r
for (list in "collar-data"){
  ggplot(mapping = aes(x =long, y= lat)) +
    geom_point(size = 5)
}
```