apply functions

Luna L Sanchez Reyes

2023-04-04

```
?lapply
?mapply
```

The `apply` functions allow us us to apply a function to a vector or list of values iteratively. This helps minimize errors in code and makes the analyses more efficient.

With `lapply()` and `sapply()` functions, we can only provide one argument to iterate on.

`sapply()` function simplifies the output to a vector (or the simplest data structure possible), while `lapply()` returns an output in the form of a list.

With `mapply()`, we can provide multiple arguments to iterate on. Be cause it is a multivariate version of `sapply()`, it probably also retruns a vector or simplified data structure as result.

Exercise 1

Write a function named `mass_from_length_theropoda()` that takes length as an argument to get an estimate of mass for Theropoda dinosaurs. Use the equation mass <- 0.73 * length^3.63. Copy and run the code below to generate the object theropoda_lengths in your R environment.

```
mass_from_length_theropoda <- function(length = 1) {
  mass <- 0.73 * length^3.63
  return(mass)
}
mass_from_length_theropoda()
```

```
## [1] 0.73
```

```
theropoda_lengths <- c(17.8013631070471, 20.3764452071665, 14.0743486294308, 25.65782386974, 26.0952008
```

Pass the entire vector to your function (by giving it as value for the length argument); this calculates the mass for each length value in the vector theropoda_lengths.

```
mass_from_length_theropoda(length = theropoda_lengths)
```

```
##  [1]  25262.027  41253.332  10767.568  95233.732 101260.017  40775.516
##  [7]  24072.130   4785.145  39129.521  29666.193  26830.297  64700.869
## [13]  42768.180  94697.262  79013.471 103955.226  92798.465  41901.983
## [19]  17439.569  41055.045  37544.201  25198.303  12928.490  36388.290
## [25]  34962.862  80307.929   8854.525  50183.194  28846.165  35735.369
## [31] 115908.187  31765.368  58958.713   5561.862  28349.410  15418.314
## [37]   9218.648   1197.666  94407.873  19552.500
```

What is the output? It is a vector of masses calculated from theropda lengths using the function we created above (called `mass_from_length_theropoda()`).

```r
theropoda_masses <- mass_from_length_theropoda(length = theropoda_lengths)

my_list <- list(theropoda_masses)
second_list <- c(my_list, list(c("Luna", "Avi", "Anita")))

second_list[[1]]
```

```
##  [1]  25262.027  41253.332  10767.568  95233.732 101260.017  40775.516
##  [7]  24072.130   4785.145  39129.521  29666.193  26830.297  64700.869
## [13]  42768.180  94697.262  79013.471 103955.226  92798.465  41901.983
## [19]  17439.569  41055.045  37544.201  25198.303  12928.490  36388.290
## [25]  34962.862  80307.929   8854.525  50183.194  28846.165  35735.369
## [31] 115908.187  31765.368  58958.713   5561.862  28349.410  15418.314
## [37]   9218.648   1197.666  94407.873  19552.500
```

```r
data.frame(theropoda_masses, c("Anita", "Avi", "Luna", "Maria"))
```

```
##    theropoda_masses c..Anita....Avi....Luna....Maria..
## 1         25262.027                              Anita
## 2         41253.332                                Avi
## 3         10767.568                               Luna
## 4         95233.732                              Maria
## 5        101260.017                              Anita
## 6         40775.516                                Avi
## 7         24072.130                               Luna
## 8          4785.145                              Maria
## 9         39129.521                              Anita
## 10        29666.193                                Avi
## 11        26830.297                               Luna
## 12        64700.869                              Maria
## 13        42768.180                              Anita
## 14        94697.262                                Avi
## 15        79013.471                               Luna
## 16       103955.226                              Maria
## 17        92798.465                              Anita
## 18        41901.983                                Avi
## 19        17439.569                               Luna
## 20        41055.045                              Maria
## 21        37544.201                              Anita
## 22        25198.303                                Avi
## 23        12928.490                               Luna
## 24        36388.290                              Maria
## 25        34962.862                              Anita
## 26        80307.929                                Avi
## 27         8854.525                               Luna
## 28        50183.194                              Maria
## 29        28846.165                              Anita
## 30        35735.369                                Avi
## 31       115908.187                               Luna
## 32        31765.368                              Maria
## 33        58958.713                              Anita
## 34         5561.862                                Avi
## 35        28349.410                               Luna
## 36        15418.314                              Maria
## 37        9218.648                            Anita
## 38        1197.666                              Avi
## 39       94407.873                             Luna
## 40       19552.500                            Maria
```

Create a new version of the function named `mass_from_length()` that uses the equation `mass <- a * length^b` and takes length, a and b as arguments. In the function arguments, set the default values for a to 0.73 and b to 3.63. If you run this function with just the length data from Part 1, you should get the same result as Part 1.

```
mass_from_length <- function(length, a = 0.73, b = 3.63){
  mass <- a * length^b
  return(mass)
}
new_masses <- mass_from_length(length = theropoda_lengths)
# rm(new_lengths) # The rm function allows to remove objects from the R environment

theropoda_masses == new_masses
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
all(theropoda_masses == new_masses) # tests that all values in a logical vector are equal tot TRUE
```

```
## [1] TRUE
```

```
all.equal(theropoda_masses, new_masses)
```

```
## [1] TRUE
```

Copy the data below into R and call your function using the vector of lengths from Part 1 (above) and these vectors of a and b values to estimate the mass for the dinosaurs using different values of a and b.

```
a_values <- c(0.759, 0.751, 0.74, 0.746, 0.759, 0.751, 0.749, 0.751, 0.738, 0.768, 0.736, 0.749, 0.746,

b_values <- c(3.627, 3.633, 3.626, 3.633, 3.627, 3.629, 3.632, 3.628, 3.633, 3.627, 3.621, 3.63, 3.631,
```

```
mass_from_length(length = theropoda_lengths, a = a_values, b = b_values)
```

```
##  [1]  26039.686  42825.603  10800.224  98273.049 104257.481  41822.386
##  [7]  24840.644   4899.022  39915.948  30937.922  26354.908  66384.865
## [13]  43837.944  97141.451  80553.856 105556.405  97374.660  42760.136
## [19]  18749.274  42109.012  40674.182  26003.425  13229.824  37472.789
## [25]  34684.033  80187.272   9460.977  51630.571  29253.772  36399.306
## [31] 117511.962  33384.288  58581.226   5462.316  28637.745  15864.172
## [37]   9284.810   1218.755  98522.609  19534.524
```

Basic functions can "naturally" iterate through values in a vector to perform calculations.

3. Create a data frame for this data using the code `dino_data <- data.frame(theropoda_lengths, a_values, b_values)`. Use dplyr to add a new masses column to this data frame (using mutate() and your function) and print the result to the console.

```
dino_data <- data.frame(theropoda_lengths, a_values, b_values)
head(dino_data)
```

```
##   theropoda_lengths a_values b_values
## 1          17.80136    0.759    3.627
## 2          20.37645    0.751    3.633
## 3          14.07435    0.740    3.626
## 4          25.65782    0.746    3.633
## 5          26.09520    0.759    3.627
## 6          20.31115    0.751    3.629
```

`<<-` the scope operator or double arrow, allows creating and modifying objects in parent variables.

Exercise 2

1. Create a new version of your mass_from_length_theropoda() function from Part 1 of Exercise 1 called mass_from_length_max(). This function should only calculate a mass if the value of length passed to the function is less than 20.

```
theropoda_lengths < 20
```

```
##  [1]  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [25]  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
## [37]  TRUE  TRUE FALSE  TRUE
```

```
mass_from_length_max <- function(length) {
  if (length < 20) {
    mass <- 0.73 * length^3.63
    return(mass)
  } else {
    return(NA)
  }
}

# mass_from_length_max(length) {
#   if (length < 20) {
#     mass <- 0.73 * length^3.63
#   } else {
#     mass <- NA
#   }
#   return(mass)
# }

# mass_from_length_max(length) {
#   if (length < 20) {
#     mass <- 0.73 * length^3.63
#     return(mass)
#   }
# }
```

2. If length is greater than 20, return NA instead. Use `sapply()` and this new function to estimate the mass for the theropoda_lengths data from Exercise 1.

```
mass_from_length_max(length = theropoda_lengths)
```

```
## Error in if (length < 20) {: the condition has length > 1
```

```
sapply(theropoda_lengths, mass_from_length_max)
```

```
##  [1] 25262.027        NA 10767.568        NA        NA        NA 24072.130
##  [8]  4785.145        NA 29666.193 26830.297        NA        NA        NA
## [15]        NA        NA        NA        NA 17439.569        NA 37544.201
## [22] 25198.303 12928.490 36388.290 34962.862        NA  8854.525        NA
## [29] 28846.165 35735.369        NA 31765.368        NA  5561.862 28349.410
## [36] 15418.314  9218.648  1197.666        NA 19552.500
```

Exercise 3

1. Download the CSV file of data on dinosaur lengths with species names into your data folder and import it using read.csv().
2. Write a function get_mass_from_length_by_name() that uses the equation mass <- a * length^b to estimate the size of a dinosaur from its length. This function should take two arguments, the length and the name of the dinosaur group. Inside this function use if/else if/else statements to check to see if the name is one of the following values and if so set a and b to the appropriate values from Seebacher 2001. Stegosauria: a = 10.95 and b = 2.64 Theropoda: a = 0.73 and b = 3.63 Sauropoda: a = 214.44 and b = 1.46 If the name is not any of these values set a = NA and b = NA.
3. Use this function and `mapply()` to calculate the estimated mass for each dinosaur. You'll need to pass the data to `mapply()` as single vectors or columns, not the whole data frame.
4. Using dplyr, add a new masses column to the data frame (using rowwise(), mutate() and your function) and print the result to the console.
5. Using ggplot2, make a histogram of dinosaur masses with one subplot for each species (remember facet_wrap()).