apply.Rmd

Damian Sanchez

4/4/2023

```
?apply
?mapply
```

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

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

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

With `mapply()`, we can provide multiple arguments.

Installing Library

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Exercise 1

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

```r
theropoda_lengths <- c(17.8013631070471, 20.3764452071665, 14.0743486294308, 25.65782386974, 26.0952008
theropod_masses <- mass_from_length_theropoda(length = theropoda_lengths)

my_list <- list(theropod_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(theropod_masses, c("Anita", "Avi", "Luna", "Maria"))
```

```
##    theropod_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
```

```
mass_from_length <- function(length, a = 0.73, b = 3.63) {
  mass <- a * length^b
  return(mass)
}
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,

new_masses <- mass_from_length(length = theropoda_lengths)
# rm(new_lengths) The rm function allows us to remove objects from the R environment.

theropod_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(theropod_masses = new_masses) # tests that all values in the logical vector are equal to TRUE
```

```
## Warning in all(theropod_masses = new_masses): coercing argument of type 'double'
## to logical
```

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

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

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

```
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.

```r
dino_data <- data.frame(theropoda_lengths, a_values, b_values)
dino_data <- dino_data %>%
  mutate(masses = mass_from_length(theropoda_lengths, a = a_values, b = b_values))

print(dino_data)
```

```
##    theropoda_lengths a_values b_values     masses
## 1          17.801363    0.759    3.627  26039.686
## 2          20.376445    0.751    3.633  42825.603
## 3          14.074349    0.740    3.626  10800.224
## 4          25.657824    0.746    3.633  98273.049
## 5          26.095201    0.759    3.627 104257.481
## 6          20.311154    0.751    3.629  41822.386
## 7          17.566324    0.749    3.632  24840.644
## 8          11.256343    0.751    3.628   4899.022
## 9          20.081903    0.738    3.633  39915.948
## 10         18.607163    0.768    3.627  30937.922
## 11         18.099189    0.736    3.621  26354.908
## 12         23.065969    0.749    3.630  66384.865
## 13         20.579885    0.746    3.631  43837.944
## 14         25.617925    0.744    3.632  97141.451
## 15         24.371433    0.749    3.628  80553.856
## 16         26.284725    0.751    3.626 105556.405
## 17         25.475378    0.744    3.639  97374.660
## 18         20.464209    0.754    3.626  42760.136
## 19         16.073826    0.774    3.635  18749.274
## 20         20.349417    0.751    3.629  42109.012
## 21         19.854399    0.763    3.642  40674.182
## 22         17.788981    0.749    3.632  26003.425
## 23         14.801642    0.741    3.633  13229.824
## 24         19.684091    0.754    3.629  37472.789
## 25         19.468589    0.746    3.620  34684.033
## 26         24.480778    0.755    3.619  80187.272
## 27         13.335996    0.764    3.638   9460.977
## 28         21.506599    0.758    3.627  51630.571
## 29         18.464030    0.760    3.621  29253.772
## 30         19.586153    0.748    3.628  36399.306
## 31         27.084752    0.745    3.628 117511.962
## 32         18.960937    0.756    3.635  33384.288
## 33         22.482917    0.739    3.624  58581.226
## 34         11.732572    0.733    3.621   5462.316
## 35         18.375885    0.757    3.621  28637.745
## 36         15.537505    0.747    3.632  15864.172
## 37         13.484875    0.741    3.627   9284.810
## 38          7.685612    0.752    3.624   1218.755
## 39         25.596335    0.752    3.634  98522.609
## 40         16.588285    0.748    3.621  19534.524
```

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

Exercise 2

```
mass_from_length_max <- function(length) {
  if (length < 20) {
    mass <- 0.73 * length^3.63
    return(mass)
  } else {
    return(NA)
  }
}
masses_max <- sapply(theropoda_lengths, mass_from_length_max)
print(masses_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

```
dino <- read.csv(file = "../Downloads/dinosaur_lengths.csv")
get_mass_from_length_by_name <- function(lengths, species) {
  if (species == "Stegosauria") {
    a <- 10.95
    b <- 2.64
  } else if (species == "Theropoda") {
    a <- 0.73
    b <- 3.63
  } else if (species == "Sauropoda") {
    a <- 214.44
    b <- 1.46
  } else {
    a <- NA
    b <- NA
  }
  mass <- a*lengths^b
  return(mass)
}

masses <- mapply(get_mass_from_length_by_name, lengths = dino$lengths, species = dino$species)
dino %>%
  rowwise() %>%
  mutate(masses = get_mass_from_length_by_name(lengths, species)) %>%
  head(10)
```

```
## # A tibble: 10 x 3
## # Rowwise:
##    species      lengths masses
```

```
##    <chr>          <dbl>  <dbl>
##  1 Stegosauria     18.5 24342.
##  2 Ankylosauria    16.4    NA
##  3 Ankylosauria    23.7    NA
##  4 Sauropoda       23.9 22114.
##  5 Ankylosauria    21.7    NA
##  6 Ankylosauria    21.4    NA
##  7 Theropoda       22.3 57349.
##  8 Theropoda       15.2 14160.
##  9 Theropoda       21.4 49678.
## 10 Stegosauria     22.8 42106.
```

```
library(ggplot2)
dino %>%
ggplot() +
  geom_histogram(mapping = aes(x=masses, color= species)) +
  facet_wrap(~species)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 138 rows containing non-finite values (`stat_bin()`).
```