Apply functions

Luna L Sanchez Reyes

2022-11-15

Exercise 1: Calculating the mass of a bunch of dinosaurs

Part 1a) 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.

```
mass_from_length_theropoda <- function(length) {
  mass <- 0.73 * length ^ 3.63
  return(mass)
}
mass_from_length_theropoda(100)
```

```
## [1] 13283816
```

Part 2b) Copy and run the code below to generate the object theropoda_lengths in your R environment. 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`.

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

Calculate the mass for each length values in the vector `theropoda_lengths`

```
c(1, 1.5, 3.7, 85) * 10
```

```
## [1]  10  15  37 850
```

```
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
```

```
sapply(X = theropoda_lengths, FUN = mass_from_length_theropoda)
```

```
##  [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
```

Part 2a) 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)
}
```

Part 2b) 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.

```
library(magrittr)
mass_from_length(length = theropoda_lengths) == mass_from_length_theropoda(length = theropoda_lengths)
```

```
##  [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
```

```
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] 2.603969e+04 4.282560e+04 1.080022e+04 9.827305e+04 1.042575e+05
##  [6] 4.182239e+04 2.484064e+04 4.899022e+03 3.991595e+04 3.093792e+04
## [11] 2.635491e+04 6.638486e+04 4.383794e+04 9.714145e+04 8.055386e+04
## [16] 1.055564e+05 9.737466e+04 4.276014e+04 1.874927e+04 4.210901e+04
## [21] 4.067418e+04 2.600342e+04 1.322982e+04 3.747279e+04 3.468403e+04
## [26] 8.018727e+04 9.460977e+03 5.163057e+04 2.925377e+04 3.639931e+04
## [31] 1.175120e+05 3.338429e+04 5.858123e+04 5.462316e+03 2.863774e+04
## [36] 1.586417e+04 9.284810e+03 1.218755e+03 9.852261e+04 1.180094e+12
```

```
a_values * theropoda_lengths ^ b_values
```

```
##  [1] 2.603969e+04 4.282560e+04 1.080022e+04 9.827305e+04 1.042575e+05
##  [6] 4.182239e+04 2.484064e+04 4.899022e+03 3.991595e+04 3.093792e+04
## [11] 2.635491e+04 6.638486e+04 4.383794e+04 9.714145e+04 8.055386e+04
## [16] 1.055564e+05 9.737466e+04 4.276014e+04 1.874927e+04 4.210901e+04
## [21] 4.067418e+04 2.600342e+04 1.322982e+04 3.747279e+04 3.468403e+04
## [26] 8.018727e+04 9.460977e+03 5.163057e+04 2.925377e+04 3.639931e+04
## [31] 1.175120e+05 3.338429e+04 5.858123e+04 5.462316e+03 2.863774e+04
## [36] 1.586417e+04 9.284810e+03 1.218755e+03 9.852261e+04 1.180094e+12
```

```
# investigate this, it shouldn't work?
mapply(mass_from_length, theropoda_lengths, a_values, b_values)
```

```
##  [1] 2.603969e+04 4.282560e+04 1.080022e+04 9.827305e+04 1.042575e+05
##  [6] 4.182239e+04 2.484064e+04 4.899022e+03 3.991595e+04 3.093792e+04
## [11] 2.635491e+04 6.638486e+04 4.383794e+04 9.714145e+04 8.055386e+04
## [16] 1.055564e+05 9.737466e+04 4.276014e+04 1.874927e+04 4.210901e+04
## [21] 4.067418e+04 2.600342e+04 1.322982e+04 3.747279e+04 3.468403e+04
## [26] 8.018727e+04 9.460977e+03 5.163057e+04 2.925377e+04 3.639931e+04
## [31] 1.175120e+05 3.338429e+04 5.858123e+04 5.462316e+03 2.863774e+04
## [36] 1.586417e+04 9.284810e+03 1.218755e+03 9.852261e+04 1.180094e+12
```

Part 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.

```r
dino_data <- data.frame(lengths = theropoda_lengths, as = a_values, bs = b_values)

dino_data %>%
  mutate(masses = mass_from_length(length = lengths, a = as, b = bs))
```

```
##      lengths    as     bs       masses
## 1  17.801363 0.759  3.627 2.603969e+04
## 2  20.376445 0.751  3.633 4.282560e+04
## 3  14.074349 0.740  3.626 1.080022e+04
## 4  25.657824 0.746  3.633 9.827305e+04
## 5  26.095201 0.759  3.627 1.042575e+05
## 6  20.311154 0.751  3.629 4.182239e+04
## 7  17.566324 0.749  3.632 2.484064e+04
## 8  11.256343 0.751  3.628 4.899022e+03
## 9  20.081903 0.738  3.633 3.991595e+04
## 10 18.607163 0.768  3.627 3.093792e+04
## 11 18.099189 0.736  3.621 2.635491e+04
## 12 23.065969 0.749  3.630 6.638486e+04
## 13 20.579885 0.746  3.631 4.383794e+04
## 14 25.617925 0.744  3.632 9.714145e+04
## 15 24.371433 0.749  3.628 8.055386e+04
## 16 26.284725 0.751  3.626 1.055564e+05
## 17 25.475378 0.744  3.639 9.737466e+04
## 18 20.464209 0.754  3.626 4.276014e+04
## 19 16.073826 0.774  3.635 1.874927e+04
## 20 20.349417 0.751  3.629 4.210901e+04
## 21 19.854399 0.763  3.642 4.067418e+04
## 22 17.788981 0.749  3.632 2.600342e+04
## 23 14.801642 0.741  3.633 1.322982e+04
## 24 19.684091 0.754  3.629 3.747279e+04
## 25 19.468589 0.746  3.620 3.468403e+04
## 26 24.480778 0.755  3.619 8.018727e+04
## 27 13.335996 0.764  3.638 9.460977e+03
## 28 21.506599 0.758  3.627 5.163057e+04
## 29 18.464030 0.760  3.621 2.925377e+04
## 30 19.586153 0.748  3.628 3.639931e+04
## 31 27.084752 0.745  3.628 1.175120e+05
## 32 18.960937 0.756  3.635 3.338429e+04
## 33 22.482917 0.739  3.624 5.858123e+04
## 34 11.732572 0.733  3.621 5.462316e+03
## 35 18.375885 0.757  3.621 2.863774e+04
## 36 15.537505 0.747  3.632 1.586417e+04
## 37 13.484875 0.741  3.627 9.284810e+03
## 38  7.685612 0.752  3.624 1.218755e+03
## 39 25.596335 0.752  3.634 9.852261e+04
## 40 16.588285 0.748 10.000 1.180094e+12
```

Exercise 2

Part 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. If length is greater than 20, return `NA` instead.

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

Part 2. Use sapply() and this new function to estimate the mass for the `theropoda_lengths` data from Exercise 1.

```
sapply(X = theropoda_lengths, FUN = 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:

Part 1

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

```
##        species  lengths
## 1  Stegosauria 18.52588
## 2 Ankylosauria 16.43598
## 3 Ankylosauria 23.73421
## 4    Sauropoda 23.93411
## 5 Ankylosauria 21.68718
## 6 Ankylosauria 21.38363
```

Part 2.

```
get_mass_from_length_by_name <- function(length, name) {
  if (name == "Stegosauria") {
    a <- 10.95
    b <- 2.64
  } else if (name == "Theropoda") {
    a <- 0.73
    b <- 3.63
  } else if (name == "Sauropoda") {
    a <- 214.44
    b <- 1.46
  } else {
    a <- NA
    b <- NA
  }

  mass <- a * length ^ b
  return(mass)

}
```

Part 3. Use `get_mass_from_length_by_name()` 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.

```
class(dino_lengths)
```

```
## [1] "data.frame"
```

```
colnames(dino_lengths)
```

```
## [1] "species" "lengths"
```

```
class(dino_lengths$species)
```

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

```
is.vector(dino_lengths$species)
```

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

```
is.vector(dino_lengths)
```

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

```
mapply(FUN = get_mass_from_length_by_name, length = dino_lengths$lengths, name = dino_lengths$species)
```

```
##   [1]  24341.681         NA         NA  22114.190         NA         NA
##   [7]  57349.470  14160.494  49677.749  42105.917  10221.747  15339.988
##  [13]  70624.102  23883.825  28552.864  18801.370  19438.673         NA
##  [19]  19607.970  16032.845         NA  50350.112  15969.078  29582.848
##  [25]  15201.456  12980.541   9937.867   9599.415  49245.963  23846.751
##  [31]  53805.661  53326.467         NA  15554.977  18544.119         NA
##  [37]         NA  82492.318  17909.041  38694.503  80303.181  19592.802
##  [43]  10614.785  29560.809  71658.477         NA  83961.661         NA
##  [49]  26284.040  21766.002  63571.873   5480.255  33917.314  22778.032
##  [55]  13819.165  21154.149  17635.099  14577.594         NA  14032.340
##  [61]  30231.694         NA  11293.886  72743.800  23679.901  64258.574
##  [67]  14931.085  16323.818         NA         NA         NA   7599.703
##  [73]         NA         NA         NA         NA  46920.035  70529.031
##  [79]   9484.528         NA  68340.494  44959.626         NA  48249.486
##  [85]  11730.174         NA  52295.177         NA         NA         NA
##  [91]  40358.292  38891.137  30878.439  19125.425         NA         NA
##  [97]   8697.216  19627.357         NA         NA  13411.390  33157.499
## [103]  10874.733  24554.930  16819.494  18421.449         NA  19645.723
## [109]  38206.241  53196.019  22346.109         NA  22685.103         NA
## [115]  13613.983  34685.790         NA  18654.525         NA 101482.428
## [121]  89149.257         NA  20820.837         NA  22232.852  59702.598
## [127]         NA  16321.774  22748.880         NA         NA         NA
## [133]         NA  25987.768  49818.253  13106.766         NA  32112.443
## [139]         NA  16984.463  10859.926  93973.020  52342.265  19151.788
## [145]         NA  13954.186         NA  15021.820  35933.327 140435.607
## [151]  20467.332  23869.639         NA         NA  15211.979  57098.945
## [157]  23588.700  27381.008  85932.513         NA   9331.295         NA
## [163]         NA  32005.502  16613.444   7904.857         NA  26352.263
## [169]  19880.480  15543.679  15493.654  13546.034         NA  36095.081
## [175]  42437.608         NA         NA  51637.913         NA  44120.181
## [181]   9535.583  59840.348         NA         NA         NA  44822.176
## [187]  14232.684  34751.496  11292.437         NA         NA         NA
## [193]  22002.082  19554.166  13223.770         NA         NA  68935.505
```

```
## [199]   9172.206  90096.476  25796.762  50594.426  61952.966  20132.528
## [205]         NA  13979.439  15481.074  12104.000  21789.436  54009.090
## [211]  13812.364   8071.939  21144.506  44097.848  16250.303  70065.996
## [217]  11170.349  22826.560  40885.088  17292.043  18394.391  50267.629
## [223]  70791.032  28464.276  41431.346         NA  14242.918         NA
## [229]         NA  52014.366  32865.058         NA  11906.150  17964.362
## [235]  14844.497  13079.836  76048.107  18843.875         NA  30737.511
## [241]  37983.026  18711.957  22636.970  29868.755  42799.606         NA
## [247]  43632.463 103600.943         NA         NA  10330.761  23659.805
## [253]  19126.024  17175.845  28017.230  54437.041         NA  20657.057
## [259]  13275.051         NA   8222.362         NA 108964.075         NA
## [265]   5845.741  26356.588         NA  59636.239  14857.582  45043.701
## [271]  47427.024         NA         NA  11807.182  27575.709  18177.367
## [277]         NA  22108.648  33908.940         NA         NA         NA
## [283]         NA  45862.941  23366.240  16165.694  10263.470         NA
## [289]  24026.928  33497.651         NA  15770.110  48190.121  33107.401
## [295]  20523.437  21387.730  15771.706  12632.938  28352.199  10401.651
## [301]  41162.369  16740.472  29576.590  28831.907  21622.906         NA
## [307]  26736.709  18663.882  10872.689  13072.222  35308.681  17145.703
## [313]  19620.530   1550.370         NA  11509.202  16574.358  94984.150
## [319]   9448.048  56370.430         NA  47899.078  27521.456  24907.229
## [325]  12800.024  34456.895         NA  19137.794   9084.302         NA
## [331]  20396.019   7636.822  15452.482         NA  11482.576         NA
## [337]  21323.042  17062.973  24482.018  19394.529  61929.256         NA
## [343]  29113.203  53044.431  17891.216  21665.733  21611.857  13917.623
## [349]  21715.000         NA  10525.601  31777.548  45932.499  16396.801
## [355]         NA  21020.829   9499.589         NA  11886.269  13597.168
## [361]         NA  32610.060  50496.496  23180.857  20838.975  27426.143
## [367]  51655.501  52241.022  27527.983  40947.425  26691.614  23152.573
## [373]  43419.737  44236.593  60396.602  15878.961  70561.697  17374.235
## [379]  10332.362  34844.884         NA  43839.492         NA  10259.928
## [385]  24344.124         NA  23490.643  15151.289  40052.674  31011.453
## [391]         NA  36300.595  28716.671  21434.730         NA  27977.292
## [397]  13912.492         NA         NA  45387.391  21638.866  12782.316
## [403]         NA         NA         NA  74279.377  19250.194  19647.872
## [409]  39022.265         NA         NA   9446.876  33097.292         NA
## [415]  23694.389  15501.027  13490.363   7311.070  63156.403  40543.550
## [421]  19942.976         NA         NA  26888.995         NA  18102.809
## [427] 125939.133         NA         NA  14393.863         NA  62045.506
## [433]  60194.052  36753.957         NA         NA  32061.537         NA
## [439]  67466.670  17627.746  24171.682  25917.752  67098.902         NA
## [445]  17699.295  18903.752  13127.745  17295.450  42209.926  23426.667
## [451] 118937.988         NA  18165.832         NA  46816.660         NA
## [457]  53237.908  23121.375  25937.746         NA  47637.068         NA
## [463] 127540.554         NA  12313.099  24276.516  15500.675  16109.794
## [469]  15965.471  54296.492         NA         NA  14365.977 153749.934
## [475]  59143.016  18524.301   6227.675  13606.978         NA         NA
## [481]  49146.996 103896.484  38059.728  41076.716         NA  30013.153
## [487]  41805.513  20113.277  24071.440         NA         NA   8489.727
## [493]  24349.181         NA         NA  44921.367  26262.993  16883.382
## [499]  14444.693         NA
```

If I try to feed the whole data frame as input:

```
mapply(FUN = get_mass_from_length_by_name, length = dino_lengths, name = dino_lengths)

Error in if (name == "Stegosauria") { : the condition has length > 1
```

Part 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.

Part 5. Using `ggplot2`, make a histogram of dinosaur masses with one subplot for each species (remember `facet_wrap()`).