Body mass of mammals — Smith et al. 2003

Smith et al. (2003) compiled a database of the body mass of mammals of the late Quaternary. Your goal is to calculate the average body mass of the species in each Family.

1. Write a script (`read_mass.R`) that reads and cleans the data.

First of all, we need to read the data into a `data.frame`. There is no header, the columns are separated by Tabs, and we want to include `stringsAsFactors = FALSE` to make sure species names are not turned into factors (categorical values, very useful for linear regressions, but not for our purposes).

```
s2003 <- read.table("../data/Smith2003_data.txt", sep = "\t",
                    stringsAsFactors = FALSE)
# Check the size
dim(s2003)
```

```
## [1] 5731    9
```

```
# Show the first few lines
head(s2003)
```

```
##   V1     V2           V3      V4         V5            V6   V7       V8
## 1 AF extant Artiodactyla Bovidae      Addax nasomaculatus 4.85  70000.3
## 2 AF extant Artiodactyla Bovidae  Aepyceros      melampus 4.72  52500.1
## 3 AF extant Artiodactyla Bovidae Alcelaphus    buselaphus 5.23 171001.5
## 4 AF extant Artiodactyla Bovidae Ammodorcas       clarkei 4.45  28049.8
## 5 AF extant Artiodactyla Bovidae Ammotragus        lervia 4.68  48000.0
## 6 AF extant Artiodactyla Bovidae Antidorcas   marsupialis 4.59  39049.9
##       V9
## 1     60
## 2 63, 70
## 3 63, 70
## 4     60
## 5     75
## 6     60
```

Now we want to put `NA` instead of `-999` for signaling missing data (always use `NA`, as R can then understand these are special values).

```
s2003[s2003 == -999] <- NA
# see that R treats NAs separately
summary(s2003[, 7])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.260   1.360   2.070   2.484   3.410   8.280    1372
```

Finally, let's chose some descriptive headers to make it simpler to reference to specific columns.

```r
colnames(s2003) <- c("Continent", "Status",
                     "Order", "Family", "Genus",
                     "Species", "LogMass",
                     "CombinedMass", "Reference")
```

See the file `read_mass.R` for the complete script.

2. Write another script (`body_mass_family.R`), which calls `read_mass.R`, and then calculates the average body mass per Family.

There are many different strategies to accomplish this. Here we explore a possible solution, and in the next section a different one.

Each species belongs to one family; if we were to sum all the body masses for the species belonging to a certain family, and then divide by the number of species in that family, we would obtain the desired result. To implement this, we could:

- cycle through the data
- if data on family and body mass of the species are present:
- add the family in case it hasn't been found before
- update the total body mass for the family
- update the number of species for the family

In practice, one would write code such as:

```r
# Empty dataframe with three columns
families_mass <- data.frame(Family = character(0),
                            AvgMass = numeric(0),
                            NumSpecies = numeric(0))

for (sp in 1:dim(s2003)[1]){
  # Extract family and mass
  my_fam <- s2003[sp,]$Family
  my_mass <- exp(s2003[sp, ]$LogMass)
  # If both are available
  if ((is.na(my_fam) == FALSE) & (is.na(my_mass) == FALSE)) {
    # Check whether the family is already in the data.frame
    # if so, return the index
    fam_num <- which(families_mass$Family == my_fam)
    # The function which returns integer(0) if no match is found
    # We can test wheter a match was found by checking the length of
    # fam_num
    if (length(fam_num) == 1) {
      # Add the mass
      families_mass[fam_num, "AvgMass"] <- families_mass[fam_num, "AvgMass"] + my_mass
      # Add to number of species
      families_mass[fam_num, "NumSpecies"] <- families_mass[fam_num, "NumSpecies"] + 1
    } else {
      # Add the family to the data.frame
      families_mass <- rbind(families_mass,
                              data.frame(Family = my_fam,
                                         AvgMass = my_mass,
                                         NumSpecies = 1))
    }
  }
}

# Now divide AvgMass by NumSpecies
families_mass$AvgMass <- families_mass$AvgMass / families_mass$NumSpecies
# And reorder the data for readability
# Because R turned families names into "factors" (a common problem), we need
# to transform them back into characters before sorting...
families_mass$Family <- as.character(families_mass$Family)
families_mass <- families_mass[order(families_mass$Family), ]
# See the first few records
head(families_mass)
```

```
##              Family  AvgMass NumSpecies
## 141    Abrocomidae 10.35712          3
## 91     Acrobatidae  5.15517          1
## 114      Agoutidae 50.24676          3
## 39    Anomaluridae 12.64420          7
## 105 Antilocapridae 95.03180          6
## 115   Aplodontidae 20.08554          1
```

The complete script is available as `body_mass_family`.

3. Measure the time it takes to run the analysis, using the command `system.time(source("body_mass_family.R"))`

Fairly straightforward:

```
system.time(source("body_mass_family.R"))
```

```
##    user  system elapsed
##   1.825   0.019   1.866
```

4. There are many ways to accomplish the task. Write the analysis using a different method, make sure it returns the same results, and time the alternative solution. Which one is more readable? Which one is faster?

A possible alternative way of performing the same task would be to:

- Take the unique families in the database
- For each family, subset the data
- Compute the average body mass

An implementation of this algorithm would look like:

```r
# Get unique families, removing NAs
fam_unique <- sort(unique(s2003$Family[is.na(s2003$Family) == FALSE]))
# Empty data frame
families_mass <- data.frame()
# Cycle through families
for (fam in fam_unique) {
  # Get body masses
  fam_logmass <- s2003[(s2003$Family == fam) &
                      (is.na(s2003$LogMass) == FALSE) &
                      (is.na(s2003$Family) == FALSE),
                      "LogMass"]
  # Add to the data.frame if there are masses
  families_mass <- rbind(families_mass,
                         data.frame(Family = fam,
                                    AvgMass = mean(exp(fam_logmass)),
                                    NumSpecies = length(fam_logmass)))
}
# See the results
head(families_mass)
```

```
##            Family  AvgMass NumSpecies
## 1    Abrocomidae 10.35712          3
## 2    Acrobatidae  5.15517          1
## 3      Agoutidae 50.24676          3
## 4   Anomaluridae 12.64420          7
## 5 Antilocapridae 95.03180          6
## 6   Aplodontidae 20.08554          1
```

This script (saved in `body_mass_family_subset.R`) is much faster:

```r
system.time(source("body_mass_family_subset.R"))
```

```
##    user  system elapsed
##   0.232   0.002   0.237
```

We have seen that there are multiple ways to solve a problem. Each come with its own speed of execution, readability, and complexity. In science, we want to strike a balance between simplicity and readability on the one side, and efficiency and speed on the other.