Exercises part 2

Course: Data analysis and visualization using R

Section 4. apply and its relatives

In this section you will encounter some exercises revolving around the different flavours of apply.

Exercise 4.1.

On the course website, you will find file whale_selenium.txt. You could download it into your working directory manually or use `download.file()`, but you can also read it directly using `read.table()` as shown here.

```
whale.selenium <- read.table(
    "https://raw.githubusercontent.com/MichielNoback/R_data_analysis_and_visualization/master/presentati
    header = T,
    row.names = 1)
head(whale.selenium)
```

```
##   liver.Se tooth.Se
## 1     6.23   140.16
## 2     6.79   133.32
## 3     7.92   135.34
## 4     8.02   127.82
## 5     9.34   108.67
## 6    10.00   146.22
```

(a) Report the means of both columns using apply().

```
apply(X = whale.selenium, MARGIN = 2, FUN = mean)
```

```
## liver.Se tooth.Se
##   20.685  156.599
```

(b) Report the standard deviation of both columns, again using apply.

```
apply(X = whale.selenium, MARGIN = 2, FUN = sd)
```

```
## liver.Se tooth.Se
## 13.44911 36.05949
```

(c) Report the standard error of the mean of both columns, again using apply. The SEM is calculated as

$$\frac{s}{\sqrt{n}}$$

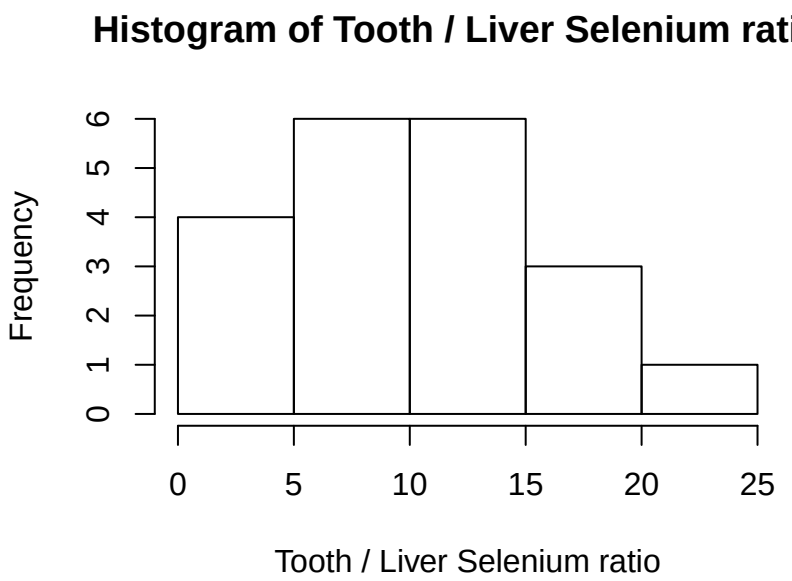
where s is the sample standard deviation and n the number of measurements. You should create the function calculating this statistic yourself.

```r
my.sem <- function(x) {
    sem <- sd(x) / sqrt(length(x))
}
apply(X = whale.selenium, MARGIN = 2, FUN = my.sem)
```

```
## liver.Se tooth.Se
## 3.007313 8.063148
```

(d) Using apply, calculate the ratio of Se_{tooth}/Se_{liver} and attach it to the whale.selenium dataframe as column `ratio`. Create a histogram of this ratio.

```r
whale.selenium$ratio <- apply(X = whale.selenium,
      MARGIN = 1,
      FUN = function(x){
          x[2] / x[1]
      })
hist(whale.selenium$ratio,
     xlab = "Tooth / Liver Selenium ratio",
     main = "Histogram of Tooth / Liver Selenium ratios")
```



(e) Using print and paste, report the mean and the standard deviation of the ratio column, but do this with an inline expression, e.g. an expression embedded in the Rmarkdown text.

The mean of the ratios is 10.565436 and the standard deviation is 5.5871469

Exercise 4.2.

This exercise revolves around the `ChickWeight` dataset of the built-in `datasets` package.

(a) Use an R expression to report the number of chickens used in the experiment.

```r
#MANY WAYS TO GET THERE
length(split(ChickWeight, ChickWeight$Chick))
```

```
## [1] 50
```

```
#OR
sum(tapply(ChickWeight$Diet, ChickWeight$Chick, FUN = function(x){1}))
```

```
## [1] 50
```

```
#OR
length(unique(ChickWeight$Chick))
```

```
## [1] 50
```

```
#OR
nrow(aggregate(x = ChickWeight, by = list(ChickWeight$Chick), FUN = function(x){x}))
```

```
## [1] 50
```

(b) Use `aggregate()` to get the mean weight of the chickens for the different Diets.

```
aggregate(formula = weight ~ Diet, data=ChickWeight, FUN = mean, na.rm = T)
```

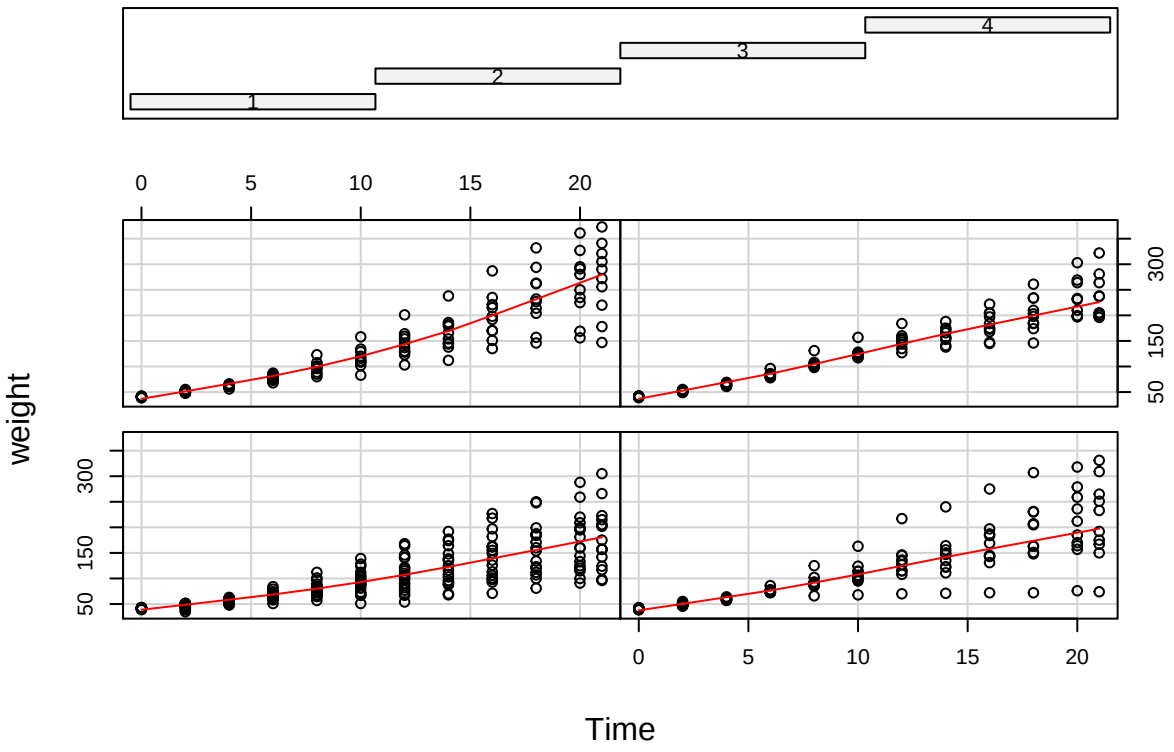
```
##   Diet   weight
## 1    1 102.6455
## 2    2 122.6167
## 3    3 142.9500
## 4    4 135.2627
```

```
#OR
aggregate(x = ChickWeight$weight, by = list(Diet = ChickWeight$Diet), FUN = mean, na.rm = T)
```

```
##   Diet        x
## 1    1 102.6455
## 2    2 122.6167
## 3    3 142.9500
## 4    4 135.2627
```

(c) Use `coplot()` to plot a panel with weight as function of Time, split over Diet.

```
coplot(weight ~ Time | Diet, data = ChickWeight, panel = panel.smooth)
```



(d)

Add a column called `weight.gain` to the dataframe holding values for the weight gain since the last measurement. Take special care with rows marking the boundaries between indivivdual chickens! You could consider using a traditional for loop here.

```
#A naive for-loop here - is this the best solution?
ChickWeight$weight.gain <- NA #create the column with missing values
for (i in 1:nrow(ChickWeight)) {
    #skip first row and rows that are preceded by values for another chick
    if (i > 1 && ChickWeight$Chick[i] == ChickWeight$Chick[i-1]) {
        ChickWeight[i, "weight.gain"] <- ChickWeight$weight[i] - ChickWeight$weight[i-1]
    }
}
#stored.weight.gain <- ChickWeight$weight.gain
#save(stored.weight.gain, file = "./data/ChickWeight_weight_gain.Rdata")
```

(e)

Split the `weight.gain` column on Diet and report the mean, median and standard deviation for each diet. Note, if you were not succesful in the previous question, load and attach the data from file `ChickWeight_weight_gain.Rdata` downloadable from `https://github.com/MichielNoback/R_data_analysis_and_visua` (click on button "Raw" to download and use `load()` to load variable `stored.weight.gain`).

```
tapply(X = ChickWeight$weight.gain, INDEX = ChickWeight$Diet, FUN = mean, na.rm = T)
```

```
##        1        2        3        4
## 11.49000 15.81818 20.86364 17.43519
```

```r
#or with aggregate
aggregate(formula = weight.gain ~ Diet, data = ChickWeight, FUN = median)
```

```
##   Diet weight.gain
## 1    1         9.0
## 2    2        13.0
## 3    3        19.5
## 4    4        17.0
```

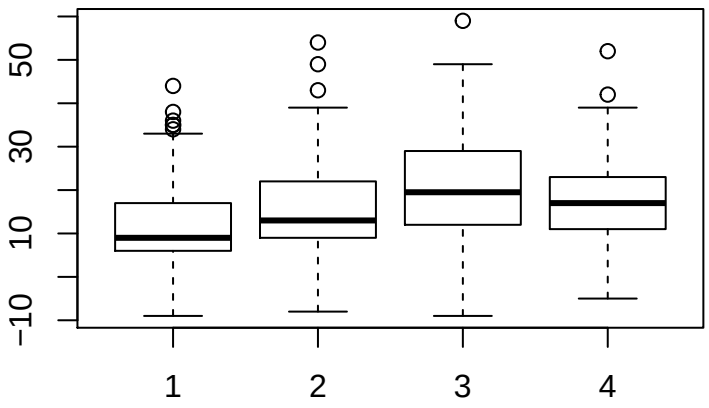
```r
#or with split and sapply
sapply(split(ChickWeight[, "weight.gain"], ChickWeight$Diet), sd, na.rm = T)
```

```
##         1         2         3         4
##  9.390999 11.030551 12.713121 10.008663
```

(f)

Create a (single-panel) boxplot for weight gain, split over Diet. Hint: read the `boxplot()` help page!

```r
boxplot(weight.gain ~ Diet, data = ChickWeight)
```



Exercise 4.3.

The food constituents dataset on the course website holds information on ingredients for different foods. Individual foods are simply marked with an id.

(a) Load the data and report the different food categories.

```r
foods <- read.table(
    "https://raw.githubusercontent.com/MichielNoback/R_data_analysis_and_visualization/master/exercises/
levels(foods$Type)
```

```
##  [1] "beverage"  "bread"     "cake"      "cheese"    "chips"
##  [6] "chocolate" "cookies"   "jam"       "meat"      "milk"
## [11] "nuts"      "pasta"     "pizza"     "potato"    "rice"
## [16] "vegetable"
```

(b) What is the mean energy content of chocolate foods?

```r
mean(foods[foods$Type == "chocolate", "kcal"])
```

```
## [1] 532.129
```

(c) What is the food category with the highest mean fat content?

```r
#aggregate over Type
mean.fat <- aggregate(formula = fat.total ~ Type, data = foods, FUN = mean)
#order and select first
mean.fat[order(mean.fat$fat.total, decreasing = T)[1], ]
```

```
##    Type fat.total
## 11 nuts      45.5
```

(d) What food category has the highest mean energy content, and which has the lowest?

```r
mean.energy <- aggregate(formula = kcal ~ Type, data = foods, FUN = mean)
mean.energy[order(mean.energy$kcal)[1], ]
```

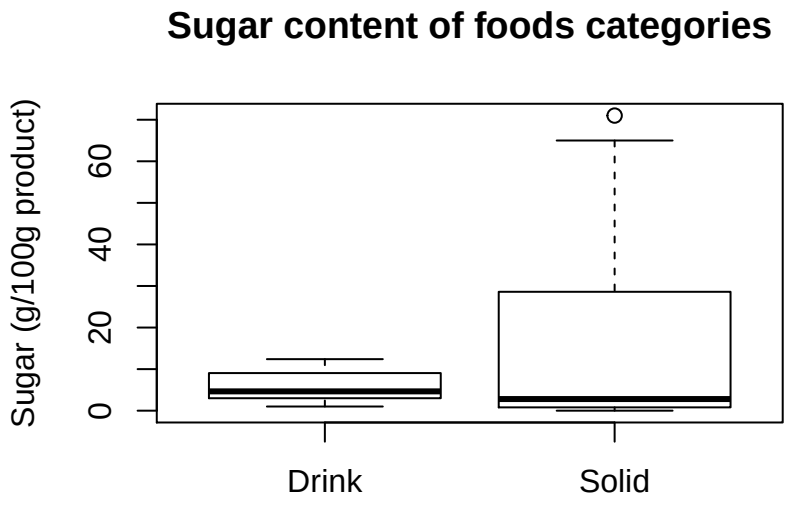
```
##       Type  kcal
## 1 beverage 26.75
```

```r
mean.energy[order(mean.energy$kcal, decreasing = T)[1], ]
```

```
##    Type kcal
## 11 nuts  592
```

(e) Challenge ahead! Create a boxplot showing the difference in sugar content between drink and solid food.

```r
#more verbose means possible; this efficient way demonstrating use of %in%
foods$solid.state <- !foods$Type %in% c("milk", "beverage")
boxplot(formula = carb.sugar ~ solid.state,
        data = foods,
        main = "Sugar content of foods categories",
        names = (c("Drink", "Solid")),
        ylab = "Sugar (g/100g product)")
```



(f) Assuming both unsaturated fats and sugar are bad for you, what food category do you consider the worst? Thiink of a means to answer this, explain it and carry it out.

```
## YOUR CODE HERE
```

Exercise 4.4.

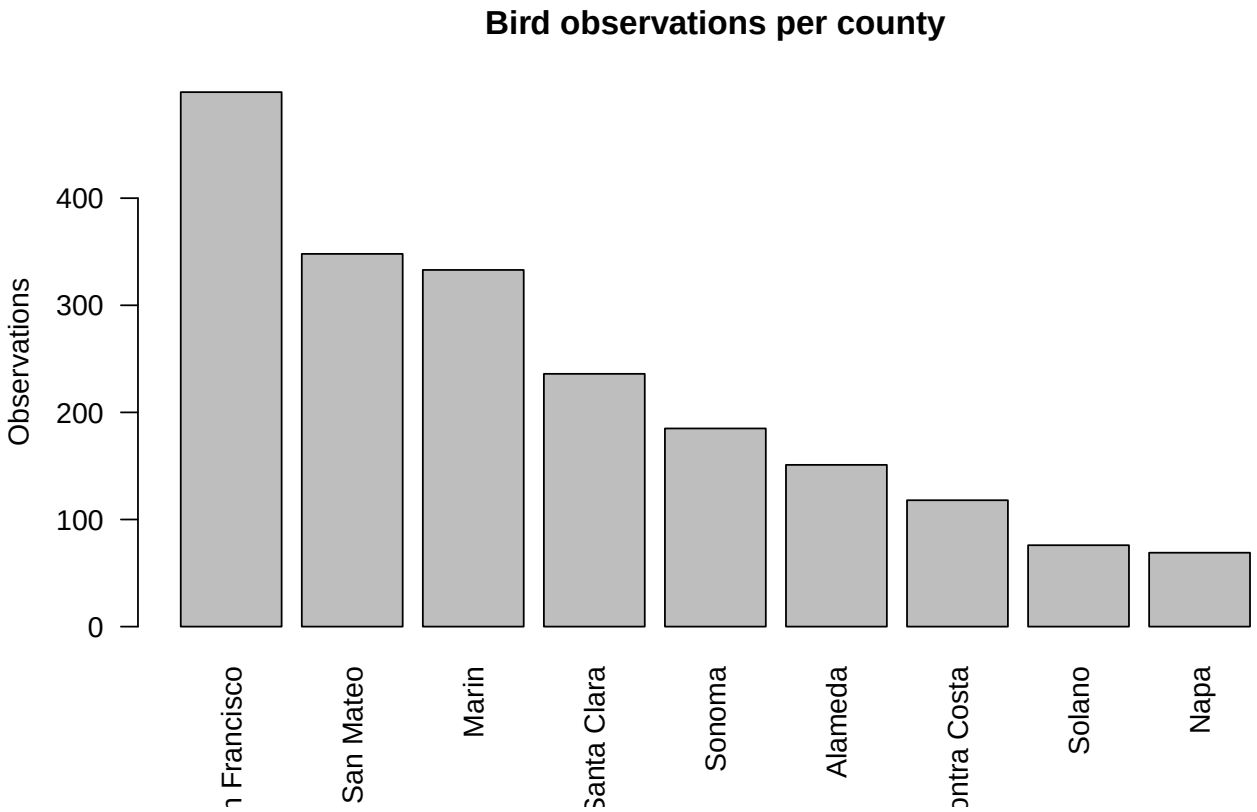
This exercise revisits the GOLDEN GATE AUDUBON SOCIETY dataset downloaded and prepared in the a previous exercise. I hope you still have the csv version of it. If it got lost, download it again. Again, open the file in Excel, replace all occurrences of the ";" character to "," and use "Save as.." to save it as ".csv" file (Comma-separated). Alternatively, download it from the course website. Load the dataset.

```
##reload the data
bird.obs <- read.table("data/Observations-Data-2014.csv",
                sep=";",
                head=T,
                na.strings = "",
                quote = "",
                comment.char = "",
                as.is = c(1, 6, 7, 8, 13))
bird.obs$Count <- as.integer(bird.obs$Number)
```

```
## Warning: NAs introduced by coercion
```

(a) Report the number of observations per `County`. Use both a textual as a barplot representation. With the barplot, you should order the bars according to observation numbers.

```
c.split <- split(x = bird.obs, f = bird.obs$County)
c.counts <- sapply(c.split, nrow)
barplot(c.counts[order(c.counts, decreasing = T)],
        main = "Bird observations per county",
        ylab = "Observations",
        las = 2)
```



(b) Report the number of observations per `Observer.1`, ordered from high to low observation count, but only those Observers with more than 10 observations into this dataset.

```
obs.split <- split(x = bird.obs, f = bird.obs$Observer.1)
obs.counts <- sapply(obs.split, nrow)
obs.counts <- obs.counts[obs.counts > 10]
obs.counts[order(obs.counts, decreasing = T)]
```

```
##               mob         Ron Thorn     Dominik Mosur
##                88                74               50
##   Alvaro Jaramillo     David Assmann     Garth Harwood
##                48                42               33
##        logan kahle  Albert Linkowski     Murray Berner
##                30                29               29
##        Hugh Cotter   Farallon Island     Brooke Miller
##                28                21               19
##         Jerry Ting      Michael Park       Russ Bright
##                18                18               18
##           Bob Dunn          Lisa Hug      Josiah Clark
##                17                17               16
##      Ruth Rudesill   Debi Shearwater      Gena Zolotar
##                16                15               15
##      Paul Saraceni      Bob Richmond Kimberly Jannarone
##                15                13               13
##       Eric Pilotte  Caroline Lambert        Dan Singer
##                12                11               11
##       Janna Pauser       Pete Dunten       Scott Sorby
##                11                11               11
```

(c) Which observer has the highest number of observations listed (and how many is that)?

```
obs.counts[order(obs.counts, decreasing = T)][1]
```

```
## mob
##  88
```

(d) Report the different observed species, using `Common.name`, for each genus. **Challenge: Report only the 5 Genera with the highest number of observed species**.

```
g.split <- split(bird.obs, bird.obs$Genus)
g.species <- lapply(g.split, function(x) {
    unique(x$Common.name)
})
#create ordering
g.species.count <- sapply(g.species, length)
g.order <- order(g.species.count, decreasing = T)
#apply order to list and select only first five
g.species[g.order[1:5]]
```

```
## $Setophaga
##  [1] American Redstart           Cape May Warbler
##  [3] Northern Parula             Magnolia Warbler
##  [5] Bay-breasted Warbler        Blackburnian Warbler
##  [7] Yellow Warbler              Chestnut-sided Warbler
##  [9] Blackpoll Warbler           Black-throated Blue Warbler
## [11] Palm Warbler                Pine Warbler
## [13] Yellow-rumped Warbler       Prairie Warbler
## [15] Black-throated Gray Warbler Townsend's Warbler
## [17] Hermit Warbler
## 329 Levels: Acorn Woodpecker Allen's Hummingbird ... Yellow-throated Vireo
##
## $Calidris
##  [1] Red Knot               Surfbird               Ruff
##  [4] Stilt Sandpiper        Red-necked Stint       Sanderling
##  [7] Dunlin                 Rock Sandpiper         Baird's Sandpiper
## [10] Pectoral Sandpiper     Semipalmated Sandpiper
## 329 Levels: Acorn Woodpecker Allen's Hummingbird ... Yellow-throated Vireo
##
## $Vireo
## [1] Yellow-throated Vireo Plumbeous Vireo       Cassin's Vireo
## [4] Blue-headed Vireo     Hutton's Vireo        Warbling Vireo
## [7] Philadelphia Vireo    Red-eyed Vireo        Yellow-green Vireo
## 329 Levels: Acorn Woodpecker Allen's Hummingbird ... Yellow-throated Vireo
##
## $Empidonax
## [1] Yellow-bellied Flycatcher Willow Flycatcher
## [3] Wilson\xd5s Snipe         Least Flycatcher
## [5] Hammond's Flycatcher      Gray Flycatcher
## [7] Dusky Flycatcher          Pacific-slope Flycatcher
## 329 Levels: Acorn Woodpecker Allen's Hummingbird ... Yellow-throated Vireo
##
## $Larus
```

```
## [1] Heermann's Gull          Mew Gull
## [3] Herring Gull             Thayer's Gull
## [5] Lesser Black-backed Gull Slaty-backed Gull
## [7] Glaucous-winged Gull     Glaucous Gull
## 329 Levels: Acorn Woodpecker Allen's Hummingbird ... Yellow-throated Vireo
```

(e) Challenge! Create a Dataframe holding the number of birds per day (use Date.start) and plot it with date on the x-axis and number of birds on the y-axis. Hint: use `as.Date()` to convert the character date to a real date field. See this page how you can do that Date Values.

```r
bird.obs$Date.start <- as.Date(bird.obs$Date.start, format = "%d-%b-%y")
date.series <- aggregate(Count ~ Date.start, data = bird.obs, FUN = sum, na.rm = T)
#2024 is an error input, remove it
date.series <- date.series[1:nrow(date.series)-1, ]
plot(x = date.series$Date.start, y = date.series$Count, ylim = c(0, 250))
```