Coding task 12.03.21 - Looping over lists

Lists in R

- in lists you can sum up all other types of data in R
- e.g a list of data frames, a list of vectors, a list of lists
- the list items do not have to have the same dimensions
- very usefull if you have two similar data sets, that you want to treat the same way (e.g two conditions/samples)

```
# example data
data("beavers")
data("airquality")
data("mdeaths")
```

```
## Warning in data("mdeaths"): data set 'mdeaths' not found
```

```
# we have data from two beavers
# beaver1
# beaver2

# and put both together in a list
beavers <- list(beaver1, beaver2)

# you can give each item within a list a name
names(beavers)<- c("Beaver1", "Beaver2")

# you can access list items via [[]]
head(beavers[[1]])
```

```
##   day time  temp activ
## 1 346  840 36.33     0
## 2 346  850 36.34     0
## 3 346  900 36.35     0
## 4 346  910 36.42     0
## 5 346  920 36.55     0
## 6 346  930 36.69     0
```

```
# or via the name
head(beavers$Beaver2)
```

```
##   day time  temp activ
## 1 307  930 36.58     0
## 2 307  940 36.73     0
## 3 307  950 36.93     0
## 4 307 1000 37.15     0
## 5 307 1010 37.23     0
## 6 307 1020 37.24     0
```

Task 1: Build a List containing the beaver list and the airquality data set and maybe a random word of you choice.

Loops in R

If you want to loop over lists it is of course helpfull is they have a similar layout. Here are three ways to loop over a list:

For Loops

The easiest loop in R is s for loop. (However, for loops are often the most coding intensive and slowest type of loops.) Here is an example on how to use a for loop on a data set

```
# initialise a variable, where you want to save the results from the loop
mean_temp <- vector()

for(i in 1:length(beavers)) # here you specify the loop variable and how often the loop is run (from 1
  {
  mean_temp[i] <- mean(beavers[[i]]$temp)
}



# note: of course with two you could still calcualte it seperatly, but you might have 10 beavers :)
```

Apply, sapply, lapply

These are base R functionalities that are faster and less text intensive then for loops. Their differ in thei output type: - lapply return a list - sapply returns a vector - apply returns a dataframe

```
# lapply, sapply
# takes as input a list and a function of what to do with each list element, here the function variable
mean_temp2_list <- lapply(beavers, function(x) mean(x$temp))
mean_temp2_vector <- sapply(beavers, function(x) mean(x$temp))

# apply cannot be used directly on the list but needs a data frame
# but you can loop over all columns at the same time
# you can specify 1 for apply this function rowwise or 2 for column wise
all_means_beaver1 <- apply(beaver1, 2, mean)

# you can combine laply an apply to get loop over all columns of all list menbers
all_mean_both_beavers <- lapply(beavers, apply,2, mean)
```

purrr::map

This is a tidyverse alternative to loop over lists.

```
library(purrr)

# map marks the function by the ~ sign and within the function the variable is always caled .x
map(beavers, ~mean(.x$temp))
beavers %>% map( ~mean(.x$temp))

# you can again change what kind of output you want to get
# a numeric vector
map_dbl(beavers, ~mean(.x$temp))

# a character vector
map_chr(beavers, ~mean(.x$temp))
```

```r
# map allows you to use dplyr within each loop
library(dplyr) ## You made a nice task on dplyr if you are not familiar with it, it might be worth look
map(beavers, ~mutate(.x, hour = substring(as.character(time),1,1)))

# and to use pipes within the loop
map(beavers, ~mutate(.x, hour = substring(as.character(time),1,1)) %>%
     arrange(hour))
```

Try to solve these tasks with all three methods

Task 2 Calucalte the sd of the 2 columns of the 2 dataframes.

Task 2b Add up the time and the temperature.

Task 3 Calcluate the rowwise mean (athough its meaningless here).

For these tasks choose your favorite method

Task 4 Caluclate the mean temp hourwise (900-950,1000-1050 etc.) **Task 5** Make a new list that contains the beaver data of both beavers only for even days **Task 6** Look at the starwars data set column films. This is a list in da data frame! Lets say you want to look at the distribution of skin color and haircolor for the different films. Can you make a list containing one element per film with all characters? How are the distributions filmwise? Make some nice plots (maybe in a loop).

```r
head(dplyr::starwars)
```

```
## # A tibble: 6 x 14
##   name  height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke~    172    77 blond      fair       blue            19   male  mascu~
## 2 C-3PO    167    75 <NA>       gold       yellow         112   none  mascu~
## 3 R2-D2     96    32 <NA>       white, bl~ red             33   none  mascu~
## 4 Dart~    202   136 none       white      yellow          41.9 male  mascu~
## 5 Leia~    150    49 brown      light      brown           19   fema~ femin~
## 6 Owen~    178   120 brown, gr~ light      blue            52   male  mascu~
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

Benchmark the different options

Task 7 There is a function microbenchmark from the package microbenchmark, which allows you to compare the efficency of different functions. Take one of the tasks before and benchmark the three loop types. Which was the best solution?