DanieleP/PA2-clarifying_instructions

README.md

Hi everyone, Assignment 2 can be very challenging to be understood. The solution is easier than what it looks, and it's strongly connected with the example given in the instructions. However, I think it's worth spending some time to understand more about the logic behind the functions given in the instruction.

1) makeVector is a function that stores a list of functions. I had no idea it was possible. Here is an easier example of a function that stores functions:

```
##function plusFunctions stores two functions:
## plustwo() : sums 2 to the given value
## plusthree() : sums 3 to the given value
plusFunctions <- function (){
    plustwo <- function(y) {
        x <- y + 2
        return(x)
    }
    plusthree <- function(y) {
        x <- y + 3
        return(x)
    }
    #the following line stores the two functions:
    list(plustwo = plustwo, plusthree = plusthree)
}
```

---

2) to use the functions stored in the main function, you need to subset the main function. To do this, you need the name of the main function + "$" + the name of the second function + (arguments)

```
## Example from the function above:
> a <- plusFunctions()
> a$plustwo(5)
[1] 7
```

```
> a$plusthree(5)
[1] 8
```

3) makeVector contains 4 functions: set, get, setmean, getmean. Let's start from get, because it's the easiest.

```
get <- function() x
```

get is a function that returns the vector x stored in the main function. Doesn't require any input.

Example:

```
> a <- makeVector(c(5,1,3))
> a$get()
[1] 5 1 3
```

4) set is a function that changes the vector stored in the main function.

```
set <- function(y) {
            x <<- y
            m <<- NULL
    }
```

We don't need to use this function unless we want to change the vector. "x <<- y" substitutes the vector x with y (the input) in the main function (makeVector). If it was "x <- y" it would have substitute the vector x with y only in the set function. "m <<- NULL" restores to null the value of the mean m, because the old mean of the old vector is not needed anymore. The new mean needs to be recalculated through the function cachemean.

Example:

```
> a <- makeVector(c(5,1,3))
```

```
> a$get()
[1] 5 1 3
> a$set(c(7,4,1,2))
>a$get()
[1] 7 4 1 2
```

5) setmean and getmean are functions very similar to set and get. They don't calculate the mean, they simply store the value of the input in a variable m into the main function makeVector (setmean) and return it (getmean).

```
setmean <- function(mean) m <<- mean
getmean <- function() m
```

This value "mean", input of seatmean, is supposed to be the mean of the vector x. However it simply stores a value, like in the following example:

```
> a<- makeVector(c(1,2,3,4))
> a$setmean(10)
> a$getmean()
[1] 10
```

As you see 10 is not the mean of vector (1,2,3,4), but because we stored it with setmean, we got it back with getmean.

6) To store the 4 functions in the function makeVector, we need the function list(), so that when we assign makeVector to an object, the object has all the 4 functions.

```
    list(set = set, get = get,
         setmean = setmean,
         getmean = getmean)
```

7) Now that makeVector is a bit more clear, let's observe the second function, cachemean. Input of cachemean is the object where makeVector

is stored.

Example:

```
> a <- makeVector(c(1,2,3,4))
> cachemean (a)
```

8) The first thing cachemean does is to verify the value m, stored previously with getmean, exists and is not NULL. If it exists in memory, it simply returns a message and the value m, that is supposed to be the mean, but not necessarily.

```
m <- x$getmean()
        if(!is.null(m)) {
                message("getting cached data")
                return(m)
        }
```

9) If it was the case, "return(m)" would have ended the function. So everything that follows this if() is a sort of else {}. data gets the vector stored with makeVector, m calculates the mean of the vector and x$setmean(m) stores it in the object generated assigned with makeVector.

```
data <- x$get()
m <- mean(data, ...)
x$setmean(m)
m
```

Hope this helps!