R: what does the small function do when it is not called from anywhere?

1

Here is the setup.

Below are two functions that are used to create a special object that stores a numeric vector and cache's its mean.

The first function, makeVector creates a special "vector", which is really a list containing a function to

set the value of the vector get the value of the vector set the value of the mean get the value of the mean

```
makeVector <- function(x = numeric()) {
        m <- NULL
        set <- function(y) {
                x <<- y
                m <<- NULL
        }
        get <- function() x
        setmean <- function(mean) m <<- mean
        getmean <- function() m
        list(set = set, get = get,
             setmean = setmean,
             getmean = getmean)
}
```

The following function calculates the mean of the special "vector" created with the above function. However, it first checks to see if the mean has already been calculated. If so, it gets the mean from the cache and skips the computation. Otherwise, it calculates the mean of the data and sets the value of the mean in the cache via the setmean function.

```
cachemean <- function(x, ...) {
        m <- x$getmean()
        if(!is.null(m)) {
                message("getting cached data")
                return(m)
        }
        data <- x$get()
        m <- mean(data, ...)
        x$setmean(m)
        m
```

Ok, I was thinking about this 2 function for 2 days. I tried to understand how it work because I will need to write similar function for different purpose. After all this thinking I could not understand one thing - why the heck we need set() function in makeVector and what the heck does it do?

Below is the description of what I managed to understand and how I understand the logic (taken from my comment on Coursera):

`makeVector` works without `set()` function?

`set()` is never called in `cachemean`. And no any function in `makeVector` is calling `set()` as well. What is really happening as I understand it:

1. `cachemean` calls `getmean()` function and assigns it to variable `m`. `m` here is local variable in `cachemean` scope
2. What `getmean()` does it returns `m` which is already from `makeVector` scope! In `makeVector` scope `m` is `NULL` at the moment of execution
3. `cachemean` checks if `m` is `NULL` or not. At the first call it is `NULL` so we skip `if` condition
4. Now we create a local variable `data` and assign it to a call of `get()` function. What `get()` does it returns the value of `x` which is in fact formal parameter of `makeVector`. In other words it returns a vector which we initially passed to `makeVector` function as an argument. After this we have that `data` is our numeric vector
5. Next step is rather simple. We reassign `cahchemean` scope `m` to mean() of `data`. That is we find mean of our numeric vector we passed as an argument to `makeVector`.
6. Now comes the step which allows us to cache result! we call `setmean()` function and pass mean of our numeric vector - `m` - to it as an argument. If you look at `makeVector` you will see that what `setmean()` does is just assigns the argument passed to it to `m` **BUT in `makeVector` scope!**.
7. Finally `cachemean` returns `m`.

vector once again.

1. The first line of `cachemean` is the key line here. We assign `getmean()` to `m` in **`cachemean` scope**. But what `getmean()` does in this case? If you look at the `makeVector`, `getmean()` just returns the value of `m` **BUT from `makeVector` scope**!. And the value of `m` in that scope is our mean of numeric vector.
2. After that it is simple. We check if `m` is NULL, it is obviously not. So we print a `message` and return `m` which is simply the mean we have found during the first call of `cachemean`.

Now comes the question which confuses me a lot. Why the heck we need to define `set()` function?

We do not call it from `cachemean`.

Moreover if you delete `set()` from `makeVector`, it still works and `cachemean` still works and returns cached value of mean of a numeric vector.

Moreover-moreover if I look closer to the `set()` function, this function does not really make sense. It assigns formal parameter `y` to parent environment variable `x`. But why we need to do this? If to get an initial numeric vector, then we do it by `get()`.

And then it assigns `NULL` to `m` which also does not make sense. Just a line before we have done the same thing.

So can please someone explain me:

1. Do I understand the logic correct?
2. What `set()` function is doing and why do we need it at all?

Thanks a lot in advance!

P.S. I am new to R and I am stupid, but I want to understand, what I am missing here.

EDIT: Really sorry for long read. But everyone on this site is asking what I have already done/tried, so I tried to describe it :-)

r

asked Jan 15 '16 at 12:49

Nikolay Dudaev
322 2 6

1 The `set` function just allows to set the values of an object returned by `makeVector` without having to call `makeVector` again. Try `a<-makeVector(1:10);cachemean(a)` and then `a$set(11:20);a$getmean()` . It changes the values of `x` and `m` in the `makeVector` scope. It's not relevant to `cachemean` however. The `get..` and `set..` functions are often called accessors and mutators methods in the OOP paradigm. – nicola Jan 15 '16 at 13:10

1 This is too long to read for me right now, but the relevant keyword is "closures". Anyway, `set` is not really necessary, but useful: `vec <- makeVector(1:10); cachemean(vec); cachemean(vec); vec$set(1:3); cachemean(vec); cachemean(vec)` – Roland Jan 15 '16 at 13:11

See adv-r.had.co.nz/Functional-programming.html for some reading material. – Roland Jan 15 '16 at 13:13

Oh yes, in this case it makes perfectly sense. I understand my thinking was a bit narrow, concentrated on only these 2 functions cachemean and makeVector. Thank you very much! Thanks for a reading @Roland ! – Nikolay Dudaev Jan 15 '16 at 13:25

@nicola: Would you mind converting your comment into an answer such that Nikolay Dudaev can accpet it? This way this question wouldn't come up as unanswered. I just read all this text just to figure out the question was answered already. – mschilli Aug 6 '18 at 14:10