14 November, 2014
6:21 AM

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

Subscribe for email updates.

ProgrammingAssignments × + Add Tag

Sort replies by: Oldest first Newest first Most popular

joseph fernando · 21 hours ago

Could someone kindly explain:

1) what does "seatmean()" do

2) is there a purpose in the way the code is written, i.e. "setmean <- function(mean) m <<- mean"

Thanks in advance.

Kind regards,
Joe

0 · flag

Bill Hilton · 20 hours ago

1) what does "seatmean()" do

This would be called by the cache function, passing in the value of mean that it calculates the first time it accesses makeVector ... any subsequent accesses it will just fetch this cached value, so it only gets set once, then read (possibly) many times

2) is there a purpose in the way the code is written, i.e. "setmean <- function(mean) m <<- mean"

He is being sneaky here ... in the slides he would write this like so:

```
setmean <- function(mean) {
    m <<- mean
}
```

You could also write it on one line like so:

```
setmean <- function(mean) { m <<- mean }
```

Because it is a one-line function you can omit the { }, which is what he does ... so it's basically a single-line function written a new way, unfortunately. A lot of students get confused by this ...

↑ 2 ↓ · flag

+ Comment

joseph fernando · 20 hours ago

Thanks Bill!

Could you help me with a few more clarifications:

1) setmean <- function(mean) m <<- mean - Is it assigning the "mean" function to "m" OR is it something else?
2) In cachemean- what does "x$setmean(m)" do.
3) I still can't work out in cachemean where "setmean()" is used...what am I missing...

Appreciate you taking time out to help all of us!

Kind regards,
Joe

↑ 0 ↓ · flag

+ Comment

Bill Hilton · 20 hours ago

1) setmean <- function(mean) m <<- mean - Is it assigning the "mean" function to "m" OR is it something else?

No, not the mean function ... 'mean' is just a token or variable he's passing in. It's confusing because it's also the name of a function (and there are far too many 'x' variables in this example too). Had he written the function like this it would be easier to understand:

```
setmean <- function(storeValue) { m <<- storeValue }
```

2) In cachemean- what does "x$setmean(m)" do.

It calls the method (an internal function is called a method) setmean() inside the object x ... 'x' is a token signifying the object we are passing it to. So if you earlier wrote bigVec <- makeVector(1:1000) you made an object with name bigVec ... now calling cachemean(bigVec) accesses that object ... first time the cached value is NULL so cachemean will calculate the mean and then store it in the bigVec object.

So in this example 'x$setmean(m)' becomes 'bigVec$setMean(calculatedMean)' ...

3) I still can't work out in cachemean where "setmean()" is used...what am I missing...

I'll post something I wrote up about these two functions when I took this class earlier ... I think it will explain it. I'm removing the set() method for now since cachemean() doesn't call it:

```
makeVector <- function(x = numeric()) {      # input x will be a vector

    m <- NULL    #  m will be our 'mean' and it's reset to NULL every
                 #    time makeVector is called

                 #  note these next three functions are not run when makeVector is called.
                 #   instead, they will be used by cachemean() to get values for x or for
                 #   m (mean) and for setting the mean.  These are usually called object 'meth
ods'

    get <- function() { x }   # this function returns the value of the original vector

    setmean <- function(mean)  # this is called by cachemean() during the first cachemean()
               { m <<- mean }  #  access and it will store the value using superassignment

    getmean <- function() { m } # this will return the cached value to cachemean() on
                                #  subsequent accesses

    list(get = get,          #  OK, this is accessed each time makeVector() is called,
         setmean = setmean,  #   that is, each time we make a new object.  These are names of
         getmean = getmean)  #   the internal methods or functions so another function knows
                             #   how to access the object created with a call to makeVector

}
```

OK, that's a stripped down version of makeVector ... call it and you create an object of type list, with variable 'm' initialized to "NULL". That's it, plus the short functions which we will access from cachemean().

Here is the second function. When called it will see if the mean has been stored. If not it will calculate the mean, store it and then return it. If the mean for this object has been calculated and stored earlier it will fetch the mean and return it. That's all it does.

```
cachemean <- function(x, ...) {    # the input is an object created by makeVector
    m <- x$getmean()               # accesses the object 'x' and gets the value of the mean
    if(!is.null(m)) {              # if mean was already cached (not NULL) ...
```

```
        message("getting cached data")  # ... send this message to the console
        return(m)                       # ... and return the mean ... "return" ends
                        #   the function cachemean(), note
    }
    data <- x$get()        # we reach this code only if x$getmean() returned NULL
    m <- mean(data, ...)   # if m was NULL then we have to calculate the mean
    x$setmean(m)           # store the calculated mean value in x (see setmean() in makeVector
    m               # return the mean to the code that called this function
}
```

Here's some sample code to exercise the functions the way I think they are supposed to be used:

```
> bV <- makeVector(1:10)   # now we have an object bV

> cachemean(bV)            # now we generated the mean and cached it
[1] 5.5
> cachemean(bV)            # reading back the cached data
getting cached data
[1] 5.5
>
> bV$set(50:60)            # this is new ... bypass calling bV <- makeVector(50:60) by using $
set
>
> cachemean(bV)            #  and as you can see it worked, m was set to NULL so cachemean(bV)
 generates
[1] 55                     #    a new mean and stores it
> cachemean(bV)
getting cached data
[1] 55
> bV$get()                 # shows that a new vector was stored
 [1] 50 51 52 53 54 55 56 57 58 59 60
```

This should help!

↑ 3 ↓ · flag

+ Comment

joseph fernando · 20 hours ago

Mate thank you so much!!!!

↑ 0 ↓ · flag

Bill Hilton · 20 hours ago

No problem. This assignment was very poorly conceived and documented, I think. It helps if you are familiar with object-oriented programming, but of course a lot of students are not.

I would just point out that you can make this work for the matrix code by changing just a couple of lines, once you understand what is going on.

Good luck!

↑ 0 ↓ · flag

+ Comment

Jim Pavlik Signature Track · 5 hours ago

Bill....that was awesome.

↑ 0 ↓ · flag

+ Comment

New post

To ensure a positive and productive discussion, please read our forum posting policies before posting.

B I Link <code> Pic Math Edit: Rich Preview

Make this post anonymous to other students

Subscribe to this thread at the same time

Add post