cachematrix.R

Badarinadh Vissapragada

2020-03-25

```
## A cache is a way to store objects in memory to accelerate subsequent
## access to the same object.We demonstarte how to make functions more effici
ent
## when expensive computations require efficient coding to reduce the
##computational load.This excercise is to demonstarte storing a function retu
rn
##value in cache, to use it further repeatedly as required saving computing t
ime.
##cache to use to avoid recomputing throughout the program.We use the lexical
##scoping of R to get this done.

## makeCacheMatrix()  builds a set of functions and returns the functions
## within a "list" to the parent environment.The input matrix=x defined in th
e
## function argument. makeCacheMatrix(),due to lexical scoping, contains
##a complete copy of the environment for makeCacheMatrix(), including any obj
ects
##that are defined within makeCacheMatrix() at design time.
##The accessible objects within makeCacheMatrix() are: set(y),y,get(),
##setsolve(solve), solve,get(solve),x, invm
## The function to be executed as follows:1. Create a non-singular square mat
rix
## example: s<-matrix(rnorm(16),nrow=4,ncol=4)
## 2. run makeCacheMatrix() 3. a<-makeCacheMatrix(s) and 4.cacheSolve(a).
## 5. The result will be an inverted matrix of "s".
## 6. If you run cacheSolve(a) again,a message ""getting cached data"

  makeCacheMatrix <- function(x = matrix()) {

  invm <- NULL
    set <- function(y) {
      x <<- y
      invm <<- NULL
    }

  get <- function() x
  setsolve <- function (solve) invm <<- solve
  getsolve <- function() invm
  list(set = set, get = get,
  setsolve = setsolve,
         getsolve = getsolve)
```

```
  }
## Access the data and functions from the parent environment of"makeCacheMatr
ix"
## and computes inverse of the matrix"s" and stores in catche. If re exicute
the
## function it will access the s-1 from cache and displays message"
## "getting cached data"

  cacheSolve <- function(x, ...) {
        ## Return a matrix that is the inverse of 'x'
    invm <- x$getsolve()
    if(!is.null(invm)) {
      message("getting cached data")
      return(invm)
    }
  data <- x$get()
  invm<- solve(data, ...)
  x$setsolve(invm)
  invm
  }
```

```
> s<-matrix(rnorm(16),nrow=4,ncol=4)
> s
           [,1]       [,2]        [,3]         [,4]
[1,] -0.99360987 -0.5742637  0.06607287  0.558588666
[2,]  0.10699516  0.5358939  0.12604015  0.003870711
[3,]  0.37501953 -0.4134942  1.01400733  0.211218680
[4,] -0.05958062 -1.3934265 -0.31917469 -0.457469701
> makeCacheMatrix()
$set
function(y) {
      x <<- y
      invm <<- NULL
    }
<environment: 0x00000265d31544e0>

$get
function() x
<environment: 0x00000265d31544e0>

$setsolve
function (solve) invm <<- solve
<environment: 0x00000265d31544e0>

$getsolve
function() invm
<environment: 0x00000265d31544e0>

> a<-makeCacheMatrix(s)
> cacheSolve(a)
           [,1]      [,2]       [,3]        [,4]
[1,] -1.4068933 -5.569845  0.2672830 -1.64159238
[2,]  0.1189600  1.953507 -0.2335950  0.05393046
[3,]  0.7092137  4.615697  0.7618385  1.25678115
[4,] -0.6739278 -8.445208  0.1451748 -3.01325701
> cacheSolve(a)
getting cached data
```

```
           [,1]      [,2]       [,3]        [,4]
[1,] -1.4068933 -5.569845  0.2672830 -1.64159238
[2,]  0.1189600  1.953507 -0.2335950  0.05393046
[3,]  0.7092137  4.615697  0.7618385  1.25678115
[4,] -0.6739278 -8.445208  0.1451748 -3.01325701
```