|  |
| --- |
| > ## The functions in this file manage the cache of matrices and makes it available when required  > ## This facilitates that the inverse calculation need not be done everytime for the same object  > ## functions do  >  > ## The function makeCacheMatrix creates an instance of the matrix and the inverse of the matrix in the memory  >  > makeCacheMatrix <- function(x = matrix()) {  +  + m <- NULL  +  + set <- function(y)  + {  + x <<- y  + m <<- NULL  + }  +  + get <- function() x  +  + setInvMatrix <- function(inverse) m <<- inverse  + getInvMatrix <- function() m  +  + list(set = set , get=get, setInvMatrix = setInvMatrix , getInvMatrix = getInvMatrix)  +  + }  >  >  > ## The function cacheSolve checks the cache for the value of the object passed. If available it shall return the inverse, else calculate  > ## the inverse and set the same in the matrix object  >  > cacheSolve <- function(x, ...) {  + ## Return a matrix that is the inverse of 'x'  +  + inv <- x$getInvMatrix()  +  + if(!is.null(inv))  + {  + message("Getting data from cache")  + return(inv)  + }  +  + srcMatrix <- x$get()  + inv <- solve(srcMatrix)  + x$setInvMatrix(inv)  +  + message("Setting data into cache")  +  + inv  + }  > y  [,1] [,2]  [1,] 1 3  [2,] 2 4  > m <- makeCacheMatrix()  > m$get()  [,1]  [1,] NA  > m$set(y)  > m$getInvMatrix()  NULL  > cacheSolve(m)  Setting data into cache  [,1] [,2]  [1,] -2 1.5  [2,] 1 -0.5  > m$getInvMatrix()  [,1] [,2]  [1,] -2 1.5  [2,] 1 -0.5  > cacheSolve(m)  Getting data from cache  [,1] [,2]  [1,] -2 1.5  [2,] 1 -0.5 |
|  |
| |  | | --- | |  | |