|  |
| --- |
|  |
|  | ## calculates the inverse of the matrix. |
|  | ## If the matrix inverse has already been calculated, it will instead |
|  | ## find it in the cache and return it, and not calculate it again. |
|  |  |
|  | makeCacheMatrix <- function(m = matrix()) { |
|  | inv\_m <- NULL |
|  | set <- function(n) { |
|  | m <<- n |
|  | inv\_m <<- NULL |
|  | } |
|  | get <- function() m |
|  | setinverse<- function(inverse) inv\_m <<-inverse |
|  | getinverse <- function() inv\_m |
|  | list(set = set, get = get, |
|  | setinverse = setinverse, |
|  | getinverse = getinverse) |
|  | } |
|  |  |
|  | ## The function cacheSolve returns the inverse of a matrix A created with |
|  | ## the makeCacheMatrix function. |
|  | ## If the cached inverse is available, cacheSolve retrieves it, while if |
|  | ## not, it computes, caches, and returns it. |
|  | cacheSolve <- function(x, ...) { |
|  | ## Return a matrix that is the inverse of 'x' |
|  | inv\_m <- m$getinverse() |
|  | if (!is.null(inv\_m)) { |
|  | message("getting cached inverse matrix") |
|  | return(inv\_m) |
|  | } else { |
|  | inv\_m <- solve(m$get()) |
|  | m$setinverse(inv\_m) |
|  | return(inv\_m) |
|  | } |
|  | } |