|  |
| --- |
| makeCacheMatrix <- function(x = matrix()) { |
|  |  | + +## This function creates a special "matrix" object that can cache its inverse |
|  |  | + |
|  |  | + +makeCacheMatrix <- function(x = matrix()) { ## define the argument with default mode of "matrix" |
|  |  | + + inv <- NULL ## initialize inv as NULL; will hold value of matrix inverse |
|  |  | + + set <- function(y) { ## define the set function to assign new |
|  |  | + + x <<- y ## value of matrix in parent environment |
|  |  | + + inv <<- NULL ## if there is a new matrix, reset inv to NULL |
|  |  | + + } |
|  |  | + + get <- function() x ## define the get fucntion - returns value of the matrix argument |
|  |  | + + |
|  |  | + + setinverse <- function(inverse) inv <<- inverse ## assigns value of inv in parent environment |
|  |  | + + getinverse <- function() inv ## gets the value of inv where called |
|  |  | + + list(set = set, get = get, setinverse = setinverse, getinverse = getinverse) ## you need this in order to refer |
|  |  | + + ## to the functions with the $ operator |
|  |  | + } |
|  |  | + |
|  |  | + |
|  |  | + -## Write a short comment describing this function |
|  |  | + +## This function computes the inverse of the special "matrix" returned by makeCacheMatrix above. |
|  |  | + +## If the inverse has already been calculated (and the matrix has not changed), |
|  |  | + +## then cacheSolve will retrieve the inverse from the cache |
|  |  | + |
|  |  | + cacheSolve <- function(x, ...) { |
|  |  | + ## Return a matrix that is the inverse of 'x' |
|  |  | + + inv <- x$getinverse() |
|  |  | + + if(!is.null(inv)) { |
|  |  | + + message("getting cached data") |
|  |  | + + return(inv) |
|  |  | + + } |
|  |  | + + data <- x$get() |
|  |  | + + inv <- solve(data, ...) |
|  |  | + + x$setinverse(inv) |
|  |  | + + inv |
|  |  | + } |