This repository ▾ Search or type a command Explore Gist Blog Help Scott-Heffron + ▾

edj-boston / **coursera-r-programming**

Watch ▾ 6 Star 5 Fork 45

branch: **master** ▾

coursera-r-programming / programming-assignment-2 / **cachematrix.R**

edj-boston on Apr 20 Adding hyphen

1 contributor

65 lines (49 sloc) | 1.546 kb

Raw Blame History

```r
## A pair of functions that cache the inverse of a matrix


## Creates a special matrix object that can cache its inverse
makeCacheMatrix <- function( m = matrix() ) {

        ## Initialize the inverse property
    i <- NULL

    ## Method to set the matrix
    set <- function( matrix ) {
            m <<- matrix
            i <<- NULL
    }

    ## Method the get the matrix
    get <- function() {
        ## Return the matrix
        m
    }

    ## Method to set the inverse of the matrix
    setInverse <- function(inverse) {
        i <<- inverse
    }

    ## Method to get the inverse of the matrix
    getInverse <- function() {
        ## Return the inverse property
        i
    }

    ## Return a list of the methods
    list(set = set, get = get,
         setInverse = setInverse,
         getInverse = getInverse)
}


## Compute the inverse of the special matrix returned by "makeCacheMatrix"
## above. If the inverse has already been calculated (and the matrix has not
## changed), then the "cachesolve" should retrieve the inverse from the cache.
cacheSolve <- function(x, ...) {

    ## Return a matrix that is the inverse of 'x'
    m <- x$getInverse()

    ## Just return the inverse if its already set
    if( !is.null(m) ) {
            message("getting cached data")
            return(m)
    }

    ## Get the matrix from our object
    data <- x$get()

    ## Calculate the inverse using matrix multiplication
    m <- solve(data) %*% data

    ## Set the inverse to the object
    x$setInverse(m)

    ## Return the matrix
    m
}
```