Programming assignment 2 - Week 3

Raphael Carvalho

21/06/2019

Assignment: Caching the Inverse of a Matrix

Matrix inversion is usually a costly computation and there may be some benefit to caching the inverse of a matrix rather than compute it repeatedly (there are also alternatives to matrix inversion that we will not discuss here). Your assignment is to write a pair of functions that cache the inverse of a matrix.

Write the following functions:

makeCacheMatrix

This function creates a special "matrix" object that can cache its inverse.

```
makeCacheMatrix <- function(x = matrix()) {
  i <- NULL
  set <- function(y) {
    x <<- y
    i <<- NULL
  }
  get <- function() x
  setInverse <- function(inverse) i <<- inverse
  getInverse <- function() i
  list(set = set, get = get,
       setInverse = setInverse,
       getInverse = getInverse)
}
```

cacheSolve

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 the cachesolve should retrieve the inverse from the cache.

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

Testing

```r
m <- matrix(c(2, 4, 5, 3), 2, 2)
matrix(c(2, 4, 5, 3), 2, 2)
```

```
##      [,1] [,2]
## [1,]    2    5
## [2,]    4    3
```

```r
cacheSolve(makeCacheMatrix(m))
```

```
##            [,1]       [,2]
## [1,] -0.2142857  0.3571429
## [2,]  0.2857143 -0.1428571
```