Problem Set 2

Martin Aragoneses

July 23, 2015

Example: Caching the Mean of a Vector

`makeVector` creates a special "vector", which is really a list containing a function to (1) set the value of the vector, (2) get the value of the vector, (3) set the value of the mean and (4) get the value of the mean.

```
makeVector <- function(x = numeric()) {
        m <- NULL                           # mean initialized as NULL
        set <- function(y) {
                x <<- y
                m <<- NULL
        }
        get <- function() x                  # gets x (vector)

        setmean <- function(mean) m <<- mean # sets m to mean
        getmean <- function() m              # gets the mean (stored in m)

        list(set = set, get = get,
             setmean = setmean,
             getmean = getmean)
}
```

The following function calculates the mean of the special "vector" created with the above function. However, it first checks to see if the mean has already been calculated. If so, it gets the mean from the cache and skips the computation. Otherwise, it calculates the mean of the data and sets the value of the mean in the cache via the setmean function.

```
cachemean <- function(x, ...) {
        m <- x$getmean()
        if(!is.null(m)) {
                message("getting cached data")
                return(m)
        }
        data <- x$get()
        m <- mean(data, ...)
        x$setmean(m)
        m
}
```

Assignment: Caching the Inverse of a Matrix

This function creates a special "matrix" object that can cache its inverse. It is really a list containing a function to (1) set the value of the matrix, (2) get the value of the matrix, (3) set the value of the inverse of the matrix and (4) get the value of the inverse.

```
makeCacheMatrix <- function(x = matrix()) {

        inv <- NULL                             # inverse initialized as NULL
        set <- function(y) {
                x <<- y
                inv <<- NULL
        }
        get <- function() x                   # gets x (mtx)

        setinv <- function(inverse) inv <<- inverse # sets m to mean
        getinv <- function() inv               # gets the inverse (stored in inv)

        list(set = set, get = get,
             setinv = setinv,
             getinv = getinv)
}
```

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'

        inv <- x$getinv()
        if(!is.null(inv)) {
                message("getting cached data")
                return(inv)
        }
        data <- x$get()
        inv <- solve(data, ...)
        x$setinv(inv)
        inv
}
```