Assignment

Gopal Nath

June 21, 2017

First Problem

```
##Caching the Inverse of a Matrix:

## Matrix inversion is usually a costly computation and there may be some  benefit to caching the invers

##For this assignment we assume that the matrix supplied is always invertible.

mcm <- 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)
}
```

2nd Problem

```
## This function computes the inverse of the special "matrix" created by
## makeCacheMatrix above. If the inverse has already been calculated (and the
## matrix has not changed), then it should retrieve the inverse from the cache. Computing the inverse of

cacheSolve <- function(x, ...) {
  i <- x$getinverse()
  if (!is.null(i)) {
          message("cached data")
          return(i)
  }
  data <- x$get()
  i <- solve(data, ...)
  x$setinverse(i)
  i
}
```

Testing my function

Code Explanation (Example)

```
A<- matrix(c(5,6,7,8),2,2)
B <- mcm(A)
cacheSolve(B) #inverse returned after computation
```

```
##      [,1] [,2]
## [1,]   -4  3.5
## [2,]    3 -2.5
```

```
cacheSolve(B) #inverse returned from cache
```

```
## cached data
```

```
##      [,1] [,2]
## [1,]   -4  3.5
## [2,]    3 -2.5
```