makeVector <- function(x = numeric()) {

m <- NULL

set <- function(y) {

x <<- y

m <<- NULL

}

get <- function() x

setmean <- function(mean) m <<- mean

getmean <- function() m

list(set = set, get = get,

setmean = setmean,

getmean = getmean)

}

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

}

Note: 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.

<https://github.com/sefakilic/coursera-rprog-assignment2/blob/master/cachematrix.R>

<http://adv-r.had.co.nz/Functions.html>

<https://stackoverflow.com/questions/2628621/how-do-you-use-scoping-assignment-in-r/2630222#2630222?newreg=6c2a38e665644c87afd574fd79340b8c>

<http://www.zhihaoding.com/r/cache>

<http://rstudio-pubs-static.s3.amazonaws.com/56438_c8c1b3a349d84e02996f91b394779b06.html>

<http://masterr.org/r/how-to-cache-a-matrix-inversion-in-r/>

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

inv <- NULL

set <- function(y) {

x <<- y

inv <<- NULL

}

get <- function() x

setinverse <- function(inverse) inv <<- inverse

getinverse <- function() inv

list(set=set, get=get, setinverse=setinverse, getinverse=getinverse)

}

# The following function returns the inverse of the matrix. It first checks if

# the inverse has already been computed. If so, it gets the result and skips the

# computation. If not, it computes the inverse, sets the value in the cache via

# setinverse function.

# This function assumes that the matrix is always invertible.

cacheSolve <- function(x, ...) {

inv <- x$getinverse()

if(!is.null(inv)) {

message("getting cached data.")

return(inv)

}

data <- x$get()

inv <- solve(data)

x$setinverse(inv)

inv

}