Programming Assignment 2

Write the following functions:

1. makeCacheMatrix: This function creates a special “matrix” object that can cache its inverse.
2. 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.

Computing the inverse of a square matrix can be done with the solve function in R. For example, if X is a square invertible matrix, then solve(X) returns its inverse.

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

### Functions

1. CacheMatrix.R:

**# This function holds the cached value or NULL if nothing is cached**

**# initially nothing is cached so set it to NULL**

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

cache <- NULL

**# store a matrix since the matrix is assigned a new value, flush the cache**

setMatrix <- function(newValue) {

x <<- newValue

cache <<- NULL

}

**# Display the stored matrix**

getMatrix <- function() {

x

}

**# cache the given argument**

cacheInverse <- function(solve) {

cache <<- solve

}

**# get the cached value**

getInverse <- function() {

cache

}

# return a list. Each named element of the list is a function

list(setMatrix = setMatrix,

getMatrix = getMatrix,

cacheInverse = cacheInverse,

getInverse = getInverse)

}

1. 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(y, ...) {

**# get the cached value**

inverse <- y$getInverse()

**# if a cached value exists return it**

if(!is.null(inverse)) {

message("getting cached data")

return(inverse)

}

**# otherwise get the matrix, caclulate the inverse and store it in the cache**

data <- y$getMatrix()

inverse <- solve(data)

y$cacheInverse(inverse)

**# return the inverse matrix**

inverse

}

**Testing**

**# create the matrix during the call of makeCacheMatrix()**

> a <- makeCacheMatrix( matrix(c(10,20,30,40), nrow = 2, ncol = 2) )

> a$getMatrix()

[,1] [,2]

[1,] 10 30

[2,] 20 40

> cacheSolve(a)

[,1] [,2]

[1,] -0.2 0.15

[2,] 0.1 -0.05

**# the 2nd time we run the function, we get the cached value**

> cacheSolve(a)

getting cached data

[,1] [,2]

[1,] -0.2 0.15

[2,] 0.1 -0.05

**#the matrix can be created after calling a makeCacheMatrix without arguments.**

> a <- makeCacheMatrix()

> a <- makeCacheMatrix( matrix(c(10,20,30,40), nrow = 2, ncol = 2) )

**#Display the matrix**

> a$getMatrix()

[,1] [,2]

[1,] 10 30

[2,] 20 40

> cacheSolve(a)

[,1] [,2]

[1,] -0.2 0.15

[2,] 0.1 -0.05

**# the 2nd time we run the function, we get the cached value**

> cacheSolve(a)

getting cached data

[,1] [,2]

[1,] -0.2 0.15

[2,] 0.1 -0.05