R Programming Assignment Week 3:

Caching the Inverse of a Matrix

The purpose of this assignment is to write a code speeding up the computation of a matrix inverse by using caching which is a mechanism for calculating objects once, storing them temporarily and reusing them when necessary instead of computing them again.

**A matrix** is a collection of data elements arranged in a two-dimensional with n rows and m columns. The standard R function for matrix inverse is **solve().** The ordinary **inverse i**s defined only for **square matrices.**

First, let us start by **makeCacheMatrix** which creates a special “matrix” object that can cache its inverse.

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

inv <- NULL

set <- **function**(y) {

x <<- y

inv <<- NULL

}

get <- **function**() x

setInverse <- **function**(solveMatrix) inv <<- solveMatrix

getInverse <- **function**() inv

list(set = set, get = get, setInverse = setInverse, getInverse = getInverse)

}

Then, let us computes **cacheSolve()** which will output the inverse of the “matrix” returned by makeCacheMatrix().

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

inv<- x$getinverse()

**if** (!is.null(inv)) {

message("getting cached data")

**return**(inv)

}

mat.data <- x$get()

inv <- solve(mat.data, **...**)

x$setinverse(inv)

inv

}

To test the caching, let us use the following matrix D.

D <- matrix( c(5, 1, 0,3,-1, 2,4, 0,-1), nrow=3, byrow=TRUE)

> D

[,1] [,2] [,3]

[1,] 5 1 0

[2,] 3 -1 2

[3,] 4 0 -1

|  |
| --- |
|  |
| |  | | --- | |  | |

As indicated above, the ordinary inverse is defined only for square matrice. To be invertible, its determinant det(D) has to be different from zero.

> det(D)

[1] 16

det(D) outputs 16 (i.e. det(D)!=0, so inverse exists)

To test our code, we start by computing the matrix D inverse without using the above code

> solve(D)

[,1] [,2] [,3]

[1,] 0.0625 0.0625 0.125

[2,] 0.6875 -0.3125 -0.625

[3,] 0.2500 0.2500 -0.500

Then we apply the code in two steps

> D1 <-makeCacheMatrix(D)

> cacheSolve(D1)

[,1] [,2] [,3]

[1,] 0.0625 0.0625 0.125

[2,] 0.6875 -0.3125 -0.625

[3,] 0.2500 0.2500 -0.500