

R Programming

Comments (-)

Share

Hide Toolbars

Thymios C

22 October 2016

Assignment: Caching the

Matrix inversion is usually a costly computation and there are many ways to compute it repeatedly (there are also alternatives to matrix inversion such as using the inverse of a matrix).

Write the following functions:

makeCacheMatrix: This function creates a special “matrix” object. The special “matrix” returned by *makeCacheMatrix* above has two fields: *set* and *get*. *set* should store the matrix and *get* should retrieve the matrix. *cacheSolve* should retrieve the inverse from the cache. Compute the inverse of the matrix. For example, if *X* is a square invertible matrix, then *solve(X)* returns the inverse of *X*.

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

The following functions are used to create a special object. *makeCacheMatrix* creates a special “matrix”, which is really a list containing

1. set the value of the matrix
2. get the value of the matrix
3. set the value of the inverse
4. get the value of the inverse

```
makeCacheMatrix <- 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)
}
```

This function computes the inverse of the special “matrix” returned by *makeCacheMatrix* above (if the matrix has not changed), then *cacheSolve* should return the inverse of the matrix.

```
cacheSolve <- function(x, ...) {
  i <- x$getinverse()
  if (!is.null(i)) {
```

