Programming Assignment 2: Lexical Scoping

Jerez

11/17/2020

Assignment: Caching the Inverse of a Matrix

Matrix inversion is usually a costly computation and there may be some benefit to caching the inverse of a matrix rather than compute it repeatedly (there are also alternatives to matrix inversion that we will not discuss here). Your assignment is to write a pair of functions that cache the inverse of a matrix.

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.

Solution

The function that we must create, must contain spaces to establish and to obtain the values of the matrix and its inverse.

```
makeCacheMatrix <- function(x = matrix()) {
    inv <- NULL           # Initialize the inverse property

    set <- function(matrix) {  # Function to set the matrix
        x   <<- matrix
        inv <<- NULL}
    get <- function() x        # Return the matrix

    setinverse <- function(inverse) # Function to set inverse of the matrix
    inv <<- inverse
    getinverse <- function() inv    # Return the inverse

    # Outputs
    list(set=set, get=get, setinverse=setinverse, getinverse=getinverse)
}
```

Now, *catchSolve*:

```r
cacheSolve <- function(s, ...) {
    inverse <- s$getInverse() # Return a matrix that is the inverse of 'x'

    if( !is.null(inverse)) { # Just return the inverse if its already set
            message("Getting cached data")
            return(inverse)
    }

    data    <- s$get()              # Get the matrix from our object
    inverse <- solve(data) %*% data # Inverse using matrix multiplication
    s$setInverse(x)                 # Set the inverse to the object
    inverse                         # Return the matrix
}
```