Test Cases for Programming Assignment 2

Stephen Schwartz

Sunday, October 26, 2014

Following are the declarations for the two functions in 'cachematrx.R' and then 3 test cases that demonstrate the functions.

Function Calls:

## this function takes an input matrix, x, and then stores it and it's inverse   
## in two global values for quick calculations later.  
  
makeCacheMatrix <- function(x) {  
   
 ## 1. store a matrix  
 ## 2. store the inverse of the current matrix  
   
 ## store the current matrix for comparison in higher function  
 ## using <<- so that matrixStore can be referred to globally  
 matrixStore <<- x  
 ## calculate the inverse for the current matrix  
 matrixInverseStore <<- solve(matrixStore)   
}  
  
  
## This function takes an input matrix, x, and then checks to see if:  
## 1. Is this matrix invertible? If not, will spit out an error msg.  
## 2. If it is invertible, then it will check to see if it's been previously processed   
## and exists in the cache. If it does, values are pulled from cache.  
## 3. if not, then makeCacheMatrix is called to generate the inverse.  
  
cacheSolve <- function(x) {  
 ### x is a user defined matrix  
 ### returns an inverse of the matrix x if it's invertible, otherwise and error msg  
  
 ## initialize matrixStore if it doesn't exist  
 ## need to figure out nrows and ncols from input matrix to initialize if necessary  
 if (!exists("matrixStore")) {matrixStore <- matrix(c(1),nrow=nrow(x),ncol=ncol(x))}  
   
 ### check to see if it's invertible  
 if (round(det(x),0) == 0) {  
 message("Matrix isn't invertible. Please try again.")  
 } else{   
 ### if it's invertable then continue  
   
 message("Matrix is invertible. Processing continues..")  
 ### check to see if it's already been declared  
 message("Checking...")  
   
 ## check to see if it matches cached matrix  
 if (!all(matrixStore == x)){  
 makeCacheMatrix(x)  
   
 message("First time this matrix was inverted. Inverse calculated and cached values set.")  
 matrixInverse <- matrixInverseStore  
 return(matrixInverse)  
 } else{  
 message("This matrix has been previously processed, retrieving from cache...")  
 matrixInverse <- matrixInverseStore  
 return(matrixInverse)  
 }  
 }  
}

**Below are the test cases:**

*Test case 1* generates a simple matrix 3x3 that IS invertible and then calls the function 2x to demonstrate cache extraction.

myMatrix <- matrix(c(-1,2,3,0,1,0,-1,0,1),   
 nrow=3,  
 ncol=3,  
 byrow=TRUE)  
dimnames(myMatrix) = list(  
 c("row1","row2","row3"),  
 c("col1","col2","col3"))  
  
## first time called, should generate  
cacheSolve(myMatrix)

## Matrix is invertible. Processing continues..  
## Checking...  
## First time this matrix was inverted. Inverse calculated and cached values set.

## row1 row2 row3  
## col1 0.5 -1 -1.5  
## col2 0.0 1 0.0  
## col3 0.5 -1 -0.5

## second time called, should pull from cache  
cacheSolve(myMatrix)

## Matrix is invertible. Processing continues..  
## Checking...  
## This matrix has been previously processed, retrieving from cache...

## row1 row2 row3  
## col1 0.5 -1 -1.5  
## col2 0.0 1 0.0  
## col3 0.5 -1 -0.5

*Test case 2* generates with with a slight modification to myMatrix to see if it re-generates the inverse rather than returning the cache. Note, this matrix IS invertible.

myMatrix <- matrix(c(-1,2,4,0,1,0,-1,0,1),   
 nrow=3,  
 ncol=3,  
 byrow=TRUE)  
dimnames(myMatrix) = list(  
 c("row1","row2","row3"),  
 c("col1","col2","col3"))  
  
## first time called, should generate  
cacheSolve(myMatrix)

## Matrix is invertible. Processing continues..  
## Checking...  
## First time this matrix was inverted. Inverse calculated and cached values set.

## row1 row2 row3  
## col1 0.3333 -0.6667 -1.3333  
## col2 0.0000 1.0000 0.0000  
## col3 0.3333 -0.6667 -0.3333

## second time called, should pull from cache  
cacheSolve(myMatrix)

## Matrix is invertible. Processing continues..  
## Checking...  
## This matrix has been previously processed, retrieving from cache...

## row1 row2 row3  
## col1 0.3333 -0.6667 -1.3333  
## col2 0.0000 1.0000 0.0000  
## col3 0.3333 -0.6667 -0.3333

*Test case 3* generatse a simple matrix 3x3 that is not invertible, which should result in an error msg.

myMatrix <- matrix(c(1,2,3,4,5,6,7,8,9),   
 nrow=3,  
 ncol=3,  
 byrow=TRUE)  
dimnames(myMatrix) = list(  
 c("row1","row2","row3"),  
 c("col1","col2","col3"))  
  
## call this one, should kick out an error msg  
cacheSolve(myMatrix)

## Matrix isn't invertible. Please try again.