**REPOSITORIO PARA SEGUNDA TAREA DE PROGRAMACIÓN**

1. En este ejemplo introducimos el operador <<- que puede utilizarse para asignar un valor a un objeto en un entorno distinto del actual. A continuación se presentan dos funciones que se utilizan para crear un objeto especial que almacena un vector numérico y almacena en caché su media.

La primera función, makeVector crea un "vector" especial, que en realidad es una lista que contiene una función para

1. establecer el valor del vector
2. obtener el valor del vector
3. establecer el valor de la media
4. obtener el valor de la media
5. makeVector <- function(x = numeric()) {
6. m <- NULL
7. set <- function(y) {
8. x <<- y
9. m <<- NULL
10. }
11. get <- function() x
12. setmean <- function(mean) m <<- mean
13. getmean <- function() m
14. list(set = set, get = get,
15. setmean = setmean,
16. getmean = getmean)
17. }

La siguiente función calcula la media del "vector" especial creado con la función anterior. Sin embargo, primero comprueba si ya se ha calculado la media. Si es así, obtiene la media de la caché y se salta el cálculo. En caso contrario, calcula la media de los datos y establece el valor de la media en la caché mediante la función setmean.

cachemean <- function(x, ...) {

        m <- x$getmean()

        if(!is.null(m)) {

                message("getting cached data")

                return(m)

        }

        data <- x$get()

        m <- mean(data, ...)

        x$setmean(m)

        m

}

La inversión de matrices suele ser un cálculo costoso y puede resultar ventajoso almacenar en caché la inversa de una matriz en lugar de calcularla repetidamente (también existen alternativas a la inversión de matrices que no trataremos aquí). Su tarea consiste en escribir un par de funciones que almacenen en caché la inversa de una matriz.

Escriba las siguientes funciones:

1. makeCacheMatrix: Esta función crea un objeto "matriz" especial que puede almacenar en caché su inversa.
2. cacheSolve: Esta función calcula la inversa de la "matriz" especial devuelta por makeCacheMatrix anteriormente. Si la inversa ya ha sido calculada (y la matriz no ha cambiado), entonces cacheSolve debe recuperar la inversa de la caché.

RESPUESTA 1. makeCacheMatrix <- function(x = matrix()) {

inv <- NULL

set <- function(y) {

x <<- y

inv <<- NULL

}

get <- function() x

setInverse <- function(inverse) inv <<- inverse

getInverse <- function() inv

list(set = set,

get = get,

setInverse = setInverse,

getInverse = getInverse)

}

RESPUESTA2.

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

## Return a matrix that is the inverse of 'x'

inv <- x$getInverse()

if (!is.null(inv)) {

message("getting cached data")

return(inv)

}

mat <- x$get()

inv <- solve(mat, ...)

x$setInverse(inv)

inv

}