Caching the Mean of a Vector

Below are two functions that are used to create a

special object that stores a numeric vector and caches its mean.

The first function, `makeVector` creates a special "vector", which is

really a list containing a function to

1. set the value of the vector

2. get the value of the vector

3. set the value of the mean

4. get the value of the mean

<!-- -->

makeVector <- function(x = numeric()) {

m <- NULL

set <- function(y) {

x <<- y

m <<- NULL

}

get <- function() x

setmean <- function(mean) m <<- mean

getmean <- function() m

list(set = set, get = get,

setmean = setmean,

getmean = getmean)

}

The following function calculates the mean of the special "vector"

created with the above function. However, it first checks to see if the

mean has already been calculated. If so, it gets the mean from the

cache and skips the computation. Otherwise, it calculates the mean of

the data and sets the value of the mean in the cache via the setmean function.

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

}

Matrix inversion is usually a costly computation and there may be some

benefit to caching the inverse of a matrix rather than computing it.

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

`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.

For this assignment, assume that the matrix supplied is always

invertible.