1. makeCacheMatrix: This function creates a special “matrix” object that can cache its inverse.

**Answer:**

## A pair of functions that cache the inverse of a matrix.

## This function creates a special "matrix" object that can cache its inverse.

makeCacheMatrix <- function(x = matrix()) {

inv <- NULL

set <- function(y){

x <<- y

inv <<- NULL

}

get <- function() x

setInverse <- function(solveMatrix) inv <<- solveMatrix

getInverse <- function() inv

list(set = set, get = get, setInverse = setInverse, getInverse = getInverse)

}

1. 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.

**Answer:**

## This function computes the inverse of the special "matrix" returned by makeCacheMatrix above.

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)

}

data <- x$get()

inv <- solve(data)

x$setInverse(inv)

inv

}