Lexical Scoping. The project demonstrates how to use lexical scoping and closures in R to implement caching of a matrix inverse. It includes two main functions:

makeCacheMatrix() – creates a special matrix object that can store its inverse.

cacheSolve() – computes the inverse of the matrix, retrieving the cached value when available.

This implementation improves computational efficiency by avoiding repeated calculations of the same matrix inverse.makeCacheMatrix <- function(x = matrix()) {

inv <- NULL # stores the cached inverse

set <- function(y) {

x <<- y # assign new matrix in parent environment

inv <<- NULL # reset cached inverse

}

get <- function() x

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

getInverse <- function() inv

list(set = set, get = get,

setInverse = setInverse,

getInverse = getInverse)

}

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

inv <- x$getInverse()

if(!is.null(inv)) {

message("Getting cached data")

return(inv) # return cached value

}

data <- x$get()

inv <- solve(data, ...) # compute inverse

x$setInverse(inv) # store in cache

inv

}

m <- matrix(c(1,2,3,4), 2, 2)

cm <- makeCacheMatrix(m)

cacheSolve(cm) # First time → calculates inverse

cacheSolve(cm) # Second time → retrieves cached value