DATA605_w2_ Matrix Manipulation

David Simbandumwe

```
library(matlib)
```

Problem set 1

(1) Show that $A^TA \neq AA^T$ in general. (Proof and demonstration.)

Proof

$A^TA \neq AA^T$
if A is not symetric then $A \neq A^T$: Definition TM (Transpose of a Matrix)
we know that $AB \neq BA$: Definition MMNC (Matrix multiplication is not commutative)
therefore $A^TA \neq AA^T$: Definition MMNC (Matrix multiplication is not commutative)

Example

A^TA - Multiply a matrix by its transposed matrix generates the following equation

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \times \begin{bmatrix} a11 & a21 \\ a12 & a22 \end{bmatrix} =$$

$$\begin{bmatrix} a11 * a11 + a12 * a12 & a11 * a21 + a12 * a22 \\ a21 * a11 + a22 * a12 & a21 * a21 + a22 * a22 \end{bmatrix}$$

AA^T **- Multiply a transposed matrix by the original matrix generates the following equation**

$$\begin{bmatrix} a11 & a21 \\ a12 & a22 \end{bmatrix} \times \begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} =$$

$$\begin{bmatrix} a11 * a11 + a21 * a21 & a11 * a12 + a21 * a22 \\ a12 * a11 + a22 * a21 & a12 * a12 + a22 * a22 \end{bmatrix}$$

As you can see the solution to the 2 equations are not equal

The following example illistrates the equation above

```
A <- matrix(c(2,1,4, 1,2,1, 1,1,2),3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    2    1    1
## [2,]    1    2    1
## [3,]    4    1    2
```

```
t(A)
```

```
##      [,1] [,2] [,3]
## [1,]    2    1    4
## [2,]    1    2    1
## [3,]    1    1    2
```

```
A%*%t(A)
```

```
##      [,1] [,2] [,3]
## [1,]    6    5   11
## [2,]    5    6    8
## [3,]   11    8   21
```

```
t(A) %*% A
```

```
##      [,1] [,2] [,3]
## [1,]   21    8   11
## [2,]    8    6    5
## [3,]   11    5    6
```

(2) For a special type of square matrix A, we get AT A = AAT . Under what conditions could this be true? (Hint: The Identity matrix I is an example of such a matrix). Please typeset your response using LaTeX mode in RStudio. If you do it in paper, please either scan or take a picture of the work and submit it. Please ensure that your image is legible and that your submissions are named using your first initial, last name, assignment and problem set within the assignment. E.g. LFulton_Assignment2_PS1.png

$A^TA = AA^T$ holds for all normal matrices. If a matrix and its transposed matrix are identical then $A^TA = AA^T$ holds

Proof

$A^TA = AA^T$
if A is symetric then $A = A^T$: Definition TM (Transpose of a Matrix)
then we can write $AA = AA$: Substitution
therefore $A^TA = AA^T$: Definition MM (Matrix Multiplication)

Example

The example below illistrates the proof abov

```
A <- matrix(c(2,1,1, 1,2,1, 1,1,2),3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    2    1    1
## [2,]    1    2    1
## [3,]    1    1    2
```

```
t(A)
```

```
##      [,1] [,2] [,3]
## [1,]    2    1    1
## [2,]    1    2    1
## [3,]    1    1    2
```

```
A%*%t(A)
```

```
##      [,1] [,2] [,3]
## [1,]    6    5    5
## [2,]    5    6    5
## [3,]    5    5    6
```

```
t(A) %*% A
```

```
##      [,1] [,2] [,3]
## [1,]    6    5    5
## [2,]    5    6    5
## [3,]    5    5    6
```

Problem set 2

(1) Matrix factorization is a very important problem. There are supercomputers built just to do matrix factorizations. Every second you are on an airplane, matrices are being factorized. Radars that track flights use a technique called Kalman filtering. At the heart of Kalman Filtering is a Matrix Factorization operation. Kalman Filters are solving linear systems of equations when they track your flight using radars. Write an R function to factorize a square matrix A into LU or LDU, whichever you prefer. Please submit your response in an R Markdown document using our class naming convention, E.g. LFulton_Assignment2_PS2.png You don't have to worry about permuting rows of A and you can assume that A is less than 5x5, if you need to hard-code any variables in your code. If you doing the entire assignment in R, then please submit only one markdown document for both the problems.

```
#' Calculates the list of Elimination Matrices for a given Matrix.
#'
#' @param A A matrix
#' @return The an array with list of Elimination Matrices.
calcEliminationMatrixArray <- function(A) {

    size_i <- dim(A)[1]
    size_j <- dim(A)[2] - 1
    n <- size_i * (size_i - 1) / 2

    U <- diag(size_i)

    m_lst <- list()
```

```
    m_arry <- array(dim = c(size_i,size_i,n))

    x <- 1

    for (j in 1:size_j) {
        for (i in 2:size_i) {
            if ( i != j) {

                U <- diag(size_i)
                x1 = 1
                while (x1 < x) {
                    U <- U %*% m_arry[,,x1]
                    x1 <- x1 + 1
                }

                U <- U %*% A
                I <- diag(size_i)
                #I[i,j] = -1 * U[i,j] * sign(U[j,j])
                I[i,j] = -1 * U[i,j] / U[j,j]

                m_arry[,,x] <- I
                x <- x+1
            }
        }
    }
    return(m_arry)
}


#' Calculates the Upper Triangular matrix from a list of Elimination Matrices.
#'
#' @param m_arry An array of Elimination Matricies
#' @param size The size of the square Matricies
#' @return The Upper Triangular matrix.
calcU <- function(m_arry,size) {

    x <- dim(m_arry)[3]
    U <- diag(size)
    for (i in x:1) {
        U <- U %*% m_arry[,,i]
    }
    U <- U %*% A
    return(U)
}
```

```
#' Calculates the Lower Triangular matrix from a list of Elimination Matrices.
#'
#' @param m_arry An array of Elimination Matricies
#' @param size The size of the square Matricies
#' @return The Lower Triangular matrix.
calcL <- function(m_arry, size) {

    x <- dim(m_arry)[3]
    L <- diag(size)
    for (i in 1:x) {
        L <- L %*% solve(m_arry[,,i])
    }
    return(L)
}
```

```
A <- matrix(c(1,2,3,1,1,1,2,0,1),nrow=3)
size <- dim(A)[1]


m_arry <- calcEliminationMatrixArray(A)
U <- calcU(m_arry,size)
print(U)
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    2
## [2,]    0   -1   -4
## [3,]    0    0    3
```

```
L <- calcL(m_arry, size)
print(L)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    2    1    0
## [3,]    3    2    1
```

```
print(L%*%U == A)
```

```
##      [,1] [,2] [,3]
## [1,] TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE
```

```
A <-matrix(c(3,-3,6, -7,5,-4, -2,1,0),nrow=3)
size <- dim(A)[1]



m_arry <- calcEliminationMatrixArray(A)
U <- calcU(m_arry,size)
print(U)
```

```
##      [,1] [,2] [,3]
## [1,]    3   -7   -2
## [2,]    0   -2   -1
## [3,]    0    0   -1
```

```
L <- calcL(m_arry, size)
print(L)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]   -1    1    0
## [3,]    2   -5    1
```

```
print(L%*%U == A)
```

```
##      [,1] [,2] [,3]
## [1,] TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE
```

```
A <-matrix(c(2,6,1,8),nrow=2)
size <- dim(A)[1]



m_arry <- calcEliminationMatrixArray(A)
U <- calcU(m_arry,size)
print(U)
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    0    5
```

```
L <- calcL(m_arry, size)
print(L)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    3    1
```

```
print(L%*%U == A)
```

```
##      [,1] [,2]
## [1,] TRUE TRUE
## [2,] TRUE TRUE
```