605-HW02-Matrix Factorization

Michael Y.

September 8, 2019

Contents

Setup

```r
knitr::opts_chunk$set(echo = TRUE)
directory = "C:/Users/Michael/Dropbox/priv/CUNY/MSDS/201909-Fall/DATA605_Larry/20190908_Week02/"
knitr::opts_knit$set(root.dir = directory)
knitr::opts_knit$set(digits=6)

### Make the output wide enough
options(scipen = 999, digits=6, width=150)

### Load some libraries
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
library(pracma)
```

Function to write Matrix, courtesy of Vinayak Patel :

```
writeMatrix <- function(x) {
  begin <- "\\begin{bmatrix}"
  end   <- "\\end{bmatrix}"
  X     <-    apply(x, 1, function(x) {
      paste(
          paste(x, collapse = "&"),
          "\\\\"
      )
    }
  )
  paste(c(begin, X, end),
        collapse = "")
  }
```

Function to write numerical Matrix, controlling decimals, adapted :

```
wnumMatrix <- function(x) {
  begin <- "\\begin{bmatrix}"
  end   <- "\\end{bmatrix}"
  X     <-    apply(x, 1, function(x) {
      paste(
          paste(format(x, digits = options()$digits), collapse = "&"),
          "\\\\"
      )
    }
  )
  paste(c(begin, X, end),
        collapse = "")
  }
```

Part 1 - (non)Commutivity of Matrix Transposes

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

```
##      [,1]      [,2]      [,3]      [,4]
## [1,] "A_{1,1}" "A_{1,2}" "A_{1,j}" "A_{1,n}"
```

```
## [2,] "A_{2,1}" "A_{2,2}" "A_{2,j}" "A_{2,n}"
## [3,] "A_{i,1}" "A_{i,2}" "A_{i,j}" "A_{i,n}"
## [4,] "A_{n,1}" "A_{n,2}" "A_{n,j}" "A_{n,n}"

##      [,1]       [,2]       [,3]       [,4]
## [1,] "A_{1,1}" "A_{2,1}" "A_{i,1}" "A_{n,1}"
## [2,] "A_{1,2}" "A_{2,2}" "A_{i,2}" "A_{n,2}"
## [3,] "A_{1,j}" "A_{2,j}" "A_{i,j}" "A_{n,j}"
## [4,] "A_{1,n}" "A_{2,n}" "A_{i,n}" "A_{n,n}"
```

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,j} & A_{1,n} \\ A_{2,1} & A_{2,2} & A_{2,j} & A_{2,n} \\ A_{i,1} & A_{i,2} & A_{i,j} & A_{i,n} \\ A_{n,1} & A_{n,2} & A_{n,j} & A_{n,n} \end{bmatrix}$$

$$A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{i,1} & A_{n,1} \\ A_{1,2} & A_{2,2} & A_{i,2} & A_{n,2} \\ A_{1,j} & A_{2,j} & A_{i,j} & A_{n,j} \\ A_{1,n} & A_{2,n} & A_{i,n} & A_{n,n} \end{bmatrix}$$

$$AA^T = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,j} & A_{1,n} \\ A_{2,1} & A_{2,2} & A_{2,j} & A_{2,n} \\ A_{i,1} & A_{i,2} & A_{i,j} & A_{i,n} \\ A_{n,1} & A_{n,2} & A_{n,j} & A_{n,n} \end{bmatrix} \begin{bmatrix} A_{1,1} & A_{2,1} & A_{i,1} & A_{n,1} \\ A_{1,2} & A_{2,2} & A_{i,2} & A_{n,2} \\ A_{1,j} & A_{2,j} & A_{i,j} & A_{n,j} \\ A_{1,n} & A_{2,n} & A_{i,n} & A_{n,n} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} A_{1,j}^2 & \sum_{j=1}^{n} A_{1,j}A_{2,j} & \dots & \sum_{j=1}^{n} A_{1,j}A_{n,j} \\ \sum_{j=1}^{n} A_{1,j}A_{2,j} & \sum_{j=1}^{n} A_{2,j}^2 & \dots & \sum_{j=1}^{n} A_{2,j}A_{n,j} \\ \dots & \dots & \dots & \dots \\ \sum_{j=1}^{n} A_{1,j}A_{n,j} & \sum_{j=1}^{n} A_{2,j}A_{n,j} & \dots & \sum_{j=1}^{n} A_{n,j}^2 \end{bmatrix}$$

$$A^TA = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{i,1} & A_{n,1} \\ A_{1,2} & A_{2,2} & A_{i,2} & A_{n,2} \\ A_{1,j} & A_{2,j} & A_{i,j} & A_{n,j} \\ A_{1,n} & A_{2,n} & A_{i,n} & A_{n,n} \end{bmatrix} \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,j} & A_{1,n} \\ A_{2,1} & A_{2,2} & A_{2,j} & A_{2,n} \\ A_{i,1} & A_{i,2} & A_{i,j} & A_{i,n} \\ A_{n,1} & A_{n,2} & A_{n,j} & A_{n,n} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} A_{i,1}^2 & \sum_{i=1}^{n} A_{i,1}A_{i,2} & \dots & \sum_{i=1}^{n} A_{i,1}A_{i,n} \\ \sum_{i=1}^{n} A_{i,1}A_{i,2} & \sum_{i=1}^{n} A_{i,2}^2 & \dots & \sum_{i=1}^{n} A_{i,2}A_{i,n} \\ \dots & \dots & \dots & \dots \\ \sum_{i=1}^{n} A_{i,1}A_{i,n} & \sum_{i=1}^{n} A_{i,2}A_{i,n} & \dots & \sum_{i=1}^{n} A_{i,n}^2 \end{bmatrix}$$

For the above to be true, we would need to have the above summations to be elementwise equal between the top and bottom matrices. This is usually not the case.

COUNTEREXAMPLE:

```
Q = c(1,2,3,6,4,5,7,8,9)
Q=matrix(Q,3,3,T)
Q
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    6    4    5
## [3,]    7    8    9
```

```
# Transpose
QT = t(Q)
QT
```

```
##      [,1] [,2] [,3]
## [1,]    1    6    7
## [2,]    2    4    8
## [3,]    3    5    9
```

```
# Q * QT
QQT = Q %*% QT
QQT
```

```
##      [,1] [,2] [,3]
## [1,]   14   29   50
## [2,]   29   77  119
## [3,]   50  119  194
```

```
# QT * Q
QTQ = QT %*% Q
QTQ
```

```
##      [,1] [,2] [,3]
## [1,]   86   82   96
## [2,]   82   84   98
## [3,]   96   98  115
```

```r
# Are the entries equal?
QQT == QTQ
```

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

As a simple counterexample, consider:

$$Q = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}; Q^T = \begin{bmatrix} 1 & 6 & 7 \\ 2 & 4 & 8 \\ 3 & 5 & 9 \end{bmatrix}$$

$$QQ^T = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 6 & 7 \\ 2 & 4 & 8 \\ 3 & 5 & 9 \end{bmatrix} = \begin{bmatrix} 14 & 29 & 50 \\ 29 & 77 & 119 \\ 50 & 119 & 194 \end{bmatrix}$$

$$Q^TQ = \begin{bmatrix} 1 & 6 & 7 \\ 2 & 4 & 8 \\ 3 & 5 & 9 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 86 & 82 & 96 \\ 82 & 84 & 98 \\ 96 & 98 & 115 \end{bmatrix}$$

Clearly, $QQ^T \neq Q^TQ$.

---

(2) For a special type of square matrix, $A^TA = AA^T$

Under what conditions could this be true? (Hint: The Identity matrix I is an example of such a matrix).

This is true for **Permutation** matrices because

$$(PP^T)_{ij} = \sum_{k=1}^{n} P_{ik}P_{kj}^T = \sum_{k=1}^{n} P_{ik}P_{jk}$$

and

$$\sum_{k=1}^{n} P_{ik} P_{jk} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

so

$$PP^T = I$$

.

PERMUTATION EXAMPLE:

```
P = c(
0,1,0,0,0,
0,0,0,1,0,
1,0,0,0,0,
0,0,1,0,0,
0,0,0,0,1
)
P=matrix(P,5,5,T)
P
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    0    0    0
## [2,]    0    0    0    1    0
## [3,]    1    0    0    0    0
## [4,]    0    0    1    0    0
## [5,]    0    0    0    0    1
```

```
PT=t(P)
PT
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    1    0    0
## [2,]    1    0    0    0    0
## [3,]    0    0    0    1    0
## [4,]    0    1    0    0    0
## [5,]    0    0    0    0    1
```

```
PPT = P %*% PT
PPT
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

```
PTP = PT %*% P
PTP
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

```
PPT==PTP
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

For example, consider:

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}; P^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$PP^T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = I_5$$

$$P^TP = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = I_5$$

Obviously commutativity is true for **symmetric** matrices because, by definition, $A = A^T$, so $AA^T = AA = A^TA$.

SYMMETRIC EXAMPLE:

```
S = c(
  1,2,4,
  2,3,5,
  4,5,6
  )
S = matrix(S,3,3,T)
S
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    4
## [2,]    2    3    5
## [3,]    4    5    6
```

```
ST=t(S)
ST
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    4
## [2,]    2    3    5
## [3,]    4    5    6
```

```
SST = S %*% ST
SST
```

```
##      [,1] [,2] [,3]
## [1,]   21   28   38
## [2,]   28   38   53
## [3,]   38   53   77
```

```
STS = ST %*% S
STS
```

```
##      [,1] [,2] [,3]
## [1,]   21   28   38
## [2,]   28   38   53
## [3,]   38   53   77
```

```
SST == STS
```

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

For example, consider:

$$S = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix}; S^T = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix}$$

$$SS^T = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 21 & 28 & 38 \\ 28 & 38 & 53 \\ 38 & 53 & 77 \end{bmatrix}$$

$$S^TS = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 21 & 28 & 38 \\ 28 & 38 & 53 \\ 38 & 53 & 77 \end{bmatrix}$$

Also this is true for **orthogonal** matrices because, by definition, $A^T = A^{-1}det(A)$,
so

$$AA^T = AA^{-1} * det(A) = I * det(A) = A^{-1}A * det(A) = A^T A$$

.

ORTHOGONAL EXAMPLE:

```
X = c(
  1,-1,
  1,1
  )
X=matrix(X,2,2,T)
X
```

```
##      [,1] [,2]
## [1,]    1   -1
## [2,]    1    1
```

```
# Compute transpose of X
XT=t(X)
XT
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]   -1    1
```

```
# Compute inverse of X
Xinv = inv(X)
Xinv
```

```
##      [,1] [,2]
## [1,]  0.5  0.5
## [2,] -0.5  0.5
```

```r
# Compute determinant of X
Xdet = det(X)
Xdet
```

```
## [1] 2
```

```r
# Compute XXT
XXT = X %*% XT
XXT
```

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

```r
# Compute XTX
XTX = XT %*% X
XTX
```

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

```r
# Do the entries match?
XXT == XTX
```

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

For example, consider **orthogonal** matrix $X = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ which has transpose $X^T = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$. Here,

$$XX^T = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$X^TX = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Note that we can make the above matrix **orthonormal** by dividing it by the square root of its determinant, which here is 2.

ORTHONORMAL EXAMPLE:

```
# make orthonormal
O = X / sqrt(det(X))
O
```

```
##           [,1]      [,2]
## [1,] 0.707107 -0.707107
## [2,] 0.707107  0.707107
```

```
# compute transpose
OT=t(O)
OT
```

```
##           [,1]     [,2]
## [1,]  0.707107 0.707107
## [2,] -0.707107 0.707107
```

```
# Compute inverse
Oinv = inv(O)
Oinv
```

```
##           [,1]     [,2]
## [1,]  0.707107 0.707107
## [2,] -0.707107 0.707107
```

```
# does transpose "equal" inverse (up to machine precision?)
OT - Oinv
```

```
##                           [,1]                       [,2]
## [1,] -0.00000000000000000111022 -0.00000000000000000111022
## [2,]  0.00000000000000000111022 -0.00000000000000000111022
```

```
abs(OT-Oinv) < 1e-12
```

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

```
# Compute OOT
OOT = O %*% OT
OOT
```

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

```
# Compute OTO
OTO = OT %*% O
OTO
```

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

```
# do the entries match?
OOT == OTO
```

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

For example, consider **orthonormal** matrix $O = \begin{bmatrix} 0.707107 & -0.707107 \\ 0.707107 & 0.707107 \end{bmatrix}$ which has transpose $O^T = \begin{bmatrix} 0.707107 & 0.707107 \\ -0.707107 & 0.707107 \end{bmatrix}$.

Here,

$$OO^T = \begin{bmatrix} 0.707107 & -0.707107 \\ 0.707107 & 0.707107 \end{bmatrix} \begin{bmatrix} 0.707107 & 0.707107 \\ -0.707107 & 0.707107 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2$$

$$O^TO = \begin{bmatrix} 0.707107 & 0.707107 \\ -0.707107 & 0.707107 \end{bmatrix} \begin{bmatrix} 0.707107 & -0.707107 \\ 0.707107 & 0.707107 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2$$

However, the above items do not include **all** matrices for which the transpose commutes under multiplication.

By the **Spectral Theorem**, the full set of matrices for which the transpose commutes are called **Normal** Matrices. These are discussed in the text book in sections NM and OD (pages 574-580.)

By **Theorem OD (Orthonormal Diagonalization)** on page 575, we have:

Suppose that A is a square matrix. Then there is a **unitary** matrix U and a **diagonal** matrix D, *with diagonal entries equal to the eigenvalues of* A, such that $U^T AU = D$ if and only if A is a **normal** matrix.

(Note: for matrices with **all real** entries, **unitary** is synonymous with **orthonormal** , and above I have used U^T to indicate **transpose**, which for real matrices is the same as the **adjoint.** The distinction only comes into play in the case of matrices with **complex** entries.)

NORMAL example (non-symmetric, etc.):

```
Z = c(
  1,1,0,
  0,1,1,
  1,0,1)
Z = matrix(Z,3,3,T)
Z
```

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

```
# Compute transpose
ZT=t(Z)
ZT
```

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

```
# Compute ZZT
ZZT = Z %*% ZT
ZZT
```

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

```
# Compute ZTZ
ZTZ = ZT %*% Z
ZTZ
```

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

```
# Are they equal?
ZZT==ZTZ
```

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

For example, consider a **normal** matrix which is neither symmetric nor orthogonal:

$$Z = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}; Z^T = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

$$ZZ^T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

$$Z^T Z = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

Part 2 - Matrix Decomposition - LU

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.Rmd 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.

MYLU function:

```
library(pracma)                       # to get functions like eye(), zeros(), ones()
debug=FALSE

MYLU = function (A) {

  n <- nrow(A);                       # how many rows (and, columns) are in the (square) matrix?
  L <- eye(n)                         # start L from the identity matrix
  U <- eye(n)                         # start U from the identity matrix

  for (k in 1:n) {
    if(debug) print(paste0("for k: ",k))
    U[k,k] <- A[k,k];
    for (i in (k+1):n) {
      if (i>n) break
      if(debug) print(paste0("(k,i): (", k,",",i,")"))
      L[i,k] <- (A[i,k])/(U[k,k]);
      U[k,i] <-  A[k,i]
    }
    for (i in ((k+1):n)) {
      if (i>n) break
```

```
      if(debug) print(paste0("(k,i): (", k,",",i,")"))
      for (j in ((k+1):n)) {
        if (j>n) break
        if(debug) print(paste0("(k,i,j): (", k,",",i,",",j,")"))
        A[i,j] <- A[i,j] - L[i,k] %*% U[k,j];
      }
    }
  }

  myresult <- list(L,U)
  names(myresult) <- c("L","U")

  return(myresult)

} # end function MYLU
```

Test a constructed 3x3 matrix

```
# Create an upper triangular matrix
testU1 <- c(
  1,1,1,
  0,1,1,
  0,0,1)
testU1 <- matrix(testU1,3,3,T)
testU1
```

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

```
det(testU1)
```

```
## [1] 1
```

```r
inv(testU1)
```

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

```r
#Create a lower triangular matrix
testL1 <- c(
  1,0,0,
  1,1,0,
  1,1,1
  )
testL1 <- matrix(testL1,3,3,T)
testL1
```

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

```r
det(testL1)
```

```
## [1] 1
```

```r
inv(testL1)
```

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

```r
# Multiply together to get a matrix for testing
testLU1 <- testL1 %*% testU1
testLU1
```

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

Testing matrix $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}$:

Get the result using my LU algorithm

```
myresult1 <- MYLU(testLU1)
myresult1
```

```
## $L
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    1    0
## [3,]    1    1    1
##
## $U
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    0    1    1
## [3,]    0    0    1
```

My result is

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}; U = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}; LU = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

```
# do my L and U match original L and U ?
myresult1$L == testL1
```

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

```
myresult1$U == testU1
```

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

Check using lu decomposition function from pracma

```
library(pracma)
theirresult1 <- pracma::lu(testLU1)
theirresult1
```

```
## $L
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    1    0
## [3,]    1    1    1
##
## $U
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    0    1    1
## [3,]    0    0    1
```

```
myresult1
```

```
## $L
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    1    0
## [3,]    1    1    1
##
## $U
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    0    1    1
## [3,]    0    0    1
```

Pracma result is

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}; U = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}; LU = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

```
# Does my result match their result?
myresult1$L == theirresult1$L
```

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

```
myresult1$U == theirresult1$U
```

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

Test a constructed 5x5 matrix

```
# Create an upper triangular matrix
testU2 <- c(
  1,2,3,4,5,
  0,2,3,4,5,
  0,0,3,4,5,
  0,0,0,4,5,
  0,0,0,0,5
  )
testU2 <- matrix(testU2,5,5,T)
testU2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    0    2    3    4    5
```

```
## [3,]    0    0    3    4    5
## [4,]    0    0    0    4    5
## [5,]    0    0    0    0    5
```

```
det(testU2)
```

```
## [1] 120
```

```
inv(testU2)
```

```
##      [,1] [,2]      [,3]      [,4]  [,5]
## [1,]    1 -1.0  0.000000  0.000000  0.00
## [2,]    0  0.5 -0.500000  0.000000  0.00
## [3,]    0  0.0  0.333333 -0.333333  0.00
## [4,]    0  0.0  0.000000  0.250000 -0.25
## [5,]    0  0.0  0.000000  0.000000  0.20
```

```
#Create a lower triangular matrix
testL2 <- c(
  1,0,0,0,0,
  1,1,0,0,0,
  1,1,1,0,0,
  1,1,1,1,0,
  1,1,1,1,1
  )
testL2 <- matrix(testL2,5,5,T)
testL2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    1    1    0    0    0
## [3,]    1    1    1    0    0
## [4,]    1    1    1    1    0
## [5,]    1    1    1    1    1
```

```r
det(testL2)
```

```
## [1] 1
```

```r
inv(testL2)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]   -1    1    0    0    0
## [3,]    0   -1    1    0    0
## [4,]    0    0   -1    1    0
## [5,]    0    0    0   -1    1
```

```r
# Multiply together to get a matrix for testing
testLU2 <- testL2 %*% testU2
testLU2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    1    4    6    8   10
## [3,]    1    4    9   12   15
## [4,]    1    4    9   16   20
## [5,]    1    4    9   16   25
```

Testing matrix $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 6 & 8 & 10 \\ 1 & 4 & 9 & 12 & 15 \\ 1 & 4 & 9 & 16 & 20 \\ 1 & 4 & 9 & 16 & 25 \end{bmatrix}$:

Get the result using my LU algorithm

```r
# Get the result using my LU algorithm
myresult2 <- MYLU(testLU2)
myresult2
```

```
## $L
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    1    1    0    0    0
## [3,]    1    1    1    0    0
## [4,]    1    1    1    1    0
## [5,]    1    1    1    1    1
##
## $U
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    0    2    3    4    5
## [3,]    0    0    3    4    5
## [4,]    0    0    0    4    5
## [5,]    0    0    0    0    5
```

My result is

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}; U = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 3 & 4 & 5 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix}; LU = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 6 & 8 & 10 \\ 1 & 4 & 9 & 12 & 15 \\ 1 & 4 & 9 & 16 & 20 \\ 1 & 4 & 9 & 16 & 25 \end{bmatrix}$$

```
# do my L and U match original L and U ?
myresult2$L == testL2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

```
myresult2$U == testU2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
```

```
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

```
# Check using lu decomposition function from pracma
library(pracma)
theirresult2 <- pracma::lu(testLU2)
theirresult2
```

```
## $L
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    1    1    0    0    0
## [3,]    1    1    1    0    0
## [4,]    1    1    1    1    0
## [5,]    1    1    1    1    1
##
## $U
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    0    2    3    4    5
## [3,]    0    0    3    4    5
## [4,]    0    0    0    4    5
## [5,]    0    0    0    0    5
```

Pracma result is

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}; U = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 3 & 4 & 5 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix}; LU = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 6 & 8 & 10 \\ 1 & 4 & 9 & 12 & 15 \\ 1 & 4 & 9 & 16 & 20 \\ 1 & 4 & 9 & 16 & 25 \end{bmatrix}$$

```
# Does my result match their result?
myresult2$L == theirresult2$L
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
```

```
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

```
myresult2$U == theirresult2$U
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

Test a different 5x5 matrix

```
# Create an upper triangular matrix
testU3 <- c(
  9,3,2,4,1,
  0,6,9,-3,5,
  0,0,4,2,-1,
  0,0,0,7,5,
  0,0,0,0,-12
  )
testU3 <- matrix(testU3,5,5,T)
testU3
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    9    3    2    4    1
## [2,]    0    6    9   -3    5
## [3,]    0    0    4    2   -1
## [4,]    0    0    0    7    5
## [5,]    0    0    0    0  -12
```

```
det(testU3)
```

```
## [1] -18144
```

```
inv(testU3)
```

```
##           [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 0.111111 -0.0555556  0.0694444 -0.1071429 -0.0643188
## [2,] 0.000000  0.1666667 -0.3750000  0.1785714  0.1750992
## [3,] 0.000000  0.0000000  0.2500000 -0.0714286 -0.0505952
## [4,] 0.000000  0.0000000  0.0000000  0.1428571  0.0595238
## [5,] 0.000000  0.0000000  0.0000000  0.0000000 -0.0833333
```

```
#Create a lower triangular matrix
testL3 <- c(
  1,0,0,0,0,
  -1,1,0,0,0,
  99,5,1,0,0,
  101,22,-66,1,0,
  77,-55,33,-22,1
  )
testL3 <- matrix(testL3,5,5,T)
testL3
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]   -1    1    0    0    0
## [3,]   99    5    1    0    0
## [4,]  101   22  -66    1    0
## [5,]   77  -55   33  -22    1
```

```
det(testL3)
```

```
## [1] 1
```

```
inv(testL3)
```

```
##         [,1]                      [,2]                      [,3]                      [,4] [,5]
## [1,]       1  -0.000000000000000025678   0.000000000000000267952   0.000000000000000102878    0
## [2,]       1   0.999999999999962252    -0.000000000000003167987  -0.000000000000000614500    0
```

```
## [3,]    -104    -4.9999999999999831246    0.99999999999999559108 -0.00000000000000000955764    0
## [4,]   -6987  -351.9999999999985220711   65.99999999999744204615  0.99999999999999940047957    0
## [5,] -150304 -7523.9999999999681676854 1418.99999999999994315658114 21.99999999999998685495939    1
```

```
# Multiply together to get a matrix for testing
testLU3 <- testL3 %*% testU3
testLU3
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    9    3    2    4    1
## [2,]   -9    3    7   -7    4
## [3,]  891  327  247  383  123
## [4,]  909  435  136  213  282
## [5,]  693  -99 -209  385 -353
```

Testing matrix $\begin{bmatrix} 9 & 3 & 2 & 4 & 1 \\ -9 & 3 & 7 & -7 & 4 \\ 891 & 327 & 247 & 383 & 123 \\ 909 & 435 & 136 & 213 & 282 \\ 693 & -99 & -209 & 385 & -353 \end{bmatrix}$:

Get the result using my LU algorithm

```
# Get the result using my LU algorithm
myresult3 <- MYLU(testLU3)
myresult3
```

```
## $L
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]   -1    1    0    0    0
## [3,]   99    5    1    0    0
## [4,]  101   22  -66    1    0
## [5,]   77  -55   33  -22    1
##
## $U
##      [,1] [,2] [,3] [,4] [,5]
```

```
## [1,]    9    3    2    4    1
## [2,]    0    6    9   -3    5
## [3,]    0    0    4    2   -1
## [4,]    0    0    0    7    5
## [5,]    0    0    0    0  -12
```

My result is

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 99 & 5 & 1 & 0 & 0 \\ 101 & 22 & -66 & 1 & 0 \\ 77 & -55 & 33 & -22 & 1 \end{bmatrix}; U = \begin{bmatrix} 9 & 3 & 2 & 4 & 1 \\ 0 & 6 & 9 & -3 & 5 \\ 0 & 0 & 4 & 2 & -1 \\ 0 & 0 & 0 & 7 & 5 \\ 0 & 0 & 0 & 0 & -12 \end{bmatrix}; LU = \begin{bmatrix} 9 & 3 & 2 & 4 & 1 \\ -9 & 3 & 7 & -7 & 4 \\ 891 & 327 & 247 & 383 & 123 \\ 909 & 435 & 136 & 213 & 282 \\ 693 & -99 & -209 & 385 & -353 \end{bmatrix}$$

```
# do my L and U match original L and U ?
myresult3$L == testL3
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

```
myresult3$U == testU3
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

```
# Check using lu decomposition function from pracma
library(pracma)
theirresult3 <- pracma::lu(testLU3)
theirresult3
```

```
## $L
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]   -1    1    0    0    0
## [3,]   99    5    1    0    0
## [4,]  101   22  -66    1    0
## [5,]   77  -55   33  -22    1
##
## $U
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    9    3    2    4    1
## [2,]    0    6    9   -3    5
## [3,]    0    0    4    2   -1
## [4,]    0    0    0    7    5
## [5,]    0    0    0    0  -12
```

Pracma result is

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 99 & 5 & 1 & 0 & 0 \\ 101 & 22 & -66 & 1 & 0 \\ 77 & -55 & 33 & -22 & 1 \end{bmatrix}; U = \begin{bmatrix} 9 & 3 & 2 & 4 & 1 \\ 0 & 6 & 9 & -3 & 5 \\ 0 & 0 & 4 & 2 & -1 \\ 0 & 0 & 0 & 7 & 5 \\ 0 & 0 & 0 & 0 & -12 \end{bmatrix}; LU = \begin{bmatrix} 9 & 3 & 2 & 4 & 1 \\ -9 & 3 & 7 & -7 & 4 \\ 891 & 327 & 247 & 383 & 123 \\ 909 & 435 & 136 & 213 & 282 \\ 693 & -99 & -209 & 385 & -353 \end{bmatrix}$$

```
# Does my result match their result?
myresult3$L == theirresult3$L
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

```
myresult3$U == theirresult3$U
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
```

```
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
## [5,] TRUE TRUE TRUE TRUE TRUE
```

In each of the examples tested, MYLU gives the same results as the LU function from the pracma package.

Such results successfully decompose the tested matrix back into the L and U constituents from which it was constructed.