D605 HW2 - Keith Colella

2023-08-25

```
library(matrixcalc)
set.seed(42)
```

Problem Set 1

Problem 1

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

Response 1

In general, we know that matrix multiplication is not commutative. We can look at Definition MM and Example MMNC from [1, Beezer] to confirm that $AB \neq BA$. Moreover, we know that, in general, $A \neq A^T$; we can again confirm this in [1, Beezer] with Definition TM and Example TM. It therefore follows that $A^TA \neq AA^T$.

We can also highlight dimensionality as proof of incongruence. For non-square matrices, the products of A^TA and AA^T will have different dimensions. For example, if M is a 3x4 matrix, then A^TA will result in a 4x4 matrix, whereas AA^T will result in a 3x3. In these cases, A^TA will clearly not equal AA^T.

We can confirm this point further with a few examples.

Example 1

```
sample_matrix <- round(matrix(runif(9,0,10), nrow = 3),0)
A_At <- sample_matrix %*% t(sample_matrix)
At_A <- t(sample_matrix) %*% sample_matrix
sample_matrix
```

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

```
A_At
```

```
##      [,1] [,2] [,3]
## [1,]  194  136  116
## [2,]  136  118   64
## [3,]  116   64   83
```

```
At_A
```

```
##      [,1] [,2] [,3]
## [1,]  171  141   93
## [2,]  141  125   97
## [3,]   93   97   99
```

```
all.equal(A_At, At_A)
```

```
## [1] "Mean relative difference: 0.1635833"
```

Example 2

```
sample_matrix <- round(matrix(runif(20,0,10), nrow = 4),0)
A_At <- sample_matrix %*% t(sample_matrix)
At_A <- t(sample_matrix) %*% sample_matrix
sample_matrix
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    7    3    1    1    5
## [2,]    5    5    5   10    4
## [3,]    7    9    6    9    9
## [4,]    9   10    9    1    4
```

```
A_At
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   85   85  136  123
## [2,]   85  191  236  166
## [3,]  136  236  328  252
## [4,]  123  166  252  279
```

```
At_A
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  204  199  155  129  154
## [2,]  199  215  172  144  156
## [3,]  155  172  143  114  115
## [4,]  129  144  114  183  130
## [5,]  154  156  115  130  138
```

```
all.equal(A_At, At_A)
```

```
## [1] "Attributes: < Component \"dim\": Mean relative difference: 0.25 >"
## [2] "Numeric: lengths (16, 25) differ"
```

Example 3

```
sample_matrix <- round(matrix(runif(16,0,10), nrow = 4),0)
A_At <- sample_matrix %*% t(sample_matrix)
At_A <- t(sample_matrix) %*% sample_matrix
sample_matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    8    7    2    4
## [2,]    7    0    9    0
## [3,]    8    8    6   10
## [4,]    4    0    4    4
```

```
A_At
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  133   74  172   56
## [2,]   74  130  110   64
## [3,]  172  110  264   96
## [4,]   56   64   96   48
```

```
At_A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  193  120  143  128
## [2,]  120  113   62  108
## [3,]  143   62  137   84
## [4,]  128  108   84  132
```

```
all.equal(A_At, At_A)
```

```
## [1] "Mean relative difference: 0.4595695"
```

Example 4

```
sample_matrix <- round(matrix(runif(18,0,10), nrow = 6),0)
A_At <- sample_matrix %*% t(sample_matrix)
At_A <- t(sample_matrix) %*% sample_matrix
sample_matrix
```

```
##      [,1] [,2] [,3]
## [1,]   10    3    2
## [2,]    9    4    3
## [3,]    6    8    5
## [4,]   10    0    7
## [5,]    6    7   10
## [6,]    3    7    8
```

```
A_At
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  113  108   94  114  101   67
## [2,]  108  106  101  111  112   79
## [3,]   94  101  125   95  142  114
## [4,]  114  111   95  149  130   86
## [5,]  101  112  142  130  185  147
## [6,]   67   79  114   86  147  122
```

```
At_A
```

```
##      [,1] [,2] [,3]
## [1,]  362  177  231
## [2,]  177  187  184
## [3,]  231  184  251
```

```
all.equal(A_At, At_A)
```

```
## [1] "Attributes: < Component \"dim\": Mean relative difference: 0.5 >"
## [2] "Numeric: lengths (36, 9) differ"
```

In all of the above cases, we find that $A^TA \neq AA^T$.

Problem 2

For a special type of square matrix A, we get $A^TA = AA^T$. Under what conditions could this be true? (Hint: The Identity matrix I is an example of such a matrix).

Response 2

Any symmetric matrix will serve as an exception to the above rule, i.e. $A^TA = AA^T$ if A is symmetric. We can prove this by simply referring to Definition SYM in [1, Beezer], which defines a symmetric matrix as a matrix that is equal to its tranpose. So, if $A = A^T$ and $AA = AA$, then $A^TA = AA^T$.

Again, we look at a few examples to confirm.

Example 5

```
sample_matrix <- matrix(c(
  2, 7,
  7, 2
), nrow = 2, byrow = TRUE)
A_At <- sample_matrix %*% t(sample_matrix)
At_A <- t(sample_matrix) %*% sample_matrix
sample_matrix
```

```
##      [,1] [,2]
## [1,]    2    7
## [2,]    7    2
```

```
A_At
```

```
##      [,1] [,2]
## [1,]   53   28
## [2,]   28   53
```

```
At_A
```

```
##      [,1] [,2]
## [1,]   53   28
## [2,]   28   53
```

```
all.equal(A_At, At_A)
```

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

Example 6

```
sample_matrix <- matrix(c(
  3,-2, 6,
  -2,2,-1,
  6,-1, 3
), nrow = 3, byrow = TRUE)
A_At <- sample_matrix %*% t(sample_matrix)
At_A <- t(sample_matrix) %*% sample_matrix
sample_matrix
```

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

```
A_At
```

```
##      [,1] [,2] [,3]
## [1,]   49  -16   38
## [2,]  -16    9  -17
## [3,]   38  -17   46
```

```
At_A
```

```
##      [,1] [,2] [,3]
## [1,]   49  -16   38
## [2,]  -16    9  -17
## [3,]   38  -17   46
```

```
all.equal(A_At, At_A)
```

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

Example 6

```
sample_matrix <- matrix(c(
  1, 0, 6, 1,
  0, 1, 0,-1,
  6, 0, 1, 0,
  1,-1, 0, 1
), nrow = 4, byrow = TRUE)
A_At <- sample_matrix %*% t(sample_matrix)
At_A <- t(sample_matrix) %*% sample_matrix
sample_matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    6    1
## [2,]    0    1    0   -1
## [3,]    6    0    1    0
## [4,]    1   -1    0    1
```

```
A_At
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   38   -1   12    2
## [2,]   -1    2    0   -2
## [3,]   12    0   37    6
## [4,]    2   -2    6    3
```

```
At_A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   38   -1   12    2
## [2,]   -1    2    0   -2
## [3,]   12    0   37    6
## [4,]    2   -2    6    3
```

```
all.equal(A_At, At_A)
```

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

Problem Set 2

Problem

Write an R function to factorize a square matrix A into LU or LDU, whichever you prefer.

Response

I'll focus on LU decomposition.

I'll start by defining a 3x3 matrix, A, and using the `lu.decomposition` function from the `matrixcalc` package to set a baseline.

```
A = matrix(c(
   1, 4,-3,
  -2, 8, 5,
   3, 4, 7
), nrow = 3, byrow = TRUE)

lu_A <- lu.decomposition(A)

lu_A$L %*% lu_A$U
```

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

I'll then create the function to perform the LU decomposition. The overall logic is to follow the "shortcut" approach, focusing on row operations to convert the lower triangle of the original matrix to zeros. We start with the first column, using the first row to zero out all values in that column. Then we move to the second column, using the the second row to zero out the one remaining value in the lower triangle. The remaining row equivalent matrix serves as the decomposed Upper, and the scalars used in the row operations are plugged into their respective places in the Lower triangle (with ones added along the diagonal).

```
lu_decomp <- function(input_matrix) {
  U <- input_matrix

  L <- matrix(
    rep(0,length(input_matrix)),
    nrow = dim(input_matrix)[1],
    ncol = dim(input_matrix)[2]
  )
  diag(L) <- 1

  start <- 1

  for (col in 1:(ncol(input_matrix)-1)){

    vector_multiplier <- U[col,]

    for (row in start:(nrow(input_matrix)-1)) {

      scalar <- -(U[row+1,col] / U[col,col])

      U[row+1,] <- U[row+1,] + scalar * vector_multiplier
      L[row+1,col] <- -scalar
    }

    start <- start + 1

  }

  return(list(L = L, U = U))

}
```

Let's test out the function. Most importantly, I want to confirm that when we multiply L by U, we get our original matrix back as the result.

```
lu_A_KC <- lu_decomp(A)

lu_A_KC$L
```

```
##      [,1] [,2] [,3]
## [1,]    1  0.0    0
## [2,]   -2  1.0    0
## [3,]    3 -0.5    1
```

```
lu_A_KC$U
```

```
##      [,1] [,2] [,3]
## [1,]    1    4 -3.0
## [2,]    0   16 -1.0
## [3,]    0    0 15.5
```

```
lu_A_KC$L %*% lu_A_KC$U
```

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

It works!

I can now define a function to evaluate my function across a variety of matrices. The function with take a matrix, decompose it using my function as well as the `matrixcalc` function. We then evaluate the function by (1) confirming that the resulting $L * U$ correctly re-composes our original matrix, and (2) confirming that our results match those of the `lu.decomposition` function, for both L and U.

```
evaluate_decomp_function <- function(input_matrix) {
  lu_KC <- lu_decomp(input_matrix)
  lu_matrixCalc <- lu.decomposition(input_matrix)

  print('Original matrix versus Reformed Matrix (L*U)')
  print(round(input_matrix,2))
  print(round(lu_KC$L %*% lu_KC$U,2))
  print(ifelse(
    all.equal(round(input_matrix,2),round(lu_KC$L %*% lu_KC$U,2)),
    'MATCH',
    'NO MATCH'
    )
  )
  cat('\n')
  print('Lower: matrixCalc versus KC')
  print(round(lu_matrixCalc$L,2))
  print(round(lu_KC$L,2))
  print(ifelse(
    all.equal(round(lu_matrixCalc$U,2),round(lu_KC$U,2)),
    'MATCH',
    'NO MATCH'
    )
  )
  cat('\n')
  print('Upper: matrixCalc versus KC')
  print(round(lu_matrixCalc$U,2))
  print(round(lu_KC$U,2))
  print(ifelse(
    all.equal(round(lu_matrixCalc$U,2),round(lu_KC$U,2)),
    'MATCH',
    'NO MATCH'
    )
  )
  cat('\n')
}
```

Then we can plug in some matrices!

```
B <- matrix(c(
  1, 3,
  4, 2
), nrow = 2, byrow = TRUE)

evaluate_decomp_function(B)
```

```
## [1] "Original matrix versus Reformed Matrix (L*U)"
##      [,1] [,2]
## [1,]    1    3
## [2,]    4    2
##      [,1] [,2]
## [1,]    1    3
## [2,]    4    2
## [1] "MATCH"
##
## [1] "Lower: matrixCalc versus KC"
##      [,1] [,2]
## [1,]    1    0
## [2,]    4    1
##      [,1] [,2]
## [1,]    1    0
## [2,]    4    1
## [1] "MATCH"
##
## [1] "Upper: matrixCalc versus KC"
##      [,1] [,2]
## [1,]    1    3
## [2,]    0  -10
##      [,1] [,2]
## [1,]    1    3
## [2,]    0  -10
## [1] "MATCH"
```

```
C <- round(matrix(c(runif(9,0,10)), nrow = 3),0)

evaluate_decomp_function(C)
```

```
## [1] "Original matrix versus Reformed Matrix (L*U)"
##      [,1] [,2] [,3]
## [1,]    6    3    2
## [2,]    8    8    0
## [3,]    2    7    1
##      [,1] [,2] [,3]
## [1,]    6    3    2
## [2,]    8    8    0
## [3,]    2    7    1
## [1] "MATCH"
##
## [1] "Lower: matrixCalc versus KC"
##      [,1] [,2] [,3]
## [1,] 1.00  0.0    0
## [2,] 1.33  1.0    0
## [3,] 0.33  1.5    1
##      [,1] [,2] [,3]
## [1,] 1.00  0.0    0
## [2,] 1.33  1.0    0
## [3,] 0.33  1.5    1
## [1] "MATCH"
##
## [1] "Upper: matrixCalc versus KC"
##      [,1] [,2]  [,3]
## [1,]    6    3  2.00
## [2,]    0    4 -2.67
## [3,]    0    0  4.33
##      [,1] [,2]  [,3]
## [1,]    6    3  2.00
## [2,]    0    4 -2.67
## [3,]    0    0  4.33
## [1] "MATCH"
```

```
D <- round(matrix(c(runif(16,0,10)), nrow = 4),0)

evaluate_decomp_function(D)
```

```
## [1] "Original matrix versus Reformed Matrix (L*U)"
##      [,1] [,2] [,3] [,4]
## [1,]    2    0    6    8
## [2,]    5    4    2    6
## [3,]    2    5    4    2
## [4,]    7    0    6    1
##      [,1] [,2] [,3] [,4]
## [1,]    2    0    6    8
## [2,]    5    4    2    6
## [3,]    2    5    4    2
## [4,]    7    0    6    1
## [1] "MATCH"
##
## [1] "Lower: matrixCalc versus KC"
##      [,1] [,2]  [,3] [,4]
## [1,]  1.0 0.00  0.00    0
## [2,]  2.5 1.00  0.00    0
## [3,]  1.0 1.25  1.00    0
## [4,]  3.5 0.00 -1.05    1
##      [,1] [,2]  [,3] [,4]
## [1,]  1.0 0.00  0.00    0
## [2,]  2.5 1.00  0.00    0
## [3,]  1.0 1.25  1.00    0
## [4,]  3.5 0.00 -1.05    1
## [1] "MATCH"
##
## [1] "Upper: matrixCalc versus KC"
##      [,1] [,2]   [,3]   [,4]
## [1,]    2    0   6.00   8.00
## [2,]    0    4 -13.00 -14.00
## [3,]    0    0  14.25  11.50
## [4,]    0    0   0.00 -14.89
##      [,1] [,2]   [,3]   [,4]
## [1,]    2    0   6.00   8.00
## [2,]    0    4 -13.00 -14.00
## [3,]    0    0  14.25  11.50
## [4,]    0    0   0.00 -14.89
## [1] "MATCH"
```

```
E <- round(matrix(c(runif(25,0,10)), nrow = 5),0)

evaluate_decomp_function(E)
```

```
## [1] "Original matrix versus Reformed Matrix (L*U)"
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    9    7    4    5
## [2,]    3    9    6    9    0
## [3,]    7    7    6   10    6
## [4,]    0    3    2    7    8
## [5,]    2    5    2    7    8
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    9    7    4    5
## [2,]    3    9    6    9    0
## [3,]    7    7    6   10    6
## [4,]    0    3    2    7    8
## [5,]    2    5    2    7    8
## [1] "MATCH"
##
## [1] "Lower: matrixCalc versus KC"
##      [,1]  [,2]  [,3] [,4] [,5]
## [1,]    1  0.00  0.00  0.0    0
## [2,]    3  1.00  0.00  0.0    0
## [3,]    7  3.11  1.00  0.0    0
## [4,]    0 -0.17 -0.14  1.0    0
## [5,]    2  0.72 -0.32 -0.3    1
##      [,1]  [,2]  [,3] [,4] [,5]
## [1,]    1  0.00  0.00  0.0    0
## [2,]    3  1.00  0.00  0.0    0
## [3,]    7  3.11  1.00  0.0    0
## [4,]    0 -0.17 -0.14  1.0    0
## [5,]    2  0.72 -0.32 -0.3    1
## [1] "MATCH"
##
## [1] "Upper: matrixCalc versus KC"
##      [,1] [,2]   [,3]  [,4]   [,5]
## [1,]    1    9   7.00  4.00   5.00
## [2,]    0  -18 -15.00 -3.00 -15.00
## [3,]    0    0   3.67 -8.67  17.67
## [4,]    0    0   0.00  5.32   7.91
## [5,]    0    0   0.00  0.00  16.82
##      [,1] [,2]   [,3]  [,4]   [,5]
## [1,]    1    9   7.00  4.00   5.00
## [2,]    0  -18 -15.00 -3.00 -15.00
## [3,]    0    0   3.67 -8.67  17.67
## [4,]    0    0   0.00  5.32   7.91
## [5,]    0    0   0.00  0.00  16.82
## [1] "MATCH"
```

```
G <- round(matrix(c(runif(36,0,10)), nrow = 6),0)

evaluate_decomp_function(G)
```

```
## [1] "Original matrix versus Reformed Matrix (L*U)"
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    8    4    6    1    8
## [2,]    5    4    9    6    8    7
## [3,]    5    4   10    9    6    8
## [4,]    0    6    2    9    1    2
## [5,]    4    6    7    6    1    9
## [6,]    6    7    9    8    5    3
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    8    4    6    1    8
## [2,]    5    4    9    6    8    7
## [3,]    5    4   10    9    6    8
## [4,]    0    6    2    9    1    2
## [5,]    4    6    7    6    1    9
## [6,]    6    7    9    8    5    3
## [1] "MATCH"
##
## [1] "Lower: matrixCalc versus KC"
##      [,1]  [,2] [,3] [,4] [,5] [,6]
## [1,]  1.0  0.00 0.00 0.00 0.00    0
## [2,]  1.0  1.00 0.00 0.00 0.00    0
## [3,]  1.0  1.00 1.00 0.00 0.00    0
## [4,]  0.0 -1.50 9.50 1.00 0.00    0
## [5,]  0.8  0.10 3.30 0.45 1.00    0
## [6,]  1.2  0.65 0.95 0.11 0.27    1
##      [,1]  [,2] [,3] [,4] [,5] [,6]
## [1,]  1.0  0.00 0.00 0.00 0.00    0
## [2,]  1.0  1.00 0.00 0.00 0.00    0
## [3,]  1.0  1.00 1.00 0.00 0.00    0
## [4,]  0.0 -1.50 9.50 1.00 0.00    0
## [5,]  0.8  0.10 3.30 0.45 1.00    0
## [6,]  1.2  0.65 0.95 0.11 0.27    1
## [1] "MATCH"
##
## [1] "Upper: matrixCalc versus KC"
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]
## [1,]    5    8    4   6.0  1.00  8.00
## [2,]    0   -4    5   0.0  7.00 -1.00
## [3,]    0    0    1   3.0 -2.00  1.00
## [4,]    0    0    0 -19.5 30.50 -9.00
## [5,]    0    0    0   0.0 -7.51  3.42
## [6,]    0    0    0   0.0  0.00 -6.89
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]
## [1,]    5    8    4   6.0  1.00  8.00
## [2,]    0   -4    5   0.0  7.00 -1.00
## [3,]    0    0    1   3.0 -2.00  1.00
## [4,]    0    0    0 -19.5 30.50 -9.00
## [5,]    0    0    0   0.0 -7.51  3.42
## [6,]    0    0    0   0.0  0.00 -6.89
## [1] "MATCH"
```

```
H <- round(matrix(c(runif(64,0,10)), nrow = 8),0)

evaluate_decomp_function(H)
```

```
## [1] "Original matrix versus Reformed Matrix (L*U)"
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    0    2    5    9    3    1    4
## [2,]    7    1    5    4    3    2    7   10
## [3,]    3    7    3    1    3    8    7    5
## [4,]    8    9    1    8    7    1    9    3
## [5,]    4    6    2    6    7    1    5    3
## [6,]    7    6    7    8    9    1    9    5
## [7,]    8    2    4    8    8    1    4    6
## [8,]    2    5    4    9    1    5    2    3
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    0    2    5    9    3    1    4
## [2,]    7    1    5    4    3    2    7   10
## [3,]    3    7    3    1    3    8    7    5
## [4,]    8    9    1    8    7    1    9    3
## [5,]    4    6    2    6    7    1    5    3
## [6,]    7    6    7    8    9    1    9    5
## [7,]    8    2    4    8    8    1    4    6
## [8,]    2    5    4    9    1    5    2    3
## [1] "MATCH"
##
## [1] "Lower: matrixCalc versus KC"
##      [,1] [,2] [,3] [,4]  [,5]   [,6]   [,7] [,8]
## [1,]    1    0 0.00 0.00  0.00   0.00   0.00    0
## [2,]    7    1 0.00 0.00  0.00   0.00   0.00    0
## [3,]    3    7 1.00 0.00  0.00   0.00   0.00    0
## [4,]    8    9 1.10 1.00  0.00   0.00   0.00    0
## [5,]    4    6 0.80 0.41  1.00   0.00   0.00    0
## [6,]    7    6 0.78 0.00  2.37   1.00   0.00    0
## [7,]    8    2 0.10 0.41 -0.16  -0.13   1.00    0
## [8,]    2    5 0.75 0.07  9.61 -56.48 -16.91    1
##      [,1] [,2] [,3] [,4]  [,5]   [,6]   [,7] [,8]
## [1,]    1    0 0.00 0.00  0.00   0.00   0.00    0
## [2,]    7    1 0.00 0.00  0.00   0.00   0.00    0
## [3,]    3    7 1.00 0.00  0.00   0.00   0.00    0
## [4,]    8    9 1.10 1.00  0.00   0.00   0.00    0
## [5,]    4    6 0.80 0.41  1.00   0.00   0.00    0
## [6,]    7    6 0.78 0.00  2.37   1.00   0.00    0
## [7,]    8    2 0.10 0.41 -0.16  -0.13   1.00    0
## [8,]    2    5 0.75 0.07  9.61 -56.48 -16.91    1
## [1] "MATCH"
##
## [1] "Upper: matrixCalc versus KC"
##      [,1] [,2] [,3]  [,4]   [,5]   [,6]  [,7]    [,8]
## [1,]    1    0    2   5.0   9.00   3.00  1.00    4.00
## [2,]    0    1   -9 -31.0 -60.00 -19.00  0.00  -18.00
## [3,]    0    0   60 203.0 396.00 132.00  4.00  119.00
## [4,]    0    0    0  23.7  39.40   2.80 -3.40    2.10
## [5,]    0    0    0   0.0  -1.76  -3.73 -0.82   -1.05
## [6,]    0    0    0   0.0   0.00  -0.54  0.82   -5.72
## [7,]    0    0    0   0.0   0.00   0.00 -3.03   -3.68
## [8,]    0    0    0   0.0   0.00   0.00  0.00 -379.86
```

```
##      [,1] [,2] [,3]  [,4]   [,5]   [,6]  [,7]    [,8]
## [1,]    1    0    2   5.0   9.00   3.00  1.00    4.00
## [2,]    0    1   -9 -31.0 -60.00 -19.00  0.00  -18.00
## [3,]    0    0   60 203.0 396.00 132.00  4.00  119.00
## [4,]    0    0    0  23.7  39.40   2.80 -3.40    2.10
## [5,]    0    0    0   0.0  -1.76  -3.73 -0.82   -1.05
## [6,]    0    0    0   0.0   0.00  -0.54  0.82   -5.72
## [7,]    0    0    0   0.0   0.00   0.00 -3.03   -3.68
## [8,]    0    0    0   0.0   0.00   0.00  0.00 -379.86
## [1] "MATCH"
```

Citations

[1] Beezer, R. A. (2013). A First Course In Linear Algebra. OpenStax CNX.