Problem Set 1

Janette Avelar

4/2/2022

## Part 3: Checking Work Using R

**Woodward Appendix A.2.2**

1. Form the transpose of each matrix:

a\_trans <- matrix(data = c(1, 2,   
 3, 4),   
 nrow = 2, ncol = 2, byrow = TRUE) %>%   
 t() %>%   
 print()

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

b\_trans <- matrix(data = c(1,  
 2,  
 3),  
 nrow = 3, ncol = 1, byrow = TRUE) %>%   
 t() %>%   
 print()

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

c\_trans <- matrix(data = c(1, 4, 3,  
 4, 1, 2,  
 3, 2, 1),  
 nrow = 3, ncol = 3, byrow = TRUE) %>%   
 t() %>%   
 print()

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

d\_trans <- matrix(data = c(4, 4, 4, 1),  
 nrow = 1, ncol = 4, byrow = TRUE) %>%   
 t() %>%   
 print()

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

1. Which of the above matrices are symmetric?

# testing a  
dim(a\_trans) == dim(t(a\_trans))

## [1] TRUE TRUE

a\_trans == t(a\_trans) #nope

## [,1] [,2]  
## [1,] TRUE FALSE  
## [2,] FALSE TRUE

# testing b  
dim(b\_trans) == dim(t(b\_trans)) #nope

## [1] FALSE FALSE

# testing c  
dim(c\_trans) == dim(t(c\_trans))

## [1] TRUE TRUE

c\_trans == t(c\_trans) #yes

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

# testing d  
dim(d\_trans) == dim(t(d\_trans)) #nope

## [1] FALSE FALSE

C and C’ are symmetric.

**Woodward Appendix A.2.5**

1. Compute a’a and aa’.

# define a for part 2  
a2 <- matrix(data = c(1,  
 2,  
 3),  
 nrow = 3, ncol = 1, byrow = TRUE) %>%   
 print()

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

# define a transposed  
a2\_trans <- matrix(data = c(1,  
 2,  
 3),  
 nrow = 3, ncol = 1, byrow = TRUE) %>%   
 t() %>%   
 print()

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

# multiply a'a  
a2\_trans %\*% a2

## [,1]  
## [1,] 14

# multiple aa'  
a2 %\*% a2\_trans

## [,1] [,2] [,3]  
## [1,] 1 2 3  
## [2,] 2 4 6  
## [3,] 3 6 9

1. Compute a’b and b’a.

# define b for part 2  
b2 <- matrix(data = c(2, 1,  
 2, 3,  
 3, 4),  
 nrow = 3, ncol = 2, byrow = TRUE) %>%   
 print()

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

# define b transposed  
b2\_trans <- matrix(data = c(2, 1,  
 2, 3,  
 3, 4),  
 nrow = 3, ncol = 2, byrow = TRUE) %>%   
 t() %>%   
 print()

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

# multiply a'b  
a2\_trans %\*% b2

## [,1] [,2]  
## [1,] 15 19

# multiply b'a  
b2\_trans %\*% a2

## [,1]  
## [1,] 15  
## [2,] 19

*Looks like I forgot to finish this question or somehow overwrote my work in the original document. Whoops.*

1. Compute cd and dc.

# define c for part 2  
c2 <- matrix(data = c(1, 2, 3,  
 4, 5, 6,  
 7, 8, 9),  
 nrow = 3, ncol = 3, byrow = TRUE) %>%   
 print()

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

# define d for part 2  
d2 <- matrix(data = c(9, 8, 7,  
 6, 5, 4,  
 3, 2, 1),  
 nrow = 3, ncol = 3, byrow = TRUE) %>%   
 print()

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

# multiply cd  
c2 %\*% d2

## [,1] [,2] [,3]  
## [1,] 30 24 18  
## [2,] 84 69 54  
## [3,] 138 114 90

# multiply dc  
d2 %\*% c2

## [,1] [,2] [,3]  
## [1,] 90 114 138  
## [2,] 54 69 84  
## [3,] 18 24 30

*Some multiplication/addition errors occurred, but on the right track by hand.*

1. Compute dI\_3 and I\_3d.

# define I3  
I3 <- matrix(data = c(1, 0, 0,  
 0, 1, 0,  
 0, 0, 1),  
 nrow = 3, ncol = 3, byrow = TRUE) %>%   
 print()

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

# multiply dI\_3  
d2 %\*% I3

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

# multiply I\_3d  
I3 %\*% d2

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

1. Compute e(c+d).

# define e for part 2  
e2 <- matrix(data = c(1, 4, 1),  
 nrow = 1, ncol = 3, byrow = TRUE) %>%   
 print()

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

# compute e(c+d)  
e2 %\*% (c2 + d2)

## [,1] [,2] [,3]  
## [1,] 60 60 60

1. Compute e(c-d).

e2 %\*% (c2 - d2)

## [,1] [,2] [,3]  
## [1,] -12 0 12

*This one was off when conducted by hand.*

1. Compute ec-ed.

(e2 %\*% c2) - (e2 %\*% d2)

## [,1] [,2] [,3]  
## [1,] -12 0 12

*This one was off when conducted by hand.*

1. Compute cdb.

c2 %\*% d2 %\*% b2

## [,1] [,2]  
## [1,] 162 174  
## [2,] 468 507  
## [3,] 774 840

*This one was off when conducted by hand.*

**Woodward Appendix A.2.9**

1. Verify that aa^-1 = I\_2 in the preceding example.

# define a for part 3  
a3 <- matrix(data = c(3, 2,  
 2, 4),  
 nrow = 2, ncol = 2, byrow = TRUE) %>%   
 print()

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

# define a inverse  
a3\_inv <- matrix(data = c(0.5, -0.25,  
 -0.25, 0.375),  
 nrow = 2, ncol = 2, byrow = TRUE) %>%   
 print()

## [,1] [,2]  
## [1,] 0.50 -0.250  
## [2,] -0.25 0.375

# define I\_2  
I2 <- matrix(data = c(1, 0,  
 0, 1),  
 nrow = 2, ncol = 2, byrow = TRUE) %>%   
 print()

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

# compute aa^-1  
a3 %\*% a3\_inv

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

# compare to I\_2  
dim(a3 %\*% a3\_inv) == dim(I2)

## [1] TRUE TRUE

(a3 %\*% a3\_inv) == I2

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

1. Compute the inverse of b, where

# define b for part 3  
b3 <- matrix(data = c(6, 1,  
 1, 3),  
 nrow = 2, ncol = 2, byrow = TRUE) %>%   
 print()

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

# compute the inverse  
solve(b3)

## [,1] [,2]  
## [1,] 0.17647059 -0.05882353  
## [2,] -0.05882353 0.35294118

# convert to fraction to see if it matched  
fractions(solve(b3))

## [,1] [,2]   
## [1,] 3/17 -1/17  
## [2,] -1/17 6/17

1. Verify that bb^-1 = I\_2 in part (b).

# assign b inverse  
b3\_inv <- solve(b3) %>%   
 print()

## [,1] [,2]  
## [1,] 0.17647059 -0.05882353  
## [2,] -0.05882353 0.35294118

# compute bb^-1  
bb1 <- round(b3 %\*% b3\_inv) %>%   
 print()

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

# compare to I\_2  
dim(b3 %\*% b3\_inv) == dim(I2)

## [1] TRUE TRUE

bb1 == I2

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

**Reduced Row Echelon Form**

Determine the reduced row echelon form of the matrix:

# define the matrix  
mat <- matrix(data = c(2, 1, 2,  
 1, -1, 2,  
 -1, -5, 2,  
 -1, -5, 2,  
 -1, 10, -8),  
 nrow = 5, ncol = 3, byrow = TRUE) %>%   
 print()

## [,1] [,2] [,3]  
## [1,] 2 1 2  
## [2,] 1 -1 2  
## [3,] -1 -5 2  
## [4,] -1 -5 2  
## [5,] -1 10 -8

# convert to rref  
rref(mat)

## [,1] [,2] [,3]  
## [1,] 1 0 1.3333333  
## [2,] 0 1 -0.6666667  
## [3,] 0 0 0.0000000  
## [4,] 0 0 0.0000000  
## [5,] 0 0 0.0000000

# compare to answer  
fractions(rref(mat))

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

*I was way off, oh well.*