Basic R: Matrices

HONG CHEN

February 15, 2018

Matrix problems

1. Suppose

$$A = \begin{bmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{bmatrix}$$

(a) Check that $A^3 = \mathbf{0}$
(b) Replace the third column of A by the sum of the second and third columns

---

First, produce A

```
A <- matrix(c(1,1,3,5,2,6,-2,-1,-3), nrow = 3, byrow = TRUE)
A
```

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

```
A %*% A %*% A
```

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

Then, add the columns 2 and 3 and assign the sum to the third column

```
A[,3] <- A[,2] + A[,3]

A
```

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

---

2. Create the following matrix B with 15 rows

$$B = \begin{bmatrix} 10 & -10 & 10 \\ 10 & -10 & 10 \\ \cdots & \cdots & \cdots \\ 10 & -10 & 10 \end{bmatrix}$$

Calculate the 3x3 matrix $B^T B$. You can make this calculation with the function crossprod(). See the documentaion.

```r
tmp <- matrix(c(10,-10,10), b=T, nc=3, nr=15)
crossprod(tmp)
```

```
##       [,1]  [,2]  [,3]
## [1,]  1500 -1500  1500
## [2,] -1500  1500 -1500
## [3,]  1500 -1500  1500
```

3. Create a 6 x 6 matrix matE with every element equal to 0. check what the functions row() and col() return when applied to matE.

Now, create the 6 x 6 matix:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Here is matE, a 6x6 matrix of 0's followed by row(matE) and col(matE)

```
matE <- matrix(rep(0,36), nrow = 6, byrow = TRUE)
# Note what the functions row() and col() do
row(matE)
```

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

```
col(matE)
```

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

```
# With a little experimentation you would see
# that the specified pattern is in the /1/'s
row(matE)-col(matE)
```

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

```r
# so you use the locations of the 1's to modify matE
matE[abs(row(matE)-col(matE))==1] <- 1
matE
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    1    0    0    0    0
## [2,]    1    0    1    0    0    0
## [3,]    0    1    0    1    0    0
## [4,]    0    0    1    0    1    0
## [5,]    0    0    0    1    0    1
## [6,]    0    0    0    0    1    0
```

---

4. Look at the help for the function outer(). Now, create the following patterned matrix:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

```r
a <- 0:4
A <- outer(a,a,"+")
A
```

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

Use outer() a little more to make sure you get it.

```r
B <- outer(a,a, "*")
B
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    0    1    2    3    4
## [3,]    0    2    4    6    8
## [4,]    0    3    6    9   12
## [5,]    0    4    8   12   16
```

```r
# and
b <- 5:10
C <- outer(a,b,"+")
C
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    6    7    8    9   10
## [2,]    6    7    8    9   10   11
## [3,]    7    8    9   10   11   12
## [4,]    8    9   10   11   12   13
## [5,]    9   10   11   12   13   14
```

```
# and finally  --  make sure you check the values.
D <- outer(b,a, "%%")
D
```

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

5. Create the following patterned matrices. Your solutions should be generalizable to enable creating larger matrices with the same structure.

(a)

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{bmatrix}$$

```
outer(0:4,0:4,"+")%%5
```

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

(b)

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

```
outer(0:9,0:9,"+")%%10
```

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

(c)

$$\begin{bmatrix} 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 \\ 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

```
outer(0:8,0:8,"-")%%9
```

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

6. Solve the following system of linear equations by setting up and solving the matrix equation $\mathbf{Ax} = \mathbf{y}$.

$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 = 7$
$2x_1 + x_2 + 2x_3 + 3x_4 + 4x_5 = -1$
$3x_1 + 2x_2 + x_3 + 2x_4 + 3x_5 = -3$
$4x_1 + 3x_2 + 2x_3 + x_4 + 2x_5 = 5$
$5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 = 17$

```
yVec <- c(7,-1,-3,5,17)
AMat <- matrix(0,nr=5, nc=5)
AMat <- abs(col(AMat)-row(AMat))+1
solve(AMat,matrix(yVec,nc=1) )
```

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

7. Create a 6 x 10 matrix of random integers chosen from 1,2,. . . ,10 by executing the following two lines of code:

```
set.seed(75)
aMat <- matrix(sample(10, size=60, replace=TRUE), nr=6)
```

Use the matrix you have created to answer these questions:

(a) Find the number of entries in each row which are greater than 4.

```
apply(aMat, 1, function(x){sum(x>4)})
```

```
## [1] 4 7 6 2 6 7
```

(b) Which rows contain exactly two occurrences of the number seven?

```
which( apply(aMat,1,function(x){sum(x==7)==2}) )
```

```
## [1] 5
```

(c) Find those pairs of columns whose total (over both columns) is greater tnan 75. The answer should be a matrix with two columns; so, for example, the row (1,2) in the output matrix means that the sum of columns 1 and 2 in the original matrix is greater than 75. Repeating a column is permitted; so, for exalmple, the final output matrix could contain the rows (1,2), (2,1), and (2,2).

What if repetitions are not permitted? Then only (1,2) from (1,2),(2,1) and (2,2) would be permitted.

```
aMatColSums <- colSums(aMat)
which( outer(aMatColSums,aMatColSums,"+")>75, arr.ind=T )
```

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

8. Calculate

(a) $\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+j)}$

```
sum((1:20)^4) * sum(1/(3+(1:5)))
```

```
## [1] 639215.3
```

```
# or

sum(outer((1:20)^4, (3+(1:5)), "/"))
```

```
## [1] 639215.3
```

(b) $\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+ij)}$

```
sum( (1:20)^4 / (3 + outer(1:20,1:5,"*")))
```

```
## [1] 89912.02
```

(c) $\sum_{i=1}^{10}\sum_{j=1}^{i}\frac{i^4}{(3+ij)}$

```
sum( outer(1:10,1:10,function(i,j){ (i>=j)*i^4/(3+i*j) }) )
```

```
## [1] 6944.743
```

exercise3

1.

(a) Write functions `tmpFn1` and `tmpFn2` such that if `xVec` is the vector $(x_1, x_2, ..., x_n)$, then `tmpFn1(xVec)` returns vector $(x_1, x_2^2, ..., x_n^n)$ and `tmpFn2(xVec)` returns the vector $(x_1, \frac{x_2^2}{2}, ..., \frac{x_n^n}{n})$.

Here is `tmpFn1`

```
tmpFn1 <- function(xVec){
  return(xVec^(1:length(xVec)))
}

## simple example
a <- c(2, 5, 3, 8, 2, 4)

b <- tmpFn1(a)
b
```

```
## [1]    2   25   27 4096   32 4096
```

and now `tmpFn2`

```
tmpFn2 <- function(xVec2){

  n = length(xVec2)

  return(xVec2^(1:n)/(1:n))
}



c <- tmpFn2(a)
c
```

```
## [1]    2.0000   12.5000    9.0000 1024.0000    6.4000  682.6667
```

(b) Now write a fuction `tmpFn3` which takes 2 arguments x and n where x is a single number and n is a strictly positive integer. The function should return the value of

$$1 + \frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{3} + ... + \frac{x^n}{n}$$

```
tmpFn3 <- function(x,n)
{
  1 + sum((x^(1:n))/(1:n))
}
```

2. Write a funciton `tmpFn(xVec)` such that if `xVec` is the vector $x = (x_1, ..., x_n)$ then `tmpFn(xVec)` returns the vector of moving averages:

$$\frac{x_1 + x_2 + x_3}{3}, \frac{x_2 + x_3 + x_4}{3}, ..., \frac{x_{n-2} + x_{n-1} + x_n}{3}$$

Try out your function. `tmpFn`(c(1:5,6:1))

```
 tmpFn <- function(xVec) {
   n <- length(xVec)
   ( xVec[ -c(n-1,n) ] + xVec[ -c(1,n) ] + xVec[ -c(1,2) ] )/3 }
tmpFn(c(1:5,6:1))
```

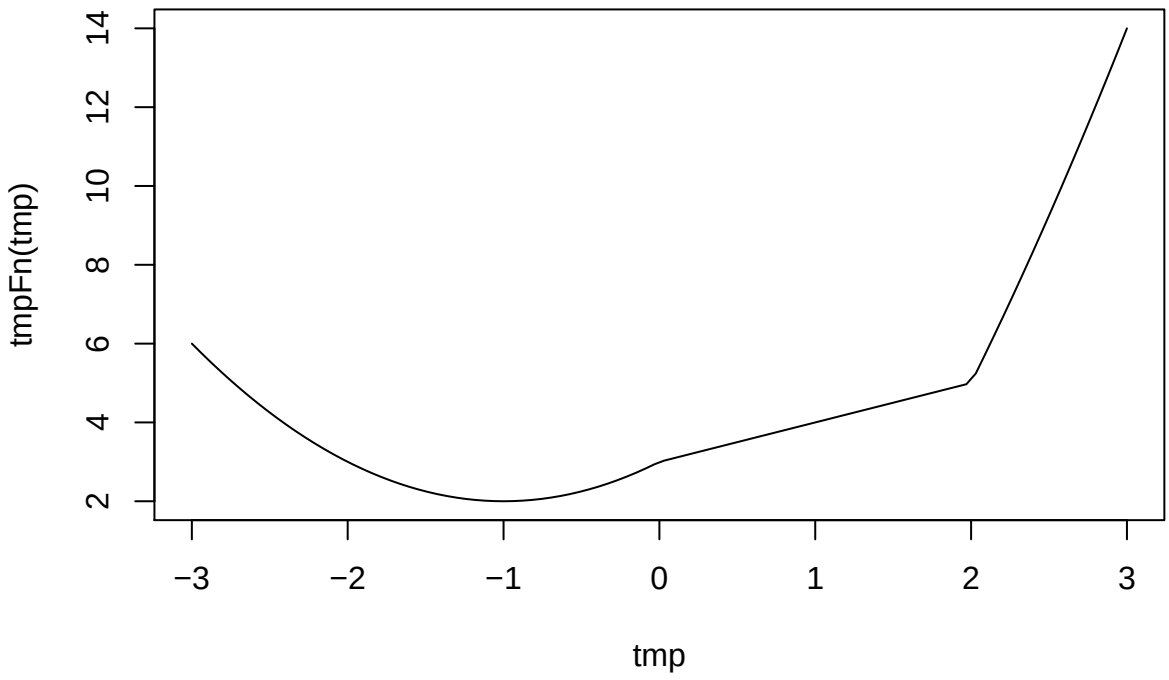
```
## [1] 2.000000 3.000000 4.000000 5.000000 5.333333 5.000000 4.000000 3.000000
## [9] 2.000000
```

3. Consider the continuous function

$$f(x) = \begin{cases} x^2 + 2x + 3 & if \quad x < 0 \\ x + 3 & if \quad 0 \leq x < 2 \\ x^2 + 4x - 7 & if \quad 2 \leq x \end{cases}$$

Write a function `tmpFn` which takes a single argument `xVec`. the function should return the vector the values of the function $f(x)$ evaluated at the values in `xVec`.
Hence plot the function $f(x)$ for $-3 < x < 3$.

```
 tmpFn <- function(x) {
ifelse(x < 0, x^2 + 2*x + 3, ifelse(x < 2, x+3, x^2 + 4*x - 7)) }
tmp <- seq(-3, 3, len=100)
plot(tmp, tmpFn(tmp), type="l")
```



4. Write a function which takes a single argument which is a matrix. The function should return a matrix which is the same as the function argument but every odd number is doubled.

Hence the result of using the function on the matrix

$$\begin{bmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{bmatrix}$$

should be:

$$\begin{bmatrix} 2 & 2 & 6 \\ 10 & 2 & 6 \\ -2 & -2 & -6 \end{bmatrix}$$

```
tmpFn <- function(mat) {
mat[mat%%2 == 1] <- 2 * mat[mat%%2 == 1]
mat
}
```

5. Write a function which takes 2 arguements n and k which are positive integers. It should return the nxn matrix:

$$\begin{bmatrix} k & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & k & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & k & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & k & \cdots & 0 & 0 \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \cdots & k & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & k \end{bmatrix}$$

```
tmpFn <- function(n, k)
{
tmp <- diag(k, nr = n)
tmp[abs(row(tmp) - col(tmp)) == 1] <- 1
tmp
}
```

6. Suppose an angle α is given as a positive real number of degrees.

If $0 \leq \alpha < 90$ then it is quadrant 1. If $90 \leq \alpha < 180$ then it is quadrant 2.
if $180 \leq \alpha < 270$ then it is quadrant3. if $270 \leq \alpha < 360$ then it is quadrant 4.
if $360 \leq \alpha < 450$ then it is quadrant 1.
And so on ...

Write a function `quadrant(alpha)` which returns the quadrant of the angle α.

```
quadrant <- function(alpha) {
1 + (alpha%%360)%/%90 }
```

7.

(a) Zeller's congruence is the formula:

$$f = ([2.6m - 0.2] + k + y + [y/4] + [c/4] - 2c) mod 7$$

where $[x]$ denotes the integer part of x; for example $[7.5] = 7$.

Zeller's congruence returns the day of the week f given:

k = the day of the month
y = the year in the century
c = the first 2 digits of the year (the century number)
m = the month number (where January is month 11 of the preceding year, February is month 12 of the preceding year, March is month 1, etc.)
For example, the date 21/07/1963 has $m = 5, k = 21, c = 19, y = 63$;
the date 21/2/63 has $m = 12, k = 21, c = 19, and y = 62$.
Write a function `weekday(day,month,year)` which returns the day of the week when given the numerical inputs of the day, month and year.
Note that the value of 1 for f denotes Sunday, 2 denotes Monday, etc.

```
weekday <- function(day, month, year) {
  month <- month - 2
  if(month <= 0) {
    month <- month + 12
```

```
    year <- year - 1 }
  cc <- year %/% 100
  year <- year %% 100
  tmp <- floor(2.6*month - 0.2) + day + year + year %/% 4 + cc %/% 4 - 2 * cc
  c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+tmp%%7] }
weekday(21,12,63)
```

```
## [1] "Friday"
```

(b) Does your function work if the input parameters day, month, and year are vectors with the same length and valid entries?

```
weekday2 <- function(day, month, year) {
  flag <- month <= 2
  month <- month - 2 + 12*flag
  year <- year - flag
  cc <- year %/% 100
  year <- year %% 100
  tmp <- floor(2.6*month - 0.2) + day + year + year %/% 4 + cc %/% 4 - 2 * cc
  c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+tmp%%7] }
weekday2( c(1,2,3), c(3,2,1), c(1993,1995,1997) )
```

```
## [1] "Monday"   "Thursday" "Friday"
```