Algebra lineal

Téllez Gerardo Rubén

5/4/2021

Opraciones

- **t(matriz):** para obtener la transpuesta de la matríz (cambia las filas por colúmnas)
- **+**: para sumar matrices
- ___*___: producto escalar por una matríz
- ___%*%___: para multiplicar matrices
- **mtr.exp(matriz, n)**: para elevar la matríz a n
  - Del paquete **Biodem**, no calcula potencias exactas, las aproxima.
- **%%**: para elevar matríces
  - Del paquete **expm**, no calcula las potencias exactas, las aproxima

```
m1 = matrix(data = seq(7, 63, by=7), nrow=3)
m1
```

```
##      [,1] [,2] [,3]
## [1,]    7   28   49
## [2,]   14   35   56
## [3,]   21   42   63
```

```
m2 = matrix(data = seq(10, 90, by=10), nrow = 3)
m2
```

```
##      [,1] [,2] [,3]
## [1,]   10   40   70
## [2,]   20   50   80
## [3,]   30   60   90
```

```
t(m2)
```

```
##      [,1] [,2] [,3]
## [1,]   10   20   30
## [2,]   40   50   60
## [3,]   70   80   90
```

```
m2 * 10
```

```
##      [,1] [,2] [,3]
## [1,]  100  400  700
## [2,]  200  500  800
## [3,]  300  600  900
```

```
m2 %*% m1 #Deben coincidir el número de filas con el número de colúmnas
```

```
##      [,1] [,2]  [,3]
## [1,] 2100 4620  7140
## [2,] 2520 5670  8820
## [3,] 2940 6720 10500
```

```
m2 %% m1
```

```
##      [,1] [,2] [,3]
## [1,]    3   12   21
## [2,]    6   15   24
## [3,]    9   18   27
```

Ejercicio

Observa qué ocurre si, siendo $A = \begin{pmatrix} 2 & 0 & 2 \\ 1 & 2 & 3 \\ 0 & 1 & 3 \end{pmatrix}$ y $B = \begin{pmatrix} 3 & 2 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, realizamos las operaciones $A\ *\ B$, $A^2\ y\ B^3$.

```
A = matrix(data = c(2, 0, 2, 1, 2, 3, 0, 1, 3), byrow = TRUE, nrow = 3)
B = matrix(data = c(3, 1, 1, 2, 0, 1, 1, 0, 1), nrow = 3)

# A*B
A * B
```

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

```
#(A)^2
A %*% A
```

```
##      [,1] [,2] [,3]
## [1,]    4    2   10
## [2,]    4    7   17
## [3,]    1    5   12
```

```
#(B)^3
B %*% B %*% B
```

```
##      [,1] [,2] [,3]
## [1,]   47   28   16
## [2,]   12    7    4
## [3,]   20   12    7
```

Operacones de álgebra lineal

- **det(matriz):** para calcular el deteminante de la matriz CUADRADA
- **qr(matriz)$rank:** para calcular el rango de una matriz DEVUELVE UNA LISTA
- **solve(matriz)**: para calcular la inversa de una matríz invertible
  - También sirve para resolver sistemas de ecuaciones lineales. Para ellos introduciremos **solve(matriz, b)**, donde *b* es el vector de términos independientes. INTRODUCIR LA MTRÍZ CUADRADA

```
M = rbind(c(1, 4, 2), c(0, 1, 3), c(1, 8, 9))

#Para obtener el determinante (producto de los valores propios)
det(M)
```

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

```
#Rango de la matríz
qr(M)$rank
```

```
## [1] 3
```

```
#Para obtener la matriz inversa
solve(M)
```

```
##      [,1] [,2] [,3]
## [1,]  3.0  4.0 -2.0
## [2,] -0.6 -1.4  0.6
## [3,]  0.2  0.8 -0.2
```

```
#Para obtener la matriz identidad
round(solve(M)%*%M, 1)
```

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

```
#Sistemas de ecuaciones lineales
vector_de_terminos_independientes <- c(2, 3, 6)
solve(M, vector_de_terminos_independientes) #Regresa el vector respuesta
```

```
## [1]  6.0 -1.8  1.6
```

Vectores y valores propios

- **eigen(matriz)**: para calcular los valores (vaps) y vectores propios (veps)
  - **eigen(matriz)$values**: devuelve el vector con los vaps de la matríz en orden decreciente de su valor absoluto y repetidos tantas veces como su multiplicidad algebraica.
  - **eigen(matriz)$vectors**: devuelve una matriz cuyas colúmnas son los veps de la matriz.

```
eigen(M)
```

```
## eigen() decomposition
## $values
## [1] 11.5677644 -1.0000000  0.4322356
##
## $vectors
##            [,1]       [,2]        [,3]
## [1,] -0.2744671  0.7427814 -0.98750842
## [2,] -0.2626036 -0.5570860  0.15481805
## [3,] -0.9250444  0.3713907 -0.02930006
```

```
eigen(M)$values
```

```
## [1] 11.5677644 -1.0000000  0.4322356
```

```
eigen(M)$vectors
```

```
##            [,1]       [,2]        [,3]
## [1,] -0.2744671  0.7427814 -0.98750842
## [2,] -0.2626036 -0.5570860  0.15481805
## [3,] -0.9250444  0.3713907 -0.02930006
```

Ejercicio

Comprueba con los datos del ejemplo anterior, que si P es la matriz de vectores propios de M en columna y D la matriz diagonal cuyas entradas son los valores propios de M, entonces se cumple la siguiente igualdad llamada **descomposición canónica:**

$$M = P \cdot D \cdot P^{-1}$$

```
M = A
P = eigen(M)$vectors
D = diag(c(eigen(M)$values), nrow = 3)

M
```

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

```
P %*% D %*% solve(P)
```

```
##                          [,1]                      [,2] [,3]
## [1,] 2.000000e+00-0.00000e+00i 3.330669e-16+1.110223e-16i 2+0i
## [2,] 1.000000e+00-0.00000e+00i 2.000000e+00+0.000000e+00i 3+0i
## [3,] 7.494005e-16+5.55112e-17i 1.000000e+00+0.000000e+00i 3+0i
```

Si hay algún vap con multiplicidad algebráica mayor que 1 (osease que aparece más de una vez), la función **eigen()** da tantos valores de este vap cmo su multiplicidad algebraica indica. Además, en este caso, R intenta que los veps asociados a cada uno de los vaps sean *linealmente independientes*. Por lo tanto, cuando como resultado obtenemos veps repetidos asociados a un vap de multiplicidad algebraica mayor que 1, es porque para este vap no existen tantos veps linealmente independientes como su multiplicidad algebraica, y por consiguiente, la matriz no es *diagonizable*.

```
m4 <- matrix(c(0, 1, 0, -7, 3, -1, 16, -3, 4), nrow=3, byrow = TRUE)
m4
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    0
## [2,]   -7    3   -1
## [3,]   16   -3    4
```

```
eigen(m4)
```

```
## eigen() decomposition
## $values
## [1] 3 2 2
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] -0.1301889 -0.1825742 -0.1825742
## [2,] -0.3905667 -0.3651484 -0.3651484
## [3,]  0.9113224  0.9128709  0.9128709
```

```
#Observese que el vep 2 es igual que el vep 3.
```