Operazioni (elementari) con Vettori e Matrici in R

1. Vettori

In R si può costruire un vettore utilizzando il comando $c(v_1,\dots,v_n)$, dove $v_1,\dots,v_n$ sono, nell'ordine, gli elementi del vettore:

```
vet1 <- c(2, 3, 4)
vet1
#[1] 2 3 4

vet2 <- c(2, 3, 4)
vet2
#[1] 2 3 4
```

1.1 Somma di vettori

Due o più vettori di uguale lunghezza n possono essere **sommati** (o sottratti). Il risultato sarà un altro vettore di lunghezza n:

```
vet3 <- vet1 + vet2
vet3
#[1] 4 6 8

vet1 + vet2 + vet3
#[1] 8 12 16

vet3 - vet2
#[1] 2 3 4
```

Si *possono* eseguire somme (e sottrazioni) con vettori di dimensioni diverse in R, ma il risultato non è *tecnicamente* una somma di vettori. In questi casi, vengono riutilizzati, in ordine, gli elementi del vettore con lunghezza minore.

```
vet_lungo <- c(1,2,3,4)
vet_corto <- c(5,6)
#dopo 1+5 e 2+6, il vet_corto viene riutilizzato per 3+5 e 4+6
vet_lungo + vet_corto
#[1] 6 8 8 10
```

1.2 Prodotto di vettori

Per calcolare il **prodotto interno** (o **scalare**) tra due vettori, utilizziamo il comando $sum()$. Il prodotto scalare è infatti dato dalla *somma dei prodotti* delle componenti i-esime di ciascun vettore.

$$[2 \quad 3 \quad 4] \cdot \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = (2 \times 2 + 3 \times 3 + 4 \times 4) = 29$$

```
vet1*vet2                          #vettore di prodotti i-esimi
#[1] 4 9 16
prod_int <- sum(vet_1*vet_2)             #prodotto scalare
prod_int
#[1] 29
#oppure usiamo l'operatore matriciale "%*% = prodotto interno"
vet1%*%vet2
#     [,1]
#[1,]   29
```

Si *può* calcolare il prodotto di due vettori di dimensioni diverse in R, ma il risultato non è *tecnicamente* un prodotto di vettori. In questo caso, vengono riutilizzati, in ordine, gli elementi del vettore con lunghezza minore, come abbiamo visto nella somma di vettori di diversa lunghezza.

Per ottenere il **prodotto esterno** (o **vettoriale**), che genera una matrice $(n_1 \times n_2)$ dei prodotti ordinati di ciascuna componente di V_1 per ciascuna componente di V_2, si utilizza invece il comando $outer()$:

$$\begin{bmatrix}2\\3\\4\end{bmatrix} \cdot [2 \quad 3 \quad 4] = \begin{bmatrix}2\times2 & 2\times3 & 2\times4\\3\times2 & 3\times3 & 3\times4\\4\times2 & 4\times3 & 4\times4\end{bmatrix}$$

```
 prod_est <- outer(vet_1, vet_2)              #prodotto vettoriale
 prod_est
#     [,1] [,2] [,3]
#[1,]    4    6    8
#[2,]    6    9   12
#[3,]    8   12   16

# oppure usiamo l'operatore matriciale "%o% = prodotto esterno"
 vet_1%o%vet_2
#     [,1] [,2] [,3]
#[1,]    4    6    8
#[2,]    6    9   12
#[3,]    8   12   16
```

2. Matrici

Per costruire una matrice si può utilizzare il comando $matrix()$:

```
 mat <- matrix(
 c(4,6,8,8,9,12,8,12,16),        #elementi della matrice
 nrow = 3,                       #numero di righe
 ncol = 3,                       #numero di colonne
 byrow = TRUE )                  #stabilisce se riempire la matrice procedendo
                                 #per riga (TRUE) o colonna (FALSE)

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

2.1 Somma di matrici

Contrariamente a quanto succede per i vettori, due matrici possono essere **sommate** solo se hanno le stesse dimensioni. La somma avviene elemento per elemento, nella stessa posizione di riga e di colonna.

Poiché entrambe le matrici che abbiamo appena trovato sono (3×3), possiamo tranquillamente sommarle:

```
 prod_est + mat
#      [,1] [,2] [,3]
#[1,]    8   12   16
#[2,]   14   18   24
#[3,]   16   24   32
```

2.2 Trasposizione di matrici

Una matrice può essere **trasposta**, scambiando le sue righe con le sue colonne, utilizzando il comando $t()$:

```
 v_test <- c(4,8,6,12,8,16)    #elementi che verranno inseriti nella matrice
 mat_A <- matrix(v_test, nrow = 3, ncol = 2, byrow = TRUE)
 mat_A
#     [,1] [,2]
#[1,]    4    8
#[2,]    6   12
#[3,]    8   16

 mat_B <- t(mat_A)             #matrice trasposta di mat_A
 mat_B
#     [,1] [,2] [,3]
#[1,]    4    6    8
#[2,]    8   12   16
```

La trasposta di una matrice trasposta è la matrice di partenza:

```
 mat_A
#     [,1] [,2]
#[1,]    4    8
#[2,]    6   12
#[3,]    8   16

 t(t(mat_A))
#     [,1] [,2]
#[1,]    4    8
#[2,]    6   12
#[3,]    8   16
```

Il comando $t()$ può anche essere utilizzato per convertire un vettore di lunghezza n in una matrice $(1 \times n)$:

```
 vet1
#   [1] 2 3 4

 t(vet1)                       # vettore diventa una matrice (1xn)
#     [,1] [,2] [,3]
#[1,]    2    3    4

 t(t(vet1))                    # viene cambiato in una matrice (nx1)
#     [,1]
#[1,]    2
#[2,]    3
#[3,]    4
```

2.3 Prodotto di matrici

Il **prodotto** tra una matrice $(r_1 \times c_1)$ e una matrice $(r_2 \times c_2)$ è possibile solo se $\boldsymbol{c_1 = r_2}$ (condizione di conformabilità). La risultante è una matrice di dimensioni $(r_1 \times c_2)$. In R si indica con %*%.

Poiché $c_{mat\_A} = r_{mat\_B}$ e $c_{mat\_B} = r_{mat\_A}$, possiamo utilizzare le matrici che abbiamo trovato prima per eseguire il prodotto:

```
mat_AB <- mat_A %*% mat_B                #A(3x2) * B(2x3) porta ad una matrice
AB(3x3)
mat_AB
#      [,1] [,2] [,3]
#[1,]   80  120  160
#[2,]  120  180  240
#[3,]  160  240  320

mat_BA <- mat_B %*% mat_A                #B(2x3) * A(3x2) porta ad una matrice
BA(2x2)
mat_BA
#      [,1] [,2]
#[1,]  116  232
#[2,]  232  464
```

2.4 Matrice inversa

Una matrice M quadrata, quindi una matrice con $n_{row} = n_{col}$, può essere **invertibile**. La matrice inversa M^{-1} è una matrice delle stesse dimensioni di M, tale per cui

$$M \cdot M^{-1} = M^{-1} \cdot M = I$$

dove I è una matrice particolare detta **identità**, caratterizzata da valori 1 lungo la diagonale, e da valori 0 al di fuori di essa:

```
mat_I <- matrix(
                c(1,0,0,0,1,0,0,0,1),
                nrow = 3,
                ncol = 3,
                byrow = TRUE )

mat_I                    #matrice identità

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

#oppure, utilizzando il comando diretto diag()

diag(1,nrow=3)
```

La matrice identità rappresenta l'uno del prodotto di matrici, fatte salve le condizioni di conformabilità:

```
 mat_AB %*% mat_I
#      [,1] [,2] [,3]
#[1,]   80  120  160
#[2,]  120  180  240
#[3,]  160  240  320
```

Per individuare la matrice inversa di M si utilizza il comando $solve()$:

```
mat_M <- matrix( c(4,8,2,3), nrow = 2, ncol = 2, byrow = TRUE )
mat_M
#      [,1] [,2]
#[1,]    4    8
#[2,]    2    3
```

```
mat_MI <- solve(mat_M)              #matrice inversa di mat_M
mat_MI
#      [,1] [,2]
#[1,] -0.75    2
#[2,]  0.50   -1
```

Vediamo che, indipendentemente dall'ordine di moltiplicazione, il loro prodotto è la matrice identità:

```
 mat_M %*% mat_MI

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

 mat_MI %*% mat_M

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

Nel caso di $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ quadrata (2×2), si può facilmente ricavare l'inversa utilizzando

$$M^{-1} = \begin{bmatrix} {}^{d}/_{\det(M)} & {}^{-b}/_{\det(M)} \\ {}^{-c}/_{\det(M)} & {}^{a}/_{\det(M)} \end{bmatrix};$$

dove $\det(M) = ad - cb$.

```
det(mat_M)

#[1] -4
```

Teorema: l'inversa di una matrice M è ammessa se e solo se il determinante di M è diverso da 0, ossia

$$\det(M) = ad - cb \neq 0.$$

L'inversa della nostra matrice M risulta quindi essere:

$$M = \begin{bmatrix} 4 & 8 \\ 2 & 3 \end{bmatrix};$$

$$M^{-1} = \begin{bmatrix} {}^{3}/_{(12-16)} & {}^{-8}/_{(12-16)} \\ {}^{-2}/_{(12-16)} & {}^{4}/_{(12-16)} \end{bmatrix} =$$

$$= \begin{bmatrix} -{}^{3}/_{4} & 2 \\ {}^{2}/_{4} & -1 \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} -0.75 & 2 \\ 0.50 & -1 \end{bmatrix};$$

2.5 Applicazione statistica – Stima dei parametri di una retta

Dati due vettori x e y tali che

$$x = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 4 \\ 5 \\ 8 \end{bmatrix}; \quad y = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 7 \\ 6 \\ 7 \end{bmatrix};$$

utilizzare solo l'algebra lineare per calcolare i parametri a e b della retta di regressione.

Per farlo, occorre applicare questa formula:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum {x_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix}.$$

a. Costruiamo i vettori

```
 x<-c(2,3,1,4,5,8)
 t(t(x))
#      [,1]
#[1,]    2
#[2,]    3
#[3,]    1
#[4,]    4
#[5,]    5
#[6,]    8

 y<-c(3,2,2,7,6,7)
 t(t(y))
#      [,1]
#[1,]    3
#[2,]    2
#[3,]    2
#[4,]    7
#[5,]    6
#[6,]    7
```

b. Costruiamo la prima matrice

```
n = length(x)       #dimensione del vettore x (pari a quella di y)
                    #elementi che verranno inseriti nella matrice
                    #sum(x) è la sommatoria degli elementi di x
                    #sum(x^2) è la sommatoria degli elementi al quadrato di x
abcd <- c(n, sum(x), sum(x), sum(x^2))

mat1 <- matrix(abcd, nrow=2, ncol=2, byrow=TRUE)
mat1
#      [,1] [,2]
#[1,]    6   23
#[2,]   23  119
```

c. Troviamone l'inversa

```
 inv_mat1 <- solve(mat1)
 inv_mat1
#           [,1]        [,2]
#[1,]  0.6432432 -0.12432432
#[2,] -0.1243243  0.03243243
```

d. Costruiamo la seconda matrice

```
ef <- c(sum(y), sum(x*y))          #elementi che verranno inseriti nella matrice
                                   #sum(y) è la sommatoria degli elementi di y
                                   #sum(x*y) è il prodotto scalare di x e y

 mat2 <- matrix(ef, nrow=2, ncol=1)
 mat2
#     [,1]
#[1,]   27
#[2,]  128
```

e. Calcoliamo infine a e b

```
 mat_ab <- inv_mat1 %*% mat2
 mat_ab
#           [,1]
#[1,] 1.4540541
#[2,] 0.7945946
```

I parametri a e b della retta di regressione dei vettori x e y sono pari a:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1.454 \\ 0.795 \end{bmatrix}.$$

```
 lm(y~x)$coeff
#(Intercept)           x
#  1.4540541   0.7945946

 plot(x,y,col="red",pch=19)
 abline(a=1.4540541,b=.7945946,col="green")
```