Il modello di regressione lineare: Stima iterativa con metodo di Fisher scoring – Applicazione in R

Definiamo in R i dati x e y su cui stimare i parametri della retta di regressione.

```
x<-c(2,3,1,4,5,8)
y<-c(3,2,2,7,6,7)
n<-length(y)
```

Costruiamo la matrice del modello lineare.

```
 X<-matrix(c(rep(1,6),x),ncol=2,byrow=FALSE)
 X
     [,1] [,2]
[1,]    1    2
[2,]    1    3
[3,]    1    1
[4,]    1    4
[5,]    1    5
[6,]    1    8
```

Identifichiamo con *j* il numero di variabili x considerate ($j = 1$; una sola variabile indipendente x) e scriviamo un vettore $(j + 2)$ di partenza (*initial trial*) per i parametri ignoti:

$$\boldsymbol{\varphi}^{(\boldsymbol{t}=\mathbf{0})} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \sigma^2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

```
j=1
phit0<-matrix(c(1,1,1))
phit0
# scorporiamo phit0 nei parametri da stimare:

B<-phit0[1:(j+1)]        # vettore di intercetta e pendenza
sigma_q<-phit0[j+2]      # varianza omoschedastica dell'errore
```

Scriviamo il vettore $\boldsymbol{u}$ delle derivate prime, di dimensione $(j + 2)\times 1$, formato dalle $Eq(21\ e\ 23)$,

$$\underset{(j+2)\times 1}{\boldsymbol{u}} = \begin{bmatrix} \frac{1}{\sigma^2}\mathbf{X}'\mathbf{y} - \frac{1}{\sigma^2}\mathbf{X}'\mathbf{X}\,\boldsymbol{\beta} \\ -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) \end{bmatrix}$$

```
u<-matrix(c(
             1/sigma_q*t(X)%*%y-1/sigma_q*t(X)%*%X%*%B,
             -n/(2*sigma_q)+1/(2*sigma_q^2)*t(y-X%*%B)%*%(y-X%*%B)
            ),ncol=1)

u
     [,1]
[1,]   -2
[2,]  -14
[3,]    3
```

Scriviamo la matrice a blocchi (Hessiana) delle derivate seconde, nella quale inseriamo i risultati delle $Eq(27,29,31)$, ottenuti sostituendo le stime di massima verosimiglianza nelle $Eq(26,28,30)$,

$$E(\boldsymbol{H}) = \begin{bmatrix} -\frac{1}{\sigma^2}\mathbf{X}'\mathbf{X} & \mathbf{0} \\ \mathbf{0} & -\frac{n}{2(\sigma^2)^2} \end{bmatrix},$$

dove $\mathbf{0}$ è un vettore (riga o colonna) «nullo» (composto da valori zero) di dimensione $(j + 1)\times 1$ e $1\times(j + 1)$.

```
H<-matrix(NA,ncol=(j+2),nrow=(j+2))
H
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
[3,]   NA   NA   NA

# Inseriamo il primo minore (2x2) con le derivate seconde del vettore B
H[1:(j+1),1:(j+1)]<- -1/sigma_q*t(X)%*%X
H
     [,1] [,2] [,3]
[1,]   -6  -23   NA
[2,]  -23 -119   NA
[3,]   NA   NA   NA

# Inseriamo l'ultimo valore in diagonale con la derivata seconda in σ^2
H[j+2,j+2]<--n/(2*sigma_q^2)
H
     [,1] [,2] [,3]
[1,]   -6  -23   NA
[2,]  -23 -119   NA
[3,]   NA   NA   -3

# Completiamo con i due vettori nulli in riga 3 e colonna 3:
H[3,1:(j+1)]<-rep(0,2) # oppure anche solo 0
H[1:(j+1),3]<-rep(0,2) # oppure anche solo 0
H
     [,1] [,2] [,3]
[1,]   -6  -23    0
[2,]  -23 -119    0
[3,]    0    0   -3
```

Aggiorniamo le stime iniziali con una prima iterazione Fisher scoring:

$$\boldsymbol{\varphi}^{(\boldsymbol{t}=\mathbf{1})} = \boldsymbol{\varphi}^{(\boldsymbol{t}=\mathbf{0})} - \left(E\left(\boldsymbol{H}^{(t=0)}\right)\right)^{-1}\boldsymbol{u}^{(t=0)}$$

```
phit1<- phit0 - solve(H)%*%u
phit1
          [,1]
[1,] 1.4540541   # OK
[2,] 0.7945946   # OK
[3,] 2.0000000   # Nel prossimo ciclo!
```

Automatizziamo in R:

```
i=0                     # Numero progressivo di iterazioni
phit0<-matrix(c(1,1,1)) # Stime iniziali dei parametri ignoti
n<-length(y)            # Grandezza del campione
FisherScoring<-c()      # data set vuoto per i risultati:
```

Notiamo che, un vettore di condizioni logiche TRUE/FALSE può essere verificato contemporaneamente mediante il comando sum(), infatti R considera TRUE = 1 e FALSE = 0:

```
c(1,2,10)>5             # Condizione logica: (TRUE = 1, FALSE = 2) «FALSE»
[1] FALSE FALSE  TRUE

sum(c(1,2,10)>5)
[1] 1                   # un solo TRUE, somma = 1

c(1,2,10)>0
[1] TRUE TRUE TRUE     # Condizione logica: (TRUE = 3, FALSE = 0) «TRUE»

sum(c(1,2,10)>0)
[1] 3                  # 3 TRUE, somma = 3

# Rendiamo il risultato «logico» con il doppio segno di verifica:
sum(c(1,2,10)>0)==3    # 3 TRUE, somma = 3 → «VERO»
[1] TRUE
```

Questo ci sarà utile per confrontare <u>contemporaneamente</u> la condizione:

$$\boldsymbol{\varphi}^{(t+1)} - \boldsymbol{\varphi}^{(t)} < .00001$$

```
phit1
          [,1]
[1,] 1.4540541
[2,] 0.7945946
[3,] 2.0000000

phit0
     [,1]
[1,]    1
[2,]    1
[3,]    1

phit1-phit0
           [,1]
[1,]  0.4540541
[2,] -0.2054054
[3,]  1.0000000

abs(phit1-phit0) < .00001
      [,1]
[1,] FALSE
[2,] FALSE
[3,] FALSE

sum(abs(phit1-phit0) < .00001)==3
[1] FALSE
```

Automatizziamo in R:

```
j=1                     # Numero di variabili indipendenti (colonne matrice X - 1)
i=0                     # Numero progressivo di iterazioni
phit0<-matrix(c(1,1,1)) # Stime iniziali dei parametri ignoti
n<-length(y)            # Grandezza del campione
FisherScoring<-c()      # data set vuoto per i risultati:

repeat{

      i=i+1 # prima iterazione (la prima sarà nulla, quindi registreremo i-1)
      B<-phit0[1:(j+1)]
      sigma_q<-phit0[j+2]

      u<-matrix(c(1/sigma_q*t(X)%*%y-1/sigma_q*t(X)%*%X%*%B,
                  -n/(2*sigma_q)+1/(2*sigma_q^2)*t(y-X%*%B)%*%(y-X%*%B)
                 ),ncol=1)

      H<-matrix(NA,ncol=(j+2),nrow=(j+2))
      H[1:(j+1),1:(j+1)]<--1/sigma_q*t(X)%*%X
      H[j+2,j+2]<--n/(2*sigma_q^2)
      H[3,1:(j+1)]<-rep(0,2) # oppure anche solo 0
      H[1:(j+1),3]<-rep(0,2) # oppure anche solo 0


      phit1<- phit0 - solve(H)%*%u

      #registriamo i risultati nel data set FisherScoring:
      Result_iter<-c(phit1[1:(j+1)],phit1[j+2])
      FisherScoring<-cbind(FisherScoring,Result_iter)

      #condizione logica per interrompere il ciclo:
      if(sum(abs(phit1 - phit0) < .0001)==3){
                    rownames(FisherScoring)<-c(paste("b",seq(1,j+1)),"sigma_q")
                    colnames(FisherScoring)<-seq(0,(i-1))
                    print(FisherScoring)
                    break
                                          }

     #aggiornamento delle stime iniziali per riprendere poi il ciclo
     phit0<-phit1
}
```

Risultato: <u>due sole iterazioni</u> (una per il vettore B e una per il parametro sigma_q):

```
                0         1         2
b 1     1.4540541 1.4540541 1.4540541
b 2     0.7945946 0.7945946 0.7945946
sigma_q 2.0000000 1.6720721 1.6720721
```

Inserendo un valore «sbagliato» nella matrice Hessiana «attesa»:

$$E(\boldsymbol{H}) = \begin{bmatrix} -\frac{n}{SQErr}\mathbf{X}'\mathbf{X} & \mathbf{0} \\ \mathbf{0} & -\frac{n}{2(\sigma^2)^2} \end{bmatrix},$$

Che rappresenta un errore in quanto:

$$E\left(-\frac{n}{SQErr}\mathbf{X}'\mathbf{X}\right) = -\frac{n}{E(SQErr)}\mathbf{X}'\mathbf{X} = -\frac{n}{n\sigma^2}\mathbf{X}'\mathbf{X} = -\frac{1}{\sigma^2}\mathbf{X}'\mathbf{X}$$

L'algoritmo (*non più Fisher scoring, perché la matrice Hessiana non è più quella attesa*) impiegherà più cicli (solo uno in più , in questo caso molto semplice).

```
                0         1         2         3
b 1     1.9081081 1.4540541 1.4540541 1.4540541
b 2     0.5891892 0.7945946 0.7945946 0.7945946
sigma_q 2.0000000 2.0000000 1.6720721 1.6720721
```

R dispone di diverse funzioni di ottimizzazione numerica, che non necessitano cioè di vettori di derivate prime e matrici Hessiane. La convergenza generalmente richiede molti cicli e sono utili nel caso di funzioni molto complesse e non derivabili. Vediamo un esempio per il nostro problema.

Iniziamo scrivendo la funzione di verosimiglianza, che moltiplicheremo per -1, dal momento che questi algoritmi ricercano un minimo (e non un massimo come nel nostro caso!):

```
#Dati:
x<-c(2,3,1,4,5,8)
y<-c(3,2,2,7,6,7)
n<-length(y)            # Grandezza del campione
j=1                     # Numero di variabili indipendenti (colonne matrice X - 1)

#Log Likelihood function con argomento «phi» = vettore di parametri ignoti.

Log_Lik<-function(phi){

         B<-phi[1:(j+1)]
         sigma_q<-phi[j+2]

         score=-n/2*log(2*pi)-n/2*log(sigma_q)-1/(2*sigma_q)*t(y-X%*%B)%*%(y-X%*%B)

        (-1)*score # algoritmi di minimizzazione!
}
```

La funzione che useremo `optim()` richiede tre argomenti: *par* = vettore iniziale di parametri, *fn* = la funzione da minimizzare, *method* = il metodo usato.

```
phit0<-c(1,1,1) #vettore di stime iniziali

#optim(par=phit0, fn=Log_Lik, method="Nelder-Mead")
#optim(par=phit0, fn=Log_Lik, method="BFGS")
#optim(par=phit0, fn=Log_Lik, method="CG")
#optim(par=phit0, fn=Log_Lik, method="L-BFGS-B")
#optim(par=phit0, fn=Log_Lik, method="SANN")
```

Ad esempio:

```
optim(par=phit0, fn=Log_Lik, method="L-BFGS-B")

$par
[1] 1.4540502 0.7945923 1.6720611   # Parametri stimati: OK!

$value                              # Valore minimo della funzione Log_Lik
[1] 10.05582

$counts
function gradient
     17       17                    # Numero di iterazioni (> di Fisher scoring!)
```

Il modello lineare generalizzato - GLM

Il metodo di stima del modello lineare per massima verosimiglianza si può ottenere in R mediante la funzione `glm()`

```
summary(                                        # per ottenere il sommario
        glm(y~x,family=gaussian(link=identity)) # funzione glm() per dati gaussiani
       )

Call:
glm(formula = y ~ x, family = gaussian(link = identity))

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.4541     1.2702   1.145   0.3161
x             0.7946     0.2852   2.786   0.0495 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 2.508108)

    Null deviance: 29.500  on 5  degrees of freedom
Residual deviance: 10.032  on 4  degrees of freedom
AIC: 26.112

Number of Fisher Scoring iterations: 2
```

Vediamo di «ritrovare» i risultati all'interno della nostra procedura iterativa di stima:

A) Coefficienti B:

```
                0         1         2
b 1     1.4540541 1.4540541 1.4540541
b 2     0.7945946 0.7945946 0.7945946
sigma_q 2.0000000 1.6720721 1.6720721
```

B) Varianza omoschedastica di errore attorno ai valori previsti *(…corretta per i gradi di libertà)* :

```
                0         1         2
b 1     1.4540541 1.4540541 1.4540541
b 2     0.7945946 0.7945946 0.7945946
sigma_q 2.0000000 1.6720721 1.6720721

k=1                             # variabili x nel modello lineare
1.6720721*n/(n-k-1)             # cambiamo il denominatore da n a (n-k-1) Equazione 24
[1] 2.508108
sqrt(1.6720721*n/(n-k-1))       # errore standard dei residui
[1] 1.583701
```

C) Errori standard dei coefficienti di regressione: Mediante il negativo dell'inversa della matrice Hessiana, aggiornata con la stima finale dei coefficienti nel processo iterativo.

```
# È noto che le stime delle varianze mediante il metodo MLE siano distorte, poiché
non corrette per i gradi di libertà:
-1*solve(H)
             [,1]        [,2]      [,3]
[1,]  1.0755491 -0.20787923 0.0000000       # Varianza distorta di (Intercetta) B_0
[2,] -0.2078792  0.05422936 0.0000000       # Varianza distorta di B_1
[3,]  0.0000000  0.00000000 0.9319417

1.0755491*n/(n-k-1)
[1] 1.613324                                # Varianza corretta di (Intercetta) B_0
sqrt(1.613324)
[1] 1.270167                                # Errore standard di (Intercetta) B_0
```

Il modello lineare generalizzato - GLM

C) …

```
-1*solve(H)
            [,1]        [,2]      [,3]
[1,]  1.0755491 -0.20787923 0.0000000       # Varianza distorta di (Intercetta) B_0
[2,] -0.2078792  0.05422936 0.0000000       # Varianza distorta di B_1
[3,]  0.0000000  0.00000000 0.9319417

0.05422936*n/(n-k-1)
[1] 0.08134404                              # Varianza corretta di B_1
sqrt(0.08134404)
[1] 0.2852088                               # Errore standard di B_1
```

Proviamo a stimare un modello più complesso:

```
 y <- c(3,2,2,7,6,7)

# Matrice del modello
 X <- matrix(
             c(1,2,2,3,
               1,3,2,7,
               1,1,1,2,
               1,4,5,6,
               1,5,9,8,
               1,8,2,9),
             ncol = 4,
             nrow = 6,
             byrow = TRUE
           )


 # Stima con funzione glm():
 # prendiamo solo le ultime 3 colonne di X
 summary(glm(y~X[,-1]))

Call:
glm(formula = y ~ X[, -1])

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.9258     1.5415   1.249    0.338
X[, -1]1      1.2969     0.5931   2.187    0.160
X[, -1]2      0.4023     0.2501   1.608    0.249
X[, -1]3     -0.6523     0.5677  -1.149    0.369

(Dispersion parameter for gaussian family taken to be 2.078125)

    Null deviance: 29.5000  on 5  degrees of freedom
Residual deviance:  4.1563  on 2  degrees of freedom
AIC: 24.824

Number of Fisher Scoring iterations: 2


###### Fisher Scoring: dati iniziali

j=3                       # Numero di variabili indipendenti (colonne matrice X - 1)
i=0                       # Numero progressivo di iterazioni
phit0<-matrix(rep(1,j+2) # Stime iniziali(intercetta + 3 beta_i + sigma_q)
n<-length(y)              # Grandezza del campione
FisherScoring<-c()        # data set vuoto per i risultati:
```

Proviamo a stimare un modello più complesso:

Operiamo alcuni cambiamenti nell'algoritmo Fisher scoring per renderlo modulare a qualsiasi dimensione della matrice modello X:

```
repeat{
       i=i+1 # prima iterazione (la prima sarà nulla, quindi registreremo i-1)
  B<-phit0[1:(j+1)]
       sigma_q<-phit0[j+2]


        u<-matrix(c(
                     1/sigma_q*t(X)%*%y-1/sigma_q*t(X)%*%X%*%B,
                    -n/(2*sigma_q)+1/(2*sigma_q^2)*t(y-X%*%B)%*%(y-X%*%B)
                    ),ncol=1)

         H<-matrix(NA,ncol=(j+2),nrow=(j+2))
         H[1:(j+1),1:(j+1)]<--1/sigma_q*t(X)%*%X
         H[j+2,j+2]<--n/(2*sigma_q^2)
         H[(j+2),1:(j+1)]<-0
         H[1:(j+1),(j+2)]<-0


         phit1<- phit0 - solve(H)%*%u


         #registriamo i risultati nel data set FisherScoring:
         Result_iter<-c(phit1[1:(j+1)],phit1[j+2])
         FisherScoring<-cbind(FisherScoring, Result_iter)


         #condizione logica per interrompere il ciclo:
         if(sum(abs(phit1 - phit0) < .0001)==(j+2)){
                       rownames(FisherScoring)<-c(paste("b",seq(1,j+1)),"sigma_q")
                       colnames(FisherScoring)<-seq(0,(i-1))
                       print(FisherScoring)
                       break
                                                  }
    phit0<-phit1
}
```

```
                  0          1          2
b 1       1.9257813  1.9257812  1.9257812
b 2       1.2968750  1.2968750  1.2968750
b 3       0.4023438  0.4023438  0.4023437
b 4      -0.6523438 -0.6523437 -0.6523438
sigma_q 115.6666667  0.6927083  0.6927083
```

«Ritroviamo» i risultati glm() all'interno della procedura iterativa...

```
k=3                       # variabili x nel modello lineare
0.6927083*n/(n-k-1)       # cambiamo il denominatore da n a (n-k-1) Equazione 24
[1] 2.078125
sqrt(2.078125)            # errore standard dei residui
[1] 1.44157

vcov.B<-(-1*solve(H))[1:(j+1),1:(j+1)]  # minore delle var/cov(distorte) di B
diag(vcov.B)                            # diagonale delle var (distorte) di B
[1] 0.79203712 0.11725532 0.02085792 0.10744646
diag(vcov.B)*n/(n-k-1)                  # var (corrette per i gdl) di B
[1] 2.37611135 0.35176595 0.06257375 0.32233938

sqrt(diag(vcov.B)*n/(n-k-1))            # errori standard dei coefficienti B
[1] 1.5414640 0.5930986 0.2501475 0.5677494
```