Corso R - Parte 2

Gianluca Mastrantonio

Contents

1 Introduzione

In questa seconda vedremo i comandi R che si possono utilizzare per fare test e la regressione. Il file precedente e questo contengono tutti i comandi che dovete conoscere usate in dei semplici esempi.

2 Test, p-value e Intervalli di confidenza

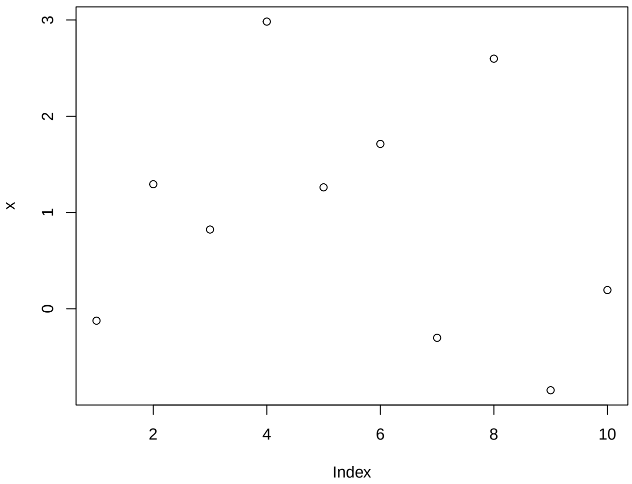
2.1 Z-test

Ipotizziamo di avere $X_i \sim N(\mu, \sigma^2)$, con $i = 1, 2, \dots, n$ e di voler testare la seguente ipotesi

$$H_0 : \mu = \mu_0 \;\; H_1 : \mu \neq \mu_0$$

Simuliamo dei dati

```
set.seed(100)
n = 10
mu = 1
sigma2 = 5
x = rnorm(n, mu, sigma2^0.5)
plot(x)
```



Abbiamo due possibili casi

- σ^2 noto
- σ^2 non noto

Nel primo caso la statistica da utilizzare è

$$Z = \frac{\bar{X} - \mu_0}{\sqrt{\frac{\sigma^2}{n}}} \sim N(0,1)$$

che ci porta alla regione di accettazione

$$\bar{X} \in \left[\mu_0 - z_{\alpha/2}\sqrt{\frac{\sigma^2}{n}}, \mu_0 + z_{\alpha/2}\sqrt{\frac{\sigma^2}{n}}\right]$$

Proviamo a vedere se accettiamo H_0 con $\mu_0 = 0$ e $\alpha = 0.05$

```
mu_0 = 0
xbar = mean(x)
alpha = 0.05
z_alpha2 = qnorm(alpha/2, 0, 1, lower.tail=F )

(xbar >= mu_0-z_alpha2*sqrt(sigma2/n)) & (xbar <= mu_0+z_alpha2*sqrt(sigma2/n))
```

```
## [1] TRUE
```

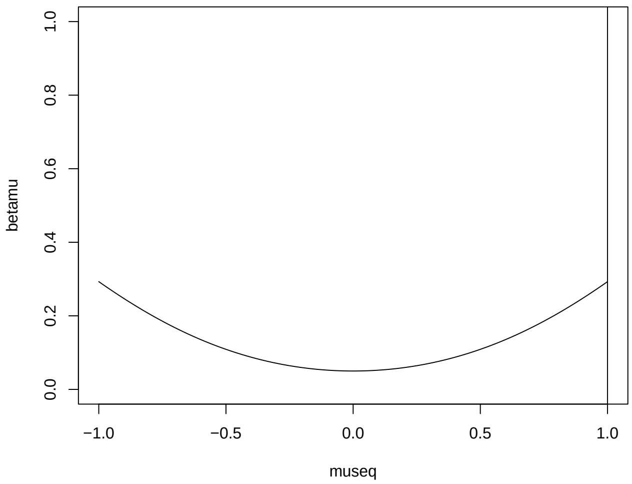
Possiamo calcolare la funzione potenza di questo test e vedere quanto era la probabilità di rifiutare H_0 con il vero valore della media. In questo caso la funzione potenza è

$$\beta(\mu) = 1 - P(\bar{X} \in [\mu_0 - z_{\alpha/2}\sqrt{\sigma^2/n}, \mu_0 + z_{\alpha/2}\sqrt{\sigma^2/n})$$

con

$$\bar{X} \sim N(\mu, \sigma^2/n)$$

```
museq = seq(-1,1, by = 0.01)
betamu =  pnorm(mu_0 -z_alpha2*sqrt(sigma2/n), museq , sqrt(sigma2/n)) +
          pnorm(mu_0 +z_alpha2*sqrt(sigma2/n), museq , sqrt(sigma2/n),
          lower.tail=F)
plot(museq,betamu, ylim=c(0,1), type="l" )
abline(v=mu)
```




```
betamu[museq==mu]
```

```
## [1] 0.2929889
```

Abbiamo quindi che

$$\beta(1) = 0.2929889$$

quindi nell maggior parte dei casi non rifiutiamo H_0

Possiamo calcolare il p-value del test appena fatto. Ricordando che nel caso di ipotesi alternativa bilaterale, la definizione è

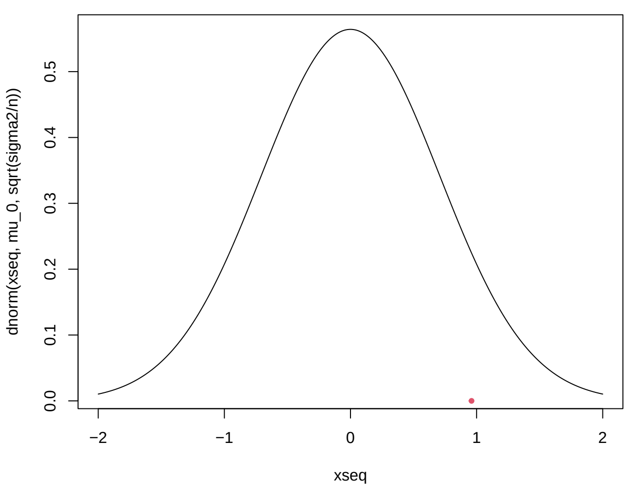
$$2\min(P(W(\mathbf{X}) \leq W(\mathbf{x})), P(W(\mathbf{X}) \geq W(\mathbf{x})))$$

Calcoliamo il p-value e vediamo dove cade il valore osservato di $W()$, che in questo caso è la media campionaria

```
pvalue = 2*min(
          pnorm(xbar, mu_0,  sqrt(sigma2/n  )),
          pnorm(xbar, mu_0,  sqrt(sigma2/n  ) ,lower.tail=F )
          )
pvalue
```

```
## [1] 0.1746452
```

```
xseq = seq(-2,2, by = 0.01)
plot(xseq, dnorm(xseq, mu_0, sqrt(sigma2/n ) ), type="l")
points(xbar, 0, col=2, pch=20)
```



Per concludere il test Z troviamo l'intervallo di confidenza di livello $1-\alpha$ di μ, che in questo caso è

$$\mu \in \left[\bar{x} - z_{\alpha/2}\sqrt{\frac{\sigma^2}{n}}, \bar{x} + z_{\alpha/2}\sqrt{\frac{\sigma^2}{n}}\right]$$

```
c(xbar - z_alpha2*sqrt(sigma2/n), xbar+z_alpha2*sqrt(sigma2/n))
```

```
## [1] -0.4260573  2.3457504
```

2.2 T-test

Rifacciamo l'esercizio ma questa volta assumendo σ^2 non nota. La statistica da utilizzare è

$$\frac{\bar{X} - \mu_0}{\sqrt{\frac{S^2}{n}}} \sim T(n-1)$$

Invece di fare i calcoli direttamente, utilizziamo la funzione R t.test. La funzione ha diversi parametri ma a noi interessano i seguenti

```
t.test(
  x,  # il campione
  y = NULL, # da usare solo se si voglino confrontare le medie di due campioni
  alternative = c("two.sided", "less", "greater"), # il tipo di H_1
  mu = 0,  # il valore sotto H_0 o il limite del dominio sotto H_0
  var.equal = FALSE, # se le varianze sono uguale quando si confronta x e y
  conf.level = 0.95 # 1-alpha, o la confidenza dell'intervallo di confidenza
  )
```

Possiamo allora applicarlo al dataset simulato e abbiamo

```
t.test(x,alternative = c("two.sided", "less", "greater")[1], conf.level = 1-alpha,mu = 0)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 2.4191, df = 9, p-value = 0.03866
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.06228871 1.85740442
## sample estimates:
## mean of x
## 0.9598466
```

e ci mostra il valore della statistica osservata

$$t = \frac{\bar{X} - \mu_0}{\sqrt{\frac{S^2}{n}}}$$

i gradi di libertà (df), il p-value, l'intervallo di confidenza e il valore della media campionaria. Basta guardare il p-value e se questo è minore di α allora rifiutiamo.

Possiamo anche vedere cosa succede se prendiamo un'ipotesi alternativa unilaterale

```
t.test(x,alternative = c("two.sided", "less", "greater")[2], conf.level = 1-alpha,mu = 0)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 2.4191, df = 9, p-value = 0.9807
## alternative hypothesis: true mean is less than 0
## 95 percent confidence interval:
##      -Inf 1.687172
## sample estimates:
## mean of x
## 0.9598466
```

```
t.test(x,alternative = c("two.sided", "less", "greater")[3], conf.level = 1-alpha,mu = 0)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 2.4191, df = 9, p-value = 0.01933
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  0.2325209       Inf
## sample estimates:
## mean of x
## 0.9598466
```

Notate come l'intervallo di confidenza diventa di lunghezza infinita.

2.3 Altri test importanti

Per completare il set di test che dovete sapere, dobbiamo vedere i test sulla varianza e test per due popolazioni normali. Per il test sulla varianza testiamo

$$H_0 : \sigma^2 = \sigma_0^2, \;\; H_1 : \sigma^2 \neq \sigma_0^2$$

e possiamo usare la statistica

$$\frac{(n-1)S^2}{\sigma_0^2} \sim \chi^2(n-1)$$

Anche qui possiamo usare una funzione R che è nel pacchetto **EnvStats** e si chiama varTest, che ha parametri simili a quelli della funzione t.test

```
varTest(
  x,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  sigma.squared = 1) # il valore sotto H_0 o il limite del dominio sotto H_0
  )
```

testiamo l'ipotesi con $\sigma_0^2 = 10$

```
library(EnvStats)
```

```
##
## Attaching package: 'EnvStats'

## The following objects are masked from 'package:stats':
##
##     predict, predict.lm
```

```
## The following object is masked from 'package:base':
##
##     print.default
```

```r
varTest(x,
  alternative = c("two.sided", "less", "greater")[1],
  conf.level = 1-alpha,
  sigma.squared = 10)
```

```
## $statistic
## Chi-Squared
##    1.416843
##
## $parameters
## df
##  9
##
## $p.value
## [1] 0.004564131
##
## $estimate
## variance
##  1.57427
##
## $null.value
## variance
##       10
##
## $alternative
## [1] "two.sided"
##
## $method
## [1] "Chi-Squared Test on Variance"
##
## $data.name
## [1] "x"
##
## $conf.int
##       LCL       UCL
## 0.7448146 5.2468114
## attr(,"conf.level")
## [1] 0.95
##
## attr(,"class")
## [1] "htestEnvStats"
```

l'output è meno compatto di quello di t.test, ma da le stesse informazioni. Fate atenzione che **Chi-Squared** indica il valore osservato della statistica

$$\frac{(n-1)S^2}{\sigma_0^2}$$

Per concludere, vediamo il test T sulle medie quando si hanno due popolazioni normali, e il test F sul rapporto delle varianza. Simuliamo un nuovo dataset y con stessa varianza di x ma media diversa

```r
set.seed(200)
m = 20
y = rnorm(m, mu+1, sigma2^0.5)
```

E per prima cosa verifichiamo che le medie sono statisticamente differenti

```r
t.test(x,y,alternative = c("two.sided", "less", "greater")[1],
      conf.level = 1-alpha, mu = 0,  var.equal =T)
```

```
##
##  Two Sample t-test
##
## data:  x and y
## t = -1.4268, df = 28, p-value = 0.1647
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.1699010  0.3881413
## sample estimates:
## mean of x mean of y
## 0.9598466 1.8507264
```

ricordatevi di specificare se le varianze sono supposte uguali o no. Il p-value è molto elevato e quindi non possiamo rifiutare H_0. Questo si vede anche dall'intervallo di confidenza che contiene lo zero. Come vedete i gradi di libertà sono m+n-2.

L'ultimo test che vediamo è il test del rapporto di varianza. Qui stiamo supponendo che $X_i \sim N(\mu_x, \sigma_x^2)$ e $Y_i \sim N(\mu_y, \sigma_y^2)$ e vogliamo fare un test del tipo

$$H_0 : \sigma_x^2/\sigma_y^2 = c \;\; H_0 : \sigma_x^2/\sigma_y^2 \neq c$$

e possiamo utilizzare la statistica

$$\frac{\sigma_y^2 S_x^2}{\sigma_x^2 S_y^2} \sim F(n-1, m-1)$$

Questo test si può fare con la funzione **var.test** che ha una sintassi simile alle altre

```r
var.test(x,
        y,
        ratio = 1, # il valore c
        alternative = c("two.sided", "less", "greater"),
        conf.level = 0.95)
```

Supponiamo che c=2 e vediamo i risultati

```r
var.test(x,
        y,
        ratio = 2, # il valore c
        alternative = c("two.sided", "less", "greater")[1],
        conf.level = 1-alpha)
```

```
##
##  F test to compare two variances
##
## data:  x and y
## F = 0.25518, num df = 9, denom df = 19, p-value = 0.04083
## alternative hypothesis: true ratio of variances is not equal to 2
## 95 percent confidence interval:
##  0.1772055 1.8798319
```

```
## sample estimates:
## ratio of variances
##          0.5103609
```

possiamo rifiutare H_0 visto che il p-value è minore di 0.05

2.4 Esercizi

1. Verificate di ottenere gli stessi risultati se utilizzate la statistica Z^2, che è distribuite come una $\chi^2(n)$, nel test Z, intermini di p-value e intervallo di confidenza per μ
2. Valutare graficamente quale n serve per poter rifiutare H_0 con probabilità 0.8 quando $\mu = 1$ nel test Z e test T
3. tramite simulazione verificate che $\beta(1) = 0.2929$ nel test Z (simulate diversi campioni di dimensione n e per ognuno verificate se accettate o no H_0. Per la legge dei grandi numeri, la proporzione di rifiuti sul totale è una stima di $\beta(1)$)
4. Rifare il test, p-value e intervalli di confidenza con un ipotesi alternativa unilaterale nel test Z e test T
5. rifare il test T (test, p-value e IC) senza l'utilizzo della funzione t.test e verificare che i vostri calcoli corrispondono a quelli di t.test
6. Applicate i test ai dataset **Morley** e **Sleep** (quelli che hanno senso) e calcolate il tutto sia con le funzioni R che "manualmente" (senza le funzioni di R che fanno direttamente i test)

3 Regressione - Anova - Ancova

3.1 Modello Regressivo

Immaginiamo di voler studiare la relazione tra una variabile x e y con un modello regressivo del tipo

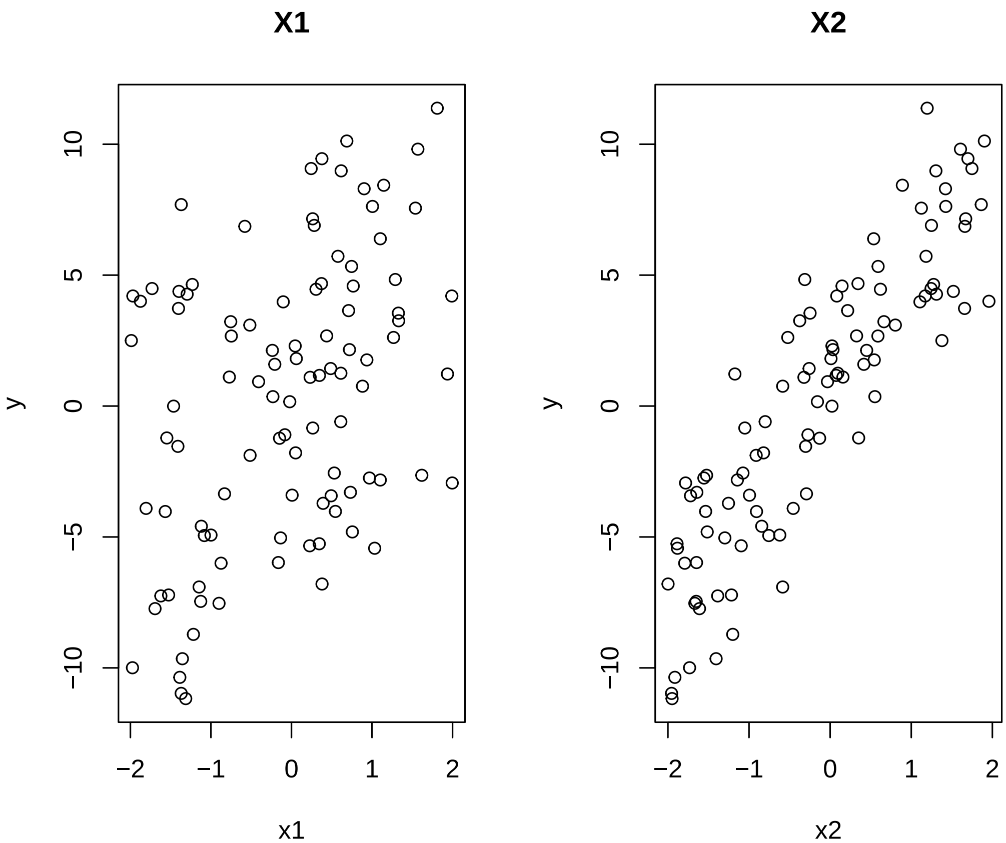
$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} + \epsilon_i$$

con $\epsilon_i \sim N(0, \sigma^2)$. Simuliamo dei dati

```
n = 100
sigma2= 2
x1 = runif(n,-2,2)
x2 = runif(n,-2,2)

beta = c(1,2,4)

y = beta[1]+ beta[2]*x1+beta[3]*x2 + rnorm(n, 0,sigma2^0.5)
par(mfrow=c(1,2))
plot(x1,y, main="X1")
plot(x2,y, main="X2")
```



Per stimare i parametri del modello possiamo usare la funzione **lm** . Nel caso specifico possiamo stimare la seguente regressione

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,2}^2 + \epsilon_i$$

```
reg = lm(y~ x1+x2+I(x2^2))
```

dove vanno specificate solo le variabili che si vogliono mettere nel modello, mentre l'intercetta è aggiunta di default. Quando si usa una funzione di una variabile, in questo caso x^2, va messa dentro I() in modo che R capisca che è una trasformazione e non il nome della variabile. Per vedere il risultato della regressione usiamo la funzione **summary()**

```
summary(reg)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + I(x2^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4588 -0.8531  0.1596  0.8521  2.4279
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.13991    0.18740   6.083  2.4e-08 ***
## x1           1.98444    0.11867  16.722  < 2e-16 ***
## x2           4.12336    0.10918  37.767  < 2e-16 ***
## I(x2^2)     -0.06623    0.10416  -0.636    0.526
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.248 on 96 degrees of freedom
## Multiple R-squared:  0.9489, Adjusted R-squared:  0.9473
## F-statistic: 594.1 on 3 and 96 DF,  p-value: < 2.2e-16
```

la colonna **Estimate** ha le stime di massima verosimiglianza, **Std.error** è la radice della varianza delle stime di massima verosimiglianza, mentre le ultime colonne sono il valore della statistica t e il p-value in test del tipo

$$H_0 : \beta_j = 0 \quad H_1 : \beta_j \neq 0$$

Dalla colonna dei p-value possiamo vedere che accettiamo l'ipotesi nulla per x^2, che quindi dobbiamo supporre avere coefficiente uguale a zero, mentre per gli altri rifiutiamo. R ci da anche il valore R^2 in **R-squared**. Molto volte è utile vedere il modello in forma multivariata

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

Possiamo chiedere a R di farci vedere la matrice **X** che utilizza con

```r
X = model.matrix(~ x1+x2+I(x2^2))
X[1:10,]
```

```
##    (Intercept)         x1          x2     I(x2^2)
## 1            1  0.3474266  0.07453588 0.005555598
## 2            1  0.8822801 -0.58482777 0.342023522
## 3            1  0.7666921  0.14693170 0.021588925
## 4            1 -1.2953982  1.31086725 1.718372949
## 5            1  1.8099035  1.19634578 1.431243234
## 6            1  0.7560476 -1.51518919 2.295798291
## 7            1 -1.3688016  1.86317152 3.471408115
## 8            1  0.3048840  0.62039025 0.384884062
## 9            1  0.6131070  0.09465507 0.008959582
## 10           1  1.3312317 -0.37493597 0.140576980
```

che come vedete ha nella prima colonna tutti 1 perchè è l'intercetta.

L'oggetto creato da **lm** ha tantissime cosa al suo interno, e per vederlo potete utilizzare

```r
str(reg)
```

```
## List of 12
##  $ coefficients : Named num [1:4] 1.1399 1.9844 4.1234 -0.0662
##   ..- attr(*, "names")= chr [1:4] "(Intercept)" "x1" "x2" "I(x2^2)"
##  $ residuals    : Named num [1:100] -0.966 0.301 1.317 0.411 1.805 ...
##   ..- attr(*, "names")= chr [1:100] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:100] -2.451 -21.852 -47.959 0.794 1.642 ...
##   ..- attr(*, "names")= chr [1:100] "(Intercept)" "x1" "x2" "I(x2^2)" ...
##  $ rank         : int 4
##  $ fitted.values: Named num [1:100] 2.136 0.457 3.266 3.861 9.57 ...
##   ..- attr(*, "names")= chr [1:100] "1" "2" "3" "4" ...
##  $ assign       : int [1:4] 0 1 2 3
##  $ qr           :List of 5
##   ..$ qr   : num [1:100, 1:4] -10 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:100] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:4] "(Intercept)" "x1" "x2" "I(x2^2)"
##   .. ..- attr(*, "assign")= int [1:4] 0 1 2 3
##   ..$ qraux: num [1:4] 1.1 1.09 1.03 1.05
```

```
##    ..$ pivot: int [1:4] 1 2 3 4
##    ..$ tol  : num 1e-07
##    ..$ rank : int 4
##    ..- attr(*, "class")= chr "qr"
##   $ df.residual  : int 96
##   $ xlevels      : Named list()
##   $ call         : language lm(formula = y ~ x1 + x2 + I(x2^2))
##   $ terms        :Classes 'terms', 'formula'  language y ~ x1 + x2 + I(x2^2)
##    .. ..- attr(*, "variables")= language list(y, x1, x2, I(x2^2))
##    .. ..- attr(*, "factors")= int [1:4, 1:3] 0 1 0 0 0 0 1 0 0 0 ...
##    .. .. ..- attr(*, "dimnames")=List of 2
##    .. .. .. ..$ : chr [1:4] "y" "x1" "x2" "I(x2^2)"
##    .. .. .. ..$ : chr [1:3] "x1" "x2" "I(x2^2)"
##    .. ..- attr(*, "term.labels")= chr [1:3] "x1" "x2" "I(x2^2)"
##    .. ..- attr(*, "order")= int [1:3] 1 1 1
##    .. ..- attr(*, "intercept")= int 1
##    .. ..- attr(*, "response")= int 1
##    .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##    .. ..- attr(*, "predvars")= language list(y, x1, x2, I(x2^2))
##    .. ..- attr(*, "dataClasses")= Named chr [1:4] "numeric" "numeric" "numeric" "numeric"
##    .. .. ..- attr(*, "names")= chr [1:4] "y" "x1" "x2" "I(x2^2)"
##   $ model        :'data.frame':   100 obs. of  4 variables:
##    ..$ y      : num [1:100] 1.17 0.758 4.583 4.272 11.375 ...
##    ..$ x1     : num [1:100] 0.347 0.882 0.767 -1.295 1.81 ...
##    ..$ x2     : num [1:100] 0.0745 -0.5848 0.1469 1.3109 1.1963 ...
##    ..$ I(x2^2): 'AsIs' num [1:100] 0.005555.... 0.342023.... 0.021588.... 1.718372.... 1.431243.... .
##    ..- attr(*, "terms")=Classes 'terms', 'formula'  language y ~ x1 + x2 + I(x2^2)
##    .. .. ..- attr(*, "variables")= language list(y, x1, x2, I(x2^2))
##    .. .. ..- attr(*, "factors")= int [1:4, 1:3] 0 1 0 0 0 0 1 0 0 0 ...
##    .. .. .. ..- attr(*, "dimnames")=List of 2
##    .. .. .. .. ..$ : chr [1:4] "y" "x1" "x2" "I(x2^2)"
##    .. .. .. .. ..$ : chr [1:3] "x1" "x2" "I(x2^2)"
##    .. .. ..- attr(*, "term.labels")= chr [1:3] "x1" "x2" "I(x2^2)"
##    .. .. ..- attr(*, "order")= int [1:3] 1 1 1
##    .. .. ..- attr(*, "intercept")= int 1
##    .. .. ..- attr(*, "response")= int 1
##    .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##    .. .. ..- attr(*, "predvars")= language list(y, x1, x2, I(x2^2))
##    .. .. ..- attr(*, "dataClasses")= Named chr [1:4] "numeric" "numeric" "numeric" "numeric"
##    .. .. .. ..- attr(*, "names")= chr [1:4] "y" "x1" "x2" "I(x2^2)"
##   - attr(*, "class")= chr "lm"
```

e per esempio possiamo anche estrarre i valori numerici dei coefficienti, che sono in **$coefficients** dentro **reg**

```
reg$coefficients
```

```
## (Intercept)          x1          x2     I(x2^2)
##  1.13991242  1.98444009  4.12335608 -0.06622736
```

Facciamo un muovo modello in cui mettiamo l'interazione tra x_1 e x_2.

```
reg2 = lm(y~ x1+x2+I(x2^2)+x1*x2)
summary(reg2)
```

```
##
## Call:
```

```
## lm(formula = y ~ x1 + x2 + I(x2^2) + x1 * x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3851 -0.8367  0.1945  0.8438  2.3953
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.14617    0.18737   6.117 2.11e-08 ***
## x1           1.96856    0.11953  16.470  < 2e-16 ***
## x2           4.10065    0.11117  36.885  < 2e-16 ***
## I(x2^2)     -0.07217    0.10424  -0.692     0.49
## x1:x2       -0.10019    0.09417  -1.064     0.29
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.248 on 95 degrees of freedom
## Multiple R-squared:  0.9495, Adjusted R-squared:  0.9474
## F-statistic: 446.5 on 4 and 95 DF,  p-value: < 2.2e-16
```

e vediamo la matrice $\mathbf{X}$.

```
X2 = model.matrix(~ x1+x2+I(x2^2)+x1*x2)
X2[1:10,]
```

```
##    (Intercept)         x1          x2     I(x2^2)       x1:x2
## 1            1  0.3474266  0.07453588 0.005555598  0.02589575
## 2            1  0.8822801 -0.58482777 0.342023522 -0.51598190
## 3            1  0.7666921  0.14693170 0.021588925  0.11265138
## 4            1 -1.2953982  1.31086725 1.718372949 -1.69809502
## 5            1  1.8099035  1.19634578 1.431243234  2.16527044
## 6            1  0.7560476 -1.51518919 2.295798291 -1.14555510
## 7            1 -1.3688016  1.86317152 3.471408115 -2.55031211
## 8            1  0.3048840  0.62039025 0.384884062  0.18914706
## 9            1  0.6131070  0.09465507 0.008959582  0.05803368
## 10           1  1.3312317 -0.37493597 0.140576980 -0.49912663
```

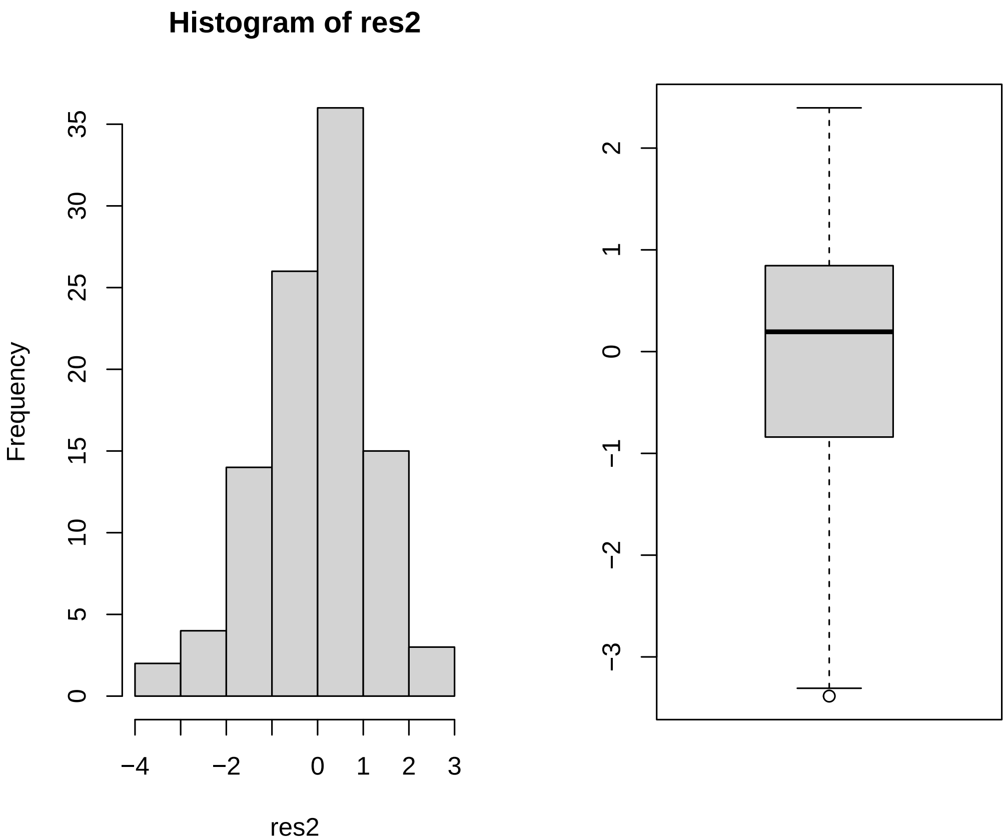
è facile verificare che l'interazione è la moltiplicazione tra le variabili x_1 e x_2 (il `:` nel nome della colonna indica l'interazione, e non è da leggere come segno di divisione)

```
X2[1:10,5] - X2[1:10,2]*X2[1:10,3]
```

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

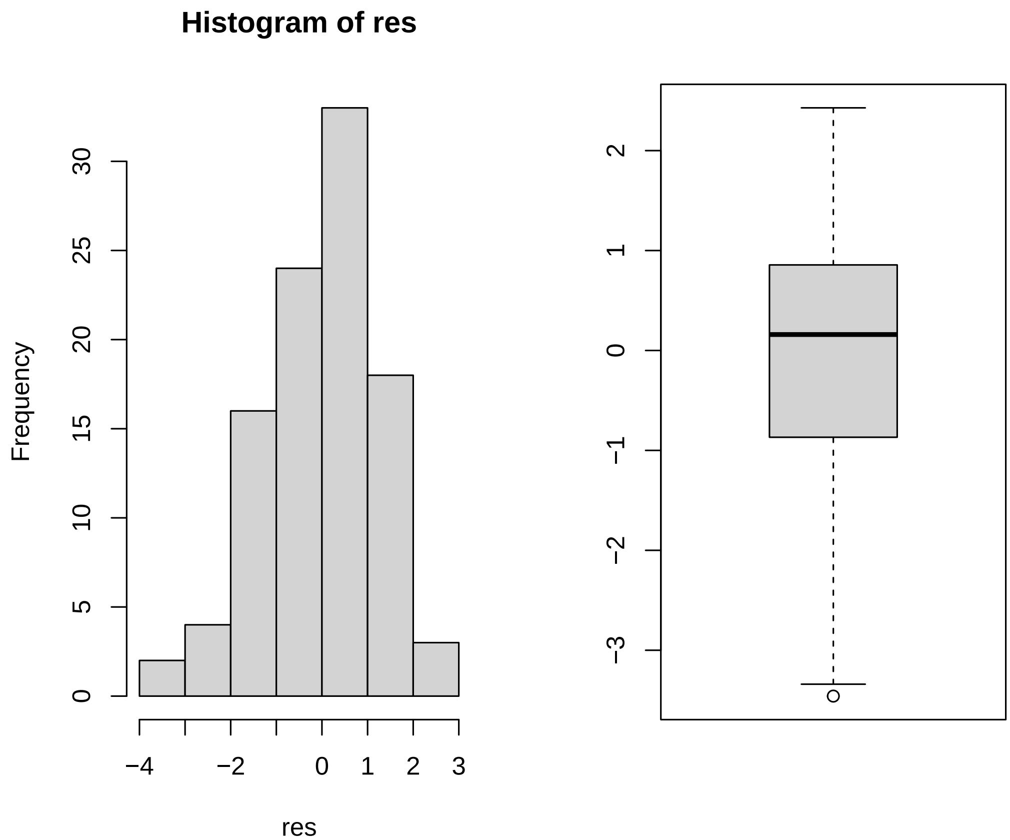
Prendiamo la seconda regressione e calcoliamo i residui

```
res2 = residuals(reg2)
par(mfrow=c(1,2))
hist(res2)
boxplot(res2)
```



e poi vediamo quelli della prima

```r
res = residuals(reg)
par(mfrow=c(1,2))
hist(res)
boxplot(res)
```



Anche se non facciamo un test, possiamo vedere che i residui sono approssimativamente normali.

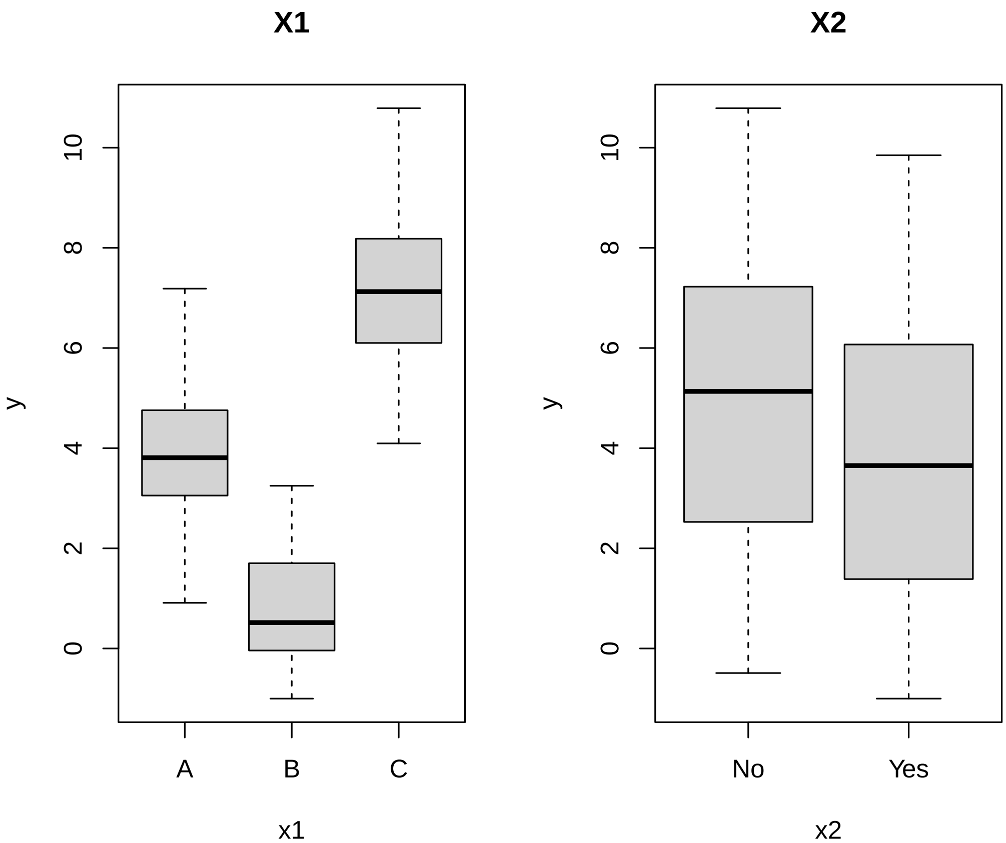
3.2 Modello Anova

Il modello ANOVA è un modello che ha solo variabili qualitative. Simuliamo un modello ANOVA, ricordando che le variabili qualitative vanno passate come fattori

```
n = 100
sigma2= 2
x1 = factor(sample(c("A","B","C"),n,replace=T))
x2 = factor(sample(c("Yes","No"),n,replace=T))

X = model.matrix(~ x1+x2)
ncov = ncol(X)

y = X%*%matrix(rnorm(ncov,0,5), ncol=1) + rnorm(n, 0,sigma2^0.5)
par(mfrow=c(1,2))
boxplot(y~ x1, main="X1")
boxplot(y~ x2, main="X2")
```



Per stimare il modello si usa la stessa funzione **lm**

```
anovareg = lm(y~ x1+x2)
summary(anovareg)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5321 -0.7853  0.0147  0.7251  3.3615
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.6157     0.2589  17.831  < 2e-16 ***
## x1B          -3.0330     0.3174  -9.555 1.34e-15 ***
## x1C           3.2947     0.3024  10.896  < 2e-16 ***
## x2Yes        -1.4232     0.2501  -5.691 1.37e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.249 on 96 degrees of freedom
## Multiple R-squared:  0.8289, Adjusted R-squared:  0.8235
## F-statistic:   155 on 3 and 96 DF,  p-value: < 2.2e-16
```

l'output è uguale a quello del modello regressivo, ma naturalmente l'interpretazione dei parametri cambia. Se vogliamo cambiare il corner (il livello della variabile mancante nell'output, per esempio A in x_1) di una della variabili qualitative, possiamo usare la funzione **relevel**

```
x1 = relevel(x1, ref="C")
anovareg2 = lm(y~ x1+x2)
summary(anovareg2)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5321 -0.7853  0.0147  0.7251  3.3615
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.9105     0.2417  32.735  < 2e-16 ***
## x1A          -3.2947     0.3024 -10.896  < 2e-16 ***
## x1B          -6.3277     0.3025 -20.915  < 2e-16 ***
## x2Yes        -1.4232     0.2501  -5.691 1.37e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.249 on 96 degrees of freedom
## Multiple R-squared:  0.8289, Adjusted R-squared:  0.8235
## F-statistic:   155 on 3 and 96 DF,  p-value: < 2.2e-16
```

Come vedete adesso nell'output della regressione "manca'' il valore ''C" di x_1. Se siamo interessati a un modello senza intercetta, possiamo stimarlo in questo modo

```
anovareg3 = lm(y~ -1+x1+x2)
summary(anovareg3)
```

```
##
## Call:
## lm(formula = y ~ -1 + x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5321 -0.7853  0.0147  0.7251  3.3615
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## x1C     7.9105     0.2417  32.735  < 2e-16 ***
## x1A     4.6157     0.2589  17.831  < 2e-16 ***
## x1B     1.5827     0.2549   6.208 1.36e-08 ***
## x2Yes  -1.4232     0.2501  -5.691 1.37e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.249 on 96 degrees of freedom
## Multiple R-squared:  0.9433, Adjusted R-squared:  0.9409
## F-statistic:   399 on 4 and 96 DF,  p-value: < 2.2e-16
```

Possiamo anche creare interazione con

```r
anovareg4 = lm(y~ -1+x1+x2 +x1*x2)
summary(anovareg4)
```

```
##
## Call:
## lm(formula = y ~ -1 + x1 + x2 + x1 * x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7117 -0.8539 -0.0378  0.7340  3.4263
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## x1C         7.9824     0.2944  27.113  < 2e-16 ***
## x1A         4.7953     0.3225  14.868  < 2e-16 ***
## x1B         1.3334     0.3123   4.270 4.67e-05 ***
## x2Yes      -1.5599     0.4058  -3.844  0.00022 ***
## x1A:x2Yes  -0.2111     0.6052  -0.349  0.72794
## x1B:x2Yes   0.6520     0.6052   1.077  0.28405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.249 on 94 degrees of freedom
## Multiple R-squared:  0.9445, Adjusted R-squared:  0.9409
## F-statistic: 266.4 on 6 and 94 DF,  p-value: < 2.2e-16
```

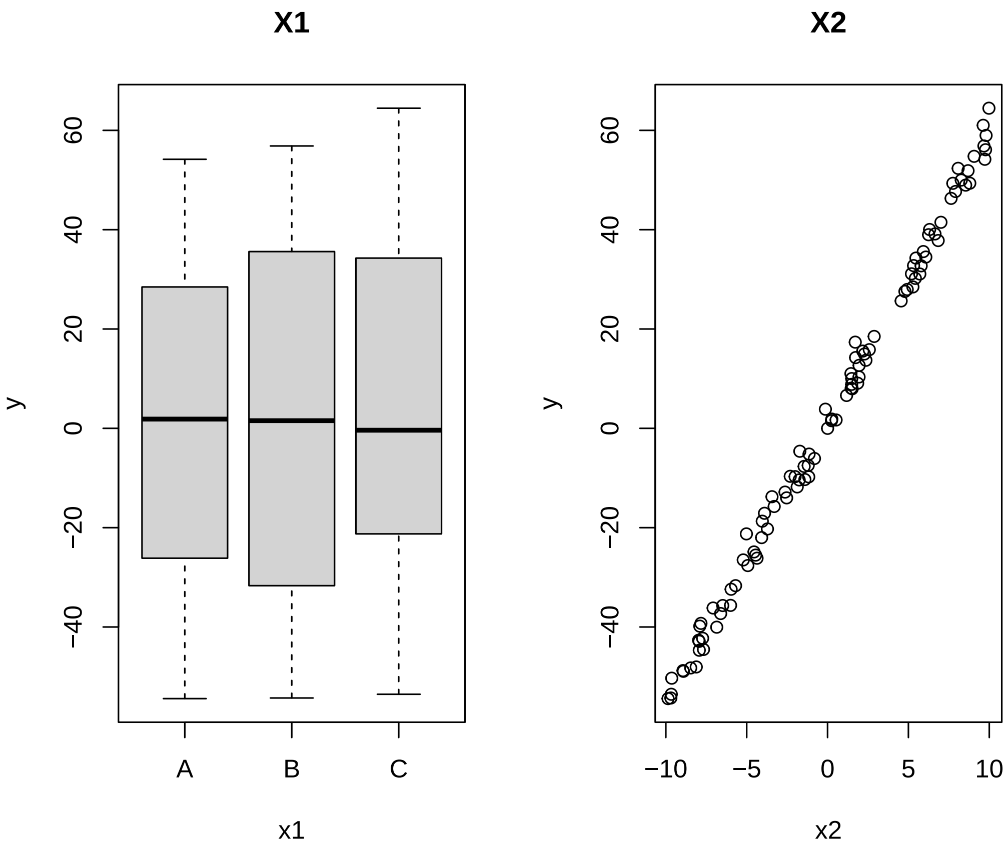
3.3 Modello Ancova

Un modello Ancova è un modello con predittori numerici e fattoriali. Simuliamo un modello di questo tipo con una variabile fattoriale e una numerica

```r
n = 100
sigma2= 2
x1 = factor(sample(c("A","B","C"),n,replace=T))
x2 = runif(n,-10,10)

X = model.matrix(~ x1+x2)
ncov = ncol(X)

y = X%*%matrix(rnorm(ncov,0,5), ncol=1) + rnorm(n, 0,sigma2^0.5)
par(mfrow=c(1,2))
boxplot(y~ x1, main="X1")
plot(x2,y, main="X2")
```



Anche per questo modello si può utilizzare la funzione **lm**. Ipotizziamo di voler stimare un modello con x_1, x_2, e la loro interazione

```
ancovareg = lm(y ~ x1+x2+x1*x2)
summary(ancovareg)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x1 * x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1146 -0.9456 -0.0233  0.8585  3.8283
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.30477    0.24670  -1.235    0.220
## x1B          1.78705    0.34891   5.122 1.61e-06 ***
## x1C          4.54695    0.38402  11.840  < 2e-16 ***
## x2           5.66302    0.04614 122.744  < 2e-16 ***
## x1B:x2       0.05104    0.06050   0.844    0.401
## x1C:x2       0.10501    0.06712   1.564    0.121
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.5 on 94 degrees of freedom
## Multiple R-squared:  0.9981, Adjusted R-squared:  0.998
## F-statistic: 1.011e+04 on 5 and 94 DF,  p-value: < 2.2e-16
```

3.4 Esercizi

1. Calcolate lo stimatore di massima verosimiglianza della prima regressione
2. Calcolare p-value e t dei test sui coefficienti regressivi senza utilizzare la funzione lm
3. Nella regressione con interazione, verificate come R calcola la colonna **x1:x2** nella matrice **X**
4. Nella regressione con interazione calcolare i residui senza utilizzare la funzione **res** e valutare che sono uguali a quelli di R
5. Verificate che anovareg, anovareg2 e anovareg3 sono lo stesso modello calcolando la media di y in tutte le possibili combinazioni di x1 e x2, usando per i calcoli i parametri del modello.
6. Calcolate il t, e il p-value delle ipotesi sui parametri regressivi di anovareg3 e anovareg4
7. descrivere come è fatta la matrice **X** in anovareg2, anovareg3 e anovareg4
8. determinare la matrice **X** nel modello ancovareg