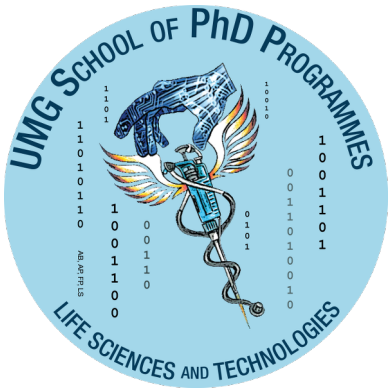
SOCIETA' DEI MATEMATICI
E NATURALISTI DI MODENA
www.socnatmatmo.unimore.it



Massimo Borelli, Ph.D.

il dataset `mieloma`
dati sigmoidali analizzati con R

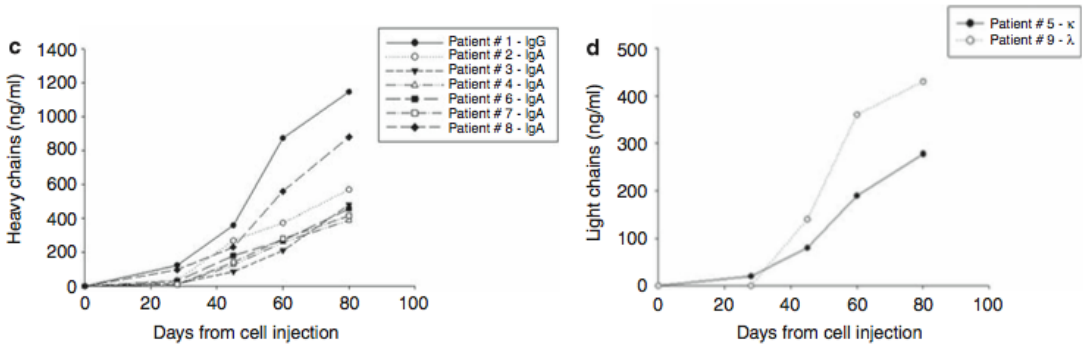
28 marzo 2017

Riproduzione pittorica di foto d'epoca a cura di Caterina Stella.

Giovanni Canestrini, nato a Revò (Tn) nel 1835 e defunto a Padova nel 1900, fu fondatore nel 1865 della Società dei Naturalisti e Matematici di Modena. Nata con lo scopo di promuovere lo studio delle Scienze Naturali e Matematiche, di favorire i legami fra studiosi, ricercatori, collezionisti e diffondere cultura scientifica a tutti i livelli per una equilibrata crescita civile, la Società dei Naturalisti e Matematici di Modena ha accolto nelle sua fila, tra gli altri, Charles Darwin, Louis Pasteur e Thomas Henry Huxley.

Indice

caro Massimo,
come vedi nelle figure (presta attenzione alle diverse scale sull'asse delle ordinate) la cinetica in entrambi i gruppi ha un andamento sigmoidale, ma la velocità di crescita ed i plateau mi sembrano ben diversi. Come posso esprimere questo giudizio in termini statistici?



1 Struttura del dataset `mieloma`

La dottoressa Teresa Calimeri ci ha cortesemente concesso i dati di uno studio relativo al mieloma multiplo [3]. Cerchiamo di capire bene quale struttura hanno i dati che sono presenti nel dataset `mieloma.csv`:

	patient	time	chain	paraprotein
1	p1	0	heavy	0
2	p1	25	heavy	122
3	p1	45	heavy	364
4	p1	60	heavy	860
5	p1	80	heavy	1145
6	p1	0	heavy	0
..	..	..	..	..
106	p9	0	light	0
107	p9	25	light	0
108	p9	45	light	150
109	p9	60	light	368
110	p9	80	light	425

```
www = "http://www.biostatisticaumg.it/dataset/mieloma.csv"
```

```
mieloma = read.csv(www, header = TRUE)
# mieloma = read.csv(file.choose(), header = TRUE)
attach(mieloma)
head(mieloma)
tail(mieloma)
str(mieloma)
```

Vi sono 110 osservazioni di 4 variabili (`patient` , `time` , `chain` e `paraprotein`), codificate nel formato *.csv* (il carattere separatore dei dati è una virgola). Vi sono 9 `patient` contrassegnati con le sigle `p1`, `p2`, .. , `p9`, e rappresentano la cosiddetta *variabile di gruppo*(rispetto alla variabile risposta `paraprotein`, che è l'oggetto di nostro interesse):

```
table(patient)
```

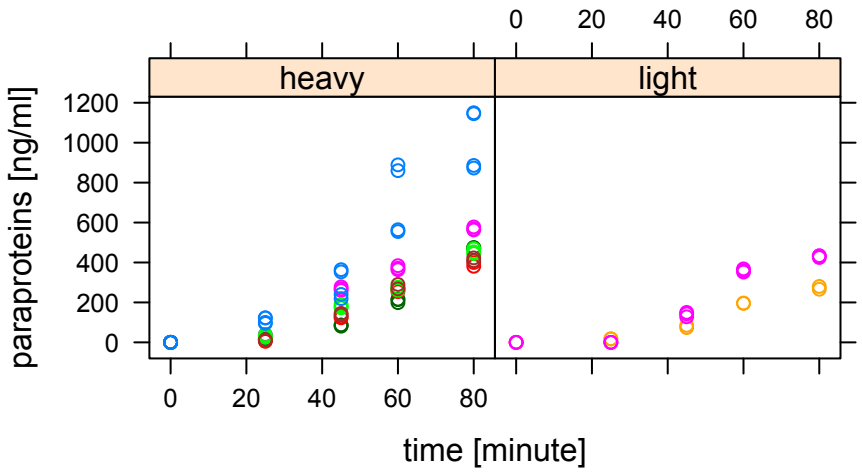
p1	p2	p3	p4	p5	p6	p7	p8	p9
10	15	15	10	10	15	10	10	15

Le frequenze 10 e 15 ci consentono di intuire che le osservazioni sono state condotte talvolta in duplicato e talvolta in triplicato. Infatti le occasioni temporali in cui sono state effettuate le misure sono cinque, rispettivamente a 0, 25, 45, 60 ed 80 minuti:

```
table(time)
```

	0	25	45	60	80
1	22	22	22	22	22

Ventidue infine sono gli esperimenti complessivamente effettuati, ossia il numero complessivo di traiettorie che vediamo raffigurate nel seguente *time plot*:




```
library(lattice)
xyplot(paraprotein ~ time | chain, type = "p",
   groups = patient, xlab = "time [minute]",
   ylab = "paraproteins [ng/ml]")
```

La prima cosa da fare, prima di procedere a formulare un modello statistico, è rendersi conto se gli ordini di grandezza delle variabili in esame siano confrontabili, per evitare problemi di congergenza degli algoritmi:

```
max(time)         # 80
max(paraprotein)  # 1150
```

Siccome la risposta `paraprotein` supera di più di un ordine di grandezza la variabile indipendente `time`, è opportuno riscalare le variabili, ad esempio con la funzione `scale` di R. Oppure, ad esempio, dividendo i valori di `paraprotein` per la mediana, e cambiando la scala del tempo in 'cinquine' di minuti:

```
response = paraproteins/150
fives = time/5
table(fives)          # 16
summary(response)     # 7.67
```

2 La funzione logistica

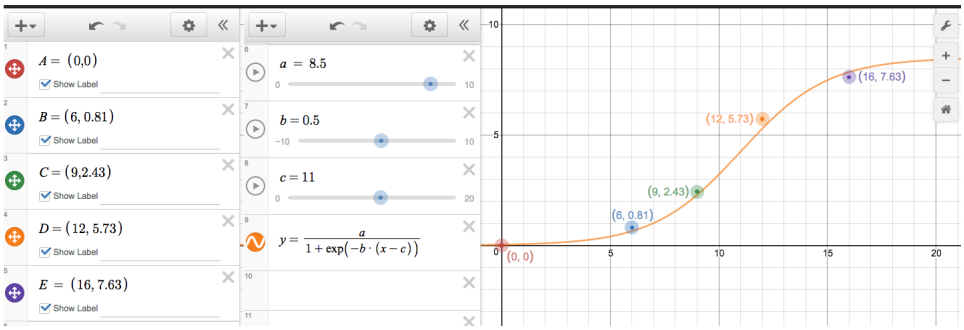
2.1 Il grafico della curva

Ricordiamo che con il termine di funzione **logistica**, o **sigmoide**, intendiamo una classe piuttosto generica di curve che hanno quindi diverse possibili espressioni matematiche. Una relazione piuttosto generale è quella della cosiddetta **logistica a quattro parametri**:

$$y = d + \frac{a - d}{1 + exp(-b(x - c))}$$

I parametri a e d sono i confini superiore ed inferiore che delimitano il grafico della sigmoide: il grafico è compreso cioè tra un asintoto inferiore (nel caso crescente $y = d$) ed uno superiore, il *plateau* (nel caso crescente $y = a$). Il parametro b determina la 'pendenza', la 'velocità di crescita' della sigmoide, mentre la costante c permette di traslare verso destra o verso sinistra la sigmoide, lasciandone inalterata la forma.

Per farsi un'idea della faccenda, la cosa migliore da fare è prendere, ad esempio, i primi cinque valori del dataset e visualizzarli con uno strumento interattivo. Ad esempio, Desmos (`https://www.desmos.com/calculator`) consente di introdurre i cinque punti, nominandoli con le lettere maiuscole, i quattro parametri e la funzione. Per mezzo degli *slider* possiamo cercare, ad occhio e croce, i valori dei parametri che meglio sembrino adattarsi ai punti raffigurati sul piano:



Nell'immagine qui sopra, si vede che abbiamo scelto di fissare $d \equiv 0$ come asintoto inferiore:

$$y = \frac{a}{1 + exp(-b(x - c))}$$

e si vede anche che rispettivamente $a = 8.5$, $b = 0.5$ e $c = 11$ possono essere dei valori dei parametri che verosimilmente approssimano i valori della sigmoide di regressione.

2.2 I parametri della logistica e la regressione nonlineare

Il software R offre varie possibilità [6] per effettuare una regressione nel caso che i parametri non siano lineari. Uno dei metodi più utilizzati, anche se forse il meno efficiente, è quello dato dalla funzione `nls` [7]. Essenzialmente, `nls` implementa un algoritmo ricorsivo che, per funzionare, necessita di una stima iniziale dei parametri che si vogliono 'fittare'. L'esperimento che abbiamo fatto con Desmos ci aiuta in tal senso: useremo proprio i valori di a, b e c che abbiamo appena trovato 'ad occhio e croce'.

Siccome ci sono 22 esperimenti, dobbiamo determinare 22 stime per i parametri in questione. Creiamo innanzitutto tre vettori vuoti:

```
stimaA = numeric(22)
stimaB = numeric(22)
stimaC = numeric(22)
```

e definiamo la funzione `logistica`:

```
logistica = function(x, a, b, c) {a/(1+exp(-b*(x-c)))}
```

Ora, con un ciclo *for* ripetuto 22 volte, leggiamo cinque valori consecutivi di `reponse` e di `fives` e ne determiniamo la migliore sigmoide di regressione, memorizzando i parametri a, b e c nei vettori `stimaA`, `stimaB` e `stimaC`.

```
for(i in 1:22)
{
y = response[c(5*i - 4, 5*i - 3, 5*i - 2, 5*i - 1, 5*i )]
x = fives[c(5*i - 4, 5*i - 3, 5*i - 2, 5*i - 1, 5*i )]
regressione = nls( y ~ logistica(x, a, b, c), start =  list(a = 8.5, b = 0.5, c = 11))
stimaA[i] = coef(regressione)[[1]]
stimaB[i] = coef(regressione)[[2]]
stimaC[i] = coef(regressione)[[3]]
}
```

Per capire meglio, supponendo che `y` ed `x` siano i vettori delle coordinate dei punti A, B, C, D ed E della figura precedente, ecco l'output del comando `nls` relativo:

```
> y = response[1:5]
> x = fives[1:5]
> regressione = nls( y ~ logistica(x, a, b, c), start =  list(a = 8.5, b = 0.5, c = 11))
> regressione
Nonlinear regression model
  model: y ~ logistica(x, a, b, c)
   data: parent.frame()
      a       b       c
 8.1469  0.5069 10.4546
 residual sum-of-squares: 0.1775

Number of iterations to convergence: 5
Achieved convergence tolerance: 2.451e-06
```

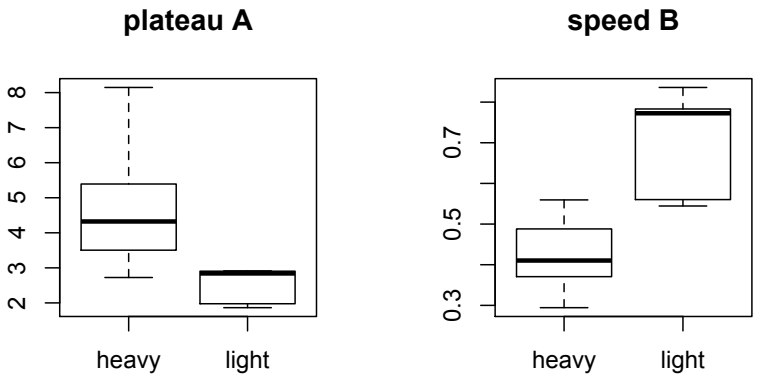
Come vedete, le stime finali di a, b e c del primo esperimento relativo a `p1` sono molto vicine a quelle che avevamo trovato 'ad occhio e croce'. In Appendice a questa dispensa, in ultima pagina, trovate l'elenco completo delle stime dei parametri, assieme alle due nuove variabili, `patient22` e `chain22`:

```
patient22 = patient[time == 0]
chain22 = chain[time == 0]
```

3 Il modello ad effetti misti

	patient22	chain22	stimaA	stimaB	stimaC
1	p1	heavy	8.15	0.51	10.45
2	p1	heavy	8.06	0.56	10.35
3	p2	heavy	4.32	0.38	10.76
4	p2	heavy	4.14	0.37	10.41
5	p2	heavy	4.07	0.41	10.21
6	p3	heavy	4.41	0.41	13.75
..	..	..	..	..	..
22	p9	light	2.84	0.84	9.75

Dunque, per mezzo della funzione `nls` ci siamo ricondotti allo studio di un nuovo dataset in cui sono stati stimati i parametri delle curve di regressione che descrivono i profili temporali dei dati del dataset `mieloma`. Siamo interessati a mostrare che la 'velocità' (`stimaB`) della sigmoide del gruppo `heavy` differisce da quella del gruppo `light`. Lo vediamo chiaramente nel boxplot che segue:



3.1 Perché non possiamo usare il t-test?

Non possiamo usare il t-test per saggiare la differenza di `stimaB` tra il gruppo `heavy` ed il gruppo `light` perché siamo in presenza di misure ripetute (e la variabile di gruppo è `patient22`), e quindi i t-value ed i p-value risulterebbero sovrastimati.

```
> moltomale = lm(stimaB ~ chain22 )
> summary(moltomale)

Call:
lm(formula = stimaB ~ chain22)

Residuals:
     Min       1Q   Median       3Q      Max
-0.15481 -0.05622 -0.01069  0.07710  0.13677

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.42293    0.02130  19.853 1.24e-14 ***
chain22light  0.27634    0.04468   6.184 4.84e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08783 on 20 degrees of freedom
Multiple R-squared:  0.6566,  Adjusted R-squared:  0.6395
F-statistic: 38.25 on 1 and 20 DF,  p-value: 4.844e-06
```

Al contrario, utilizziamo il pacchetto `lme4`[1] per valutare un **modello lineare ad effetti misti**, introducendo la variabile di gruppo `patient22` come un **effetto casuale**[4]:

```
library(lme4)
modelloB1 = lmer(stimaB ~ chain22 + (1| subject22))
summary(modelloB1)
```

```
> library(lme4)
> modelloB1 = lmer(stimaB ~ chain22 + (1| patient22))
> summary(modelloB1)
Linear mixed model fit by REML ['lmerMod']
Formula: stimaB ~ chain22 + (1 | patient22)

REML criterion at convergence: -56.8

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.56555 -0.64015 -0.03457  0.67740  1.41869

Random effects:
 Groups    Name        Variance  Std.Dev.
 patient22 (Intercept) 0.0081244 0.09014
 Residual              0.0009302 0.03050
Number of obs: 22, groups:  patient22, 9

Fixed effects:
             Estimate Std. Error t value
(Intercept)   0.43328    0.03489  12.417
chain22light  0.24262    0.07398   3.279
```

Come vediamo, il t-value si è praticamente dimezzato rispetto a quello errato del **modello lineare ad effetti fissi** (t test). Non avendo però a disposizione un p-value, siamo forse in dubbio nel decidere se efettivamente tale differenza sia significativa. Abbiamo però due strade possibili da percorrere per trarci dagli impicci ..

3.2 Il metodo della selezione del modello minimale adeguato

L'idea è questa. Proviamo a vedere se un `modelloB2` più semplice, in cui non c'è l'effetto di `chain22`, risulti equivalente in termini statistici al `modelloB1`. Per effettuare il confronto in maniera esatta, dobbiamo ricorrere al metodo di stima di massima verosimiglianza, `REML = FALSE` [4]:

```
> ml.modelloB1 = lmer(stimaB ~ chain22 + (1| patient22), REML = FALSE)
> ml.modelloB0 = lmer(stimaB ~   1     + (1| patient22), REML = FALSE)
> anova(ml.modelloB1, ml.modelloB0)
Data: NULL
Models:
ml.modelloB0: stimaB ~ 1 + (1 | patient22)
ml.modelloB1: stimaB ~ chain22 + (1 | patient22)
             Df     AIC     BIC logLik deviance  Chisq Chi Df Pr(>Chisq)
ml.modelloB0  3 -51.256 -47.983 28.628  -57.256
ml.modelloB1  4 -57.597 -53.233 32.798  -65.597 8.3405      1   0.003877 **
```

Il p-value 0.004 ci fa capire che il `modelloB0` non è adeguato a descrivere i dati e che l'effetto che `chain22` ha sulla `stimaB` è realmente significativo. Ossia, l'apparizione delle paraproteine nei gruppi `heavy` e `light` avviene con velocità diversa.

3.3 Il metodo bayesiano

Un'altra strada percorribile è quella bayesiana[2] e del pacchetto R-INLA[5].

```
library(INLA)
formula = stimaB ~ chain22 + f(patient22, model = "iid")
output = inla(formula, family = "gaussian", data = data.frame(patient22, chain22, stimaB)
summary(output)
```

```
> library(INLA)
> formula = stimaB ~ chain22 + f(patient22, model = "iid")
> output = inla(formula, family = "gaussian", data = data.frame(patient22, chain22, stimaB))
> summary(output)

Call:
c("inla(formula = formula, family = \"gaussian\", data = data.frame(patient22, ",  "    chain22,
stimaB))")

Time used:
 Pre-processing    Running inla Post-processing           Total
         0.5206          0.1863          0.0770          0.7838

Fixed effects:
               mean     sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept)  0.4331 0.0342     0.3648   0.4330     0.5018 0.4329   0
chain22light 0.2432 0.0725     0.0973   0.2434     0.3877 0.2439   0

Random effects:
Name    Model
 patient22   IID model

Model hyperparameters:
                                          mean     sd 0.025quant 0.5quant 0.975quant    mode
Precision for the Gaussian observations 1219.93 449.56     535.54  1157.96    2271.21 1036.64
Precision for patient22                  165.69  84.95      51.94   149.35     375.67  117.24

Expected number of effective parameters(std dev): 8.569(0.2938)
Number of equivalent replicates : 2.567

Marginal log-Likelihood:  16.32
```

Come si vede all'interno del corposo output, l'intervallo di fiducia al 95% dell'effetto dato dal gruppo `light` è $[0.097; 0.388]$; e tale intervallo di fiducia non contiene lo zero. E dunque, in termini classici, con un livello di significatività del 5 per cento possiamo rifiutare l'ipotesi nulla che `chain22` non abbia effetto su `stimaB`.

Riferimenti bibliografici

[1] Douglas Bates, Martin Maechler, Ben Bolker, and Steven Walker. *lme4: Linear mixed-effects models using Eigen and S4*, 2014. R package version 1.1-7.

[2] Marta Blangiardo and Michela Cameletti. *Spatial and spatio-temporal Bayesian models with R-INLA*. John Wiley & Sons, 2015.

[3] T Calimeri, E Battista, F Conforti, P Neri, MT Di Martino, M Rossi, U Foresta, E Piro, F Ferrara, A Amorosi, et al. A unique three-dimensional scid-polymeric scaffold (scid-synth-hu) model for in vivo expansion of human primary multiple myeloma cells. *Leukemia*, 25(4):707–712, 2011.

[4] Julian J Faraway. *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC press, 2005.

[5] Finn Lindgren and Håvard Rue. Bayesian spatial modelling with r-inla. *Journal of Statistical Software*, 63(19), 2015.

[6] John C Nash. *Nonlinear parameter optimization using R tools*. John Wiley & Sons, 2014.

[7] Christian Ritz and Jens Carl Streibig. *Nonlinear regression with R*. Springer Science & Business Media, 2008.

	patient22	chain22	stimaA	stimaB	stimaC
1	p1	heavy	8.15	0.51	10.45
2	p1	heavy	8.06	0.56	10.35
3	p2	heavy	4.32	0.38	10.76
4	p2	heavy	4.14	0.37	10.41
5	p2	heavy	4.07	0.41	10.21
6	p3	heavy	4.41	0.41	13.75
7	p3	heavy	4.48	0.40	13.88
8	p3	heavy	5.39	0.36	15.06
9	p4	heavy	2.72	0.50	10.90
10	p4	heavy	2.88	0.49	10.92
11	p6	heavy	3.50	0.37	11.45
12	p6	heavy	4.37	0.29	12.97
13	p6	heavy	3.62	0.36	11.59
14	p7	heavy	3.14	0.45	11.25
15	p7	heavy	2.87	0.51	10.49
16	p8	heavy	6.85	0.43	11.67
17	p8	heavy	7.00	0.39	11.82
18	p5	light	1.98	0.56	10.87
19	p5	light	1.86	0.54	10.45
20	p9	light	2.90	0.78	10.14
21	p9	light	2.91	0.77	9.96
22	p9	light	2.84	0.84	9.75