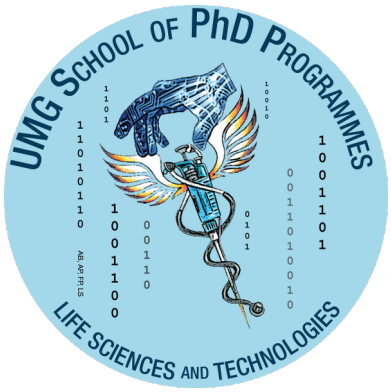
SOCIETA' DEI MATEMATICI
E NATURALISTI DI MODENA
www.socnatmatmo.unimore.it



Massimo Borelli, Ph.D.

il dataset `istologia` analizzato con R

2 novembre 2016

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.

caro Massimo,
avrei un dubbio metodologico da chiarire e forse tu riesci ad aiutarmi. Ho due gruppi di partecipanti ad uno studio di tipo cross-section; in un gruppo una certa `caratteristica` *è* ***presente****, mentre nell'altro gruppo essa è* ***assente****. Inoltre, questi partecipanti vengono classificati secondo una certa* `scala` *ordinale, a quattro livelli:* ***uno****,* ***due****,* ***tre*** *e* ***quattro****. Quello che io vorrei riuscire a decidere è se vi sia una associazione tra le variabili* `caratteristica` *e* `scala`*. Come dovrei procedere?*

1 Struttura del dataset `istologia`

Si tratta di un dataset di 46 osservazioni di 2 variabili, in formato *.csv*: `scala` è un fattore a quattro livelli (in ordine alfabetico, `due`, `quattro`, `tre` ed `uno`; `caratteristica` invece è un fattore a livelli (`assente` e `presente`). Importiamo il dataset in R[3] con i seguenti comandi:

```
www = "http://www.biostatisticaumg.it/dataset/istologia.csv"
istologia = read.csv(www, header = TRUE)
# istologia = read.csv(file.choose(), header = TRUE)
attach(istologia)
head(istologia)
str(istologia)
```

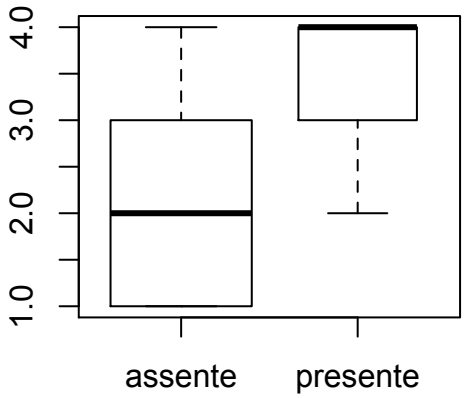
La prima cosa che vogliamo modificare è l'ordine alfabetico della `scala`, trasformandolo in un ordine (di gravità) crescente; a tale proposito utilizzeremo il comando `ordered`:

```
scala = ordered(scala, levels = c("uno", "due", "tre", "quattro"))
scala
```

2 Visualizziamo il dataset `istologia`

Abbiamo molte possibilità per visualizzare i dati del nostro dataset, dalle più spartane alle più ricercate. La pirma possibilità ci viene offerta da un boxplot:

```
boxplot(scala ~ caratteristica,  varwidth = TRUE)
```



Intuiamo immediatamente che l'assenza o la presenza della `caratteristica` giochi un ruolo sulla `scala` di gravità della patologia, visto che le **mediane** dei boxplot sono molto discoste. Vediamo che il boxplot di sinistra (`presente`, 19 partecipanti) è un pochino più magrolino del boxplot di destra (`assente`, 27 partecipanti); infatti:

```
table(scala, caratteristica)
table(caratteristica)
```

	assente	presente
uno	9	0
due	9	1
tre	5	6
quattro	4	12

	caratteristica
assente	27
presente	19

3 Il test di Wilcoxon / Mann / Whitney

Il test della somma dei ranghi di Wilcoxon (equivalente al test dei ranghi con segno di Mann e Whitney[1]) è la soluzione nonparametrica che classicamente viene utilizzata per affrontare questo tipo di decisione dal punto di vista statistico. Il test di Wilcoxon richiede che i dati siano di tipo numerico, e quindi convertiamo la variabile `scala` in questo modo:

```
> wilcox.test(as.numeric(scala) ~ caratteristica)

   Wilcoxon rank sum test with continuity correction

data:  as.numeric(scala) by caratteristica
W = 76.5, p-value = 3.191e-05
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x = c(2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4,  :
  impossibile calcolare p-value esatto in presenza di ties
```

L'output di R ci fornisce un p-value che non lascierebbe spazio ai dubbi ($p = 0.00003191$). L'unica cautela potrebbe riguardare quel messaggio in colore rosso: siccome sono presenti molti 'ties', molti ex-æquo nella classifica dei ranghi, potremmo essere in presenza di un p-value sovrastimato?

4 Il modello lineare generalizzato

Se pensiamo a `scala` come ad un'informazione in input, ossia ad un predittore della risposta (output) di tipo binomiale `caratteristica`, allora un modello lineare generalizzato (conosciuto anche con il nome di regressione logistica) può fare al caso nostro:

```
modello = glm(caratteristica ~ as.numeric(scala), family = binomial)
summary(modello)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-5.2295	1.5371	-3.40	0.0007
as.numeric(scala)	1.6496	0.4719	3.50	0.0005

Anche in questo caso il p-value si attesta attorno ad un valore di 0.0005, comunque 'altamente significativo'. E non ci sono più messaggi di errore. Stiamo tranquilli allora, decidiamo che i due gruppi provengono da popolazioni statistiche diverse?

5 Un dubbio sconcertante: c'è o non c'è associazione?

	uno	due	tre	quattro
assente	9	9	5	4
presente	0	1	6	12

Alla luce del messaggio di attenzione che avevamo ricevuto, riguardiamo con maggiore attenzione la distribuzione dei dati: ci rendiamo conto che in parecchie celle della tabella le frequenze osservate sono esigue. Per decidere in effetti se i due caratteri siano statisticamente indipendenti o dipendenti, l'approccio classico del test del Chi Quadrato mostra tutti i suoi limiti, legati all'approssimazione asintotica:

```
> chisq.test(table(scala, caratteristica))

   Pearson's Chi-squared test

data:  table(scala, caratteristica)
X-squared = 18.664, df = 3, p-value = 0.0003208

Warning message:
In chisq.test(table(scala, caratteristica)) :
  L'approssimazione al Chi-quadrato potrebbe essere inesatta
```

Osserviamo il risultato che si otteniene con il medesimo test di indipendenza, ma realizzato in modo **esatto**, ossia basandosi su un approccio combinatorio[4]:

```
> library(coin)
> independence_test(scala ~ caratteristica,
+             ytrafo = rank_trafo,
+             distribution = exact())

   Exact General Independence Test

data:  scala (ordered) by caratteristica (assente, presente)
Z = 0.62564, p-value = 0.5359
alternative hypothesis: two.sided
```

Apparentemente, quest'ultimo test si rivela essere particolarmente adatto al caso di nostro interesse: '`independence_test` *provides a general independence test for two sets of variables measured on arbitrary scales*' (Zeileis et al. [4]). Allora, a chi dobbiamo credere? C'è o non c'è indipendenza statistica?

6 No! .. e lo garantiscono i test di permutazione.

Se ricordiamo che le categorie 1, 2, 3 e 4 altro non sono che un proxy della misura geometrica dell'estensione di una neoplasia, allora **no**, non c'è differenza di centralità tra le due popolazioni, e lo garantiscono i test di permutazione. Seguiamo passo per passo quello che ci insegnano Pesarin e Salmaso nel paragrafo 1.10 del loro libro [2], adattando il nostro dataset al codice pubblicato a pagina 27. Innanzitutto importiamo il dataset `istologiaP`:

```
www = "http://www.biostatisticaumg.it/dataset/istologiaP.csv"
istologiaP = read.csv(www, header = TRUE)
attach(istologiaP)
```

	X	Y
1	1	1
2	1	1
3	1	1
..	..	..
10	2	1
11	2	1
..	..	..
19	3	1
..	..	..
24	4	1
..	..	..
28	2	2
29	3	2
..	..	..
35	4	2
..	..	..
46	4	2

Vediamo che la `caratteristica` adesso si chiama `Y`, ed `assente` e `presente` vengono codificati con in numeri 1 e 2; anche la `scala` ora è un numero intero, `X`.

```
n = table(Y)
n
```

```
C = length(n)
C
contr = rep(1/n, n)
contr
```

Definiamo la dimensione `n` dei due gruppi `assente` e `presente`, diciamo `C` la quantità di popolazioni in esame (che sono 2), e creiamo il vettore `contr` detto in linguaggio statistico 'dei contrasti', come il peso di ciascun elemento della media ponderata.

```
contr[-c(1:n[1])] = - contr[-c(1:n[1])]
contr
round(contr, digits = 3)
```

Con un intelligente trucco da esperti informatici, scambiamo i segni del secondo gruppo dei contrasti, rendendoli negativi.

```
B = 10000
T = array(0, dim = c((B+1), 1))
```

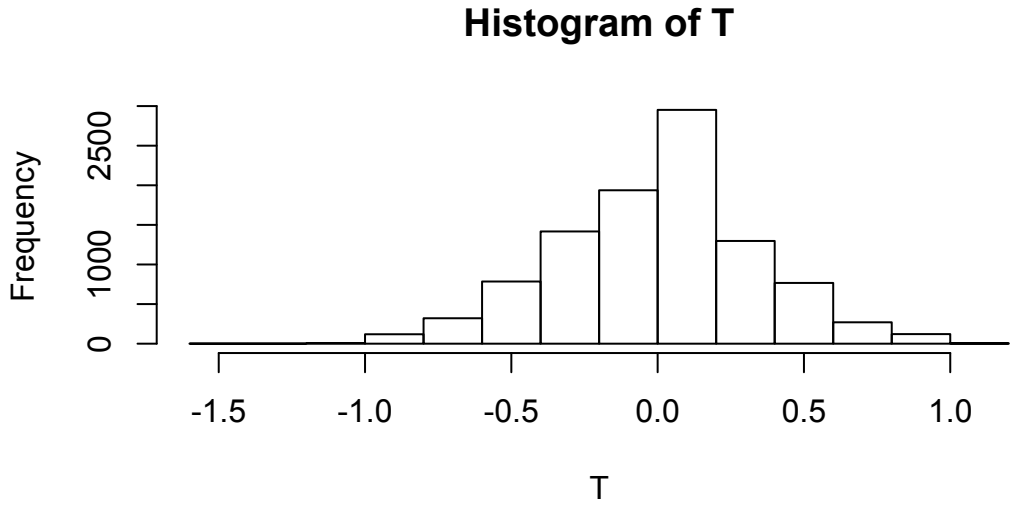
Creiamo ora una matrice verticale di una colonna sola con 10001 'zeri'.

```
T[1] = X %*% contr
T[1]
mean(X[1:27]) - mean(X[28:46])
```

Definiamo il primo elemento `T[1]` di questa matrice `T` come la differenza delle medie tra `assente` e `presente`, ossia -1.43. Questa operazione viene fatta per mezzo del prodotto scalare tra due vettori, che in R si effettua con l'operatore `%*%`. Questo valore è cruciale perchè rappresenta il **consuntivo** del test statistico. Si tratta ora di decidere se questo consuntivo sia 'grande' o 'piccolo' in senso statistico. Entrano ora in gioco i metodi di simulazione Monte Carlo: permutiamo casualmente il campione `X` con il comando `sample`, ottenendo per 10000 volte un nuovo campione chiamato `X.star` e calcoliamo la differenza delle medie mediante il prodotto scalare `X.star %*% contr`:

```
for (bb in 2 : (B+1) ) {
X.star = sample(X)
T[bb] = X.star %*% contr
}
```

Vediamo di visualizzare con un diagramma a barre cosa abbiamo ottenuto in queste 10000 simulazioni:



In questi diecimila esperimenti aleatori, che scambiavano di volta in volta le misure `X` attribuendole a caso in un gruppo `Y`, la differenza tra le medie rimaneva grossomodo confinata tra i valori di circa un'unità. Ma sapevamo che nei nostri data la differenza tra le medie valeva -1.43. Non ci stupiremo allora se non troviamo differenza significativa, ossia un p-value particolarmente alto. Per questo, si crea una funzione `t2p` (nel senso scherzoso di *conversion t to p*):

```
t2p = function(T){
if(is.null(dim(T))){T = array(T, dim = c(length(T), 1) )}
oth = seq(1:length(dim(T)))[-1]
B = dim(T)[1]-1
p = dim(T)[2]
if(length(dim(T)) == 3){C = dim(T)[3]}
rango = function(x){
r = 1 - rank(x[-1], ties.method = "min")/B+1/B
return(c(mean(x[-1]>=x[1]),r))
}
P = apply(T, oth, rango)
return(P)
```

```
}
```

la quale valuta i p-value relativi ai consuntivi ottenuti. In particolare, il primo valore del vettore `P` è il p-value di nostro interesse:

```
P = t2p(T)
P[1]
```

```
> P = t2p(T)
> P[1]
[1] 1
```

Riferimenti bibliografici

[1] William Jay Conover and WJ Conover. *Practical nonparametric statistics*. Wiley New York, 1980.

[2] Fortunato Pesarin and Luigi Salmaso. *Permutation tests for complex data: theory, applications and software*. John Wiley & Sons, 2010.

[3] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[4] Achim Zeileis, Mark A Wiel, Kurt Hornik, and Torsten Hothorn. Implementing a class of permutation tests: The coin package. *Journal of Statistical Software*, 28(8):1–23, 2008.