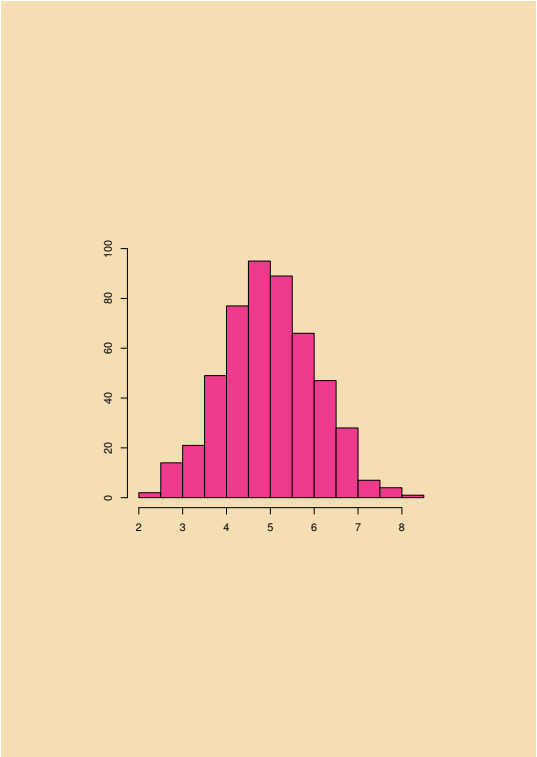
L'A B C di R



Stefano Leonardi ©
Dipartimento di Bioscienze
Università di Parma
Parma, 6 dicembre 2017

Indice

[1] Capitolo scritto in collaborazione con Franco Sartore - Dipartimento di Scienze Ambientali - Università di Parma

Capitolo 1

Impariamo R

1.1 Cos'è R?

R è un software di pubblico dominio sviluppato da un gruppo di volontari ed è reperibile in rete al sito `http://www.r-project.org` e nei vari `mirror` sparsi per il mondo. È sostanzialmente un pacchetto di programmi per l'analisi dei dati (statistica) e per la grafica. È molto potente, flessibile e facile da imparare. In realtà è un linguaggio vero e proprio e questo permette di usarlo sia in modo interattivo, sia di scrivere veri e propri programmi. Funziona bene in Windows, MacIntosh e Unix (Linux). È particolarmente usato in ambiente scientifico e il numero delle pubblicazioni dove è utilizzato è in continua crescita.

Il grafico seguente è tratto da un lungo post su un blog sulla popolarità di R (`http://r4stats.com/articles/popularity/`) e di altri pacchetti statistici nel mondo scientifico.

Una delle sue caratteristiche principali è la sua estensibilità, infatti sono state sviluppate centinaia di pacchetti aggiuntivi (oltre al modulo base) per obiettivi specifici: analisi statistiche e numeriche molto sofisticate, vari tipi di statistica multivariata, bootstrap, analisi filogenetiche, analisi di microarray, analisi QTL, analisi delle serie temporali, fitting non-lineare, statistica spaziale e tanti altri. Esistono anche alcuni pacchetti per l'analisi di dati "ecologici".

1.2 Storia di R

R è una differente implementazione di un pacchetto statistico molto noto e diffuso: S e S-PLUS, sviluppato nei laboratori Bell (USA). Molti dei program-

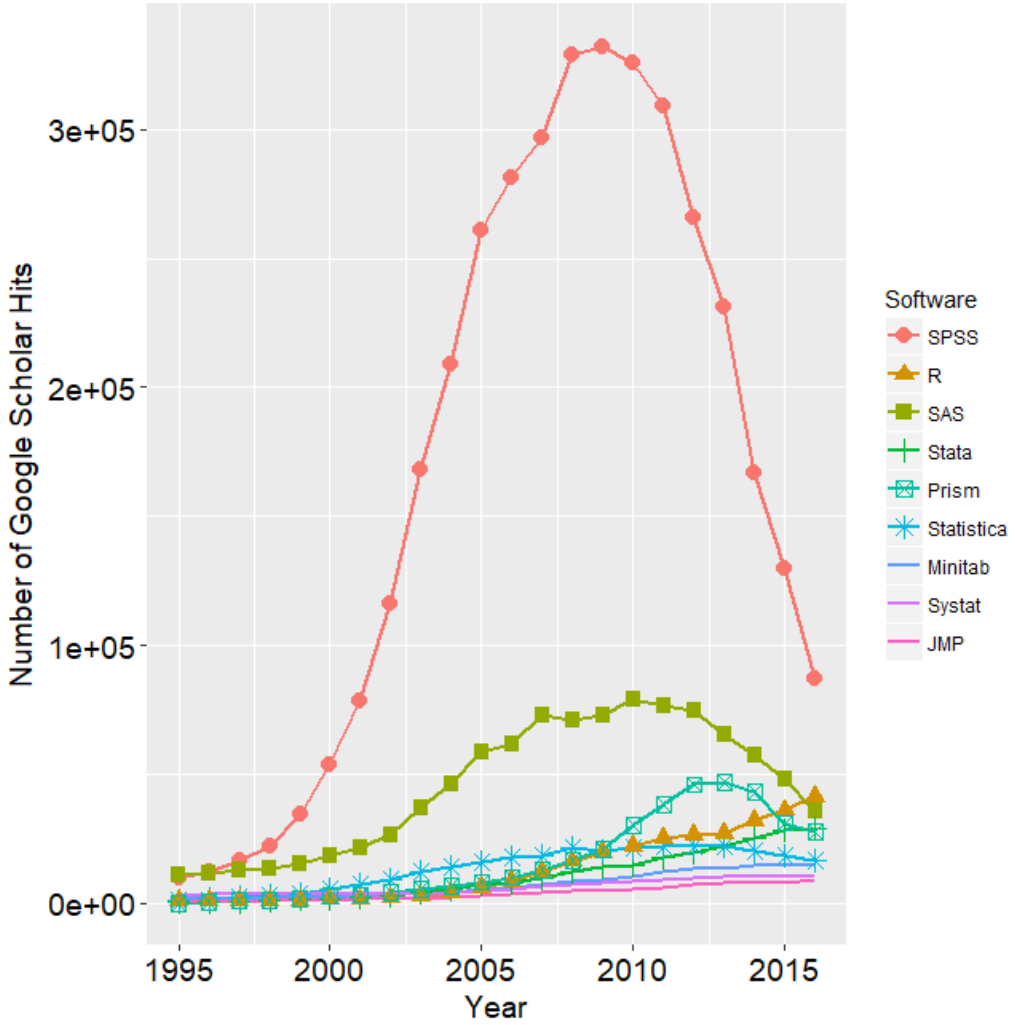


Figura 1.1: Popoloratità di R e di altri pacchetti statistici negli ultimi anni (tratto da: http://r4stats.com/articles/popularity/)

mi, libri e manuali scritti per S funzionano in R ma esistono delle "piccole" differenze (vedi `R-FAQ`).

R è un programma relativamente nuovo ed in continuo aggiornamento, anche se dalla versione 1.0.0 in poi può essere ritenuto stabile. Ogni anno vengono rilasciate diverse *release*, che aggiungono nuove *features* e correggono gli errori riscontrati nelle versioni precedenti. In questo momento siamo alla 3.4.2

1.3 Documentazione

R ha un buon sistema di "*help*" implementato nel programma e consultabile facilmente e immediatamente.

Per un'introduzione generale è altamente consigliato leggersi *An Introduction to R* (`R-intro.pdf`) e le FAQ (Frequently Answered Questions) che sono compresi nel programma e sono disponibili, assieme ad altra documentazione, sul sito di R .

Il sito del CRAN `http://cran.r-project.org/` è il sito principale di R e vi si possono trovare molte informazioni, mailing list archiviate e tanti link utili.

Ovviamente però il modo migliore per imparare R è provare ad usarlo. Fatelo senza timori, in R è praticamente impossibile arrecare danni al calcolare o perdere dati.

1.4 Start, Quit e Help

Per usare R in Windows, dopo averlo installato, cliccate sull'icona o lanciatelo dal solito menu *Start*, oppure doppio click sul file `Rgui.exe`. In UNIX dare semplicemente il comando `R`.

Appena lanciato, il programma presenta un banner inziale, leggetelo.

Per uscire digitate `q()` e per l'help *on line* potete digitare `help()`, mentre `help.start()` fa partire un browser (Explorer o Firefox) con un help ipertestuale (molto utile). Una funzione utile per cercare un qualsiasi aiuto partendo da una parola è `help.search("paroladacercare"`.

In UNIX premere la barra spaziatrice per scorrere l'help e `q` per uscire dall'help. In Windows l'help viene aperto in un nuova finestra o nel browser.

Quando si esce dal programma con `q()` ci viene chiesto se vogliamo salvare lo "**workspace**", cioè tutti gli oggetti (variabili e funzioni create) e la "storia dei comandi" che abbiamo in memoria. Rispondendo affermativamente, la prossima volta che si rientra nel programma si ritroveranno tutti

gli oggetti e la storia dei comandi che abbiamo dato e che avevamo al momento dell'uscita. Gli oggetti vengono salvati nel file `.RData` e la storia dei comandi nel file `.Rhistory` ed entrambi vengono salvati nella directory corrente. In Windows potete cambiare la directory corrente dal menù `File - Change Working Directory`. Il modo piú semplice per ritornare nelle stesse condizioni di quando avete chiuso il programma è quindi di fare doppio-click sul file `.RData`.

1.5 Una sessione tanto per provare...

Nonostante nella versione per Windows sia possibile richiamare alcuni comandi da menù, per sfruttare pienamente le potenzialità di R , consigliamo di digitare i comandi direttamente sulla linea di comando dopo il prompt '`>`'. La presenza del prompt '`>`' indica che il programma ha finito di eseguire le istruzioni precedenti ed è in attesa di un nostro input. Naturalmente ogni comando deve terminare con un `Return`.

Ora proviamo a digitare alcuni comandi:

```
> 1 + 2
```

e il programma risponderà

```
[1] 3
```

(per ora tralasciamo quanto è riportato nelle parentesi quadrate); vediamo che il programma ha risposto correttamente `3`. Il numero di spazi attorno all'operatore '`+`' non è importante. Semplicemente uno spazio prima e dopo l'operatore aumenta la leggibilità del codice.

Ora proviamo qualcosa di leggermente più complicato:

```
> 10 * 2 / 4 * log(10)
```

il programma risponde

```
[1] 11.51293
```

evidentemente ha usato i logaritmi naturali. Per avere i logaritmi in base 10 consultate l'help:

```
> help(log)
```

e rifate il conto con i logaritmi in base 10.
Per richiamare il comando precedente premete la freccia verticale ↑ sulla vostra tastiera.

Ora proviamo qualcos'altro:

```
> x <- c(2,5,8,12,18)
```

abbiamo creato un vettore `x` utilizzando la funzione `c()` che **c**-oncatena i numeri separati dalla virgola, crendo un vettore. L'operatore '`<-`' assegna quello che c'è a destra, ad una variabile di nome `x` che sta a sinistra dell'operatore. Il nome di una variabile in gergo informatico corrisponde ad un

indirizzo nella memoria (RAM) del computer dove vengono memorizzati i dati.
Per rivedere il contenuto di `x` semplicemente digitiamo `x`

```
[1] 2 5 8 12 18
```

Per avere la media di `x`:

```
> mean(x)
```

e la varianza:

```
> var(x)
```

Per la somma di tutti gli elementi di `x`:

```
> sum(x)
```

Per sapere quanti elementi ci sono in `x` e quindi per sapere quanti dati abbiamo:

```
> length(x)
```

Un modo più elaborato ma più istruttivo di calcolare la media di `x` è quindi:

```
> sum(x)/length(x)
```

Per avere gli elementi di `x` al quadrato:

```
> x^2
```

oppure

```
> x * x
```

quindi per avere la somma degli scarti dalla media al quadrato o *devianza* che ha la seguente formula:

$$Devianza = \sum_{i=1}^{n} (x_i - \overline{x})^2$$

in R si può usare il seguente comando:

```
> sum((x - mean(x))^2)
```

Se iniziamo il comando con il carattere '#' possiamo scrivere quello che ci pare ed il comando verrà ignorato e non iterpretato in nessun modo. È un "trucco" molto utile per scrivere commenti o appunti direttamente nella console di R .
Esempio:

```
> #Vorrei tanto uscire invece che stare qui a lezione!
```

e poi date invio e guardate cosa succede. Niente! Ora provate a ridare il comando qui sopra senza il '#'.
Vi ricordo che potete sempre copiare e incollare il testo della console di R in un file di testo o nel vostro programma di scrittura preferito. In questo caso è meglio usare un font non proporzionale (es: Courier) per avere una impaginazione corretta.

Ora creiamo un altro vettore `y`:

```
> y <- c(42,61,70,100,122)
```

e rivediamolo.

Un minimo di algebra fra due vettori:

```
> 2*x + y
```

Proviamo la nostra prima regressione. La funzione da chiamare è `lm()` (Linear Model) ed è una delle più importanti di R .

```
> xy.lm <- lm(y ~ x)
```
[1]

Non viene prodotto nessun output perché il risultato della regressione viene memorizzato in un "oggetto" che noi abbiamo chiamato `xy.lm`.

Con il comando `ls()` (*list*) si possono listare gli oggetti che abbiamo in memoria. Se vogliamo cancellare un oggetto in memoria possiamo utilizzare il comando `rm()` (*remove*). Abbiamo così imparato a creare, listare, stampare sul video e cancellare un oggetto.

Interroghiamo `xy.lm` per vedere se ci dà qualche informazione in più:

```
> summary(xy.lm)
```

Stavolta l'ouput dà un po' di soddisfazione:

```
Call:
lm(formula = y ~ x)

Residuals:
     1      2      3      4      5
-1.551  2.256 -3.936  5.808 -2.577

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  33.4231     3.8974   8.576  0.00333 **
x             5.0641     0.3679  13.763  0.00083 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.596 on 3 degrees of freedom
Multiple R-Squared: 0.9844,     Adjusted R-squared: 0.9792
F-statistic: 189.4 on 1 and 3 DF,  p-value: 0.00083
```

[1] Il carattere $\sim$ o *tilde* è molto usato in R . In Windows per chi ha la tastiera italiana dovrebbe prima di tutto assicurarsi di avere il tasto `Num Lock` (o `Block Num`) attivato in modo che digitando i numeri sul tastierino numerico (chi non ha il tastierino numerico separato ma ha i tasti con i numeri blu o colorati provare anche con `Fn`) effettivamente escano dei numeri. Quindi si può immettere con `Alt 1 2 6`, cioè tenendo premuto il tasto `Alt` e digitando i numeri uno dopo l'altro usando il tastierino numerico, oppure con `AltGr +`. Chi non ha il tastierino numerico separato può provare con `Alt Fn 1 2 6`. Per chi usa Apple si immette con `Alt 5` o `Alt n`. Chi usa Linux provi con `AltGr ì`

Ora proviamo il nostro primo grafico:

```
> plot(y ~ x)
```

aggiungiamo la retta di regressione

```
> abline(xy.lm)
```

Direi che per ora possiamo ritenerci soddisfatti. Come avete visto R tende a produrre un output piuttosto scarno, soprattutto se confrontato con altri programmi di statistica molto diffusi (SAS, SPSS). I risultati sono di solito memorizzati in "oggetti" che possono essere salvati e sono "interrogati" successivamente con delle funzioni apposite. Tipicamente sono interrogati con comandi come `summary()`, `print()`, `plot()`, `residuals()`, `predict()` ecc.

Alcuni settaggi preliminari

È altamente consigliabile usare R in **inglese**. I messaggi di errore in inglese sono più frequentemente riportati in rete e quindi è più facile trovare delle risposte in caso di difficoltà. Normalmente R però usa il linguaggio del sistema operativo installato e quindi nella maggior parte dei casi per noi userà l'italiano.

Settare la lingua inglese in R è un'operazione che si fa una volta e non si dovrebbe più ripetere fino a quando non si cambia computer o sistema operativo. In Windows ^{TM}conviene fare partire R e da menù scegliere `Modifica-Preferenze Interfaccia` e inserire la parola '`en`' nella casella `Language for menus and messages` in alto a destra. Quindi cliccare su `Salva` assicurandosi di salvare nella propria "home directory" che di solito è la cartella `Documenti` sul disco. Quindi dare `OK`. Uscire da R e rientrare, dovremmo così avere la lingua in inglese.

In linux si deve creare un file di nome `.Renviron` nella propria "home directory" contenente una riga con: `LANGUAGE=en`. Per farlo in modo veloce aprire un terminale e dare il seguente comando

```
echo LANGUAGE=en >> ~/.Renviron
```

Facendo partire R con il comando `R` da terminale si dovrebbe avere R in inglese.

In Windows (^{TM}) è utile vedere **i nomi dei file per esteso** (anche le estensioni come `.doc`, `.xls`, `.txt` altrimenti i file creati da R come `.RData` che hanno un nome con la sola estensione potrebbero apparire senza nome nelle cartelle. Per **Windows Vista, Windows 7 e Windows Server 2008**:

- Avviare `Esplora risorse` (che può essere avviato aprendo una cartella qualunque).

- Fare clic su `Organizza`.
- Fare clic su `Opzioni cartella e ricerca`.
- Fare clic sulla scheda `Visualizzazione`.
- Scorrere in basso fino a `Nascondi le estensioni per i tipi di file conosciuti` e deselezionare questa riga facendo clic sulla casella di controllo.
- Fare clic su `OK`.

Per **versioni più recenti di Windows**:

- Avviare `Esplora risorse` (che può essere avviato aprendo una cartella qualunque).
- Fare clic su `Aspetto e personalizzazione` e quindi su `Opzioni cartella`
- Fare clic sulla scheda `Visualizza`, quindi effettuare una delle operazioni seguenti in `Impostazioni avanzate`:
- Per visualizzare le estensioni di file, deselezionare la casella di controllo `Nascondi le estensioni per i tipi di file conosciuti`, quindi fare clic su `OK`.

Nei sistemi operativi **Apple**, `Finder` di default non mostra i files il cui nome inizia con un punto, per cui non mostra i files `.RData` e `.Rhistory`. Un modo per farglieli mostrare è quello di aprire un `Terminal` (si può trovare nel gruppo di programmi chiamato `Altro`) e dare il seguente comandi [2]:

```
defaults write com.apple.finder AppleShowAllFiles TRUE
killall Finder
```

Un modo facile per tenere i dati in ordine

Quando si sta lavorando in R spesso si devono importare ed esportare dati o file vari. Anche semplicemente uscendo da R e rispondendo "Sì" a "Save you Workspace?" si creano due files: `.RData` che contiene gli oggetti che avevamo nel nostro Workspace e `.Rhistory` con la storia di tutti i comandi che abbiamo dato. Ma dove vanno a finire questi file? E dove devono essere i files che devo importare affinché R li possa trovare? Devono essere nella

[2] Per altre possibili soluzioni si veda `https://apple.stackexchange.com/questions/5870/how-to-show-hidden-files-and-folders-in-finder`

Working Directory cioè la cartella o directory dove sto lavorando. Il nome e il percorso della cartella dove sto lavorando si ottiene in R con comando `getwd()`. Posso spostare la working directory in un altra cartella precedentemente creata con il comando `setwd("c:\Percorso\Miacartella")` ma è più facile usare il comando da menù `File-Change dir`. Gli utenti Apple trovano questo comando sotto al menù `Misc`. R cercherà ed esporterà i files in questa directory/cartella.

È utile avere una directory e un file `.Rdata` per ogni progetto di ricerca. Per esempio se avete la directory dove svolgete le analisi riguardanti i dati chimici di un lago e un'altra directory dove avete i vostri dati riguardanti i dati di una stazione marina. Immaginiamo che vogliate tenere le analisi completamente separate nelle direcory `lago` e `mare`. Quando svolgete la prima analisi dei dati di lago cambiate la *working directory* con il comando `File - Change Working Directory` e settatela su `lago`, effettuate le analisi e chiudete con `q()` e rispondete `y` alla domanda se volete salvare il workspace. Se il giorno seguente volete analizzate i dati marini per la prima volta, settate di nuovo *working directory* su `mare`, e poi uscite e salvate con `q()` e `y`. Quando volete riprendere in mano i dati di lago, fate doppio click sul file `.RData` nella directory `lago`, mentre se volete proseguire le analisi con il mare, fate doppio click sul file `.RData` nella directory `mare`. È molto più complicato spiegarlo che farlo!

1.6 Tipi principali di dati

Semplificando si può dire che in R esistono tre tipi **mode** base di dati:

- numerici (interi, reali o complessi)
- logici (vero o falso)
- carattere (stringhe di carattere).
- fattori (una sorta di "ibrido" fra tipo numerico e carattere)

I dati sono raggruppati in vari oggetti. I più comuni sono:

vettori sono "colonne" di dati omogenei tra loro. Ne abbiamo già visti alcuni.

matrici o **array** sono come i vettori ma sono caratterizzate dall'avere due (o più) dimensioni

liste sono collezioni di dati anche eterogenei tra loro

dataframe sono sostanzialmente dei *data set*, o collezioni di dati, in cui le colonne rappresentano le variabili e le righe rappresentano le osservazioni. Le colonne di solito sono vettori e i dataframe possono essere usati come se fossero matrici. In un dataframe le colonne/vettori hanno tutte la stessa lunghezza, ma possono essere di tipo diverso (numeriche, carattere o logiche).

In R il semplice numero 3 è in realtà un vettore di interi di lunghezza uno.

Per essere un po' più precisi in R è meglio distingure in **class** e **mode** di un oggetto. Il **mode** di un'oggetto ha a che fare con il modo come è memorizzato l'oggetto nella memoria del computer. La **class** indica alle funzioni generiche come "trattare" l'oggetto. Nell'esempio sotto **a** e **b** hanno entrambi un *mode* numerico ma *class* diversa e infatti vengono stampati in modo diverso dalla funzione generica `print()`

```
> a <- 1:12
> b <- matrix(1:12, 2, 6)

> class(a)
[1] "integer"
> mode(a)
[1] "numeric"

> class(b)
[1] "matrix"
> mode(b)
[1] "numeric"

> print(a)
[1] 1 2 3 4 5 6 7 8 9 10 11 12
> print(b)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    5    7    9   11
[2,]    2    4    6    8   10   12
```

Esistono anche altri tipi di oggetti. Un tipo di oggetto che useremo spesso è il vettore di **fattori**. Sono variabili (vettori) di tipo descrittivo-classificatorio e sono usati per le analisi statistiche (tipicamente nell'analisi della varianza) e descrivono a quale gruppo/trattamento appartengono le osservazioni. Sono facilmente convertibili in vettori di carattere con

`as.character()`, o un vettore (carattere o numerico) può essere facilmente convertito in vettore di fattori con `as.factor()`. Da ricordare che quando si importano i dati in R , di default, tutte le stringhe vengono automaticamente trasformate in fattori.

1.7 Operatori

R usa gli abituali operatori **matematici** simili a quelli del linguaggio "C".

`+`	somma	`-`	sottrazione
`*`	moltiplicazione	`/`	divisione
`^`	elevamento a potenza	`%%`	resto di una divisione
`%*%`	moltiplicazione di matrici	`%x%`	prodotto di Kroneker

Inoltre sono disponibili tutte le comuni funzioni matematiche `log()`, `exp()`, `sin()`, `cos()`, `tan()`, `sqrt()`, eccetera.

Ecco invece una breve lista di operatori **tipici** di R :

`<-`	assegnazione	`:`	sequenze di numeri (es: `1:5` equivale al vettore `c(1,2,3,4,5)`
`?`	help	`$`	subset di una lista
$\sim$	si usa nelle formule dei modelli	`[`	indice di vettore o lista
`[[`	indice di lista		

Una lista di operatori **logici**:

`!`	negazione	`&`	e
`|`	o	`==`	uguale
`!=`	diverso	`>`	strettamente maggiore
`<`	strettamente minore	`<=`	minore uguale
`>=`	maggiore uguale		

1.8 Le funzioni

Abbiamo già visto l'uso di alcune semplici funzioni come `mean()`, `var()`, `sum()`, `c()`, `summary()`. Esse operano sugli oggetti passati come argomenti (dentro alle parentesi) per restituirci un risultato.

Il risultato può essere "salvato" in una variabile o può essere semplicemente scritto sul video.

Per esempio:

```
> x <- c(2,5,8,12,18)
> max(x)
```

produce: [3]

```
[1] 18
```

mentre

```
> massimo<-max(x)
```

non produce nessun output ma da questo momento in avanti avremo in memoria una variabile (un vettore di lunghezza uno) di nome `massimo`.

Per avere una lista di tutte le funzioni guardate le pagine dell'help. Ce ne sono moltissime. Solo nel pacchetto base ce ne sono più di mille. In realtà quelle che si usano di solito non sono poi tantissime.

Gli oggetti tra le due parentesi, eventualmente separati da una virgola, sono chiamati gli **argomenti** (Es: chiamando `mean(x)`, `x` è l'agomento della funzione `mean()`). Alcuni argomenti sono necessari e senza di loro la funzione non potrebbe operare (es: nel caso precedente, senza l'argomento `x` la funzione `mean()` produce un errore). Altri argomenti hanno un loro *deafult* e quindi non è necessario passarli (metterli fra le due parentesi) a meno che non si voglia cambiare il default. Nell'help di ogni funzione c'è l'indicazione su quali argomenti hanno o non hanno default.

Per esempio notate la differenza fra:

```
> mad(x)
[1] 5.9304
```

e

```
> mad(x, constant=1)
[1] 4
```

Anche l'ordine degli argomenti nelle parentesi può essere importante. Se non si specifica "`argomento=`", si deve rispettare l'ordine degli argomenti previsto per la nostra funzione e che è riportato nell'help. Assicuratevi di capire bene perché i seguenti output sono così:

```
> seq(from=1, to=5, by=0.5)
```

[3] `[1]` indica solo che l'ouput di quella riga inizia con il primo elemento di un vettore. È un'indicazione utile nei casi in cui l'ouput si prolunghi su più righe. In questi casi, per ogni riga, il programma stampa la posizione del primo elemento della riga.

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> seq(1, 5, 0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> seq(5, 1, 0.5)
Error in seq.default(5, 1, 0.5) :  wrong sign in 'by' argument
> seq(to=5, from=1, by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Come esercizio potremo studiare cosa fanno le seguenti funzioni:

```
> sample(1:10)
> sample(1:10) # i risultati sono diversi dal comando precedente
> sample(1:10, 2)
> runif(10)
> runif(10, min=0, max=100)
> rnorm(7)
> rnorm(7, m=12, s=2)
```

Sono tutte funzioni che estraggono numeri casuali e il risultato sarà diverso ogni volta che diamo il comando e fra un computer ed un altro. Per avere una sequenza di risultati uguali in tutta la classe si deve usare `set.seed()`.

Per esempio studiate il differente risultato grafico dei due comandi:

```
> hist(runif(500, 0, 100))
```

e

```
> hist(rnorm(500, 50, 8))
```

1.8.1 Crearsi le proprie funzioni

È molto facile crearsi una funzione da soli. Per esempio creiamo una funzione che duplica il valore passato come argomento:

`> duplica <- function(arg) { return(2* arg) }` crea una funzione che si chiama `duplica()`. Applicando `duplica()` ad una costante otteniamo:

```
> duplica(3.5)
[1] 7
```

mentre su un vettore otteniamo:

```
> duplica(x)
[1] 4 10 16 24 36
```

Notate che il vettore `x` è rimasto inalterato.

```
> x
[1] 2 5 8 12 18
```

In sostanza, quando una funzione viene chiamata, viene creata una copia dell'argomento (in questo caso una copia di `x`) ed è come se venisse assegnata ad `arg`. Su questa copia di `x` vengono effettuate le operazioni specificate dentro al **corpo della funzione** (la parte fra le due parentesi graffe) e la funzione termina con il `return()` che "ritorna" il risultato dell'operazione al chiamante della funzione. Sia gli argomenti, sia le variabili temporanee create dentro al corpo della funzione esistono solo per il brevissimo tempo in cui dura l'esecuzione della funzione e poi scompaiono dalla memoria.

Per esempio la funzione `somma2v()` effettua la somma di due vettori elemento per elemento

```
> somma2v <- function(v1, v2) {
     somma <- v1 + v2
     return(somma)
}
```

In quest'ultima funzione il corpo della funzione è spezzato in più righe e viene creato l'oggetto `somma`, che sarà un vettore numerico e che cesserà di esistere alla fine della chiamata della funzione.

Nell'esempio di `duplica`, `x` rimane inalterato anche nel caso di una funzione duplica creata così:

```
> duplica <- function(x) {
     x <- 2 * x
     return(x)
}

> duplica(x)
[1]  4 10 16 24 36

> x
[1]  2  5  8 12 18
```

Come dovremmo fare se effettivamente volessimo avere un `x` che è il doppio di quello precedente? Occorre assegnare il risultato di duplica di nuovo a `x`:

```
> x <- duplica(x)
```

In generale dentro al corpo della funzione conviene lavorare solo sugli argomenti della funzione stessa o su oggetti che derivano direttamente dagli argomenti (come in questo caso il vettore `somma`). Dentro al corpo di una funzione, si raccomanda fortemente di non fare mai riferimento ad oggetti esterni alla funzione che non siano gli argomenti stessi della funzione.

Un altro ottimo suggerimento è quello di creare delle funzioni che funzionino bene in qualunque situazione e che siano quindi il più generali possibili.

Di seguito un esempio con la funzione `multiplo()` che esegue il prodotto di un vettore per una costante `k`, che di default avrà valore 2. Cioè di default darà il risultato uguale alla funzione `duplica()`, ma potrà moltiplicare il vettore anche per valori diversi da 2.

```
 > multiplo <- function(arg, k=2) { return(k * arg) }
> multiplo(x)
[1] 4 10 16 24 36
> multiplo(x, k = 3)
[1] 6 15 24 36 54
```

Ma quando conviene crearsi una funzione da soli? Conviene farlo quando ci ritroviamo ad eseguire sempre lo stesso calcolo più e più volte e quindi quando, l'avere una funzione che faccia il calcolo che vogliamo, ci fa risparmiare tempo. Più avanti nel corso ci saranno diverse occasioni simili. Tenete presente che le funzioni in R sono il "cuore" del linguaggio stesso, che è fatto di oggetti e funzioni che vi operano sopra per darci le informazioni che vogliamo.

1.9 Input e output da/su file

Ora che abbiamo un minimo di conoscenza degli elementi di R (strutture dati e funzioni) possiamo passare a qualcosa di concreto.

Input di dati da file

Dobbiamo prima di tutto assicurarsi che il file che vogliamo importare sia nella nostra *working directory* ottenibile con il comando `getwd()`. Quindi abbiamo due possibilità: o spostiamo il file che vogliamo importare nella working directory o spostiamo la working directory (con il comando da menù `File-Change dir`) nella cartella dove c'è effettivamente il file che vogliamo importare. Ora siamo pronti per importare il file.

Importiamo i dati da un file esterno formato testo separato da spazi o tabulatori. Si usa la funzione `read.table()` che restituisce un dataframe.

```
> exp1.df <- read.table("exp1.txt", header=TRUE)
```

I dati vegono memorizzati nel dataframe `esp1.df`. L'argomento `header=TRUE` indica che nel file la prima riga riporta i titoli delle colonne.

Controlliamo il dataframe appena creato [4]

[4]Richiamando semplicemente il nome di un oggetto in realtà si invoca implicitamente la funzione `print()` su quell'oggetto (es: `exp1.df` è equivalente a `print(exp1.df)`).

```
> exp1.df
```

possiamo anche averne una descrizione sommaria:

```
> summary(exp1.df)
```

Il dataframe ha due variabili/vettori (`weight` e `group`) e 27 osservazioni. La prima è un vettore numerico e si può "accedervi" separatamente dall'altra semplicemente usando l'operatore '`$`': `exp1.df$weight`. Per esempio se voglio vedere l'ottavo elemnto di `weight`: `exp1.df$weight[8]`).

La seconda colonna è un vettore di caratteri che è stato automaticamente trasformato in un *fattore* da `read.table` con tre livelli: `Ctrl, Trt1 e Trt2` rispettivamente con 10, 8 e 9 osservazioni.

```
> summary(exp1.df)
     weight       group
 Min.   :3.590   Ctrl:10
 1st Qu.:4.570   Trt1: 8
 Median :5.170   Trt2: 9
 Mean   :5.108
 3rd Qu.:5.560
 Max.   :6.310
```

Il `summary` del data frame è un modo veloce per controllare di avere letto bene i dati. Per le colonne numeriche il `summary` riporta il minimo, il massimo, la media, la mediana ecc. Le colonne di stringhe, come abbiamo detto, vengono trasformate in fattori da `read.table` e il `summary` ne riporta le frequenze per ogni livello del fattore. Attenti che è sufficiente avere un solo numero non letto correttamente per far diventare tutto il vettore un fattore, anche se doveva essere numerico. Questo produce spesso risultati inattesi. I vettori di fattori sono "ibridi" e possiamo intenderli sia di tipo numerico, sia di tipo carattere. Per vederne la parte numerica, per esempio, si può fare:

```
> as.numeric(exp1.df$group)
```

mentre per vederne la parte carattere:

```
> as.character(exp1.df$group)
```

Per controllare se un vettore è un veramente fattore si può usare `is.factor()` oppure `class()`.

Eseguiamo un'analisi della varianza per confrontare la media tra i tre gruppi:

```
> anova(lm(weight ~ group, data=exp1.df))
```

confrontiamo i gruppi anche graficamente con un boxplot:

```
> boxplot(weight ~ group, data=exp1.df, col="salmon")
```

possiamo evitare di digitare tutte le volte `data=exp1.df` se prima "attacchiamo" il dataframe con il comando `attach()`

```
> attach(exp1.df)
```

Dopo l'attach è inteso che il programma cerca le variabili dentro al dataframe "attaccato". Attenzione però che dopo l'attach:

```
> weight <- weight * 1000 #per passare a milligrammi
```

crea un nuovo vettore `weight` e non rimpiazza il vettore `weight` del dataframe `exp1.df`. Per ottenere quello che vogliamo:

```
> exp1.df$weight <- weight * 1000
```

Ricapitolando, le tre chiamate a `lm()`:

```
> lm(exp1.df$weight ~ exp1.df$group)
> lm(weight ~ group, data=exp1.df)
> attach(exp1.df)
> lm(weight ~ group)
```

sono equivalenti e producono lo stesso risultato.

Per "staccare" un dataframe si usa `detach()` che per default stacca l'ultimo dataframe attaccato.

Esistono pacchetti appositi per estrarre dati da Microsoft Excel o da databases (Posgresql, Mysql, ecc.).

Importare i dati da Excel ™

Esistono diversi modi per importare i dati da Microsoft Excel ™, ma il modo più semplice e più sicuro è il seguente, che si adatta bene anche ad altri fogli elettronici come Openoffice o Libreoffice Calc.

La premessa è che esistono diverse versioni di Excel ™, e che anche la versione sistema operativo e della lingua determinano variazioni sulle modalità precise per esportare i dati da Excel ed importarli in R . Le operazioni descritte di seguito possono quindi variare leggermente da caso a caso.

Nel foglio elettronico, prima di exportare i dati conviene:

- Controllare che le colonne abbiano tutte un titolo nella prima riga e che il titolo sia formato da una sola stringa (parola).
- É meglio, ma non indispensabile, sostituire tutti gli spazi con un `'_'` o un `'.'`.
- Non devono esserci celle vuote. Riempire le celle vuote riempirle con un `'NA'`.
- Eliminare le righe e le colonne vuote sopra, a sinistra e in mezzo ai dati.

Quindi salvare il file in formato `CSV` (Comma Separated Value), dal menù `File`, `Salva con nome`, tipo di file.

Aprire il file salvato con un editor di testo, come il Blocco Note (Non fare doppio click sul file!). Possibilmente non modificate il file e quindi non salvatelo: esaminatelo e basta! Occorre capire con quale carattere Excel ha separato le varie colonne (**Separatore di colonna**) e con quale carattere ha usato come **Separatore decimale**, cioè se ha usato il punto o la virgola. Spesso Excel usa il `';'` come separatore di colonna e la virgola come separatore decimale, ma può usare anche la vigola e il punto rispettivamente. Se usa il carattere `Tabulatore` l'editor di testo lo sostituisce con una serie di spazi.

Quindi importare il file con il solito comando

```
> d.df <- read.table("nomefile.csv", header=T, sep=";", dec=",")
```

dove vengono specificati il carattere separatore di colonna con l'argomento `sep=` ed il separatore decimale con `dec=` racchiusi tra virgolette. Naturalmente voi dovete usare i caratteri separatori che avete dentro al vostro file `.csv`. Se il separatore di colonna è il tabulatore o un spazio non occorre specificarli nel `read.table` in quanto sono i separatori di colonna di default.

I fogli elettronici Openoffice Calc e LibreOffice Calc permettono di editare i filtri di esportazione (`Edit filter settings` nel menù `File - Save as` nella versione in inglese). Quindi si può usare il tabulatore (`Tab`) come separatore di colonna e di mettere le virgolette a tutte le colonne di testo con `Quote All Text Cells` nella versione in inglese. In questo caso ci si può permettere di avere di avere tranquillamente degli spazi nelle stringhe e di non specificare il separtore di colonna nel `read.table`. Infine controllare di avere importato correttamente i dati stampando il dataframe sullo schermo o facendo un `summary`.

Output su file

Il modo più comodo per fare un ouput su file varia da sistema a sistema. In Windows forse il sistema più comodo è fare un "*copy and paste*" da una finestra di R ad una di Word (o qualche altro programma). Questo sistema dovrebbe funzionare anche per la grafica.

Altrimenti esistono soluzioni più generali.

Una funzioni più utili è `write.table()` per fare l'output di oggetti, tipicamente un dataframe, che abbiamo creato in R . In genere io consiglio si salvarlo con il seguente comando:

```
> write.table(miodataframe.df, file="nomefile.txt",
+ row.names=F,quote=F,sep="\t")
```

Il file `nomefile.txt` è facilmente importabile in qualsiasi foglio elettronico.

Un'altra soluzion generale è quella di usare il comando `sink("nomefile.txt")` per redirigere l'output su *nomefile.txt*. Per esempio per "salvare" i dati e l'ultima analisi della varianza sul file "`exp1_anova.txt`". [5]

```
> sink("exp1_anova.txt", append=TRUE)
> exp1.df #stampo i dati
> anova(lm(weight ~ group, data=exp1.df))
> sink() #chiudo il sink
```

Non si vede l'ouput sullo schermo in quanto l'output viene "dirottato" al file, quindi conviene redirigere l'output solo quando il risultato è certo e soddisfacente. Per ritornare a vedere l'output sullo schermo è necessario chiudere il "sink" con `sink()`

Per la grafica si usa lo stesso approccio e si usano i comandi `postscript()` o `pdf()`. In Windows è utile il comando `win.metafile()` che crea un file che può essere importato in Word).

```
> postscript("exp1_box.eps")
> par(bg="peachpuff") #per avere lo sfondo colorato

> boxplot(weight ~ group, data=exp1.df, col="salmon",
 + xlab="Trattamento", ylab="Peso (g)")

> dev.off() #chiudo l'output a postscript
```

In questo modo ho ottenuto il file `exp1_box.eps` con un grafico colorato con una qualità da pubblicazione (Vedi Figura 1.2). In Windows potrei aver ottenuto lo stesso risultato con `win.metafile()` ed includere il file ottenuto in un documento word. Per generare delle immagini *bitmap* si possono usare i comandi `jpeg()` o `png()`.

1.10 Un po' di più sui vettori

Come si crea un vettore

Il modo più semplice per creare un vettore è di usare la funzione `c()` (*concatenare*). Per creare un vettore di caratteri è sufficiente:

```
> nomi<-c("Franco", "Stefano", "Pio", "Tizio", "Marco")
> ages<-c(102, 26, 25, 48, 19)
```

Per creare vettori più lunghi sono molto utili l'operatore ':' e le funzioni `rep()` e `seq()`. Proviamo i seguenti comandi:

[5] Se non speficate `append=T` un eventuale file `exp1_anova.txt` già esistente viene rimpiazzato da quello nuovo senza chiedervi alcunché.

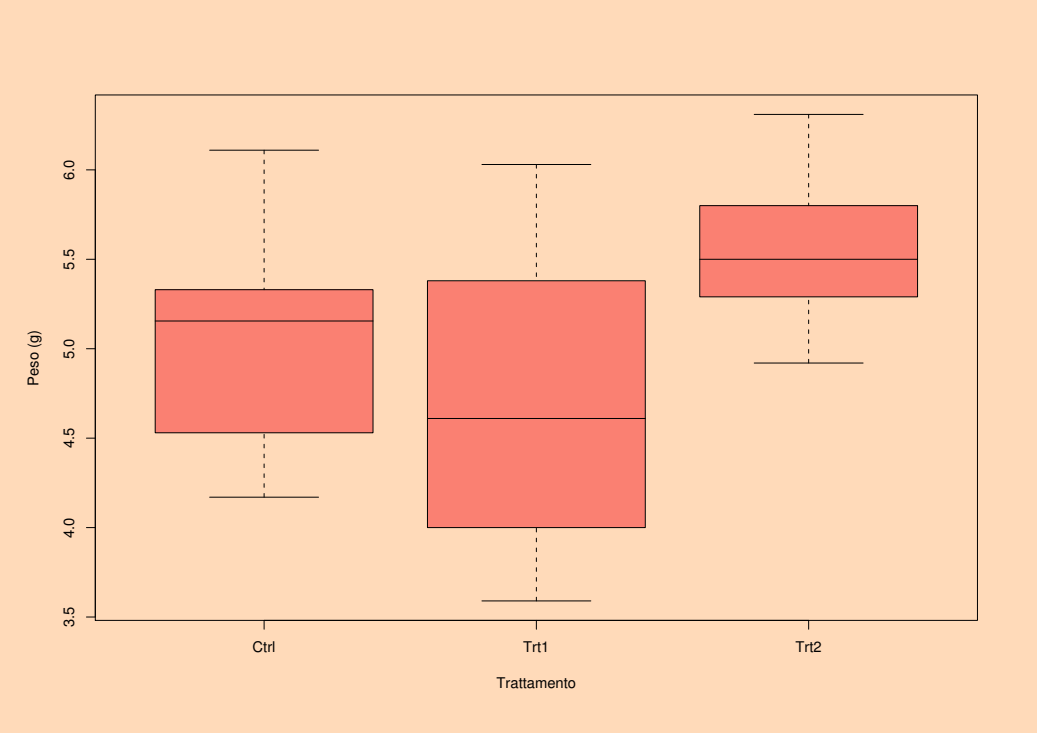


Figura 1.2: Esempio di “boxplot”

```
> rep(1:3, times=5)
> rep(nomi, c(2,3,1,4)) #times= può venire omesso
> seq(from=0,to=10,by=0.5)
```

In classe useremo spesso **vettori numerici** creati con il generatore di numeri casuale di R . R infatti è in grado di generare dei numeri chiamati **pseudo-casuali**. Sono chiamati così perché simulano molto bene delle serie di numeri casuali, anche se in realtà casuali non sono. Noto infatti il primo *seme*, gli altri sono determinati. Per esempio, se tutta la classe settasse il seme a un certo numero (ad esempio a 1) con `set.seed(1)`, tutta la classe avrà la stessa sequenza di numeri (a parità di "estrazioni" a sorte). I numeri pseudo-casuali sono estremamente importanti nelle simulazioni.

I numeri casuali possono essere estratti da diverse distribuzioni:

```
> runif(10, 0, 100)
```

estrarrà 10 numeri casuali reali compresi da 0 e 100 da una distribuzione uniforme, mentre

```
> rnorm(10, 5, 2)
```

estrarrà 10 numeri casuali da una distribuzione normale con $\mu = 5$ e $\sigma = 2$ (teoricamente i numeri estratti da una distribuzione normale possono sempre andare da $-\infty$ a $+\infty$).

I **vettori logici** sono vettori che contegono valori che sono `TRUE`, `FALSE` (abbreviabili con `T` e `F` e sostanzialmente equivalenti a `1` e `0`) Possiamo creare vettori logici con la solita funzione `c()` ma più spesso sono usati quando si controllano alcune osservazioni. Vediamo come:

```
> ages < 30
```

produce

```
[1] FALSE TRUE TRUE FALSE TRUE
```

Uso degli indici nei vettori

Un uso appropriato dei vettori in R permette di apprezzare quanto semplice e potente sia questo linguaggio.

```
> nomi[2]
 [1] "Stefano"
```

estrae il secondo elemento dal vettori nomi, mentre

```
> nomi[-2]
 [1] "Franco" "Pio"    "Tizio"  "Marco"
```

estrae tutti gli elementi di `nome` tranne il secondo:

```
> nomi[c(5,4,3,2,1)]
```

```
 [1] "Marco" "Tizio"   "Pio"     "Stefano" "Franco"
```

produce il vettore `nomi` con l'ordine invertito.

Notate anche:

```
> nomi[c(1,2,2,3,3,6)]
 [1] "Franco"  "Stefano" "Stefano" "Pio"     "Pio"     NA
```

Abbiamo già visto cosa produce `ages < 30` ma ora vediamo cosa produce

```
> ages[ages < 30]
[1] 26 25 19
```

e possiamo anche provare

```
> nomi[ages < 30]
 [1] "Stefano" "Pio" "Marco".
```

In sostanza quando viene passato un vettore logico all'interno delle parentesi quadre (come indice) si determina l'estrazione delle osservazioni in corrispondenza del valore `TRUE`. Assicuratevi di avere capito bene questo meccanismo e l'uso dei vettori logici che permette di selezionare facilmente le osservazioni che ci interessano e sostanzialmente trasformano R in un database con una sintassi molto semplice ma potente.

È utile sapere che la condizione fra parentesi quadre (indicizzazione con vettore logico) può esserci anche "a sinistra dell'assegnazione (`<-`)"

Ad esempio per portare a 49 l'età di `Stefano`, a 40 quella di `Pio` e a 45 quella di `Marco` (tutti quelli che hanno l'età minore di 30)

```
> ages[ages < 30] <- c(49, 20, 45)
> ages
[1] 102 49 20 48 45
```

Il numero degli elementi a sinistra dell'assegnazione (`<-`) dovrebbe essere uguale a quello a destra, altrimenti il vettore di sinistra, se è più corto, viene riciclato [6].

Vettori indicizzati da stringhe

I *ragged arrays* o vettori indicizzati da stringhe sono simili ai vettori indicizzati da numeri interi tranne per il fatto, ovviamente, di essere indirizzati da "nomi" con il vantaggio di essere molto più facili da memorizzare. Come esempio vogliamo stimare la crescita demografica di 6 regioni (Piemonte, Lombardia, Veneto, Emilia-Romagna, Toscana, Umbria, Marche). Abbiamo dati di natalità e di mortalità per mille abitanti:

```
> natalita <- c(8.4, 9.4, 9.6, 8.5, 8.0, 8.1, 8.5)
> mortalita <-c(11.3, 9.4 , 9.3, 11.4, 11.6, 10.8, 10.6)
```

controlliamo `natalita`

[6] Vedi regola del *Riciclaggio dei vettori* negli esercizi

```
> natalita
[1] 8.4 9.4 9.6 8.5 8.0 8.1 8.5
```

l'output è poco leggibile

Diventa più chiaro se associamo a ciascuno dei due vettori un vettore di nomi

```
> nomiregione <- c("Piemonte", "Lombardia",
  + "Veneto", "Emilia-Romagna", "Toscana", "Umbria", "Marche")
```

con la funzione `names()`:

```
> names(natalita) <- nomiregione
> names(mortalita) <- nomiregione
```

ora ricontrolliamo `natalita`

```
> natalita
      "Piemonte"      "Lombardia"     "Veneto" "Emilia-Romagna"
             8.4              9.4          9.6              8.5
       "Toscana"       "Umbria"       "Marche"
             8.0              8.1          8.5
```

l'output è molto più leggibile. Ora calcoliamo la differenza

```
> natalita - mortalita
      "Piemonte"      "Lombardia"     "Veneto" "Emilia-Romagna"
            -2.9              0.0          0.3             -2.9
       "Toscana"       "Umbria"       "Marche"
            -3.6             -2.7         -2.1
```

La funzione `names`, se usata con l'operatore `<-`, serve per associare `nomi` ai vettori , mentre, senza l'operatore, stampa semplicemente eventuali nomi già assegnati.

Se vogliamo estrarre la natalità del Veneto, non è necessario ricordarsi che il Veneto è il terzo elemento del vettore, basta usare `"Veneto"` come indice:

```
> natalita["Veneto"]

 "Veneto"
    9.6
```

naturalmente avremmo ottenuto lo stesso risultato con

```
> natalita[3]

 "Veneto"
    9.6
```

1.11 Liste

Le liste sono usate per dati eterogenei fra loro (es: numerici e carattere). È meglio avere un'idea di cosa sono e come si usano perchè molti degli oggetti restituiti dalle funzioni statistiche di R (es: `lm()`) sono liste.

Ecco un esempio:

```
> dip <- list(dipendente="Anna", coniuge="Fred", nfigli=3,
  + etafigli=c(4,7,9))
```

La lista `dip` contiene 4 elementi: due stringhe di carattere, uno scalare (in realtà un vettore con un elemento) e un vettore con 3 elementi.

```
> dip
$dipendente
[1] "Anna"

$coniuge
[1] "Fred"

$nfigli
[1] 3

$etafigli
[1] 4 7 9
```

Si puo accedere ai singoli elementi usando indici numerici come per i vettori ma si può/è meglio usare l'operatore '`[[ ]]`':

```
> dip[[2]]
[1] "Fred"
```

oppure, meglio ancora, si può usare l'operatore '`$`':

```
> dip$conigue
[1] "Fred"
> dip$etafigli[2]
[1] 7
```

Come per i vettori, la funzione `names()` può essere usata per settare/leggere i nomi degli elementi. La funzione `unlist` converte una lista in un vettore di caratteri se c'è almeno un elemento della lista di modo carattere, mentre la funzione `c()` concatena anche le liste.

1.12 Dataframes

In un dataframe le colonne devono avere lo stesso numero di osservazioni (vedi la sezione 1.14 per la codifica dei dati mancanti) ma le colonne possono essere di tipo diverso.

Abbiamo già visto che la funzione `read.table()` legge dati da un file esterno e restituisce un dataframe.

Si può creare un dataframe anche associando due o più vettori usando la funzione `data.frame()` [7] :

```
> eta.df <- data.frame(nomi,ages)

> eta.df
     nomi   ages
1  Franco  102
2 Stefano   40
3     Pio   40
4   Tizio   48
```

Per accedere alle due colonne separatamente si può usare l'operatore '`$`' (es: `eta.df$nomi` oppure `eta.df$ages`). Oppure si possono usare indici numerici, esattamente come nelle matrici:

```
> eta.df[ , 2]
[1] 102  40  40  48
```

Lasciando vuoto il primo campo dentro alla parentesi quadrata si selezionano tutte le righe, ma solo le colonne indicate nel secondo campo (dopo la virgola), mentre lasciando vuoto il secondo campo si selezionano tutte le colonne, ma solo le righe indicate nel primo campo.

```
> eta.df[ 2:4, ]
     nomi ages
2 Stefano   40
3     Pio   40
4   Tizio   48
```

Ovviamente specificando sia la riga che la colonna si accede ad un solo dato:

```
> eta.df[2,4]
[1] 48
```

[7] Attenzione che anche in questo caso, nel nuovo dataframe, i vettori di tipo carattere vegnono trasformati automaticamente in fattori.

Subset di dataframe

Per selezionare solo alcune osservazioni (righe), ed eseguire le analisi statistiche solo su una parte del dataframe, è molto utile la funzione `subset()`.

```
> eta.piu45.df<- subset(eta.df, ages > 45)
> eta.piu45.df
    nomi ages
1 Franco  102
4  Tizio   48
```

1.13 Matrici

Le matrici sono essenzialmente dei vettori con in più l'attributo delle **dimensioni**.

Un modo per creare una matrice 4×3 è il seguente

```
> X<-c(1:12)
> dim(X)<-c(4,3)
> X
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

La funzione `dim()` può essere, quindi, usata per settare (o visualizzare non usando " `<-` ") le dimensioni della matrice.

Notate l'intestazione delle colonne e delle righe nell'output (es: per accedere solo alla prima colonna è sufficiente: `X[ ,1]`).

Notate anche che la progressione dei numeri è dall'alto verso il basso (per colonna) e non da destra a sinistra (per riga). Per ottenere una matrice con una progressione "per riga" si può usare la funzione `matrix()` con l'argomento `byrow=TRUE`:

```
> matrix(1:12,nrow=3,byrow=T)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

È possibile traformare un dataframe interamente numerico in una matrice con la funzione `as.matrix(dataframe)`. Se nel dataframe sono presenti fattori (o vettori carattere) otterremo una matrice in modo carattere.

Esistono diverse funzioni e alcuni operatori per l'algebra matriciale:

`%*%` operatore che effettua il prodotto di matrici

`t()` ritorna la trasposta di una matrice

`det()` ritorna il determinante di una matrice

`diag()` se applicata ad una matrice ritorna la sua diagonale, se applicata ad un vettore ritorna una matrice quadrata con il vettore sulla diagonale e zero altrove, mentre se applicata ad uno scalare ritorna una matrice identità quadrata con ordine pari allo scalare.

`crossprod(A,B)` ritorna il prodotto incrociato di due matrici (lo stesso di `t(A) %*% B` ma più velocemente)

`solve()` usabile con uno o due argomenti: `solve(A)`, se possibile, ritorna la matrice `A` invertita, mentre, `solve(A,y)` se possibile, risolve `A %*% b = y` e ritorna il vettore `b` (vedere `help(solve)` per maggiori dettagli)

`eigen()` calcola autovalori e autovettori e ritorna una lista con due elementi `values` e `vectors`. (es: gli autovalori di `A` sono ottenibili con `eigen(A)$values`)

`qr()` fornisce la decomposizone `QR` di una matrice (sono routine di livello inferiore usate molto spesso in statitisca)

Il libreria `Matrix` fornisce alteriori e più sofisticate funzioni per l'algebra matriciale.

1.14 Dati mancanti

In R i dati mancanti sono indicati con `NA`. Se, per esempio, nel file dei dati che vogliamo importare, i dati mancanti sono indicati con il carattere '`-`' o con la stringa '`MANCA`' occorrerà usare l'argomento `na.strings` nella funzione `read.table()`

```
> exp1.df <- read.table("exp1.txt", header=T,
  + na.strings=c("-", "MANCA"))
```

Il dato mancante si controlla con la funzione `is.na()` che ritorna `TRUE` o `FALSE`. Nel prossimo esempio il vettore di caratteri '`lett`' presenta un dato mancante al sesto elemento

```
>lett
[1] "a" "b" "c" "d" "e" NA  "g" "h" "i" "j"
> is.na(lett)
[1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

La funzione `length()` si usa molto frequentemente e serve per contare gli elementi di una vettore o di una lista. Per contare il numero di dati presenti si può usare `!is.na()` con `sum()` oppure `length()` con la funzione `na.omit()`

```
> length(lett)
[1] 10
> length(!is.na(lett)) #occhio!
[1] 10
> sum(!is.na(lett)) #corretto
[1] 9
> length(na.omit(lett)) #corretto
[1] 9
> length(lett[!is.na(lett)]) #complicato ma corretto
[1] 9
```

In generale ogni operazione su un `NA` diventa `NA`. È sbagliato usare `lett==NA`, in quanto `NA` non è uguale a niente o nessuno, neanche a `NA`!

1.15 Strutture di controllo

R come tutti i linguaggi di programmazione ha delle funzioni che permettono di effettuare cicli (`for` e `while`) e verificare condizioni (`if` e `else`).

L'uso di `for` è piuttosto semplice. Per esempio per stampare, uno alla volta il quadrato di alcuni interi si possono eseguire i seguenti comandi:

```
> a <- c(1,3,5,6,8,4,10)
> a
[1]  1  3  5  6  8  4 10

> for (i in 1:length(a)){
+ print(a[i]^2)
```

```
+ }

[1] 1
[1] 9
[1] 25
[1] 36
[1] 64
[1] 16
[1] 100
```

Ovviamente lo stesso risultato poteva essere ottenuto con il semplice comando `print(a^2)`, che utilizza la semplice algebra elemento per elemento dei vettori di R . Ci sono problemi però che non possono essere risolti senza ricorrere ai cicli. Per esercizio provate a fare dei cicli che stampino gli elementi della prima colonna della matrice seguente moltiplicati per 1, gli elementi della seconda colonna $\times 2$, quella della terza colonna $\times 3$ ecc. Poi cercate di farlo in un doppio ciclo (un `for` dentro un altro `for`)

```
> #creo la matrice da trasporre
> X <- matrix(1:12, nrow=3, byrow=TRUE)
> X
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

Proviamo ora a creare una funzione che trasponga una matrice qualsiasi. Occorreranno due cicli `for`, uno dentro l'altro, per "scannerizzare" le righe e le colonne di una matrice. (Ricordo che in R c'è già la funzione `t()` che svolge egregiamente questo compito).

```
>trasponi <- function(M){
  #creo una matrice vuota delle dimensioni appropriate
  trM <- matrix(rep(NA,length(M)),nrow=ncol(M))
  for(i in 1:nrow(M)) { #ciclo sulle righe di M
    for(j in 1:ncol(M)){ #ciclo sulle colonne di M
      trM[j,i] <- M[i,j] #notare lo scambio tra i e j
    }
```

```
  }
  return(trM)
}
> trasponi(X)
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

Ora proviamo un esempio difficilmente risolvibile senza l'uso dei cicli. Poniamo di avere degli alberi con coodinate x e y:

```
x <- c(45.7, 21.4,  9.2, 35.0, 12.3, 33.1, 18.8, 11.4, 33.3, 11.1)
y <- c(32.9, 33.5, 39.2, 36.7, 33.1, 16.1,  9.2, 21.6, 30.0,  6.9)
```

```
> cbind(x,y)
         x    y
 [1,] 45.7 32.9
 [2,] 21.4 33.5
 [3,]  9.2 39.2
 [4,] 35.0 36.7
 [5,] 12.3 33.1
 [6,] 33.1 16.1
 [7,] 18.8  9.2
 [8,] 11.4 21.6
 [9,] 33.3 30.0
[10,] 11.1  6.9
```

Prima di tutto creiamo una funzione che calcoli la distanza euclidea fra due alberi qualsiasi applicando il teorema di Pitagora: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

```
> dist <- function(xa,ya,xb,yb) return(sqrt((xa-xb)^2 + (ya-yb)^2))
>#proviamo la funzione calcolando la distanza fra i primi due punti
> dist(x[1],y[1],x[2],y[2])
[1] 24.30741
```

creiamo la matrice 10×10 che contenga le distanze fra tutti gli alberi (compresa la distanza fra un albero e se stesso)

```
D <- matrix(rep(NA, 10*10), nrow=10)
> D
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [2,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [3,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [4,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [5,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [6,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [7,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [8,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [9,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA
[10,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA

>#ora eseguiamo i calcoli delle distanze e memorizziamole
>#nella matrice D

> for( i in 1:length(x)){
+    for( j in 1:length(y)){
+       D[i,j] <- dist(x[i],y[i],x[j],y[j])
+    }
+ }

> D
          [,1]      [,2]      [,3]      [,4]      [,5]     [,6]      [,7]
 [1,]  0.00000 24.307406 37.039708 11.354735 33.400599 21.00000 35.851081
 [2,] 24.30741  0.000000 13.465883 13.971399  9.108787 20.96783 24.438699
 [3,] 37.03971 13.465883  0.000000 25.920841  6.842514 33.23883 31.498571
 [4,] 11.35473 13.971399 25.920841  0.000000 22.983690 20.68744 31.916923
 [5,] 33.40060  9.108787  6.842514 22.983690  0.000000 26.86336 24.768125
 [6,] 21.00000 20.967833 33.238833 20.687436 26.863358  0.00000 15.877657
 [7,] 35.85108 24.438699 31.498571 31.916923 24.768125 15.87766  0.000000
 [8,] 36.11343 15.543809 17.736967 28.017316 11.535164 22.38616 14.440222
 [9,] 12.73460 12.404032 25.796318  6.912308 21.227576 13.90144 25.355276
[10,] 43.28002 28.524551 32.355834 38.200131 26.227467 23.84617  8.036168
          [,8]      [,9]     [,10]
 [1,] 36.11343 12.734599 43.280018
 [2,] 15.54381 12.404032 28.524551
 [3,] 17.73697 25.796318 32.355834
 [4,] 28.01732  6.912308 38.200131
 [5,] 11.53516 21.227576 26.227467
```

```
 [6,] 22.38616 13.901439 23.846174
 [7,] 14.44022 25.355276  8.036168
 [8,]  0.00000 23.455703 14.703061
 [9,] 23.45570  0.000000 32.038258
[10,] 14.70306 32.038258  0.000000
```

Notate che gli elementi sulla diagonale sono sempre 0.

Se, per esempio, vogliamo automatizzare l'analisi della varianza vista precedentemente (sezione 1.9) in modo che venga velocemente fatta su 10 file testo diversi ma con la stessa struttura[8] possiamo scrivere:

```
files <- c("exp1.txt","exp2.txt","exp3.txt","exp4.txt",
+ "gennaio.txt","febbraio.txt","marzo.txt","anno2002.txt",
+ "nono.txt","decimo.txt",)

for (f in files) {
+ data.df <- read.table(f, header=T, na.strings="-")
+ print(anova(lm(weight ~ group, data=data.df)))
}
```

In questo ciclo `for`, `f` assume, nel primo ciclo, il valore `"esp1.txt"`, nel secondo `"exp2.txt"`, e via via tutti i valori contenuti in `files` fino all'ultimo `"decimo.txt"`. Ad ogni ciclo il programma leggerà i dati dal nomefile contenuto in `f` ed eseguirà un'analisi della varianza, stampandola.

In qualche secondo si eseguono 10 analisi della varianza su 10 files diversi... provate a fare lo stesso in Excel e misurate il tempo impiegato (`:)`).

All'interno di un ciclo la stampa funziona solo chiamando esplicitamente `print()`. Quindi se fate un ciclo e non vedete nessun output, semplicemente aggiungete dei `print()`.

Per l'uso di `if`, `else`, `while`, per altro molto semplice, potete guardare l'help on line (`help("if")`) o `R-intro.pdf`.

1.16 Tapply, apply, ecc.

Chi è esperto di programmazione si accorgerà ben presto che i cicli (`for` e `while`) in R sono relativamente lenti.

Un buon conoscitore di R cerca di evitare i cicli `for` usando l'algebra matriciale o usando alcune funzioni che sono molto più veloci e comode del `for`.

[8] In realtà è sufficiente che vi siano almeno le colonne `weight` e `group` ma possono esserci altre e diverse colonne e il numero di righe può essere diverso da file a file.

Una di queste è la funzione `tapply` che prende 3 argomenti: il primo è un vettore numerico sul quale vogliamo effettuare i calcoli, il secondo è un vettore di fattori (o una variabile carattere) che dice su quali gruppi, ciclicamente, effettuare il calcolo, e il terzo argomento è una funzione che fa il calcolo che interessa. Per esempio vogliamo calcolare la media di `weight` separatamente per ogni gruppo `group` del dataframe `exp1.df`:

```
> tapply(exp1.df$weight, exp1.df$group, mean)
    Ctrl     Trt1     Trt2
5.032000 4.700000 5.555556
```

Se al posto della media vogliamo calcolare la deviazione standard:

```
> tapply(exp1.df$weight, exp1.df$group, sd)
     Ctrl      Trt1      Trt2
0.5830914 0.8896227 0.4588331
```

Il valore restituito da `tapply()` è un semplice *ragged arrary* (vedi sezione 1.10).

Nel caso volessimo effettuare un'operazione su dei gruppi basati sulla combinazione di più di un fattore si può usare una **lista** dei fattori come secondo argomento. Se, per esempio con il data set `ChickWeight`, volessimo fare una medie del peso dei pulcini per ogni giorno in cui sono stati misurati (`Time`) e per le quattro diete (`Diet`) a cui sono stati sottoposti possiamo dare il comando

```
> data(ChickWeight)
> d.df <- ChickWeight
> tapply(d.df$weight,list(d.df$Time,d.df$Diet),mean)
```

L'oggetto ritornato sarà una matrice che avrà sulle righe le medie dei tempi (0, 2, 4 ecc.) e sulle colonne le medie delle quattro diete.

```
            1     2     3        4
0   41.40000  40.7  40.8  41.0000
2   47.25000  49.4  50.4  51.8000
4   56.47368  59.8  62.2  64.5000
6   66.78947  75.4  77.9  83.9000
8   79.68421  91.7  98.4 105.6000
10  93.05263 108.5 117.1 126.0000
12 108.52632 131.3 144.4 151.4000
14 123.38889 141.9 164.5 161.8000
```

```
16 144.64706 164.7 197.4 182.0000
18 158.94118 187.7 233.1 202.9000
20 170.41176 205.6 258.9 233.8889
21 177.75000 214.7 270.3 238.5556
```

La funzione `apply()` è forse più semplice e prende anch'essa tre argomenti: il primo è un array (di solito una matrice), il secondo di solito è il numero 1 per effettuare operazioni sulle righe, 2 per le colonne e c(1,2) per righe e colonne, mentre il terzo argomento è la funzione che vogliamo applicare alla nostra matrice. Ecco un esempio per calcolare le somme per riga e per colonna su una matrice `X`:

```
row.sums <- apply(X, 1, sum)
col.sums <- apply(X, 2, sum)
```

1.17 Un po' di grafica

Si può avere un'idea delle capacità grafiche di R eseguendo il comando `demo(graphics)` come suggerito dal banner iniziale.

Concretamente però abbiamo già visto un'applicazione della funzione `plot()`. L'applicazione più semplice è `plot(x,y)` per plottare il vettore `x` contro quello `y` oppure `plot(y ~ x)`. Ecco un breve elenco di altri argomenti che possono essere passati a `plot()`:

`type="p"`	per avere punti
`type="l"`	per avere linee
`type="b"`	entrambi
`type="n"`	niente, produce solo la "cornice"
`xlim=c(0,10)`	l'ascissa parte da 0 e arriva a 10
`ylim=c(-5,10)`	l'ordinata parte da -5 e arriva a 10
`main="Titolo"`	titolo del grafico
`xlab="Label X"`	titolo sull'asse x
`ylab="Label Y"`	titolo sull'asse y
`pch=16`	usa il simbolo 16 per i punti (cerchio pieno)
`lty="dashed"`	linea tratteggiata
`col="red"`	simboli o linee di colore rosso

Questa è solo una minima parte dei possibili argomenti, date un'occhiata a `help(plot)` e a `help(par)` per altre possibilità.

Altre funzioni utili (*low-level*) sono:

`points(x, y,pch=14)`	aggiunge punti ad un grafico già esistente
`lines(x, y,lty="dotted", col="green")`	aggiunge una linea tratteggiata verde ad un grafico già esistente
`text(x, y, stazioni)`	aggiunge un vettore di stringhe sul grafico nei punti con coordinate x e y (utile per la statistica multivariata)
`legend(x,y,"Questa Legenda",pch=16)`	aggiunge la legenda
`axis(1, ...)`	per aggiungere un asse x
`abline()`	aggiunge una retta

Altre funzioni che creano grafici di tipo diverso sono:

`boxplot(y ~ group)`	boxplot (come in Figura 1.2)
`hist(x)`	distribuzione di frequenza
`barplot(x)`	istogrammi
`coplot(y ~ x | group)`	produce uno scatter x y per ogni gruppo di `group`
`pie(x)`	grafici "a torta"

Naturalmente esistono molte altre funzioni e altri argomenti di cui potete trovare descrizioni nell'help o nei manuali.

Se volete farvi dei grafici "personalizzati", sarete molto probabilmente costretti ad esaminare l'help della funzione `par()`. `par()` accetta moltissimi parametri (alcuni li abbiamo già esaminati (es: `pch` `lty`, `col`) e cambia i "default" grafici per tutta la sessione successiva (es: `par(bg="peachpuff")` setta il background in colore "peachpuff" per tutti i grafici che farete in seguito).

Qualche grafico come esempio

Nel data set `Iris` che possiamo ottenere con il domando `data(iris)` sono presenti i dati di lunghezza e larghezza di petali e sepali di tre specie di iris.

Nel caso volessimo fare uno scatterplot con le medie della lunghezza dei petali e il rispettiva barra di errore per le tre specie di iris, per prima cosa

daremo il seguente comando per calcolare le medie e le deviazioni standard sul vettore della lunghezza dei petali:

```
> mPet <- tapply(iris$Petal.Length, iris$Species, mean)
> mPet
    setosa versicolor  virginica
     1.462      4.260      5.552
> dsPet <- tapply(iris$Petal.Length, iris$Species, sd)
> dsPet
    setosa versicolor  virginica
 0.1736640  0.4699110  0.5518947
```

poi creaiamo un vettore con i tre nomi delle specie usando il comando `unique` e un vettore numerico per l'asse delle ascizze della lunghezza opportuna

```
> specie <- as.character(unique(iris$Species))
> specie
[1] "setosa"     "versicolor" "virginica"
> x <- 1:length(specie)
> x
[1] 1 2 3
>
```

ora possiamo creare il grafico avendo cura di:

- creare dapprima un plot con le sole linee `type="l"`
- creare un titolo al grafico con `main=`
- cambiare la scala dell'asse `x` e `y` con `xlim=` e `ylim=`
- mettere delle etichette opportune agli assi con le unità di misura possibilmente con i comandi `xlab=` e `ylab=`
- abolire l'asse `x` con il comando `xaxt="n"`.
- creare un'asse `x` con il comando `axis`, in modo da mettere i *ticks* dove vogliamo noi (`at=`) e con l'etichetta che vogliamo noi `label=`
- aggiungere le barre d'errore alle medie lunghe ± 2 volte la deviazione standard con il comando `arrows`.

- plottando dei bei punti in rosso con il comando `points`

Quindi in sequenza i comandi sono:

```
plot(mPet ~ x, type="l", main="Iris - Petali", xlim=c(0.3,3.8),
ylim=c(0,8), xlab="Specie", ylab="Lunghezza (mm)", xaxt="n")

axis(1,at=x,label=specie)

arrows(x, mPet-2*dsPet, x, mPet+2*dsPet, angle=90,length=0.04, code=3)

points(mPet ~ x, pch=21, col = "black", bg="red", cex=1.8)
```

Il risultato è nella figura 1.3.

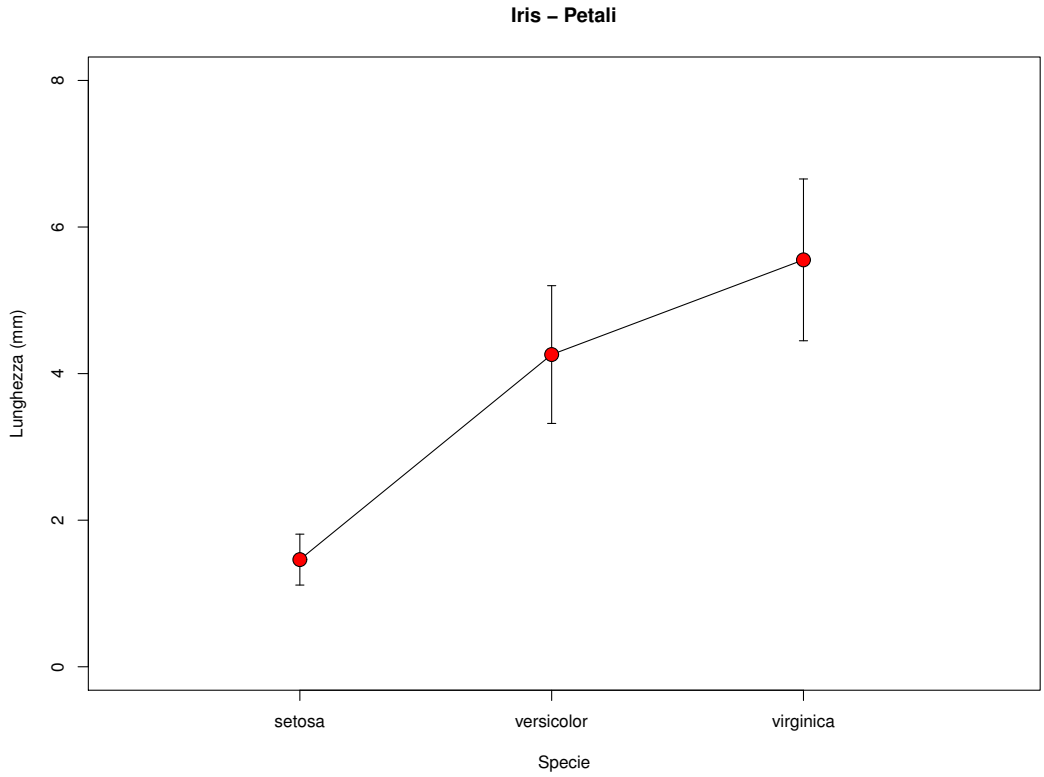


Figura 1.3: Scatterplot con barre d'errore che rappresentano 2 volte la deviazione standard

Il comando `arrows` prende come argomento le `x` di partenza delle frecce, le `y` di partenza e la `x` e la `y` di arrivo, l'angolo della "freccia" (90^o per avere una "T"), `length` per la lunghezza del segmento superiore della "T" e con `code=3` la "freccia" viene fatta sia in alto, sia in basso.

Quando, come in questo caso, lungo le ascisse è rappresentato un fattore senza un vero e proprio valore numerico, uno dei grafici più usati è il `barplot`. Se volessimo produrre un barplot il comando porebbe essere il seguente:

```
xc <-barplot(mPet, main="Iris - Petali", ylim=c(0,8),
xlab="Specie", ylab="Lunghezza (mm)",
col=heat.colors(length(specie)))

arrows(xc, mPet, xc, mPet+2*dsPet, angle=90,length=0.08)
```

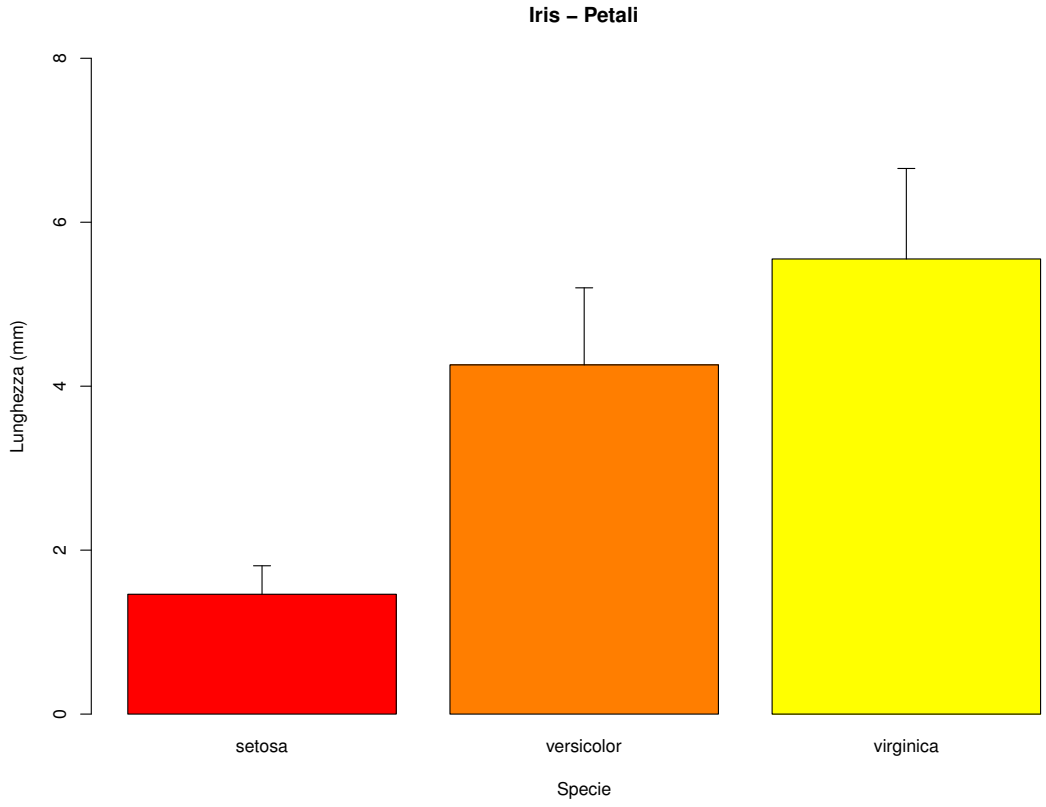


Figura 1.4: Barplot con barre d'errore che rappresentano 2 volte la deviazione standard

Notate che `barplot` restituisce un vettore `xc` con le coordinate sull'asse delle ascisse dei punti centrali delle barre, che sono molto utili per poter disegnare successivamente le barre d'errore al centro delle barre colorate con il comando `arrows`. Stavolta però le disegnamo solo verso l'alto. Il risultato è nella figura 1.4. Non è stato necessario ridisegnare l'asse `x`.

Nella figura 1.5 è mostrato un esempio di grafico un po' più complesso relativo al dataset `ChickWeight`. Si vuole illustrare la crescita nel tempo

di un certo numero di pulcini sottoposti a 4 diete diverse. Abbiamo già calcolato la crescita media per dieta, per giorno nel paragrafo dedicato alla funzione `tapply`. Qui per disegnare le barre d'errore calcoliamo, oltre alla media, anche la deviazione standard.

```
> data(ChickWeight)
> d.df <- ChickWeight
> m<- tapply(d.df$weight,list(d.df$Time,d.df$Diet),mean)
> m
           1     2     3        4
0   41.40000  40.7  40.8  41.0000
2   47.25000  49.4  50.4  51.8000
4   56.47368  59.8  62.2  64.5000
6   66.78947  75.4  77.9  83.9000
8   79.68421  91.7  98.4 105.6000
10  93.05263 108.5 117.1 126.0000
12 108.52632 131.3 144.4 151.4000
14 123.38889 141.9 164.5 161.8000
16 144.64706 164.7 197.4 182.0000
18 158.94118 187.7 233.1 202.9000
20 170.41176 205.6 258.9 233.8889
21 177.75000 214.7 270.3 238.5556

> s<- tapply(d.df$weight,list(d.df$Time,d.df$Diet),sd)
> s
            1         2         3         4
0   0.9947229  1.494434  1.032796  1.054093
2   4.2781575  2.875181  2.412928  1.932184
4   4.1280668  2.299758  2.780887  2.549510
6   7.7572829  4.168666  5.704774  5.065131
8  13.7761978 14.802778 12.348639  9.335714
10 22.5424875 24.295633 20.190482 11.430952
12 32.6179372 37.458273 27.072742 15.276707
14 37.3837632 43.697063 34.503623 15.732486
16 45.0374028 52.824763 44.580015 25.302613
18 49.2194964 63.331667 57.587518 33.557413
20 55.4358400 70.252244 65.243901 37.568086
21 58.7020726 78.138126 71.622545 43.347754
```

In entrambe queste matrici abbiamo le diete sulle colonne e il tempo (giorni) sulle righe. Per cui:

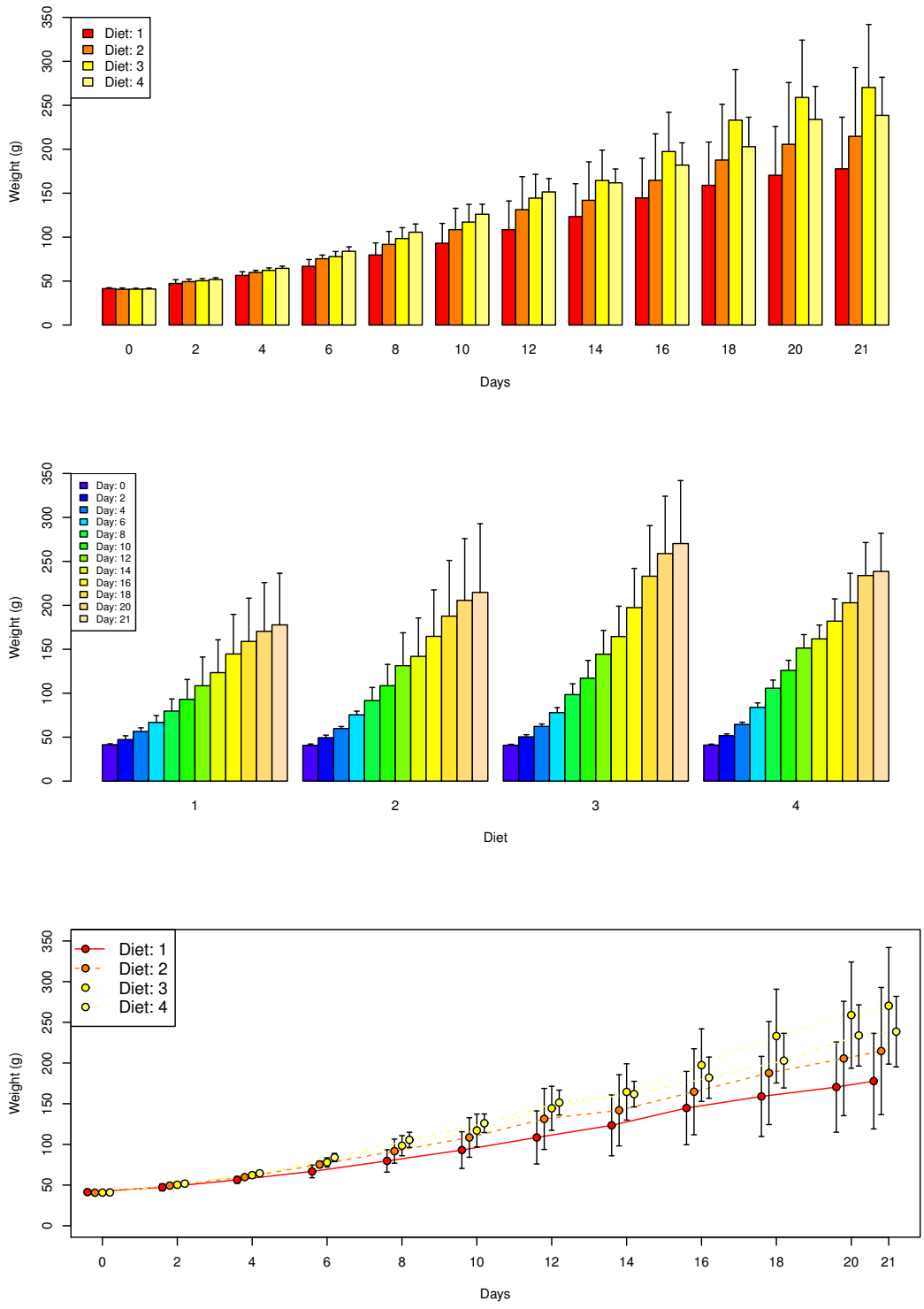


Figura 1.5: Dataset `ChickWeight`. Le barre rappresentano ± una deviazione standard. Quale è la rappresentazione più chiara?

```
> diets <- colnames(m)
> days <- rownames(m)
```

aggiungiamo un vettore di colori per identificare le diete:
`>colori <- heat.colors(length(diets))`

Quindi tracciamo tre grafici su una stessa pagina uno sopra l'altro (una matrice di grafici di 3×1):
`>par(mfrow=c(3,1))`

Nel primo grafico mettiamo un barplot utilizzando le trasposte della matrice `m` ed `s` per avere i `days` sulle ascisse e le barre colorate relative alle diverse diete (`diet`) affiancate a gruppi `beside=T`.

```
> xc <-barplot(t(m), beside=T, col=colori, xlab="Days", ylab="Weight (g)",
+ ylim=c(0, 350))
> arrows(xc, t(m), xc, t(m)+t(s), angle=90, length=0.02)
> legend("topleft", fill=colori, legend=paste("Diet:",diets))
```

Quindi aggiungiamo un altro barplot stavolta senza traspost e quindi con le diete sull'asse delle ascisse. I colori stavolta identificano i giorni e ne servono tanti quanti sono i giorni dell'esperimento.

```
> colori2 <- topo.colors(length(days))
> xc <-barplot(m, beside=T, col=colori2, xlab="Diet", ylab="Weight (g)",
+ ylim=c(0, 350))
> arrows(xc, m, xc, m+s, angle=90, length=0.02)
> legend("topleft", fill=colori2, legend=paste("Day:",days))
```

Infine proviamo un grafico per punti con le medie e le barre d'errore sopra e sotto:

```
days <- as.numeric(days)
## produco un grafico vuoto e senza asse delle ascisse
plot(m[,1] ~ days, type="n", xlab="Days", ylab="Weight (g)", ylim=c(0, 350),
     xaxt="n")
## aggiungo l'asse x e i ticks in corrispondenza dei giorni dell'esperimento
axis(1, at=days,lab=days)

## con un ciclo for sulle diete aggiungo le linee e le barre d'errore
 for (d in 1:length(diets)){
## sposto leggermente le x per non far sovrapporre le barre
   jdays <- days-0.6+d*0.2
   lines(m[,d] ~ jdays, col=colori[d], lty=d)
```

```
   arrows(jdays, m[,d]-s[,d], jdays,  m[,d]+s[,d], angle=90, length=0.02, code=3
}
## traccio i punti per ultimi
 for (d in 1:length(diets)){
   jdays <- days-0.6+d*0.2
   points(m[,d] ~ jdays, col=colori[d], pch=16, cex=1.2)
}
## legenda
legend("topleft", lty=1:length(diets), col=colori,pch=16,
       legend=paste("Diet:",diets),cex=1.2)
```

poi riporto il numero di grafici per pagina al default

```
>par(mfrow=c(1,1))
```

1.18 Esempi ed esercizi per voi

1. Dato un vettori di dati `weight` in grammi, standardizzatelo, togliendo la media e dividendo il risultato per la deviazione standard. Confrontate il risultato con quello della funzione `scale()`.
2. Se `x<-c(2,2,3)` e `y<-1:6`. Cosa succede se provo `x * y`?
3. Perchè il risultato di `x * y`, dove `x<-c(2,2,3)` e `y<-1:7`, è così ed ho un `warning`? Perché nel caso precedente non ho il `warning`?
4. Cosa succede se moltiplico/sommo/divido/sottraggo due vettori di lunghezza diversa? Questa è la "regola del riciclaggio dei vettori" che si applica quando i vettori sono di lunghezza diversa.
5. Utilizzando la funzione `rnorm` per creare due variabili casuali (anche correlate) `x` e `y` (per esempio: `x <- rnorm(10,mean=50, sd=5)` verificate empiricamente che:
   (a) $E(k\ x) = k\ E(x)$
   (b) $var(k\ x) = k^2\ var(x)$
   (c) $sd(k\ x) = k\ sd(x)$
   (d) $E(x + y) = E(x) + E(y)$
   (e) $var(x + y) = var(x) + var(y) + 2cov(x, y)$
   (f) $E(x\ y) = E(x)\ E(y) + cov(x, y)$ (Vedi nota:[9])

[9] Funziona esattamente solo se la $cov(x, y)$ al denominatore ha n e non $n - 1$

(g) $E\left(\frac{1}{x}\right) \neq \frac{1}{E(x)}$

(h) $var(x\ y) \neq var(x)\ var(y) + 2cov(x, y)$

dove k è una costante, $E()$ è il valore atteso (media), $var()$ è la varianza, $sd()$ è la deviazione standard, $cov()$ è la covarianza.

Attenzione che a volte l'operatore '`==`' restituisce `FALSE` anche per piccoli ed inevitabili errori di arrotondamento interni alla macchina. Una delle soluzioni più semplici è controllare la differenza tra la parte a destra e a sinistra delle eguaglianze. Se la differenza è molto molto piccola, allora si può ritenere verificata l'uguaglianza. Un'altra soluzione è usare la funzione `all.equal()`

6. Che differenza c'è fra `rep(1:3,3)` e `rep(1:3,c(3,3,3))`?

7. Cosa producono le espressioni `sum(ages < 30)`, `sum(ages[ages < 30])` e `sum(nomi[ages < 30])` e perché?

8. Cosa produce la seguente espressione
   `nomi[nomi=="Stefano" | nomi=="Franco"]`?

9. Perchè `abs(z)` ha lo stesso effetto di `z[z < 0] <- -z[z < 0]`?

10. Come posso estrarre nella matrice `X` l'i, j-esimo elemento?

11. Come posso estrarre tutte le colonne che vanno da 3 a 6?

12. Come posso estrarre tutte le righe tranne la i-esima?

13. Cosa produce `diag(3)`? e `diag(1:3)`?

14. Scrivere due funzioni che calcolino la traccia di una matrice (somma degli elementi sulla diagonale). Nella prima usare la funzione apposita di R per estrarre la diagonale e nella seconda usare un cicolo *for* per accedere agli elementi della matrice.

15. Scrivere utilizzando ciclo `for` una funzione che ritorni un vettore per fare la media degli elementi di ogni riga di una matrice. Aiuto: usate la funzione `nrow()` per contare le righe della matrice. Fare una seconda funzione che faccia la media delle colonne della matrice. Confrontare i risultati con le funzioni di sistema `colMeans` e `rowMeans`.

Capitolo 2

Primi approcci alla statistica con R

Alcune semplici applicazioni con R per capire alcuni concetti di statistica. I seguenti esempi fanno uso di cicli `for`.

2.1 Il test di permutazione

Il test di permutazione è uno dei tanti testi che prevedono il ricampionamento delle osservazioni e ha molti dei vantaggi, fra cui quello di non avere bisogno di una distribuzione teorica di riferimento. Il concetto principale del test *ad hoc* che svolgiamo qui sotto è quello di costruirsi una distribuzione empirica delle differenze tra le medie sotto l'ipotesi nulla. Tale distribuzione di riferimento è ottenibile con un'attribuzione casuale degli individui ai due gruppi (Trattato= `tr` e Controllo = `co`) attraverso una permutazione, appunto casuale, effettuata con la funzione `sample()`.

Il ragionamento potrebbe essere simile al seguente: se fosse vera l'ipotesi nulla, cioè che il trattato e il controllo non differiscono nella sopravvivenza dei topi, allora attribuire un topo ad un gruppo o all'altro sarebbe indifferente. Posso quindi crearmi una distribuzione di differenze tra le medie dei due trattamenti sotto ipotesi nulla, attribuendo a caso le osservazioni di sopravvivenza ai due gruppi e usarla come distribuzione di riferimento. Se la probabilità di osservare la mia effettiva (quella vera) differenza tra le medie, rispetto alla distribuzione empirica ottenuta, è sufficientemente bassa, posso rigettare l'ipotesi nulla. Se la probabilità, al contrario, fosse sufficientemente alta, posso non rigettare/accettare l'ipotesi nulla. Per convenzione, nel caso di test a due code, si usano i limiti di probabilità del 2.5% e del 97.5%.

```
##Inserisco i dati dei trattati
tr<-c(94,197,16,38,99,141,23)
##inserisco i dati deicontrolli
co<-c(52,104,146,10,50,31,40,27,26)
##creo un vettore di stringhe "tr" e "co"
gr<- c(rep("tr",length(tr)),rep("co",length(co)))
gr
##creo un dataframe gr e surv (concatenazione di tr e co)
m.df <- data.frame(gr,surv=c(tr,co))
##creo un vettore vuoto grande mille elementi
diffs<-numeric(1000)
##studiate cosa fa sample
sample(m.df$gr)
##ciclo che fa le permutazioni

for (i in 1:length(diffs)) {
  ms<-tapply(m.df$surv,sample(m.df$gr),mean)
  diffs[i]<-ms["co"] - ms["tr"]
##  print(paste("Ciclo: ", i))
}

##ottengo una distributione di frequenza di diffs
hist(diffs)

##stampa il 2.5% e il 97.5% percentile
quantile(diffs,prob=c(0.025,0.975))

##plotto i limti
abline(v=quantile(diffs,prob=c(0.025,0.975)))
## altro modo per calcolare i percentili
sort(diffs)[c(25,975)]

##calcola la le medie vere
msvero<-tapply(m.df$surv,m.df$gr,mean)
##calcola la differenza fra le medie vere
diffvero<-msvero["co"] - msvero["tr"]
print(diffvero)

##plotta un linea verticale in corrispondenza della diff vera
abline(v=diffvero, col="red")
### Ora posso trarre le conclusioni del test.
```

```
## calcolo il P della mia differenza osservata vera
2*mean(diffs < diffvero)

## Per conferma effettuiamo anche una
## batteria di test parametrici e non parametrici
anova(lm(surv ~ gr,data=m.df))
t.test(surv ~ gr,data=m.df, var.equal=T)
t.test(surv ~ gr,data=m.df)
wilcox.test(surv ~ gr,data=m.df)
```

2.2 Dimostrazione empirica del teorema centrale del limite

Prima di discutere il teorema centrale del limite è utile ripassare alcuni concetti importanti quando si vuole stimare o misurare una certa quantità. Si possono commettere due tipi di errore: un errore di tipo *sistematico* e un errore di tipo *casuale*.

Il primo determina una mancanza di **accuratezza** ed ha a che fare con un tipo di errore che tende ad essere costante, sistematico, sempre nella stessa direzione rispetto al valore vero (sempre maggiore o sempre minore). Un esempio può essere quello della bilancia starata che tende sempre a ritornare un peso più grande di circa 800 grammi rispetto al peso reale. Questo tipo di errore tende a dare una sovra-stima o una sotto-stima del valore reale. Un modo per misurare questo tipo di errore è la differenza fra il valore reale (supponendo di conoscerlo) e la media delle misure ripetute effettuate.

L'errore di tipo casuale invece determina un errore di **precisione** ed è sostanzialmente una scarsa ripetibilità della misura. Per esempio pesando ripetutamente lo stesso oggetto la bilancia tende a ritornare valori leggermente diversi da una volta all'altra. L'errore che si commette è semplicemente legato ad una certa variabilità casuale della stima attorno ad un valore medio e quest'ultimo potrebbe anche essere molto vicino al valore reale (se non lo fosse mancherebbe di accuratezza) e quindi in teoria non conduce a nessuna sovra o sottostima del valore reale. Un modo per misurare questo tipo di errore è ricavabile dalla differenza fra le singole misure e la medie delle misure stesse (es: la deviazione standard delle misure effettuate).

Le seguenti figure dovrebbero chiarire il concetto.

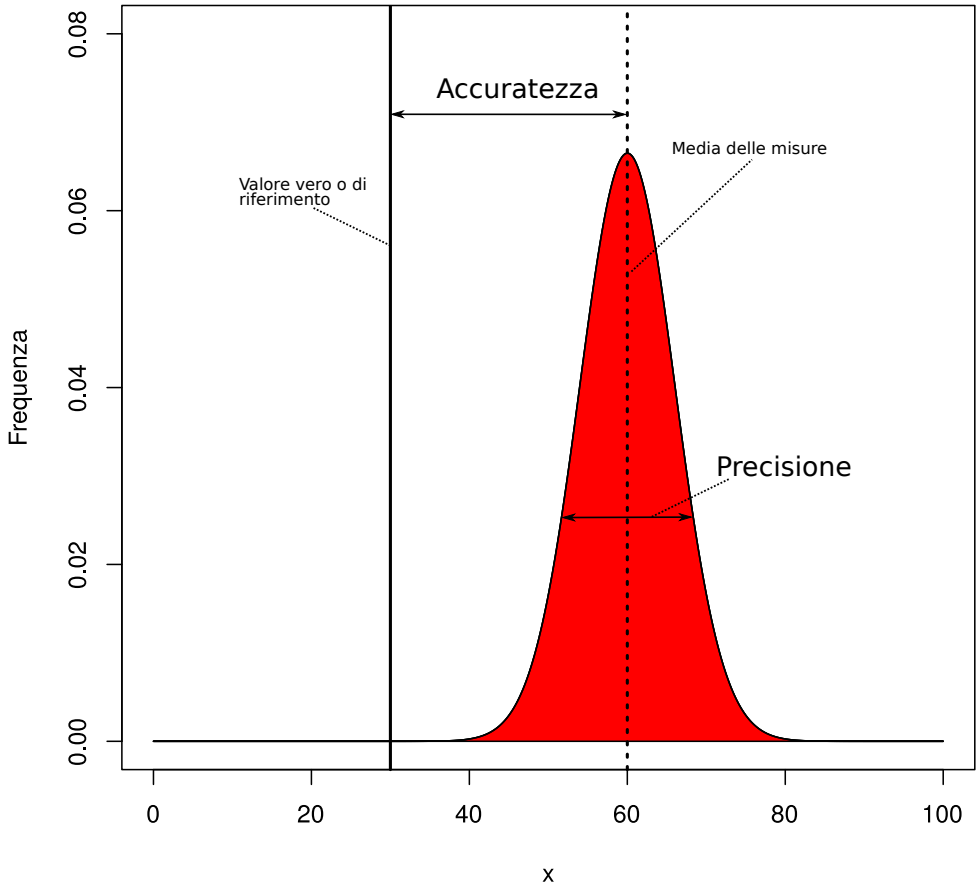


Figura 2.1: Illustrazione dei possibili errori di accuratezza e precisione. L'area rossa rappresenta la distribuzione di frequenza delle misure effettuate.

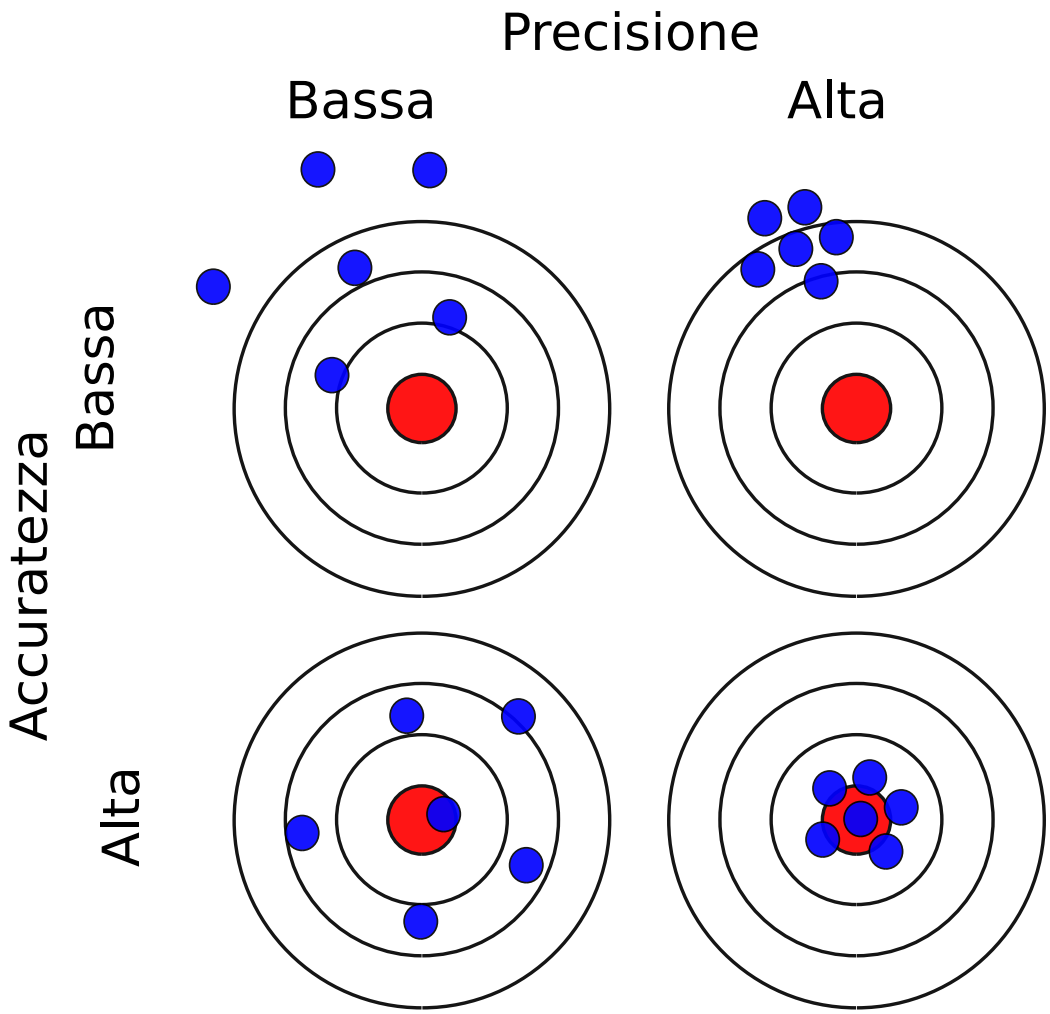


Figura 2.2: Un'altra illustrazione dei possibili errori di accuratezza e precisione

In questa sezione ci chiediamo:

- Cosa succede se aumento le dimensioni del campione?
- In particolare cosa succede alla media se la dimensione del campione aumenta?
- E alla variabilità dei dati? La deviazione standard aumenta o diminuisce all'aumentare della dimensione del campione?
- Se prendo tanti campioni e calcolo tante medie cosa succede alla variabilità di queste medie?
- Perchè la distribuzione normale è così importante in Ecologia e nelle scienze in generale?

Immaginiamo di avere a disposizione tutti i dati del nostro universo (quella che normalmente viene chiamata una popolazione). Immaginiamo per una volta di non dover campionare, ma di poter fare un censimento esaustivo in modo da conoscere esattamente (non stimare) media e deviazione standard.

Immaginiamo poi di prendere dei campioni a dimensioni crescenti da questa popolazione e toglierci la curiosità di vedere cosa succede alle medie di questi campioni e controllare se stimano bene o male la media vera del nostro universo.

Stabiliamo che il nostro universo `u` sia costituito da 10000 individui e di prendere campioni a dimensioni crescenti (5, 10, 15, 25, 50, 80, 120, 200, 400, 800, 1200, 2000). Rifacciamo questa procedura molte volte (500). Avremo, per esempio, una matrice che sarà larga 12 colonne e lunga 500 righe. Ogni colonna conterrà una medie riferita ad una delle 12 dimensioni diverse dei nostri campioni. Nella prima colonna avremo le medie prese dai campioni con 5 individui, nella seconda quelle relative a 10 individui, nella terza quella con 15 e così via fino alla 12esima colonna in cui avremo i campioni con 2000 individui. Ogni riga invece conterrà una replica delle 500 volte in cui ripeteremo queste serie di 12 estrazioni. Avremo anche altre due matrici con le stesse dimensioni: una contenete la deviazione standard e una con l'errore standard.

Proviamo ad eseguire il seguente script cambiando la distribuzione dei dati della popolazione - universo:

- prima con popolazione con dati **normali**
- poi con popolazione con dati a distribuzione piatta (**uniforme**)
- e poi popolazione con dati a distribuzione **esponenziale**

R ci permette di fare questo esperimento in modo molto veloce.

```
stderr <- function(x) {sd(x)/sqrt(length(x))}
set.seed(1)
u <-round(rnorm(10000,m=50,sd=15))
#u <-round(runif(10000,min=0,max=100))
#u <- round(rexp(10000,r=0.05))
# lines(1:100,length(u)*dnorm(1:100,m=mean(u), s=sd(u)))

#u[u < 0] <- 0
mean(u)
sd(u)
#postscript("stderr.eps", hor=F)
#par(mfcol=c(2,1))
hist(u, main="Distribuzione di frequenza nella popolazione", ylab="Frequenza")

radenne<-seq(5,50,4)
enne <- c(5,10,15,25,50,80,120,200,  400,800,1200,2000)
length(enne)


ncampioni<- 500

m <- matrix(NA, nrow=ncampioni,ncol=length(enne))
sds <- m
se <- m


#set.seed(1)
for (j in 1:length(enne)){
  for (i in 1:ncampioni) {
    s <-sample(u,(enne[j]))
    m[i,j] <- mean(s)
    se[i,j] <- stderr(s)
    sds[i,j] <- sd(s)
  }
}

limiti <- range(m)
##plot(m[1,] ~enne, ylim=limiti, pch=".", ylab="Medie dei campioni",
```

```
##   xlab="Dimensione del campione (N)")
##for (i in 2:ncampioni) points(m[i,] ~enne, pch=".")

matplot(enne, t(m), pch=16, col="black", cex=0.4, ylab="Medie dei campioni",
xlab="Dimensione del campione (N)")
points(apply(m,2,mean) ~ enne, col="red", type="p",pch=16)
abline(mean(u),0, col="green")
legend("topright",pch=c(16,16,-1), lty=c(NA,NA,1),pt.cex=c(0.4,1,NA),
   col=c("black","red", "green"),
   legend=c("Singole Medie", "Media delle medie",
   "Media della popolazione"))


##plot(sds[1,] ~enne, ylim=c(10,40), pch=".",

matplot(enne, t(sds), pch=16, col="black", cex=0.4,
  ylab="Deviazione Standard dei campioni",
  xlab="Dimensione del campione (N)")
##for (i in 2:ncampioni) points(sds[i,] ~enne, pch=".")
points(apply(sds,2,mean) ~ enne, col="red", type="b",pch=16)
abline(sd(u),0, col="green")
legend("topright",pch=c(16,16,-1), lty=c(NA,NA,1),pt.cex=c(0.4,1,NA),
   col=c("black","red", "green"),
   legend=c("Singole Dev. Standard", "Media delle Dev. Standard",
   "Deviazione standard della popolazione"))


print(paste(c("s.e. Min", "s.e. Max"), range(se)))


#par(mfcol=c(3,2))
limiti <- range(m)
for ( j in 1:length(enne))
  {
    hist(m[,j], freq=F,xlim=limiti,
        main=paste("Distribuzione di ",ncampioni,
        " medie su\n campioni di ", enne[j], "individui." ),
        xlab="")
    curve(dnorm(x, m=mean(m[,j]), s=sd(m[,j])),add=T)
  }
```

```
#par(mfcol=c(2,1))

##plot(se[1,] ~enne, ylim=c(0,9), pch=".",

matplot(enne, t(se), pch=16, col="black", cex=0.4,
ylab="Valore stimato", xlab="Dimensione del campione (N)")

#for (i in 2:ncampioni) points(se[i,] ~enne, pch=".")
points(apply(se,2,mean) ~ enne, col="red", type="b",pch=16)
lines(apply(m,2,sd) ~ enne, col="green")
legend("topright",pch=c(16,16,NA), lty=c(NA,NA,1),pt.cex=c(0.4,1,NA),
    col=c("black","red", "green"),
    legend=c("Singoli Errori standard",
    "Media degli errori standard",
    "Deviazione standard delle medie"))


#dev.off()
```

Da questi grafici si evince che, anche da distribuzioni dell'universo che non sono affatto normali, si ottengono medie che si distribuiscono normalmente. Quindi, in un certo senso, è l'operazione del "fare la media" che determina il fatto che il risultato sia distribuito secondo una normale. In effetti è l'operazione di **somma** che determina la distribuzine normale.

Questo effetto "somma" è evidente nella figura seguente dove mostriamo l'effetto della somma del lancio di uno, due, tre, ... otto dadi. Man mano che si sommano più dadi la distribuzione dei possibili risultati diventa via via più normale.

Spero sia evidente a questo punto l'importanza della distribuzione normale e del perchè viene usata come riferimento.

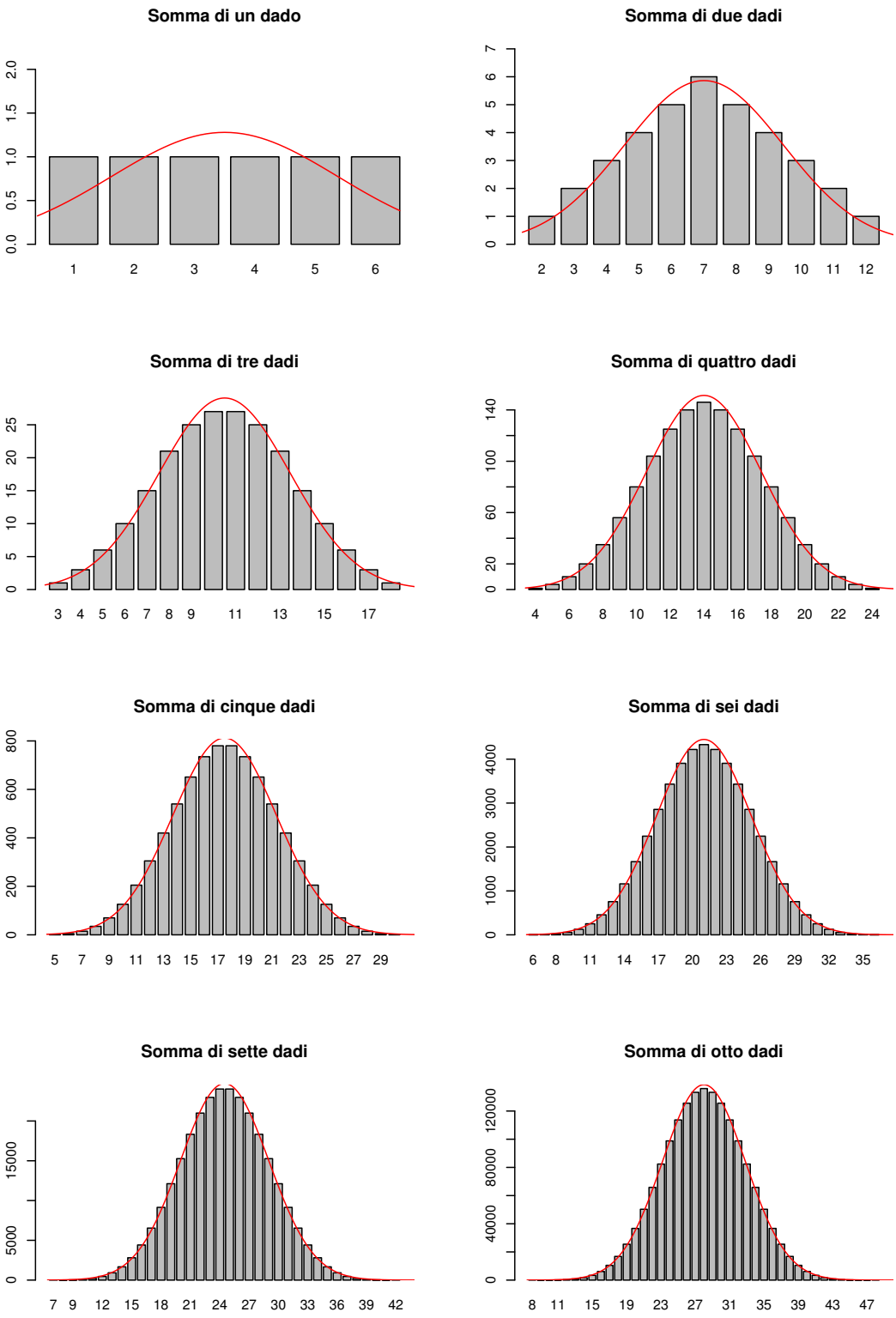


Figura 2.3: Distribuzione dei possibili risultati della somma di uno, due, tre, ..., otto dadi. La linea rossa indica il risultato predetto dalla curva gaussiana.

Curve di Rarefazione con R

L'obiettivo di questa lezione è quello di famigliarizzare, ancora una volta, con l'uso di R , per rispondere ad alcune semplici frequenti in campo ecologico.

Immaginiamo di avere raccolto i dati di una comunità vegetale di alberi, cioè la specie e l'abbondanza di ciascuna specie, su una certa area. Immaginiamo di volerne misurare la diversità biologica.

Ci poniamo le seguenti domande:

- Come di misura la diversità biologica?
- Avremmo campionato un'area sufficientemente grande?
- Avremmo potuto campionare un'area più piccola? Che conseguenze ne avremmo pagato?
- Come confrontiamo la diversità del nostro campione con quello di campioni su aree più piccole o aree più grandi? È noto infatti che la diversità dipende anche dallo sforzo campionario. Più grande è lo sforzo di campionamento, maggiore la diversità.
- Come sarà la relazione fra sforzo di campionamento e diversità?

Alla prima domanda si può rispondere in vari modi: si può utilizzare il numero di specie, o un indice di diversità. L'indice di diversità più noto è quello di Shannon-Wiener e solitamente si indica con H

$$H = -\sum_{i=1}^{s} p_i \log(p_i)$$

dove p_i è la proporzione della i-esima specie ($\sum_i p_i = 1$) e s è il numero delle specie.

Tale indice di diversità è derivato dalla teoria dell'informazione,

Dividendo H con il valore massimo possibile $H_{max} = \log(s)$, si ottiene un indice compreso tra 0 e 1.

Utilizzando i dati nel file `alberi.txt`, cerchiamo di rispondere alle domande successive. L'assunzione generale è che il numero di individui campionati sia una misura dello sforzo di campionamento, che pensiamo quindi essere in qualche modo proporzionale all'area campionata. Dopo lunga e ampia discussione abbiamo deciso che ci attendiamo che l'andamento della curva curva che lega la diversità alle dimensioni crescenti dei campioni dovrebbe essere una curva crescente che però vada a *plateau*.

Importiamo i dati in R e li vediamo sullo schermo:

```
> d.df <- read.table("alberi.txt", h=T)
```

```
> d.df
```

Calcoliamo il totale del numero di individui campionati:

```
> sum(d.df$ni)
```

Prepariamo una funzione che implementi l'indice di Shannon su un vettore di abbondanze e la applichiamo sul nostro campione:

```
> shannonH <- function(arg) {
  #calcola l'indice di shannon su un qualsiasi vettore numerico
  #contenente i valori di abbondanza degli individui
  #prima però lo trasforma in frequenze (pi)
  pi <- arg/sum(arg)
  -1* sum(pi * log10(pi))
}
> shannonH(d.df$ni)
```

Non possiamo dire niente su campioni più grandi del nostro, ma potremmo vedere se campioni più piccoli "catturano" la stessa diversità del nostro campione totale di 1996 indivui. Come detto sopra ci attendiamo che la curva raggiunga un *plateau*. Il valore delle dimensioni del campione in corrispondenza del raggiugimento del *plateau* è la nostra dimensione minima del campione.

L'idea è quindi di prendere dei campioni più piccoli dal nostro campione totale e di vedere l'andamento della diversità in funzione della dimensione crescente del nostro campione piccolo. Predisponiamo quini le dimensioni crescenti dei campioni:

```
> ssize <- c(5,seq(10,90,10),seq(100,900,100),seq(1000,1996,250))
> ssize
> ## manca la dimensione di 1996
> ssize<-c(ssize,1996)
```

Dobbiamo però preparare prima la "comunità" da cui estrarre il nostro campione. Vogliamo una comunità in cui le specie più abbondanti abbiamo maggior probabilità di essere estratte rispetto alle specie meno abbondanti. Un metodo facile per implementare questo concetto in R con i dati sopra è quello di utilizzare la funzione `rep()`:

```
>com <-rep(as.character(d.df$specie),d.df$ni)
>com
```

Ora utilizzando la funzione `sample` e la funziona `table` possiamo calcolare la diversità di un campione piccolo (ad esempio di 10 individui) estratto dalla comunità totale `com`:

```
> s <- sample(com,10)
> s
> st <-table(s)
> ##st è un hash array con le abbondanze di s
> st
> shannonH(st)
```

oppure in un solo comando:

```
> shannonH(table(sample(com,10)))
```

Dovremmo poi ripetere la stessa operazione per campioni di 20, 30, 40 ecc e memorizzare i risultati in un vettore lungo tanto quante sono le dimensioni diverse dei campioni che voglio estrarre. Per fare questo preparo il vettore `H` vuoto della lunghezza appropriata:

```
> H<-rep(NA, length(ssize))
```

quindi faccio un ciclo su tutte le dimensioni dei campioni

```
> for (i in 1:length(ssize)){
    H[i]<-shannonH(table(sample(com, ssize[i])))
}
> H
```

Ora faccio un plot per illustrare l'andamento di `H` in funzione della dimensione del campione `ssize`

```
> plot(H ~ ssize,type="b", pch=16, ylim=c(0.3,1.2), ylab=
"Diversity (H)", xlab="Sample size")
```

La curva è crescente e sembra raggiungere effettivamente un *plateau*, l'andamento è piuttosto irregolare. Occorre fare più ripetizioni dei campioni alle varie dimensioni e fare la media. L'andamento della media probabilmente seguirà un andamento più regolare. Proviamo ad eseguire 100 replice di ogni campione e a calcolare la media e la deviazione standard. Servirà una struttura diversa per immagazzinare i dati della diversità: La matrice `H` avrà un numero di colonne pari alle diverse dimensioni del campioni che vogliamo estrarre (23 dimensioni diverse in questo caso) e un numero di righe pari al numero delle repliche di ogni campione (100 in questo caso)

```
>H<- matrix(NA, nrow=100,ncol=length(ssize))
```

Servirà un ciclo in più per estrarre ogni campione 100 volte:

```
> for (ciclo in 1:100) {
      for (i in 1:length(ssize)){
         H[ciclo,i]<-shannonH(table(sample(com, ssize[i])))
      }
}
```

infine trovo le medie dell'indice di Shannon e le deviazioni standard tra le 100 repliche:

```
> mH <- apply(H,2,mean)
> sdH <- apply(H,2,sd)
```

Il plot delle medie con le deviazioni standard è ora molto più regolare:

```
> plot(mH ~ ssize, type="b", pch=16, ylim=c(0.3,1.2), ylab=
  "Mean Diversity (H)", xlab="Sample size")
> arrows(ssize, mH-sdH, ssize, mH+sdH, angle=90, length=0.02,
  code=3)
> abline(h=shannonH(d.df$ni), lty=2)
```

In linea generale si potrebbe dire che già con circa 300 individui sono vicini al *plateau* di diversità indicato dalla linea tratteggiata.

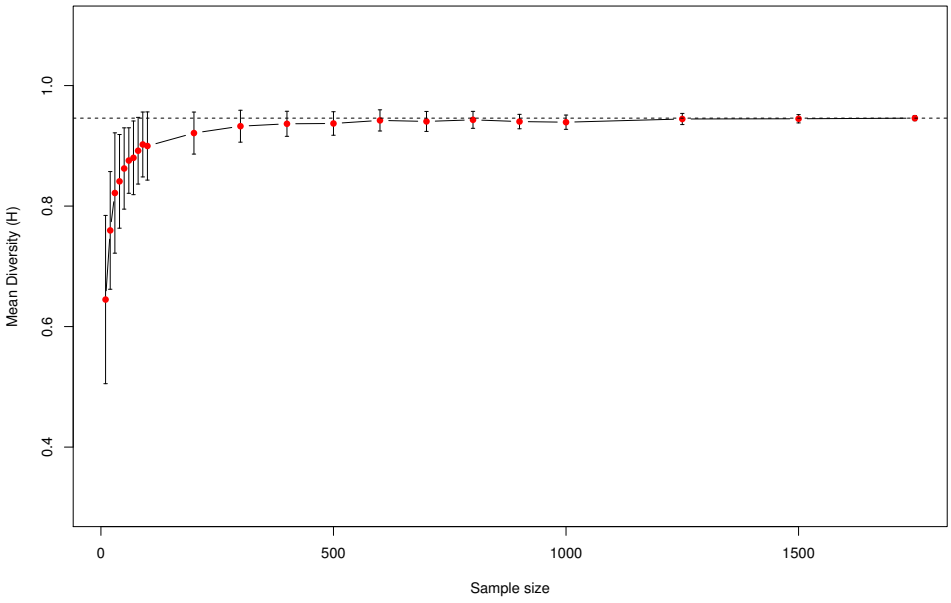


Figura 2.4: Curva di rarefazione con 100 repliche di campionamento a dimensioni crescenti. Le barre d'errore indicano ±1 deviazione standard.

Capitolo 3

Statistica con R

3.1 Definizione di Scienza e ruolo della statistica

La scienza è un processo per imparare a conoscere la natura, nel quale le *idee* su come funziona il mondo sono *in competizione* e sono *misurate* contro le osservazioni (Feynmann, 1965).

Le nostre descrizioni del mondo e le nostre misurazioni sono spesso incerte e inaccurate, per cui gli scienziati hanno bisogno di metodi per stabilire la concordanza fra le idee in competizione e le osservazioni. Questi metodi costituiscono il campo della statistica.

Gli scienziati si devono sforzare di essere il più neutrali possibile verso le idee in competizione.

Abbiamo visto che il **confronto delle ipotesi con i dati** è l'elemento chiave per il progresso della scienza.

Abbiamo già detto che in ecologia i dati sono spesso di "scarsa qualità" e cerchiamo di capire perchè spesso i dati presentano questa caratteristica.

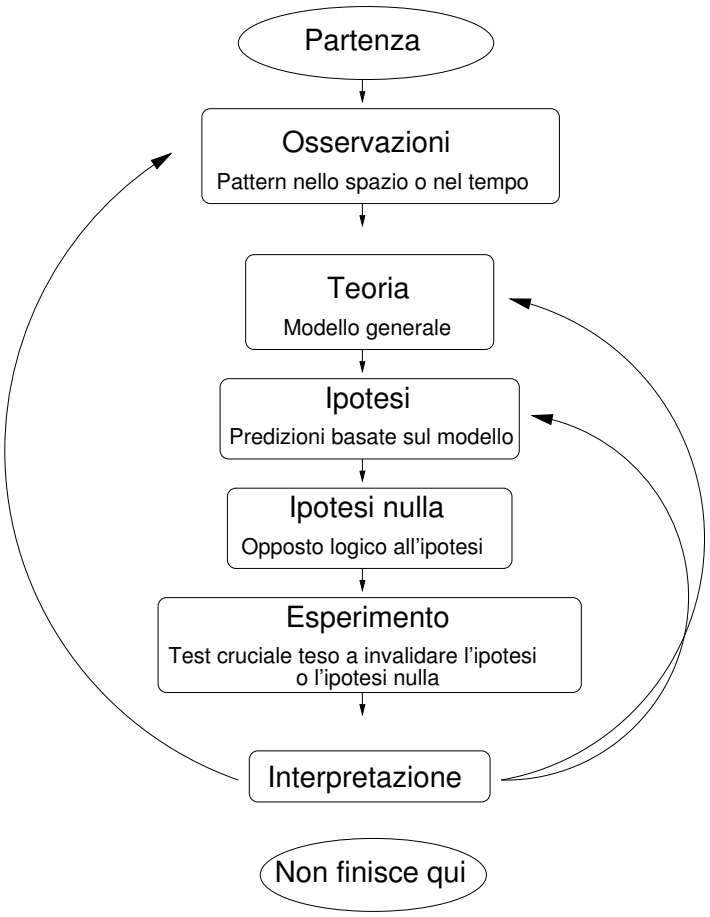
Per esempio cerchiamo di pensare ad un esperimento per capire la dinamica di una popolazione di balene azzurre.

- Avete idea di come sia possibile condurre un esperimento "manipolativo" (che modifichi le condizioni ambientali o la popolazione stessa) per testare ipotesi precise?

- Avete idea delle repliche possibili?

Quindi generalizzando è frequente che i sistemi ecologici presentino le seguenti caratteristiche che rendono la sperimentazione e la raccolta di "buoni dati" particolarmente difficile.

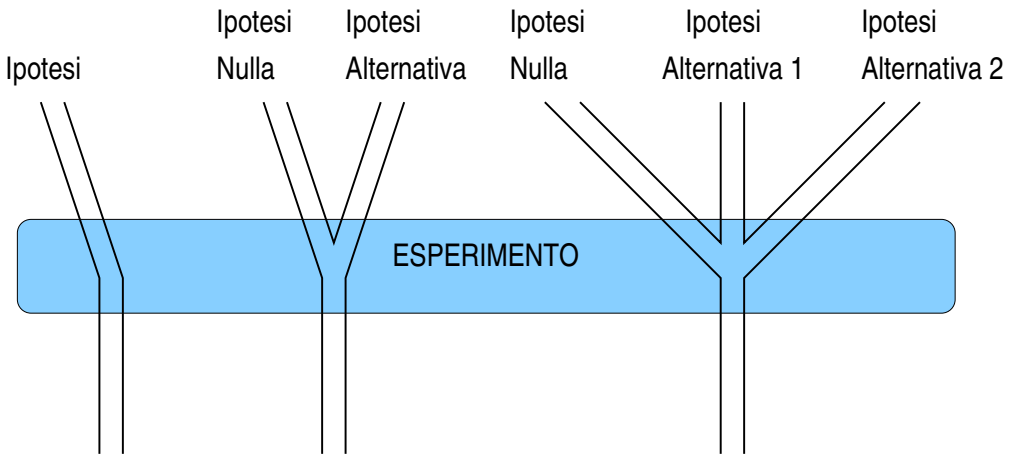
- scala temporale molto lunga
- scala spaziale molto grande
- scarsa replicazione
- incapacità di controllo dei vari fattori e aspetti implicati nell'esperimento



Popper (1979) ha rivoluzionato la filosofia della scienza nel ventesimo secolo sostenendo che le ipotesi non possono essere provate, ma solo *invalidate*. Anche se i dati si adattano bene alla teoria non si può dire che la teoria sia corretta e si immagina che ci possano essere teorie che possono spiegare ancora meglio i dati raccolti. Quindi secondo Popper il carattere essenziare di una teoria che si possa definire scientifica è la sua possibile *falsificabilità*. Alcune presunte "teorie scientifiche" secono Popper non sono in realtà da considerare veramente scientifiche, ma "pseudo-scienza" in quanto non sono falsificabili

(es: psiconalisi di Freud, o la teoria della storia di Marx), altre teorie invece hanno questa caratteristica importante e sono ritenute pientamente scientifiche (l'esempio preferito di Popper era la teoria della relatività di Einstein). Anche l'ecologia ha avuto e ha delle teorie ritenute non falisficabili come la teoria trofo-dinamica di Lyndemann.

L'essenza dell'idea di Popper è quella di sfidare un'ipotesi ripetutamente con esperimenti critici (cruciali). Se l'ipotesi "sta in piedi" e resiste a ripetuti esperimenti non viene validata, ma piuttosto acquisisce un grado di rispetto e in pratica viene trattata come se fosse vera.



Qualcuno ha paragonato questa procedura all'arrampicarsi su un albero dove ogni biforcazione del ramo corrisponde ad un possibile risultato di un esperimento e la "direzione di salita" viene decisa dagli esperimenti stessi.

Chamberlain alla fine del 1890 sostiene l'esigenza di sviluppare *ipotesi multiple* rispetto a ipotesi singole per evitare di incorrere in guai.

Contemporaneamente alle idee di Popper vengono presentati i lavori di **Ronald Fisher** and **Karl Pearson** e altri che hanno sviluppato molta della moderna statistica e la teoria associata con i "*test delle ipotesi*". In sostanza nei test delle ipotesi viene calcolata la probabilità che i dati siano stati osservati qualora l'ipotesi nulla fosse vera. Se la probabilità è sufficientemente piccola (normalmente 0.01 o 0.05) allora noi "rigettiamo" l'ipotesi nulla. Per completare il lavoro però noi dovremmo anche calcolare la potenza del test (probabilità di rigettare l'ipotesi nulla quando questa è falsa).

Thomas Kuhn (1962) ha introdotto i concetti di "*scienza normale*", "*paradigma scientifico*" e "*rivoluzioni scientifiche*". Secondo Kuhn, gli scienziati normalmente operano entro specifici paradigmi, che non sono altro che ampie descrizioni delle modalità di funzionamento della natura. Quindi

il "paradigma scientifico" secondo Kuhn è un insieme di assunzioni teoriche fondamentali che tutti i membri della comunità scientifica accettano in un dato momento, ma è qualcosa di più di una mera teoria: quando condividono un paradigma gli scienziati concordano sul modo di fare ricerca nel futuro nel loro campo, su quali siano i problemi principali da indagare, su quali siano metodi appropriati per risolverli. Il paradigma riassume un'intera prospettiva scientifica.

La scienza normale si effettua raccogliendo dati entro il contesto normale del paradigma esistente. *La scienza normale non si confronta con i paradigmi esistenti, ma piuttosto li "abbellisce".* Il paradigma detta quale tipo di esperimenti condurre e quali dati raccogliere e come interpretare i dati. Il periodo di scienza normale dura decenni e a volte secoli. Nella visione di Kuhn il cambiamento reale avviene quando:

1. si accumula una grande massa di dati che i paradigmi esistenti non possono spiegare;

2. si scopre un paradigma alternativo che può spiegare le discrepanze fra i vecchi paradigmi e le osservazioni.

Kuhn sostiene che gli esperimenti sono raramente (quasi mai) decisivi al livello dei paradigmi in uso e qualora vengano trovate discrepanze fra osservazioni e paradigmi, queste vengono interpretate come problema o incertezza nelle misurazioni. Solo l'accumularsi di dati contraddittori dei paradigmi conduce alle rivoluzioni scientifiche. Ne sono esempi di queste ultime: la rivoluzione copernicana rispetto alla visione tolemaica del movimento dei pianeti, la transizione fra la fisica newtoniana e quella einsteiniana.

3.1.1 Domande per un breve dibattito sul ruolo della scienza

1. Come funziona la "scienza reale" di tutti i giorni?

2. Come funziona la scienza ecologia?

3. Mi posso aspettare una "rivoluzione scientifica" nella mia tesi di laurea?

4. Qual è il ruolo delle riviste scientifiche?

5. Che relazione c'è fra pubblicazioni e carriera scientifica?

6. Qual è il ruolo della scienza nella società?

7. Qual è il ruolo di un biologo? E di un ecologo? E di un ecologista?

8. Che tipo di rapporto c'è fra i media e la scienza?
9. Che ruolo hanno le politiche di finanziamento della ricerca?

3.2 Esempio pratico di scelta fra due o più ipotesi

Facciamo un'ipotesi:

Ipotesi: Uccelli in stormi numerosi sono più efficaci nel procurarsi il cibo rispetto agli uccelli in stormi piccoli

Ipotesi nulla: Non c'è relazione fra la dimensione dello stormo e l'efficienza nel procurarsi il cibo

Indichiamo con S la dimensione dello stormo, misurata come numero medio di storni appartenenti allo stormo nell'arco del periodo di studio, e indichiamo con C il numero medio di semi di granoturco mangiati da ciascun individuo nel periodo di studio.

Il modello "concettuale" esposto nell'ipotesi si può così tradurre in un linguaggio un po' più simbolico:

$$C \text{ cresce al crescere di } S;$$

Il ruolo dello scienziato ecologo è spesso quello di entrare sempre più nel dettaglio dell'ipotesi, di cercare modi per invalidare l'ipotesi o fare un'ipotesi più *avanzata*. Entriamo nel dettaglio del il modello concettuale e a tradurlo in un modello "quasi matematico". Analizziamo cioè cosa significa "C cresce al crescere di S". Subito possiamo pensare a vari modi (o modelli) con cui "C cresce al crescere di S"
Per esempio possiamo enunciare alcune ipotesi più specifiche:

1. C cresce *linearmente* e indefinitamente al crescere di S. (Vi viene in mente qualche conseguenza?)

2. C cresce al crescere di S ma fino ad un certo livello, poi C non cresce più;

3. C cresce al crescere di S ma fino ad un certo livello, poi C diminuisce al crescere di S;

Traducendo in linguaggio matematico i modelli sopra esposti possono avere la seguente forma:

Modello 1:

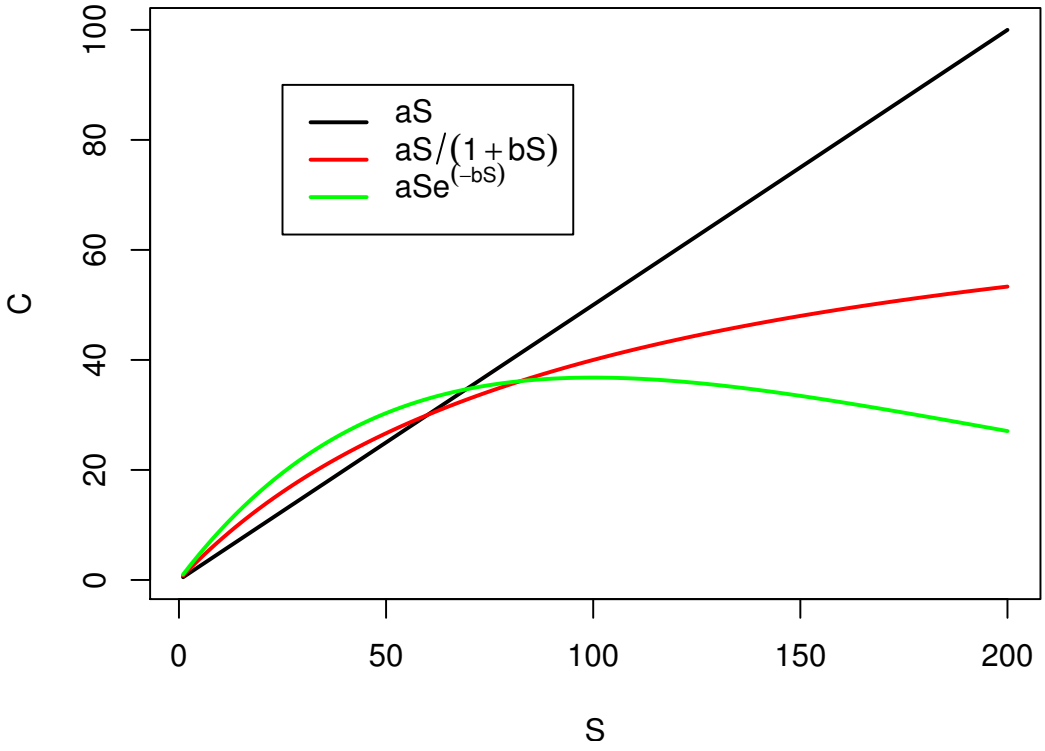
$$C = aS$$

Modello 2:

$$C = \frac{aS}{1 + bS}$$

Modello 3:

$$C = aSe^{-bS}$$



3.3 Script in R per testare questo esempio

Vi verranno forniti dei dati di S e C che voi dovrete interpretare e valutare per decidere quale dei 3 modelli esposti precedentemente "spiega" meglio i dati e quale/i modello/i viene/vengono rigettato/i. Dovrete fornire una breve spiegazione del perchè (Max 10 righe).

Uno dei modi per svolgere questo compito è eseguire un fitting non-lineare sui dati con ciascuno dei 3 modelli e vedere per quale modello c'è meno differenza fra valore predetto di C e valore osservato.

Eseguire tutti i comandi riportati qui sotto. (N.B. Le righe che iniziano con "#" sono dei commenti per voi e non vengono interpretati dal programma).

```
#Programma in linguaggio R per fitting non lineare
#sono testati 3 modelli per studiare la relazione
#flock Size vs. foraggiamento


#Lettura dati
#sostituire C6.txt con il proprio file di dati
d1.df <- read.table("C6.txt", header=T)
#memorizzo i dati in un "data frame" chiamato d1.df

#vedo i dati
print(d1.df)

#stampo un breve sommario di d1.df
summary(d1.df)

#attacco il data frame d1.df
attach(d1.df)

#provo il fitting del primo modello
# C = a * S
#per iniziare setto il parametro a=1
modello1.nls <- nls( C ~ a * S, start=list(a=1), trace=T)

#stampo un breve sommario del fitting
summary(modello1.nls)


#provo ora il fitting del secondo modello
# C = a * S/(1 + b * S)
#per iniziare setto i parametro a=1 e b=1
#se non converge provare a settare b=0.1 o a=0.1 o b=0.01 o a=10
modello2.nls <- nls( C ~ a * S/(1 + b * S), start=list(a=1,b=1), trace=T)

summary(modello2.nls)
#vedo se il modello 2 diminuisce significativamente l'errore rispetto
```

```
#al modello 1
anova(modello1.nls,modello2.nls)


#provo ora il fitting del terzo modello
# C = a * S * e^(-b * S)
#per iniziare setto il parametro a=1 e b=0.1
#se non converge provare a settare b=0.01 o a=0.1 o b=1 o a=10...
modello3.nls <- nls( C ~ a * S * exp(-b * S), start=list(a=1,b=0.1), trace=T)

summary(modello3.nls)

#faccio un grafico di S vs C con simbolo 16(cerchio pieno)
plot(S,C, pch=16)
#aggiungo una linea che unisce i punti predetti dal modello1
lines(S,predict(modello1.nls))
#aggiungo una linea rossa che unisce i punti predetti dal modello2
lines(S,predict(modello2.nls), col="red")
#aggiungo una linea verde che unisce i punti predetti dal modello2
lines(S,predict(modello3.nls), col="green")

#Confront l'AIC dei tre modelli
AIC(mdello1.nls)
AIC(mdello1.nls)
AIC(mdello1.nls)
#la preferenza va al modello con l'AIC più basso
```

In sostanza dall'output di R abbiamo ben 5 criteri per valutare la scelta del modello "migliore" fra i tre che abbiamo esaminato. Ribadisco che i dati sono sempre gli stessi, cambia il modello applicato e si cerca il modello che è "in qualche modo più aderente" ai dati.

Il principio generale che guida la scelta dei modelli è quello di scegliere il modello più semplice possibile che dia approssimativamente la stessa capacità predittiva o esplicativa di modelli complicati.

I cinque criteri sono:

1. Scegliere il modello che produce **l'errore minore**. L'errore è definito come la somma dei quadrati degli scarti fra la y osservata e la $\hat{y}$ predetta dal modello (in base alla x). A volte però le differenze nell'errore

possono essere minime e il criterio potrebbe favorire modelli intutilmente complicati solo perché producono un errore leggermente minore di modelli più semplici.

2. **Se**, soltanto se, **due modelli sono *nested***, si può utilizzare l'**anova** per testare se il modello più complicato permette una diminuzione dell'errore significativa. Due modelli sono *nested* quando un modello è un caso particolare dell'altro, cioè fissando un parametro ad un determinato valore per il primo modello, il modello si semplifica e diventa uguale al secondo. In questo caso si dice che il secondo modello è *nested* nel primo. Nel nostro caso il primo modello è *nested* dentro il secondo e anche dentro al terzo. Infatti ponendo $b = 0$ sia il secondo, sia il terzo modello diventano uguali al primo. Il secondo e il terzo modello non sono *nested* fra loro e hanno anche gli stessi gradi di libertà. I modelli *nested* hanno sempre gradi di libertà diversi.

3. Valutare la possibilità che uno o più **parametri** nei vari modelli possano **non essere diversi da zero**. Questo test si effettua con il `summary`. Se uno dei parametri non risultasse significativamente diverso da zero, avremmo indicazioni che il modello potrebbe essere inutilmente complicato.

4. Valutare l'**AIC**. Semplicemente si sceglie il modello che ha l'AIC (Akaike Information Criteria) più basso. L'AIC permette di scegliere il modello migliore anche nei casi di modelli *non nested*. L'AIC valuta la *likelihood* o verosimiglianza del modello penalizzandola per il numero di parametri usati (modelli più complicati hanno mediamente più parametri da stimare rispetto a modelli più semplici). La *likelihood* è semplicemente la probabilità di trovare i dati che abbiamo raccolto, in base al modello statistico usato. Spesso, ma non sempre, si usa la probabilità data dalla curva normale di ottenere l'errore (predetto - osservato) di ogni misura. La *likelihood* è la produttoria delle singole probabilità, ma più spesso si usa la *Log-likelihood* che è la somma dei logaritmi delle singole probabilità di ogni dato/errore. In R la *Log-likelihood* si ottiene con la funzione `LogLik`. L'AIC è semplicemente calcolato in base alla seguente formula:

$$AIC = -2\ LogLik(modello) - 2\ k$$

dove k è il numero di parametri stimati. Con l'AIC si preferisce un modello "A" con una *LogLikelihood* leggermente più bassa di un secondo modello "B", semplicemente perche il modello "A" ha un numero di

parametri minore e quindi una penalità più bassa. Il BIC è un criterio simile ma assegna una penalità ancora maggiore ai modelli con più parametri.

5. **Valutare graficamente** il modello. Forse questo è il criterio più importante di cui bisogna sempre tenere conto. Si guarda il grafico e si valuta se il comporamento dei modelli (linee) è adeguato ai ai dati (punti) e si confrontano i modelli diversi, avendo sempre in mente che, a parità di capacità predittive, è sempre preferibile un modello più semplice.

Non sempre i vari criteri danno risposte coincidenti. Spetta quindi al ricercatore la valutazione definitiva, ma il criterio usato dovrebbe essere sempre trasparente e plausibile alla luce del processo ecologico che ha generato i dati.

3.4 L'errore standard

L'errore standard o deviazione standard della media ($\sigma_{\overline{y}}$) misura la variabilità della media campionaria ($\overline{y}$) attorno alla media della popolazione (μ) e si stima anche da una sola media media campionaria. È una misura della variabilità che si avrebbe facendo ripetuti campionamenti della popolazione e calcolando la media ogni volta.

$$Var(\overline{y}) = \frac{\sigma^2}{n}$$

$$\sigma_{\overline{y}} = \frac{\sigma}{\sqrt{n}}$$

Ma queste due formule da dove derivano? Per la loro dimostrazione si sfrutta i fatti a noi già noti (lo abbiamo dimostrato empiricamente in uno dei primi compiti assegnati)

$$\mathrm{Var}(x + y) = \mathrm{Var}(x) + \mathrm{Var}(y) + \mathrm{Cov}(x, y)$$

e per una k costante:

$$\mathrm{Var}(ky) = k^2 \,\mathrm{Var}(y)$$

Si assume inoltre che le singole osservazioni siano indipendenti una dall'altra e che quindi $\mathrm{Cov}(x, y) = 0$ e che le osservazioni siano identicamente distribuite e che tutte con varianza σ^2

$$Var(\overline{y}) = \text{Var}(\frac{1}{n}\sum y_i) = \frac{1}{n^2}\sum \text{Var}(y_i) = \frac{1}{n^2} \times \sum \sigma^2 = \frac{n}{n^2}\sigma^2 = \frac{\sigma^2}{n}$$

Un'analoga dimostrazione sfrutta la definizione di Var($\overline{y}$) che viene definita come il valore atteso ("$E()$" o valore medio) dello scarto della media campionaria $\overline{y}$ dalla media della popolazione μ:

$$var(\overline{y}) = E\{(\overline{y} - \mu)^2\} =$$
$$E\left(\frac{y_1 + y_2 + \cdots + y_n}{n} - \mu\right)^2 =$$
$$E\left[\frac{(y_1 - \mu) + (y_2 - \mu) + \cdots + (y_n - \mu)}{n}\right]^2 =$$
$$\frac{1}{n^2}\left\{E\left[(y_1 - \mu)^2 + (y_2 - \mu)^2 + \cdots + (y_n - \mu)^2\right] + \right.$$
$$\left. +2\,E\left[(y_1 - \mu)(y_2 - \mu)\right] + \cdots + 2\,E\left[(y_{n-1} - \mu)(y_n - \mu)\right]\right\}$$

e poiché

$$E\left[(y_i - \mu)^2\right] = \sigma^2$$

e se y_i e y_j sono indipendenti

$$E\left[(y_i - \mu)(y_j - \mu)\right] = 0$$

allora:

$$\text{Var}(\overline{y}) = \frac{1}{n^2}\, n\, \sigma^2 = \frac{\sigma^2}{n}$$

3.5 L'Analisi della Varianza

Siete invitati a riguardarvi sul libro di statistica i capitoli riguardanti l'analisi della varianza. In particolare siete tenuti a conoscere le assunzioni dell'analisi della varianza.

L'analisi della varianza è uno dei più importanti modelli per l'analisi dei dati. Viene usato in tutte le scienze e anche in Ecologia per la sua potenza e versatilità. Nonostante i suoi concetti di base siano piuttosto semplici e comprensibili, l'analisi può rilevarsi talvolta apparentemente complicata.

Rivediamo ora alcuni dei concetti di base. In generale la prima **ipotesi nulla** che si va a verificare con l'analisi della varianza è che le medie di a gruppi campionati siano non significativamente diverse:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_a \tag{3.1}$$

L'**analisi della varianza** si basa su modelli lineari molto semplici e ciascuna osservazione j-esima appartenente al gruppo i-esimo può essere definita come:

$$Y_{ij} = \mu_i + \epsilon_{ij} \tag{3.2}$$

con $i = 1, ...a$ gruppi o trattamenti e $j = 1, ...n_j$ dove μ_j è la media del trattamento/gruppo $j-$esimo e ϵ_{ij} è il residuo o errore sperimentale. Il modello precedente può essere espresso introducendo il parametro A_i cioè lo scostamento della media $i-$esima dalla media generale della popolazione (μ)

$$A_i = \mu_i - \mu$$

Il modello generale diviene:

$$y_{ij} = \mu + A_i + \epsilon_{ij}$$

dove A_i viene definito l'*effetto* dell'$i-$esimo trattamento.

Come è noto l'analisi della varianza assume che:

$$\epsilon_{ij} \sim \quad \text{i.i.d.} \quad N(0, \sigma^2)$$

che significa che l'errore è indipendente e identicamente distribuito e che è distribuito secondo una distribuzione normale con media 0 e varianza σ^2.

Quale sarà, in termini dell'equazione 3.1 l'ipotesi nulla?

3.5.1 La scomposizione della varianza

Lo scostamento di ciascun dato dalla media generale del campione ($\overline{y}$) cui appartiene viene scomposto in due termini:

- scostamento dalla media del gruppo a cui appartiene
- scostamento della media del gruppo dalla media generale

$$y_{ij} - \overline{y} = y_{ij} - \overline{y}_i + \overline{y}_i - \overline{y}$$
$$y_{ij} - \overline{y} = (y_{ij} - \overline{y}_i) + (\overline{y}_i - \overline{y})$$

e, come abbiamo visto, $\overline{y}_i - \overline{y} = A_i$.

Al fine di ottenere la **somma degli scarti quadratici** si eleva al quadrato e si somma su tutti i a gruppi o trattamenti e all'interno di ogni gruppo si somma sugli n_j dati di quel gruppo:

$$\sum_{i=1}^{a} \sum_{j=1}^{n_j} [(y_{ij} - \overline{y}_i) + (\overline{y}_i - \overline{y})]^2$$

$$= \sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij} - \overline{y}_i)^2 + \sum_{i=1}^{a}\sum_{j=1}^{n_j}(\overline{y}_i - \overline{y})^2 +$$

$$+2 \sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij} - \overline{y}_i)(\overline{y}_i - \overline{y})$$

Questo non è che la semplice applicazione di un quadrato di un polinomio con l'ultimo termine uguale al doppio prodotto. Questo ultimo termine però si annulla in quanto la somma degli scarti (non quadratici) da un media equivale a zero.

Per cui la **somma degli scarti al quadrato totale** (SST) è data da:

$$SST = \sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij} - \overline{y}_i)^2 + \sum_{i=1}^{a}\sum_{j=1}^{n_j}(\overline{y}_i - \overline{y})^2$$

La somma dei quadrati totale è stata scomposta in due termini:

- il primo termine misura la variabilità **entro** i gruppi (SSE) in quanto è dato dagli scarti delle singole osservazioni dalla media di ogni gruppo,
- il secondo termine misura la variabilità **tra** delle medie dei gruppi attorno alla media generale (SSA).

Da cui possiamo ricavare:

$$SST = SSE + SSA \tag{3.3}$$

e possiamo, per esempio ricavare la somma dei quadrati entro come differenza fra le altre due somme dei quadrati.

$$SSE = SST - SSA$$

Se l'ipotesi nulla fosse vera e non vi fosse errore campionario quest'ultimo termine sarebbe zero. Purtroppo quest'ultima ipotesi non è mai vera nella realtà.

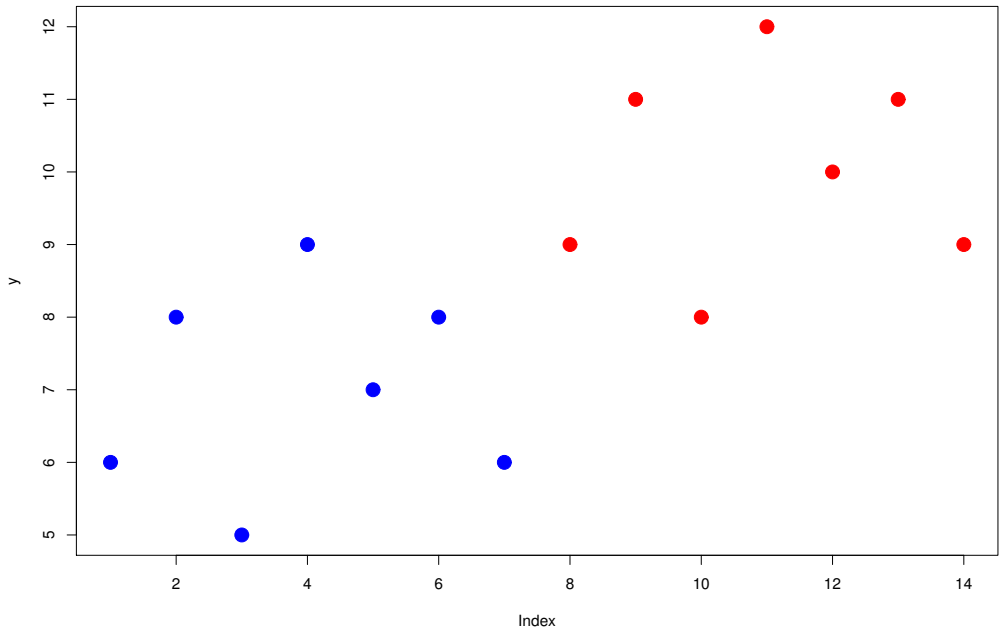


Figura 3.1: I dati sono divisi in due gruppi: rosso e blu. Sull'asse delle ascisse c'è solamente un indice per non "ammucchiare" i dati.

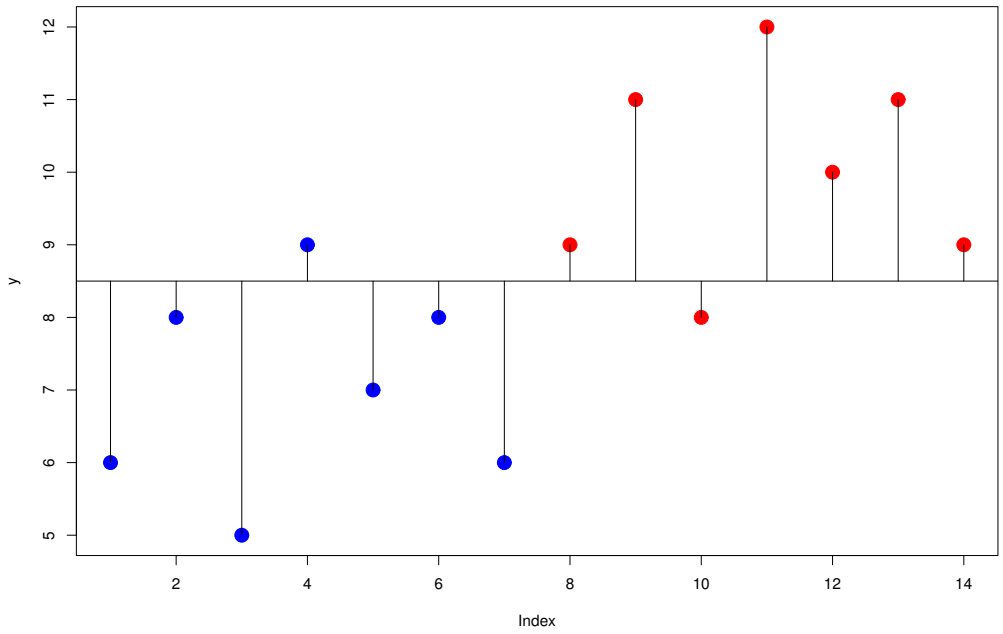


Figura 3.2: I segmenti verticali neri indicano gli scarti dalla media generale $(y_i - \overline{y})$ per dare SST.

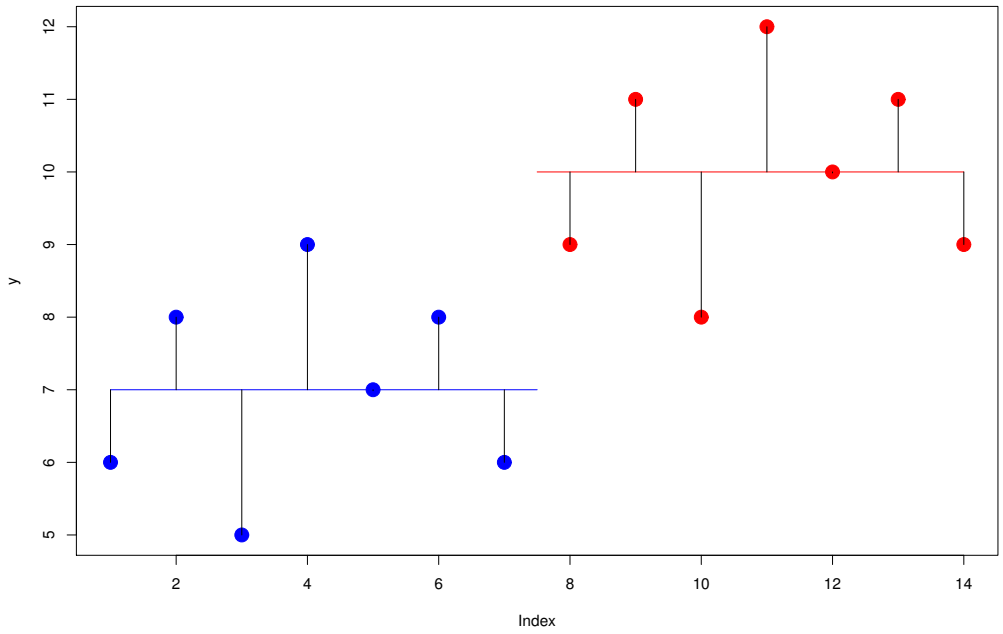


Figura 3.3: I segmenti verticali indicano gli scarti dalla media dei gruppi $(y_i - \overline{y}_i)$ per dare SSE.

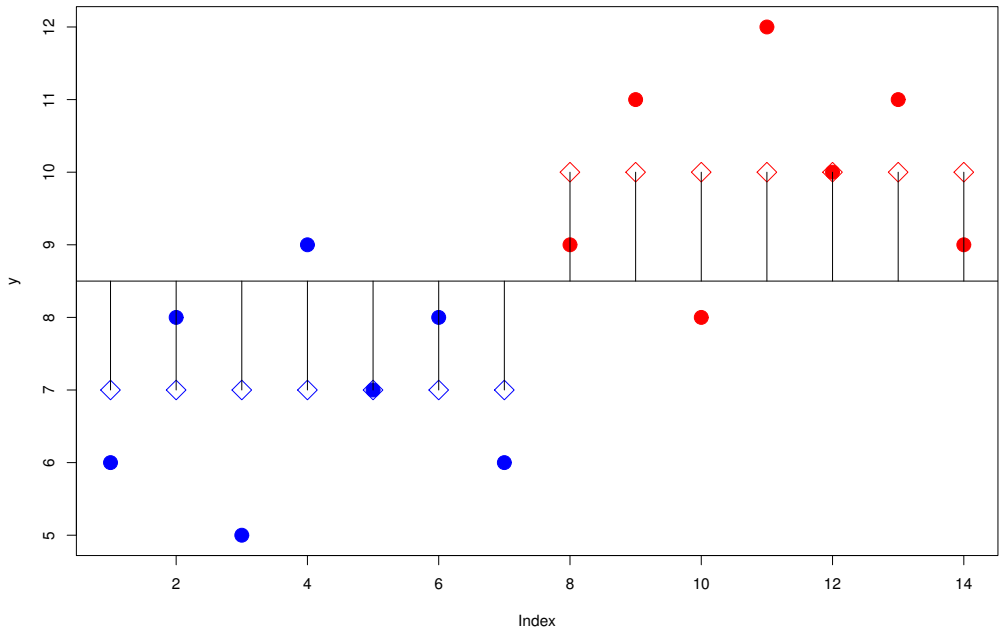


Figura 3.4: I segmenti verticali indicano gli scarti della media del gruppo dalla media generale $(\overline{y}_i - \overline{y})$ per dare SSA.

Per capire intuitivamente come questa scomposizione della varianza può essere utilizzata come test concentratevi su SST e SSE.

Se le *medie* dei due gruppi *fossero uguali*, allora anche SST e SSE sarebbero uguali, perchè le linee orrizontali della figura 3.3 coinciderebbero con la linea orrizontale della figura 3.2.

Se le *medie* dei due gruppi *fossero diverse*, allora SSE sarebbe piú piccola di SST. I segmenti verticali sarebbero piú corti nella figura 3.3 rispetto alla 3.2. In teoria SSE potrebbe addirittura annullarsi se le repliche all'interno di ciascun gruppo fossero identiche, ma questo, come abbiamo detto sopra, non è mai vero nella realtà.

Il comportamento di SSA è ovviamente una conseguenza del comportamento di SST e SSE vista l'equazione 3.3: SSA è tanto piú piccola quanto piú SSE è vicina a SST.

Dovrebbe a questo punto risultarvi abbastanza intuitivo come si possano usare i rapporti fra le varianze per testare delle ipotesi sulle medie.

Un passaggio preliminare è quello di passare dalle devianze (Somma dei quadrati o SS) alle varianze (**Quadrati Medi** o MS) dividendo le devianze per i rispettivi **gradi di liberta**.

In effetti l'analisi della varianza si basa su un principio molto semplice: **si stima la varianza della popolazione in due modi e poi si fa il rapporto fra le due stime. Se il rapporto non è significativamente diverso da 1 allora è vera l'ipotesi nulla, altrimenti è vera l'ipotesi alternativa**.

Intuitivamente possiamo pensare al caso dell'ipotesi nulla in questi termini: se partendo da un'unica popolazione facessi dei gruppi (due o più) a caso (attribuendo il fattore casualmente, proprio come nel caso del test di permutazione) otterrei delle medie leggermente diverse, ma tutto sommato abbastanza simili tra loro. Come abbiamo visto per il teorema centrale del limite, il grado di similitudine delle medie dipende dalla variabilità in origine dei dati e dalle dimensioni dei campioni. Però, sempre nel caso dell'ipotesi nulla di un'unica popolazione, la variabilità (*varianza*) delle medie (attorno alla media generale o *tra*) sarebbe stimabile a partire da quella prevista dalla variabilità della popolazione (*varianza entro*).

Al contrario, se le due stime di varianza non fossero simili, significherebbe che il criterio applicato per formare i gruppi (fattore *non* attribuito a caso) sarebbe la causa della varianza fra gruppi sia più alta dell'atteso. In questo secondo caso quindi non sarei di fronte ad un'unica popolazione, ma due (o più) popolazioni diverse.

Riepilogando possiamo dire che l'analisi della varianza si basa sul rapporto fra due modi di stimare la varianza della popolazione. Se le due stime sono

simili allora ne deduco che sono di fronte ad una sola popolazione. Se le due stime divergono deduco che ho più di una popolazione.

Le due stime sono fatte:

- in base alla variabilità interna ad ogni gruppo (*entro*)
- in base alla variabilità tra le medie dei gruppi (*tra*)

La **prima stima** della varianza della popolazione si fa in base ad una stima "cumulata" o "ponderata" basata sugli scarti dalla media di ciascun gruppo, sommando però per tutti i gruppi. Potrete intuitivamente capire che avendo a gruppi potrei avere a stime della varianza della popolazione, ma ciascuna sarebbe imprecisa perchè basata su un campione piccolo. Si ovvia facendo un'unica stima su tutti i campioni rispetto alla loro media. Per semplicità affrontiamo il caso in cui i campioni sono tutti equamente numerosi (tutti gli n_j sono uguali a n) per cui $N = a\ n$ dove N è il numero totale dei dati misurati.

$$MSE = \frac{SSE}{a\ (n-1)}$$

$$MSE = \frac{\sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij} - \overline{y}_i)^2}{a\ (n-1)}$$

Per la **seconda stima** si sfrutta la relazione fra la varianza della popolazione σ^2 e la varianza delle medie attorno alla media generale ${\sigma_{\overline{y}}}^2$ (quadrato dell'errore standard) riportata all'inzio:

$$var(\overline{y}) = \frac{\sigma^2}{n}$$

da cui

$$\sigma^2 = n\ var(\overline{y}).$$

Inoltre ricordo che la definizione di $var(\overline{y})$ era:

$$var(\overline{y}) = E\{(\overline{y} - \mu)^2\}$$

che nel nostro caso stimiamo con

$$var(\overline{y}) = \frac{\sum_{i=1}^{a}(\overline{y}_i - \overline{y})^2}{a-1}$$

per cui, partendo dalla somma dei quadrati tra gruppi (SSA) dividiamo per i gradi di libertà pari al numero dei gruppi (a) - 1 :

$$MSA = \frac{SSA}{a-1}$$

$$MSA = \frac{\sum_{i=1}^{a}\sum_{j=1}^{n}(\overline{y}_i - \overline{y})^2}{a-1}$$

che, nel caso che tutti gli n_j siano uguali a n, non è altro che:

$$= \frac{n\ \sum_{i=1}^{a}(\overline{y}_i - \overline{y})^2}{a-1}$$

ma questa non è altro che la n volte la stima di $var(\overline{y})$ cioè σ, quindi possiamo concludere che anche MSA stima σ e possimo riscrivere

$$MSA = \frac{n}{a-1}\sum_{i=1}^{a}(\overline{y}_i - \overline{y})^2$$

Abbiamo quindi ottenuto due stime indipendenti di σ la varianza della della nostra popolazione, se il rapporto fra queste due stime

$$F = \frac{MSA}{MSE}$$

è piú grande in modo significavo di 1 allora si conclude che MSA contiene "qualcosa in più" della semplice varianza della popolazione e che quindi non stiamo parlando di un'unica popolazione ma abbiamo evidenze che esistano popolazioni con medie diverse.

Se, viceversa, il rapporto è non significativamente diverso da 1 allora abbiamo evidenze dell'esistenza di un unica popolazione con un'unica media.

Il rapporto fra le due stime (F) è un normale rapporto di varianze e si distribuisce, appunto, come una distribuzione F che è nota con gradi di libertà $a-1$ e $a(n-1)$ ovvero $a-1$ e $N-a$ nel caso generale.

I valori soglia di F possono essere calcolati in R con la funzione `qf()` (es: per un α del 5%: `qf(0.95, a-1, N-a)`) mentre la probabilità di osservare un dato valore di F si usa la la funzione 1-`pf()` (es: `1-pf(F,a-1,N-a)`.

Naturalmente devono essere verificate tutte le assunzioni dell'analisi della varianza, che voi ormai conoscete benissimo `:)`!

In una tipica tabella dell'analisi della varianza troviamo riassunto il tutto: appaiono più evidenti:

Sorgenti di Var.	**Somma dei quadrati**	**g.d.l.**	**Quadrati Medi**
Fra gruppi	$\sum_{i=1}^{a}\sum_{j=1}^{n_j}(\overline{y}_i-\overline{y})^2$	$a-1$	$\frac{\sum_{i=1}^{a}\sum_{j=1}^{n_j}(\overline{y}_i-\overline{y})^2}{a-1}$
Entro gruppi	$\sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij}-\overline{y}_i)^2$	$N-a$	$\frac{\sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij}-\overline{y}_i)^2}{N-a}$
Totale	$\sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij}-\overline{y})^2$	$N-1$	$\frac{\sum_{i=1}^{a}\sum_{j=1}^{n_j}(y_{ij}-\overline{y})^2}{N-1}$

Applichiamo l'analisi ad un caso concreto con i dati nel file `exp1.txt` e vediamo come calcolare queste quantità velocemente in R per poi confrontarle con il risultato della funzione "ufficiale" di R `anova` da applicarsi ad un oggetto restituito da `lm`.

```
   > d.df <- read.table("exp1.txt", h=T)
> summary(d.df)
```

Con i dati di `exp1.txt` la nostra y è il vettore `d.df$weight`. Le prime quantità da calcolare sono le medie: quella generale $\overline{y}$ e quelle dei vari gruppi $\overline{y}_i$

```
> d.df <- read.table("exp1.txt", h=T)
> summary(d.df)
```

```
     weight       group
 Min.   :3.590   Ctrl:10
 1st Qu.:4.570   Trt1: 8
 Median :5.170   Trt2: 9
 Mean   :5.108
 3rd Qu.:5.560
 Max.   :6.310
> ym <- mean(d.df$weight)
> ym
[1] 5.108148
> ymi <- tapply(d.df$weight, d.df$group, mean)
> ymi
    Ctrl     Trt1     Trt2
5.032000 4.700000 5.555556
```

`ymi` è un hash array per cui posso usare `d.df$group` come indice del vettore e produrre una matrice che mi permetta di capire come calcolare SSA, SSE e SST

```
> ymi[as.character(d.df$group)]
    Ctrl     Ctrl     Ctrl     Ctrl     Ctrl     Ctrl     Ctrl     Ctrl
5.032000 5.032000 5.032000 5.032000 5.032000 5.032000 5.032000 5.032000
    Ctrl     Ctrl     Trt1     Trt1     Trt1     Trt1     Trt1     Trt1
5.032000 5.032000 4.700000 4.700000 4.700000 4.700000 4.700000 4.700000
    Trt1     Trt1     Trt2     Trt2     Trt2     Trt2     Trt2     Trt2
4.700000 4.700000 5.555556 5.555556 5.555556 5.555556 5.555556 5.555556
    Trt2     Trt2     Trt2
5.555556 5.555556 5.555556

> cbind(d.df$weight,ymi[as.character(d.df$group)],ym)
                          ym
Ctrl 4.17 5.032000 5.108148
Ctrl 5.58 5.032000 5.108148
Ctrl 5.18 5.032000 5.108148
Ctrl 6.11 5.032000 5.108148
Ctrl 4.50 5.032000 5.108148
Ctrl 4.61 5.032000 5.108148
Ctrl 5.17 5.032000 5.108148
Ctrl 4.53 5.032000 5.108148
Ctrl 5.33 5.032000 5.108148
Ctrl 5.14 5.032000 5.108148
```

```
Trt1 4.81 4.700000 5.108148
Trt1 4.17 4.700000 5.108148
Trt1 4.41 4.700000 5.108148
Trt1 3.59 4.700000 5.108148
Trt1 5.87 4.700000 5.108148
Trt1 3.83 4.700000 5.108148
Trt1 6.03 4.700000 5.108148
Trt1 4.89 4.700000 5.108148
Trt2 6.31 5.555556 5.108148
Trt2 5.12 5.555556 5.108148
Trt2 5.54 5.555556 5.108148
Trt2 5.50 5.555556 5.108148
Trt2 5.37 5.555556 5.108148
Trt2 5.29 5.555556 5.108148
Trt2 4.92 5.555556 5.108148
Trt2 6.15 5.555556 5.108148
Trt2 5.80 5.555556 5.108148
```

La prima colonna corrisponde alla nostra y_{ij}, la seconda alle medie dei vari gruppi $\overline{y}_i$ e la terza alla media generale $\overline{y}$ per cui posso calcolarmi le quantità di cui ho bisogno sommando i vari scarti al quadrato.

```
> SSE <- sum((d.df$weight - ymi[as.character(d.df$group)])^2)
> SSA <- sum((ymi[as.character(d.df$group)]-ym)^2)
> SST <- sum((d.df$weight-ym)^2)
```

verifico anche che $SSA + SSE = SST$

```
> SSA+SSE
[1] 13.47641
> SST
[1] 13.47641
```

calcolo i gradi di libertà e i quadrati medi:

```
> a <-length(ymi)
> a
[1] 3
> N <- length(d.df$weight)
> N
```

```
[1] 27
> ni <- tapply(d.df$weight, d.df$group, length)
> ni
Ctrl Trt1 Trt2
  10    8    9
> MSA <- SSA/(a-1)
> MSE <- SSE/(N-a)
> MSA
[1] 1.596113
> MSE
[1] 0.4285076
```

Infine calcolo il rapporto F con MSE sempre al denominatore e la probabilità di ottenerlo sotto ipotesi nulla:

```
> F <- MSA/MSE
> F
[1] 3.724818
> 1-pf(F, a - 1, N - a)
[1] 0.03900745
```

Ora controllo che i conti siano corretti confrontandoli con quelli dell'`anova` di R .

```
> m.lm <- lm(weight ~ group, data=d.df)
> anova(m.lm)
Analysis of Variance Table

Response: weight
          Df  Sum Sq Mean Sq F value  Pr(>F)
group      2  3.1922 1.59611  3.7248 0.03901 *
Residuals 24 10.2842 0.42851
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Da questi risultati è chiaro che, adottando il limite del 5%, possiamo smentire l'ipotesi nulla che le medie dei gruppi siano tutte uguali, ma non sappiamo niente su quale o quali gruppi abbiamo medie diverse dagli altri.

Una risposta, anche se non completa, si ha dall'analisi dei **contrasti**. In questa fase ci limitiamo a spiegare i contrasti di default di R , che però, volendo, si possono cambiare. I contrasti di default di R sono chiamati `contr.treatment` e assumono come riferimento, non la media generale come

nell'esposzione precedente, ma il primo livello in ordine alfabetico del fattore in esame. In questo caso viene preso come riferimento il livello `Ctrl`. Nel `summary` la riga con `Intercept` riporta il valore stimato della medie di questo livello. Nelle righe successive sono riportate le differenze degli altri due gruppi rispetto alla media del gruppo di riferimento `Ctrl`. É importante notare che viene testato se queste differenze sono diverse da zero, che ha il significato ovvio di testare se i due gruppi `Trt1` e, separatamente, `Trt2` hanno medie diverse dal gruppo `Ctrl`.

```
> summary(m.lm)

Call:
lm(formula = weight ~ group, data = d.df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.11000 -0.46878 -0.01556  0.27122  1.33000

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.0320     0.2070  24.309   <2e-16 ***
groupTrt1    -0.3320     0.3105  -1.069   0.2956
groupTrt2     0.5236     0.3008   1.741   0.0945 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6546 on 24 degrees of freedom
Multiple R-squared:  0.2369,Adjusted R-squared:  0.1733
F-statistic: 3.725 on 2 and 24 DF,  p-value: 0.03901
```

Da questa tabella si può dedurre che sia la media di `Trt1`, sia quella di `Trt2` non differiscono significativamente dal controllo (`Ctrl`). In questo caso manca il confronto tra il `Trt1` e `Trt2` che, nel caso in esame, è probabilmente la differenza tra le medie che determina la significatività dell'ANOVA generale.

Per capire come avere gli stessi valori del `summary` è sufficiente calcolare le differenze delle medie dei gruppi rispetto al gruppo di riferimento

```
> ##per avere le estimate del summary
> ## usando il primo livello in ordine alfabetico come riferimento
> myestimates <- ymi - ymi[1]
> myestimates
```

```
      Ctrl       Trt1       Trt2
 0.0000000 -0.3320000  0.5235556
```

e rimpiazzare il primo valore di 0 con la media del primo gruppo in ordine alfabetico.

```
> myestimates[1] <- ymi[1]
> myestimates
      Ctrl       Trt1       Trt2
 5.0320000 -0.3320000  0.5235556
```

ed ecco che le otteniamo le stesse stime di `summary` di `lm`

```
> m.lm

Call:
lm(formula = weight ~ group, data = d.df)

Coefficients:
(Intercept)    groupTrt1    groupTrt2
     5.0320      -0.3320       0.5236
```

Ricordate che l'analisi dei contrasti ha senso ed è legittima solo se l'ANOVA generale è significativa. Se non lo fosse, sarebbe scorretto procedere all'analisi dei contrasti. L'analisi dei contrasti di default di R non fa tutti i possibili confronti, ma confronta solo tutti i livelli con il primo in ordine alfabetico.

Un metodo per analizzare le differenze tra tutte le possibili coppie di livelli è il *Tukey's Honest Significant Difference method*. Il comando per avere subito un risultato è il seguente:

```
> plot(TukeyHSD(aov(weight ~ group, data = d.df)))
```

Il metodo di Tukey ha il pregio di tenere conto del fatto che i confronti non sono tra loro indipendenti. Nel nostro caso, visto che sappiamo che le medie sono differenti tra loro, ma nè la seconda, nè la terza media sono significativamente diverse dalla prima, deve per forza esistere una differenza significativa tra la seconda e la terza media.

Dati appaiati

Affrontiamo il caso concreto in cui uno studente della Magistrale, nell'ambito del suo lavoro di tesi, voglia analizzare gli effetti dello scarico dei depuratori sulla diversità bentonica dei fiumi. Voi come disegnereste l'esperimento?

Uno dei disegni più ovvi è quello chiamato *BACI - Before-After Contol Impact* in cui vengono confrontati di dati prima e dopo la costruzione di un certo impianto. Spesso i dati prima della costruzione non sono dispnibili per cui si usano i dati "a monte" dello scarico come controllo.

Quindi lo studente analizzerà la diversità bentonica a `valle` e a `monte` dello scarico, utilizzando i dati raccolti in più transetti o plots. L'analisi statistica consisterà nel confrontare la diversità fra i plot a valle e a monte. Se la differenza risultasse significativa si potrebbe inferire che lo scarico ha un effetto sulla diversità o meglio si rifiuterebbe l'ipotesi nulla che lo scarico non ha nessun effetto.

Se decideste di analizzare un solo fiume con un solo scarico, anche aumentando il numero dei plots o dei transetti andreste incontro ad un problema, molto controverso e molto discusso, noto in ecologia come *pseudo-replicazione*. In effetti studiando un solo fiume e un solo scarico anche intensamente (con tante repliche) non si potrebbe generalizzare agli scarichi. I dati raccolti sarebbero riferibili ad un solo scarico in un solo fiume. Se volessi generalizzare dovrei ripartire le repliche in più fiumi, allora potrei essere sicuro che il *pattern* trovato non è il risultato particolare riferibile solo ad un caso, ma estendibile a più fiumi.

Non sempre si riesce a replicare alla scala spaziale appropriata, ma naturalmente gli ecologi dovrebbero farlo sempre quando fosse possibile al fine di produrre dei risultati che abbiano una validità il più generale possibile.

Ora analizziamo il caso concreto tratto dai dati di Crawley di 9 diversi fiumi (`Location` in cui sono stati raccolti a monte e a valle degli scarichi.

```
> p.df <- read.table("paired.txt", h=T)
```

```
> p.df
  Location Upstream Downstream
1        A     5.68       5.36
2        B     5.60       5.41
3        C     5.43       5.39
4        D     5.90       5.85
5        E     5.94       5.82
6        F     5.70       5.73
7        G     5.66       5.58
8        H     5.98       5.89
9        I     5.85       5.78
```

riorganizziamo i dati in modo che si possa fare sia il `t.test` sia l'`anova`

```
> p2.df <-data.frame(
           Location=c(as.character(p.df$Location),as.character(p.df$Location)),
```

```
           UpDown=rep(c("Upstream", "Downstream"),c(9,9)),
           Diversity=c(p.df$Upstream,p.df$Downstream))
> p2.df
   Location     UpDown Diversity
1         A   Upstream      5.68
2         B   Upstream      5.60
3         C   Upstream      5.43
4         D   Upstream      5.90
5         E   Upstream      5.94
6         F   Upstream      5.70
7         G   Upstream      5.66
8         H   Upstream      5.98
9         I   Upstream      5.85
10        A Downstream      5.36
11        B Downstream      5.41
12        C Downstream      5.39
13        D Downstream      5.85
14        E Downstream      5.82
15        F Downstream      5.73
16        G Downstream      5.58
17        H Downstream      5.89
18        I Downstream      5.78
```

Il `t-test` ci dice che le diversità a monte e a valle non sono diverse

```
> t.test(Diversity ~ UpDown, data=p2.df,var.equal=T)

Two Sample t-test

data:  Diversity by UpDown
t = -1.1085, df = 16, p-value = 0.284
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.30095618  0.09428951
sample estimates:
mean in group Downstream   mean in group Upstream
                5.645556                 5.748889
```

anche l'`anova` conferma questo risultato come è ovvio in quanto nei casi di due soli gruppi $F = t^2$ e il P è identico nei due test.

```
> anova(lm(Diversity ~ UpDown, data=p2.df))
 Analysis of Variance Table

Response: Diversity
          Df  Sum Sq  Mean Sq F value Pr(>F)
UpDown     1 0.04805 0.048050  1.2287  0.284
Residuals 16 0.62571 0.039107
```

Come al solito la somma dei quadrati, i quadrati medi e F possono essere calcolati come segue:

```
> ##medie
> mgen <- mean(p2.df$Diversity)
> mupdo <- tapply(p2.df$Diversity, p2.df$UpDown, mean)
>
> ##gradi di libertà
> dfa <- length(mupdo) -1
> dfe <- length(p2.df$Diversity) -  dfa - 1
>
> ## somma dei quadrati
> SST <- sum((p2.df$Diversity - mgen)^2)
> SSE <- sum((p2.df$Diversity -
+                         mupdo[as.character(p2.df$UpDown)])^2)
> SSA <- sum((mupdo[as.character(p2.df$UpDown)] - mgen)^2)
>
> ##controllo che SSA + SSE sia uguale a SST
> SSA
[1] 0.04805
> SSE
[1] 0.6257111
> SSA+SSE
[1] 0.6737611
> SST
[1] 0.6737611
>
> ##quadrati medi
> MSA <- SSA/dfa
> MSE <- SSE/dfe
>
> MSA
[1] 0.04805
> MSE
```

```
[1] 0.03910694
>
> F <- MSA/MSE
> F
[1] 1.228682
>
> 1-pf(F, dfa, dfe)
[1] 0.2840482
```

Riguardando i dati però il risultato sembra "strano" in quanto in ben 8 casi su 9 la diversità *donwstream* è minore di quella *upstream*. Evidentemente non abbiamo sfruttato qualche informazione. In particolare non abbiamo sfruttato il fatto che i dati sono appaiati per `Location`. Il nostro risultato sarebbe stato identico se noi avessimo cambiato l'ordine delle misure entro ad ogni gruppo `Upstream` o `Downstream`.

Un modo semplice per sfruttare questa informazione è eseguire il `t.test` per dati appaiati (`paired=T`).

```
> t.test(Diversity ~ UpDown, data=p2.df, paired=T)

Paired t-test

data:  Diversity by UpDown
t = -3.077, df = 8, p-value = 0.01519
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.18077449 -0.02589217
sample estimates:
mean of the differences
             -0.1033333
```

Il risultato stavolta è significativo al livello del 5%.

Secondo l'ipotesi nulla mi attendo che la distribuzione delle differenze `Upstream-Downstream` sia centrata sullo zero. Un modo identico di eseguire il test precedente è vedere la probabilità che lo zero sia dentro l'intervallo di confidenza della media delle differenze fra le coppie e si esegue tramite un T-test per un campione:

```
## calcolo le differenze
> diffs <-p.df$Upstream - p.df$Downstream
## oppure
> diffs <- p2.df$Diversity[1:9] - p2.df$Diversity[10:18]
```

```
> t.test(diffs, mu=0)

One Sample t-test

data:  diffs
t = 3.077, df = 8, p-value = 0.01519
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.02589217 0.18077449
sample estimates:
mean of x
0.1033333
```

Quindi il test per dati appaiati (o quello sulle differenze) è un modo semplice per tenere conto delle informazioni raggruppate secondo un un fattore `UpDo`, ma anche del secondo fattore `Location` Ma come si esegue l'analisi della varianza con due fattori?

3.6 Analisi della Varianza a due fattori

Quando ci sono due fattori l'analisi si complica leggermente. Comunque anche con l'ANOVA si riesce e replicare l'analisi precedente tenendo conto di sia di `UpDo`, sia di `Location` Anzi l'ANOVA a due fattori è un test più generale in quanto può esser usato anche quando i livelli sono più di due. Il *t-test*, come è noto, si applica solo quando ci sono due livelli (nel nostro caso: `Upstream` e `Downstream`).

Alcuni conteggi sono del tutto identici a quelli precedenti però stavolta dobbiamo anche tener conto delle medie per `Location` e di SSA di `Location` e relativo MSA

```
> ##medie
> mgen  <- mean(p2.df$Diversity)
> mupdo <- tapply(p2.df$Diversity, p2.df$UpDown, mean)
> mloc  <- tapply(p2.df$Diversity, p2.df$Location, mean)
> mgen
[1] 5.697222
> mupdo
Downstream   Upstream
  5.645556   5.748889
> mloc
```

```
    A     B     C     D     E     F     G     H     I
5.520 5.505 5.410 5.875 5.880 5.715 5.620 5.935 5.815
>
>
> ##gradi di libertà
> dfa <- length(mupdo) -1
> dfb <- length(mloc) -1
> dfe2 <- length(p2.df$Diversity) -  dfa - dfb - 1
> dfa
[1] 1
> dfb
[1] 8
> dfe2
[1] 8
>
> ## somma dei quadrati
> SST  <- sum((p2.df$Diversity - mgen)^2)
> SSA  <- sum((mupdo[as.character(p2.df$UpDown)]   - mgen)^2)
> SSB  <- sum((mloc[as.character(p2.df$Location)] - mgen)^2)
>
> SSE2 <- SST - SSA -SSB
>
> SST
[1] 0.6737611
> SSA
[1] 0.04805
> SSB
[1] 0.5851111
> SSE2
[1] 0.0406
>
>
> ##y = m + A + B + e
> ##y = m + (ma - m) + (mb - m) +e
> ##y = ma + mb - m + e
> ##y - ma - mb + m = e
>
> ##calcolo SSE in un altro modo
> sum(( p2.df$Diversity -  mupdo[as.character(p2.df$UpDown)]-
+       mloc[as.character(p2.df$Location)] + mgen)^2)
[1] 0.0406
```

```
>
>
> ##quadrati medi
> MSA <- SSA/dfa
> MSB <- SSB/dfb
> MSE2 <- SSE2/dfe
>
> MSA
[1] 0.04805
> MSB
[1] 0.07313889
> MSE2
[1] 0.005075
>
>
> F1 <- MSA/MSE2
> F1
[1] 9.46798
> F2 <- MSB/MSE2
> F2
[1] 14.4116
>
> 1-pf(F1, dfa, dfe2)
[1] 0.01518564
> 1-pf(F2, dfb, dfe2)
[1] 0.0005289203
```

Questa analisi corrisponde alla seguente `anova` con due fattori

```
> anova(lm(Diversity ~ Location + UpDown, data=p2.df))
Analysis of Variance Table

Response: Diversity
          Df  Sum Sq  Mean Sq F value    Pr(>F)
Location   8 0.58511 0.073139  14.412 0.0005289 ***
UpDown     1 0.04805 0.048050   9.468 0.0151856 *
Residuals  8 0.04060 0.005075
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Il passaggio fondamentale è quello in cui si calcola la nuova seconda somma dei quadrati per l'errore ($SSE2$). E' più comodo calcolarla da SST togliendo sia SSA sia SSB

$$SSE2 = SST - SSA - SSB$$

Un modo analogo per calcolare $SSE2$ deriva dal fatto che se definisco l'effetto del gruppo i-esimo (`Upstream` o `Downstream`

$$A_i = \overline{y}_{i.} - \overline{\overline{y}}_{..}$$

mentre definisco l'effetto del gruppo j-esimo (`Location: A` o `B` o .. `I`)

$$B_j = \overline{y}_{.j} - \overline{\overline{y}}_{..}$$

per cui l'osservazione z-esima appartenente al gruppo i-esimo e alla località j-esima sarà data da:

$$\begin{aligned} y_{ijk} &= \overline{\overline{y}}_{..} + A_i + B_j + \epsilon_k \\ y_{ijk} &= \overline{\overline{y}}_{..} + (\overline{y}_{i.} - \overline{\overline{y}}_{..}) + (\overline{y}_{.j} - \overline{\overline{y}}_{..}) + \epsilon_k \\ y_{ijk} &= \overline{y}_{i.} + \overline{y}_{.j} - \overline{\overline{y}}_{..} + \epsilon_k \\ \epsilon_k &= y_{ijk} - \overline{y}_{i.} - \overline{y}_{.j} + \overline{\overline{y}}_{..} \end{aligned}$$

sommando per tutte le osservazioni ed elevando al quadrato:

$$SSE2 = \sum \left(y_{ijk} - \overline{y}_{i.} - \overline{y}_{.j} + \overline{\overline{y}}_{..} \right)^2$$

che equivale al comando `sum(( p2.df$Diversity - mupdo[as.character( p2.df$UpDown)] - mloc[as.character(p2.df$Location)] + mgen)^2)`

Comunque, di nuovo, il concetto importante è che, considerando anche la località, l'errore diminuisce e quindi anche l'effetto "monte-valle" diventa significativo perchè cala il denominatore. Il concetto è del tutto generale ed è quello che rende l'analisi della varianza un straordinario strumento per disegnare gli esperimenti perché, non solo permette di studiare l'effetto di più fattori contemporaneamente, ma il farlo rende più potente l'analisi dei singoli fattori.

Quindi in sostanza per diminuire la variabilità *entro* esistono due modi:

- **sperimentalmente** cioè adottando pratiche sperimentali che rendano più uniformi, meno variabili le osservazioni entro ciascun gruppo (o combinazione di gruppi)

- **statisticamente** in cui si aggiungono ulteriori fattori o covariate nel modello riducendo così l'errore *entro*.

In un certo senso c'è un prezzo da pagare per questa procedura: i gradi di libertà dell'errore diminuiscono, perchè sono utilizzati dai fattori/covariate aggiunti e quindi SSE viene diviso per un numero più piccolo per dare MSE. Quindi conviene inserire nel modello solo ulteriori fattori che siano realmente efficaci nel diminuire l'errore.

3.6.1 Esperimenti fattoriali

Ora complichiamo ulteriormente l'analisi. Immaginiamo di avere due fattori: il Fattore A con tre livelli e il Fattore B con due livelli. Per esempio potrebbero essere due genotipi coltivati a tre quote diverse. Inizialmente si fanno i conti calcolando separatamente le medie per i livelli del Fattore A e poi per i livelli del FattoreB e si calcola anche la media totale.

Per il calcolo dell'**interazione** occorre calcolare anche tutte le **medie per le combinazioni dei due fattori** (in nero nella figura successiva).

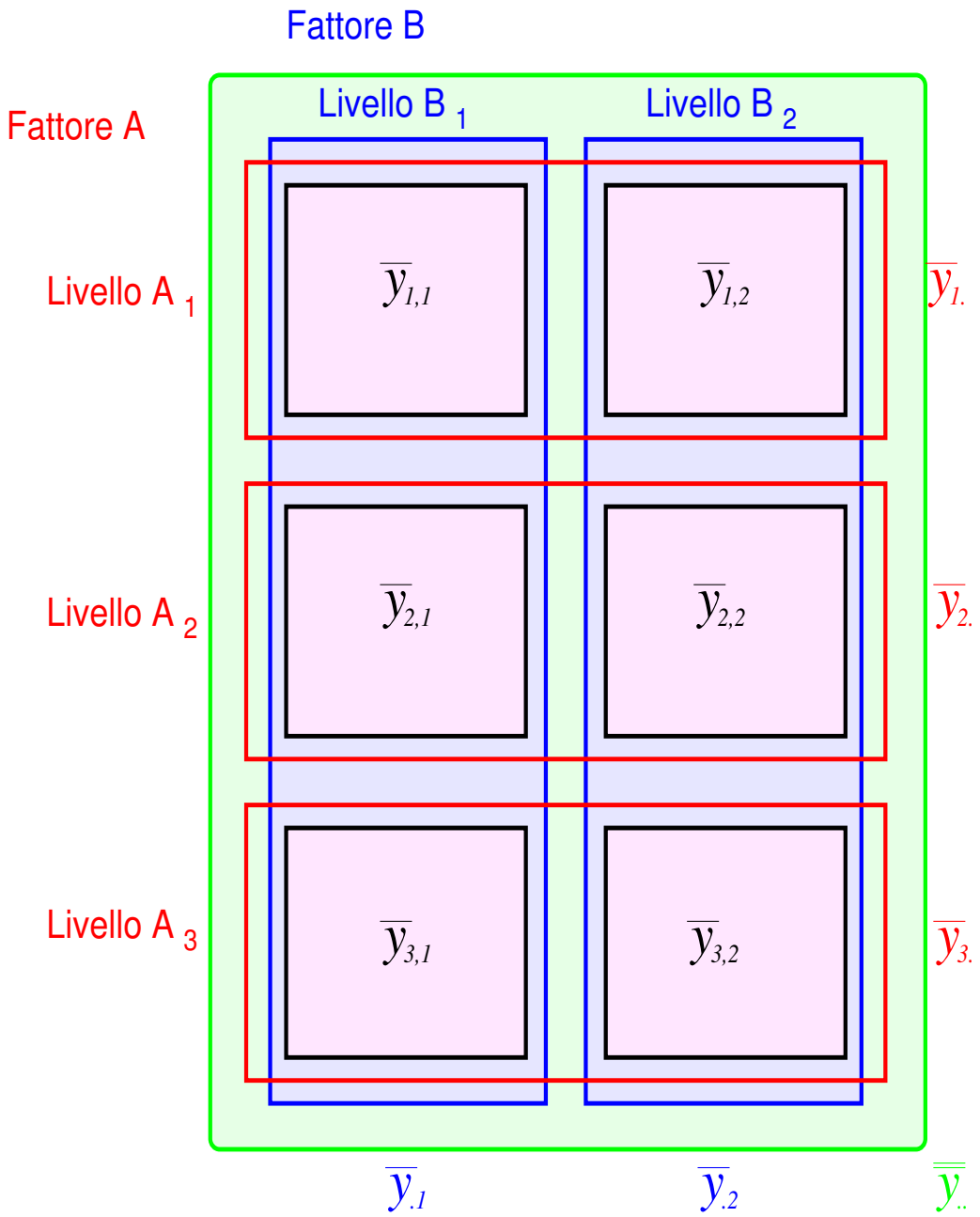


Figura 3.5: Illustrazione grafica del calcolo delle medie per un disegno sperimentale a due fattori

Dalle medie si passa poi al calcolo delle varianze intese proprio come deviazioni dalle varie medie appena calcolate:

Tabella 3.1: Tabella dell'analisi della varianza a due fattori senza interazione. Esistono a livelli (gruppi) per il Fattore A e b livelli (gruppi) per il fattore B e n osservazioni per combinazione dei fattori A e B. Diverse volte in questo disegno $n = 1$, cioè non ci sono repliche entro le combinazioni dei gruppi.

Sorgenti di Var.	**g.d.l.**	**Somma dei quadrati**
Fra livelli di A	$a-1$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(\overline{y}_{i.}-\overline{\overline{y}}_{..})^2$
Fra livelli di B	$b-1$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(\overline{y}_{.j}-\overline{\overline{y}}_{..}\right)^2$
Entro gruppi	$abn-(a+b-1)$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(y_{ijk}-\overline{y}_{i.}-\overline{y}_{.j}+\overline{\overline{y}}_{..}\right)^2$
Totale	$abn-1$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(y_{ijk}-\overline{\overline{y}}_{..})^2$

Quando si hanno repliche sufficienti e il disegno è appropriato si può calcolare l'interazione che aggiunge una riga in più alla tabella precedente.

Tabella 3.2: Tabella dell'analisi della varianza a due fattori con interazione. Si assume che il disegno sia bilanciato, cioè che le combinazioni di tutti i livelli a del fattore B e tutti i livelli b del fattore B abbiano lo stesso numero di osservazioni (n).

Sorgenti di Var.	**g.d.l.**	**Somma dei quadrati**
Fra livelli di A	$a-1$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(\overline{y}_{i.}-\overline{\overline{y}}_{..})^2$
Fra livelli di B	$b-1$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(\overline{y}_{.j}-\overline{\overline{y}}_{..}\right)^2$
Interazione A×B	$(a-1)(b-1)$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(\overline{y}_{ij}-\overline{y}_{i.}-\overline{y}_{.j}+\overline{\overline{y}}_{..}\right)^2$
Entro gruppi	$ab(n-1)$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(y_{ijk}-\overline{y}_{ij})^2$
Totale	$abn-1$	$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(y_{ijk}-\overline{\overline{y}}_{..})^2$

Il test statistico si fa come al solito: prima dividendo le somme dei quadrati per i rispettivi gradi di libertà e poi facendo il rapporto fra i quadrati medi relativi ai fattori A e B e l'interazione con i quadrati medi d'errore calcolati come al solito. Il valore di F così ottenuto ci dirà se i fattori o l'interazione sono significativi o no.

3.6.2 L'interazione statistica

In ecologia ci si trova spesso ad avere a che fare con un'*interazione* fra due o più fattori. Si calcola come spiegato nella terza riga della tabella precedente.

Il ragionamento è il seguente: se i fattori A e B fossero completamente additivi, partendo dalla media generale ($\overline{\overline{y}}$), aggiungendo l'effetto del fattore A calcolato come ($\overline{y}_{i.} - \overline{\overline{y}}$) e aggiungendo l'effetto del fattore B calcolato come ($\overline{y}_{.j} - \overline{\overline{y}}$), si riuscirebbero a prevedere tutte le medie delle combinazioni dei due fattori ($\overline{y}_{ij}$), corrispondenti ai quadrati con il bordo nero nella figura precedente. Se le medie delle combinazioni dei fattori ($\overline{y}_{ij}$) non differiscono da quelle previste, allora l'interazione non è significativa e i fattori A e B sono detti completamente additivi.

Se invece le medie delle combinazioni dei due fattori A e B sono significativamente diverse da quelle predette allora l'interazione è significativa. In altri termini se la somma al quadrato delle differenze $\overline{y}_{ij} - (\overline{\overline{y}}_{..} + (\overline{y}_{i.} - \overline{\overline{y}}_{..}) + (\overline{y}_{.j} - \overline{\overline{y}}_{..}))$, cioè la somma al quadrato di $\overline{y}_{ij} - \overline{y}_{i.} - \overline{y}_{.j} + \overline{\overline{y}}_{..}$ differisce significativamente dalla stima della varianza *entro* allora siamo in presenza di un'interazione significativa.

La causa dell'interazione è appunto la *non additività* dei due fattori: ci può essere qualche combinazione dei due fattori che produce un risultato maggiore (o minore) della somma degli effetti separati. Nel nostro esempio ci può essere un qualche genotipo che in qualche situazione ambientale (es: una certa quota altimetrica) produce delle prestazioni superiori o inferiori alla somma dell'effetto del genotipo i-esimo e dell'ambiente j-esimo.

Lo studio delle interazioni assume un significato particolarmente importante in ecologia: si pensi all'interazione genotipo×ambiente. A volte la sua valutazione diventa proprio l'obiettivo principale della ricerca.

Le norme di reazione sono un modo grafico per illustrare l'interazione genotipo × ambiente. Nel grafico delle norme di reazione si mettono gli ambienti sull'asse delle ascisse e la variabile dipendente sull'asse delle ordinate. Le linee connettono le medie degli stessi genotipi nei vari ambienti.

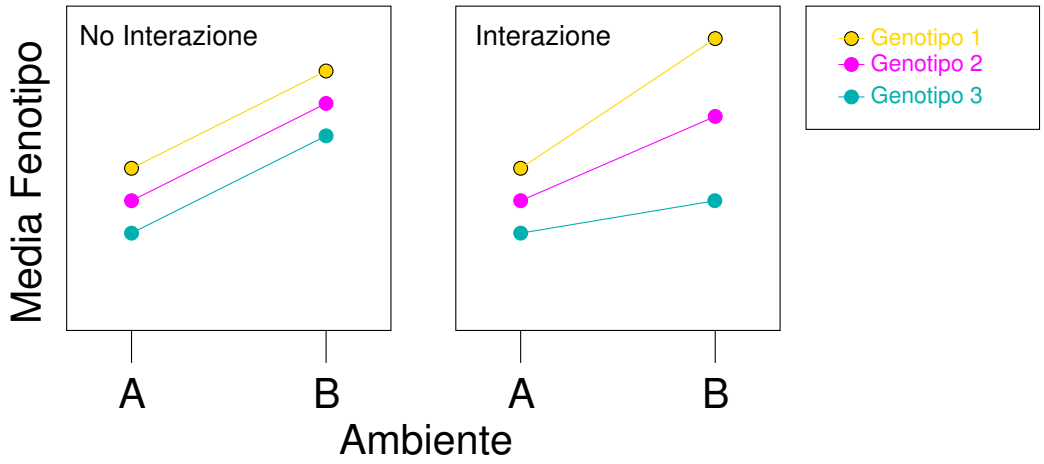


Figura 3.6: Norme di reazione nel caso in cui c'è e non c'è interazione significativa.

Quando non c'è interazione (interazione non significativa) le linee che collegano i genotipi sono parallele: la distanza (differenza fra le medie) dei genotipi è la stessa nei vari ambienti oppure la differenza fra i vari ambienti è la stessa per tutti i genotipi. Mentre la presenza di un'interazione significativa si evidenzia dall fatto che le linee **non sono parallele**.

Ecco un esempio di calcolo con R applicato al file `Picea.txt`:

```
> ##ANOVA A DUE VIE CON INTERAZIONE
> ## Funzione per l'errore standard
> myse <- function(x)return(sd(x, na.rm=T)/sqrt(sum(!is.na(x))))
> ## Importo i dati
> p.df <-read.table("Picea.txt", h=T)
> pdf("Picea_grafici.pdf", paper="a4", heig=18/2.54, wid=18/2.54)
> ##Calcolo le medie e gli se
> mfamdens <- tapply(p.df$Growth, list(p.df$Density,p.df$Family), mean)
> mfamdens
          Fam1   Fam2   Fam3   Fam4
High    113.12 126.92 114.72 129.78
Interm. 125.58 114.90 127.58 117.28
Low     132.64 102.62 139.14 101.48
> s <- tapply(p.df$Growth, list(p.df$Density,p.df$Family), myse)
> s
            Fam1     Fam2     Fam3     Fam4
High    4.314325 4.037499 3.159494 3.267170
Interm. 2.004345 6.846605 2.498279 5.994864
Low     7.222091 5.083542 3.981407 3.374078
```

```
> ### GRAFICO
> coldens <-rainbow(ncol(mfamdens))
> xc1<-barplot(t(mfamdens), beside=T, col=coldens,ylim=c(0,180),
+      xlab="Density", ylab="Growth (g)")
> arrows(xc1,t(mfamdens),xc1,t(mfamdens)+2*t(s), angle=90, length=0.05)
> legend("topleft",fill=coldens, legend=colnames(mfamdens))
> box()
> colfam <-heat.colors(nrow(mfamdens))
> xc2<-barplot(mfamdens, beside=T, col=colfam,ylim=c(0,180),
+     xlab="Family", ylab="Growth (g)")
> arrows(xc2,mfamdens,xc2,mfamdens+2*s, angle=90, length=0.05)
> legend("topleft",fill=colfam, legend=rownames(mfamdens))
> box()
> with(p.df,interaction.plot(Density,Family,Growth))
> dev.off()
null device
          1
>
> ### MODELLO lm CON INTERAZIONE  E
> ### CON CONTRASTI DI DEFAULT: contr.treatment
>
> m.lm <-lm(Growth ~ Family*Density, data=p.df)
> anova(m.lm)
Analysis of Variance Table

Response: Growth
               Df Sum Sq Mean Sq F value   Pr(>F)
Family          3 1589.0  529.68  5.0072 0.004221 **
Density         2   68.8   34.40  0.3252 0.723958
Family:Density  6 5887.9  981.32  9.2766 9.48e-07 ***
Residuals      48 5077.6  105.78
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(m.lm)

Call:
lm(formula = Growth ~ Family * Density, data = p.df)

Residuals:
    Min      1Q  Median      3Q     Max
-25.040  -5.455   0.010   5.885  21.900
```

```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                113.120      4.600  24.593  < 2e-16 ***
FamilyFam2                  13.800      6.505   2.121  0.03907 *
FamilyFam3                   1.600      6.505   0.246  0.80676
FamilyFam4                  16.660      6.505   2.561  0.01363 *
DensityInterm.              12.460      6.505   1.915  0.06140 .
DensityLow                  19.520      6.505   3.001  0.00426 **
FamilyFam2:DensityInterm.  -24.480      9.199  -2.661  0.01056 *
FamilyFam3:DensityInterm.    0.400      9.199   0.043  0.96550
FamilyFam4:DensityInterm.  -24.960      9.199  -2.713  0.00922 **
FamilyFam2:DensityLow      -43.820      9.199  -4.763 1.80e-05 ***
FamilyFam3:DensityLow        4.900      9.199   0.533  0.59673
FamilyFam4:DensityLow      -47.820      9.199  -5.198 4.09e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.29 on 48 degrees of freedom
Multiple R-squared:  0.5978,Adjusted R-squared:  0.5056
F-statistic: 6.485 on 11 and 48 DF,  p-value: 1.812e-06

>
> ## COME VENGONO CALCOLATE LE ESTIMATES DEL SUMMARY?
> ## calcolo le differenze fra le medie rispetto alla combinazione
> ## famiglia e densità di riferimento (Fam1High)
> mfamdens
       High Interm.    Low
Fam1 113.12  125.58 132.64
Fam2 126.92  114.90 102.62
Fam3 114.72  127.58 139.14
Fam4 129.78  117.28 101.48
> mfamdens  - mfamdens[1,1]
      High Interm.    Low
Fam1  0.00   12.46  19.52
Fam2 13.80    1.78 -10.50
Fam3  1.60   14.46  26.02
Fam4 16.66    4.16 -11.64
> ## Per esempio la riga
> ## Coefficients:
> ##                           Estimate Std. Error t value Pr(>|t|)
```

```
> ## ....
> ## FamilyFam3:DensityInterm.    0.400      9.199   0.043  0.96550
> ## .....
>
> ## l'estimate di 0.4 si ottiene facendo:
>
> ## che  equivalgono rispettivamente:
> ##media osservata Fam3Interm
> ##                - (media Fam1Hig + ScartoFam3Hig + ScartFam1Interm)
> 127.58 - (113.12+  1.600 + 12.460)
[1] 0.4
>
> mgen <-mean(p.df$Growth)
> mfam<-with(p.df, tapply(Growth, Family, mean))
> mdens<-with(p.df, tapply(Growth, Density, mean))
> ### il prossimo comando è identico a quello dato per il grafico
> ### ma lo ripetiamo per chiarezza
> mfamdens<-with(p.df, tapply(Growth, list(Family,Density), mean))
> ##medie e scarti dalla medie del primo livello per entrambi i fattori
> mgen
[1] 120.48
> mfam
    Fam1     Fam2     Fam3     Fam4
123.7800 114.8133 127.1467 116.1800
> #mfam- mfam[1]
>
> mdens
   High Interm.     Low
121.135 121.335 118.970
> #mdens-mdens[1]
>
>
>
>
>
> ## NEL CASO DI UN MODELLO SENZA INTERAZIONE
> ## contrasti di default (contr.treatment)
> ## Nel modello senza interazione il livello di riferimento
> ## è di nuovo il primo livello in ordine alfabetico per entrambi
> ## i fattori (in questo caso Fam="Fam1" e Dens="High") ma in questo
> ## caso, con i contrasti di default (contr.treatment), non viene
```

```
> ## usata la media osservata per quella combinazione di fattori ma quella
> ## stimata in base alla somma additiva dei due effetti
> ## Intercept = media generale +( primo livello del primo fattore -
> ## media generale)+ (primo livello del secondo fattore -
> ## media generale)
>
> mnoint.lm <-lm(Growth ~ Family+Density, data=p.df)
> anova(mnoint.lm)
Analysis of Variance Table

Response: Growth
          Df  Sum Sq Mean Sq F value  Pr(>F)
Family     3  1589.0  529.68  2.6084 0.06088 .
Density    2    68.8   34.40  0.1694 0.84461
Residuals 54 10965.5  203.07
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(mnoint.lm)

Call:
lm(formula = Growth ~ Family + Density, data = p.df)

Residuals:
     Min       1Q   Median       3Q      Max
-26.6350 -11.5688   0.2983  10.9900  29.7300

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     124.435      4.506  27.614   <2e-16 ***
FamilyFam2       -8.967      5.203  -1.723   0.0906 .
FamilyFam3        3.367      5.203   0.647   0.5204
FamilyFam4       -7.600      5.203  -1.461   0.1499
DensityInterm.    0.200      4.506   0.044   0.9648
DensityLow       -2.165      4.506  -0.480   0.6329
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.25 on 54 degrees of freedom
Multiple R-squared:  0.1313,Adjusted R-squared:  0.0509
F-statistic: 1.633 on 5 and 54 DF,  p-value: 0.1671
```

```
> ## l'intercetta è calcolata così:
>  120.48 +(123.7800-120.48) + (121.135-120.48)
[1] 124.435
> ## Nel caso del modello senza interazione naturalmente non vengono
> ## stimate le medie per le combinazioni dei fattori ma solo quelle dei
> ## fattori separatamente
>
> ## MODELLO CON INTERAZIONE CON CONTRASTI RISPETTO ALLA
> ## MEDIA GENERALE (contr.sum)
> options("contrasts"=c("contr.sum", "contr.poly"))
> m.contr2.lm <-lm(Growth ~ Family*Density, data=p.df)
> anova(m.contr2.lm)
Analysis of Variance Table

Response: Growth
               Df Sum Sq Mean Sq F value   Pr(>F)
Family          3 1589.0  529.68  5.0072 0.004221 **
Density         2   68.8   34.40  0.3252 0.723958
Family:Density  6 5887.9  981.32  9.2766 9.48e-07 ***
Residuals      48 5077.6  105.78
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## L'anova è identica al caso precedente con contrasti di default
##
> summary(m.contr2.lm)

Call:
lm(formula = Growth ~ Family * Density, data = p.df)

Residuals:
    Min      1Q  Median      3Q     Max
-25.040  -5.455   0.010   5.885  21.900

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      120.4800     1.3278  90.736  < 2e-16 ***
Family1            3.3000     2.2998   1.435 0.157802
Family2           -5.6667     2.2998  -2.464 0.017375 *
Family3            6.6667     2.2998   2.899 0.005633 **
Density1           0.6550     1.8778   0.349 0.728756
Density2           0.8550     1.8778   0.455 0.650932
```

```
Family1:Density1 -11.3150     3.2524  -3.479 0.001081 **
Family2:Density1  11.4517     3.2524   3.521 0.000954 ***
Family3:Density1 -13.0817     3.2524  -4.022 0.000203 ***
Family1:Density2   0.9450     3.2524   0.291 0.772647
Family2:Density2  -0.7683     3.2524  -0.236 0.814257
Family3:Density2  -0.4217     3.2524  -0.130 0.897388
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.29 on 48 degrees of freedom
Multiple R-squared:  0.5978,Adjusted R-squared:  0.5056
F-statistic: 6.485 on 11 and 48 DF,  p-value: 1.812e-06

> ## il riferimento stavolta è la media generale
>
> mgen
[1] 120.48
> mfam  - mgen
     Fam1      Fam2      Fam3      Fam4
 3.300000 -5.666667  6.666667 -4.300000
> mdens - mgen
   High Interm.     Low
  0.655   0.855  -1.510
> ##mfamdens - mgen
>
> ## l’estimates della Fam3 alla densità intermedia (qui chiamata Dens2)
> ## è data da
> 127.58 - (120.48 +6.6667 +0.8550)
[1] -0.4217
> ## media della Fam3Int - (media generale +
> ##     differenza tra media di tutte le Fam3 e la media generale +
> ##     differenza tra media di tutte le DensInter (Dens2) e la media generale)
>
```

3.7 L’analisi della varianza nella regressione

L’analisi della varianza di una regressione si effettua analogamente all’analisi della varianza sui gruppi. Con il calcolo di SSE somma delle devianze d’errore, SSR somma della devianza dovuta alla regressione e SST somma della

devianza totale. Il calcolo della significatività si effettua come al solito sui **quadrati medi** cioè dividendo le devianze per i rispettivi gradi di libertà.

In R si possono eseguire i calcoli in svariati modi più o meno comodi. Usando l'algebra lineare è utile avere a disposione alcune matrici per il calcolo delle devianze.

Partendo dal modello

$$\mathbf{Y} = \mathbf{Xb} + \epsilon$$

I valori attesi saranno dati da:

$$\hat{\mathbf{Y}} = \mathbf{Xb}$$

abbiamo già detto che

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

da cui

$$\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

$$\hat{\mathbf{Y}} = \mathbf{H}\ \mathbf{Y}$$

per cui definiamo $\mathbf{H}$ detta anche "hat matrix" come:

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$$

Un'altra matrice utile è la matrice $\overline{\mathbf{J}}$ data dal prodotto di $\mathbf{1}$ un vettore colonna di "1" con tanti elementi quanti sono i elementi di $\mathbf{Y}$ e $\mathbf{X}$ per un vettore riga di "1" con lo stesso numero di elementi. Il tutto viene diviso per n appunto il numero dei dati (elementi di $\mathbf{Y}$ e $\mathbf{X}$).

$$\mathbf{J} = \mathbf{1}\ \mathbf{1}'$$

$$\overline{\mathbf{J}} = \frac{1}{n}\mathbf{J}$$

Inoltre servirà la matrice identità $\mathbf{I}$.

Forme quadratiche

Definiamo le seguenti matrici:

$$\mathbf{A}_{SST} = \mathbf{I} - \overline{\mathbf{J}}$$

$$\mathbf{A}_{SSE} = \mathbf{I} - \mathbf{H}$$

$$\mathbf{A}_{SSR} = \mathbf{H} - \overline{\mathbf{J}}$$

e allora è possibile dimostrare che

$$SST = \mathbf{Y}' \, \mathbf{A}_{SST} \, \mathbf{Y}$$

$$SSE = \mathbf{Y}' \, \mathbf{A}_{SSE} \, \mathbf{Y}$$

$$SSR = \mathbf{Y}' \, \mathbf{A}_{SSR} \, \mathbf{Y}$$

Dividendo per gli appropriati gradi di liberta troveremo i quadrati medi.

3.7.1 Errore standard dei coefficienti della regressione

Un'ulteriore matrice molto utile è la seguente:

$$var(\mathbf{b}) = MSE \, (\mathbf{X}'\mathbf{X})^{-1}$$

i cui elementi sulla diagonale sotto radice quadrata sono gli errori standard dei coefficienti di regressione (pendenza e intercetta).

L'analisi della varianza in R

In R l'analisi della varianza è basata sull'uso della funzione `lm` che esegue il fitting e stima i parametri del vettore **b**. È utile assegnare il risultato del fitting ad un'oggetto:

```
> fit1.lm <- lm(Gas ~ Temp +Insul, data=whiteside)
```

Il risultato del fitting può quindi essere interrogato con diverse funzioni: `add1, coef, effects, kappa, predict, residuals , alias, deviance, family, labels, print, step , anova, drop1, formula, plot, proj, summary` (vedere la sezione 11 di *An Introduction to R* nel pacchetto di R). Le funzioni più importanti sono: `coef, summary, anova`. Le prime due stampano i coeffcienti e `summary` effettua anche il test per vedere se sono diversi da zero.

La funzione `anova` applicata ad un solo fitting (`anova(fit1.lm)`) effettua il test per l'analisi della varianza e ne stampa la tabella. Se applicata a due o più fitting (`anova(fit1.lm, fit2.lm)`) ne confronta i risultati effettuando un'analisi della varianza sui residui. Affinchè il test riesca è necessario che il numero dei parametri e i gradi di libertà nei due modelli differiscano.

Un importante considerazione è che `anova` applicato ad un solo oggetto effettua il test tenendo conto dell'ordine dei fattori nel modello. Differenze fra un'ordine e un'altro sono riscontrabili solo se i dati non sono ortogonali, cioè l'esperimento non è bilanciato e i vari livelli dei fattori non hanno lo

stesso numero di osservazioni o addirittura mancano del tutto dati per uno o più livelli. Esempio se il modello è il seguente:

```
> m1.lm <- lm(Y ~ A + B + C)
```

e

```
> anova(m1.lm)
```

produrrà una tabella in cui verrà testata prima la significatività di `A`, poi la significatività di `B` al netto di `A`, cioè si testa se il fattore `B` è significativo anche dopo l'effetto di `A`. L'effetto di `C` è testato al netto sia di `A` sia di `B`, cioè dopo che si è tenuto conto di `A` e `B`. Questo risultato è analogo al risultato "Type I Sum of Square" del SAS o di SPSS. L'ordine in cui si inseriscono i fattori è importante a meno che i fattori non siano fra loro ortogonali, nel qual caso un ordine diverso non cambia i risultati.

Se vogliamo testare anche `A` e `B` per ultimi possiamo rifare il fitting cambiando l'ordine e mettendoli per ultimi (si può usare la funzione `update`). Oppure si effettua il fitting senza il fattore che vogliamo considerare e si testano con `anova` i modelli con e senza il fattore che interessa e si vede se l' aumento/decremento di R^2 è significativo o no. Nel nostro esempio per testare `A` è significativo al netto di `B` e `C` è suffciente effettuare

```
> m2.lm <- lm(Y ~ B + C)
```

e

```
> anova(m1.lm,m2.lm).
```

La tabellina con l'analisi effettuata in questo modo può essere ottenuta mediante la funzione `drop1` che di default toglie solo i fattori che è "lecito" togliere:

```
> drop1(m1.lm, test="F")
```

mentre

```
> drop1(m1.lm, scope=.~., test="F")
```

toglie tutti i fattori uno alla volta, ma attenzione alle interazioni! (Es: Non ha senso dire che il fattore `A` non è significativo o non ìmportante, quando una delle sue interazioni (`A:B` o `A:C` è significativa.)

L'output di questo anova è analogo quello riscontrabile nel "Type III Sum of Square" del SAS o di SPSS.

La funzione `aov` è molto simile ad `anova` ma essa richiama automaticamente `lm` per cui non è necessario salvare il risultato del fitting ottenuto da `lm`. Essa produce direttamente una tabellina dell'analisi della varianza. In `aov` è possibile specificare il fattore da mettere al denominatore per il calcolo dell'F. Questa opzione si sfrutta solo in taluni tipi di analisi dove almeno uno dei fattori è casuale, cioè nei casi di analisi della varianza modello II. Il risultato di un `aov` si usa nei confronti multipli per vedere quale dei livelli di un fattore è diverso da quali altri livelli (Metodo di Tukey).

```
> m4.aov <- aov(Y ~ A)
> TukeyHSD(m4.aov)
> plot(TukeyHSD(m4.aov)).
```

Dal plot si capisce facilmente per quale coppia di livelli l'intervallo di confidenza non "copre" lo zero.

3.8 Effetti Fissi ed Effetti casuali

Il modello di analisi della varianza che abbiamo fino ad ora analizzato è un modello a *effetti fissi* (detta anche ANOVA Modello I). Esiste però una categoria di fattori, detti fattori *casuali* o *random*, (ANOVA Modello II) che sono sostanzialmente differenti dai primi nella "filosofia" dell'esperimento, in particolare sono diversi nello schema di campionamento e nei parametri di interesse allo sperimentatore.

Un fattore è detto **fisso** se i livelli del fattore sono scelti *non casualmente* dallo sperimentatore o se consistono in tutti i possibili livelli presenti nella popolazione.

Un fattore è detto **casuale** (o **random**) quando i livelli che lo compongono sono un *campione casuale* di tutti i possibili livelli presenti nella popolazione.

Attenzione a non confondere la scelta delle unità sperimentali, che deve essere sempre casuale anche in un disegno a effetti fissi. In un disegno ad effetti casuale è la scelta dei fattori (es: i tipi di trattamento, il tipo di gruppi) che è scelta casualemente.

Esempi di fattori casuali:

- La scelta di un campione casuale di piante in una popolazione di cui analizzare i semi;
- Le repliche di un esperimento di analisi chimica quantitativa, quando su ciascuna replica effettuo la stessa determinazione analitica più volte.

3.8.1 Obiettivi diversi

Modello random Normalmente in un'analisi con fattori **random** non sono interessato all'effetto di uno specifico livello ma sono interessato a *misurare la variabilità generale* legata a quel fattore (Es: sono interessato a stimare la variabilità *fra famiglie* (fra piante madri) del peso del seme; oppure a vedere quanta variabilità esiste nella determinazione del contenuto di azoto di foglie di faggio scelte a caso su una o più piante). In un modello random tipicamente si stimano le **componenti di varianza** (es: % di varianza *tra* e % di varianza *entro*.

Modello effetti fissi Al contrario in un analisi a fattori **fissi** sono interessato a *misurare l'effetto dello specifico livello* che ho sperimentato. (Es: mi interessa vedere se i semi della famiglia (pianta madre) A pesano di più di quelli della famiglia B; oppure se le foglie "da luce" contengono più azoto delle foglie "da ombra").

Quindi tutto dipende da come ho scelto i livelli su cui effettuo l'esperimento.

Modello misto Se in un esperimento in cui entrambi i tipi di fattori (*fissi* e *casuali*) sono presenti allora si dice che il modello di ANOVA è **misto**.

La distinzione però non è soltanto "filosofica" ma ha dei risvolti importanti per la stima dei parametri dei modelli e viene usata in molti esperimenti di genetica quantitativa (es: nella stima dell'ereditabilità di un tratto fenotipico).

Innanzitutto nei modelli **random** lo sperimentatore se vuole aumentare la potenza del test ha interesse a aumentare, oltre al numero delle osservazioni, anche il numero di livelli nell'esperimento per avere una stima migliore della variabilità legata al fattore in oggetto. Mentre in un modello **fisso** il numero di fattori è "fisso" e lo sperimentatore può aggiungere solo altre osservazioni (non livelli).

Spesso quando i livelli sono tanti (es: quando sono troppi i parametri da stimare) conviene interpretare il fattore come casuale, anche se di per sè sarebbe fisso. Viceversa quando i livelli sono pochi (per esempio meno di 5) allora conviene interpretare il fattore come fisso, anche se di per sè sarebbe casuale, in quanto la variabilità generale tra livelli sarebbe stimata piuttosto male.

In una tipica tabella dell'analisi della varianza le differenze appaiono più evidenti:

Sorg. di Variazione	Quadrati Medi
Effetti fissi:	
Fra Trattamenti	$\sigma_e^2 + \frac{n\sum_{i=1}^{a}(A_i - \overline{A})}{(a-1)}$
Effetti casuali:	
Fra Trattamenti	$\sigma_e^2 + n\sigma_A^2$
Entrambi:	
Entro Trattamenti	σ_e^2

dove n è la dimensione del campione (numero di osservazioni), a è il numero di livelli (trattamenti), σ_e^2 è la stima della varianza *entro* (errore) e σ_A^2 è la stima della varianza "*extra*" dovuta ai trattamenti.

3.9 Modelli gerarchici (Nested)

Nei disegni sperimentali che coinvolgono modelli gerarchici (o nested) ovviamente ci devono essere più di due fattori, ma uno deve essere "innestato" nell'altro oppure potremo dire che uno "comprende" l'altro, cioè sono su due scale gerarchiche diverse (esempi di fattori nested: stato, regione, provincia, comune).

Vi sono cioè almeno due tipi di unità sperimentali di gerarchia diversa (superiore e inferiore) o dimensioni diverse: una più grande e una più piccola (innestata in quella più grande).

- In un modello nested si possono avere sia effetti fissi sia casuali (dipende ovviamente da come scelgo o considero le mie unità sperimentali), *tipicamente però il modello di livello inferiore nella gerarchia è di tipo casuale* (cioè viene scelto casualmente fra una lista di possibili livelli).

- In un modello nested *non ci sono interazioni* fra livelli gerarchici diversi, semplicemente perchè un livello inferiore non può appartenere a più di un livello superiore (es: un comune non può appartenere contemporaneamente a due province). La seguente figura dovrebbe chiarire quest'ultimo concetto.

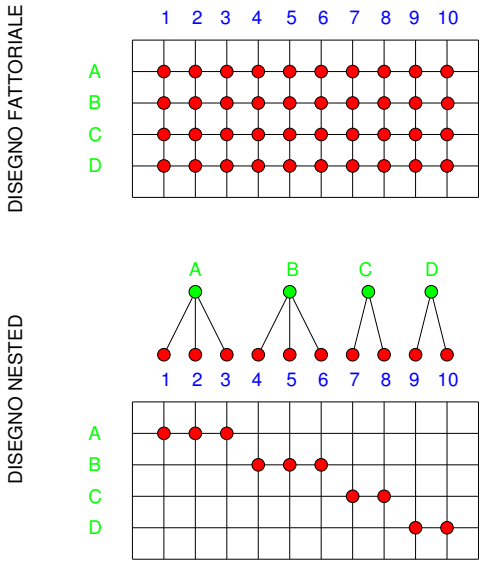


Figura 3.7: Illustrazione grafica fra un disegno sperimentale fattoriale (crossed) e un gerarchico (nested)

In un modello nested il modello additivo che andrò a testare è:

$$Y_{ijk} = \mu + A_i + B_{j(i)} + e_{ijk}$$

dove $B_{j(i)}$ è l'effetto del j-esimo fattore "innestato" (*entro*) il fattore i-esimo gerarchicamente superiore.

Nei modelli nested in cui il fattore gerarchicamente inferiore $B(A)$ è random, il test F per la significatività del fattore superiore A viene calcolato diversamente dal solito:

$$F = \frac{QMdiA}{QMdiB(A)}$$

mentre il fattore inferiore $B(A)$ viene testato normalmente contro l'errore come al solito.

Comunque nei modelli misti la scelta di come condurre il test F, in particolare di cosa mettere al denominatore, è sempre molto complicata e conviene riferirsi ad un testo specialistico

Capitolo 4

Regressione con le Matrici [1]

4.1 Cenni di calcolo matriciale

4.1.1 Definizioni preliminari

- **Definizione di matrice**. Una matrice è un insieme di numeri organizzati per righe e per colonne. Noi considereremo solo matrici costituite di numeri reali.

- **Matrici quadrate e matrici rettangolari**. Quando il numero di righe è uguale al numero di colonne la matrice si dice **quadrata**. Altrimenti la matrice si dice **Rettangolare**.

- **Vettori**. Una matrice costituita da una sola riga è chiamata **vettore riga**:

$$\begin{pmatrix} 83 & 15 & 21 & 3 & 45 \end{pmatrix}$$

Una matrice costuita da una sola colonna è chiamata **vettore colonna**:

$$\begin{pmatrix} 2 \\ 3 \\ 12 \\ 13 \\ 21 \\ 16 \end{pmatrix}$$

È immediato il collegamento con i vettori della geometria (e della fisica).

[1] Capitolo scritto in collaborazione con Franco Sartore - Dipartimento di Scienze Ambientali - Università di Parma

- **Ordine di una matrice**. L'ordine di una matrice è dato da suo numero di righe e di colonne. La matrice che segue è di ordine 2×4.

$$\begin{pmatrix} 2 & 4 & 5 & 6 \\ 3 & 1 & 21 & 3 \end{pmatrix}$$

- Una matrice si indica generalmente con una **lettera maiuscola** in grassetto eventualmente munita di pedici che ne indicano l'**ordine**.

$$\mathbf{A_{3,4}} = \begin{pmatrix} 12 & 34 & 5 & 6 \\ 2 & 5 & 7 & 4 \\ 23 & 1 & 21 & 3 \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} 1 & 4 & 5 \\ 5 & 2 & 7 \\ 2 & 1 & 1 \end{pmatrix}$$

- In forma simbolica un elemento di matrice si indica con una **lettera minuscola** munita di indici non separati da virgola che ne individuano la **riga** e **colonna** di appartenenza.

$$\mathbf{B_{n,p}} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2p} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ b_{i1} & b_{i2} & \cdots & b_{ij} & \cdots & b_{ip} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nj} & \cdots & b_{np} \end{pmatrix}$$

Più sinteticamente possiamo scrivere:

$$\mathbf{B_{n,p}} = (b_{ij})$$

o anche:

$$\mathbf{B} = (b_{ij})$$

$$[i = 1, 2, ..., n; j = 1, 2, ..., p]$$

La matrice rappresentata sopra è una generica matrice $\mathbf{B}$ di ordine $n \times p$ che potremmo anche chiamare $\mathbf{B_{np}}$. Il suo **elemento generico**, come si può notare, è b_{ij}.

- Un vettore si indica in genere con una lettera minuscola (o maiuscola) in grassetto, eventualmente munita di indice quando il vettore è una colonna di matrice.

$$\mathbf{a} = \begin{pmatrix} 2 \\ 3 \\ 12 \\ 13 \\ 21 \\ 16 \end{pmatrix}$$

$$\mathbf{x_2} = \begin{pmatrix} 4 \\ 2 \\ 1 \end{pmatrix}$$

  Il vettore $\mathbf{x_2}$ è la seconda colonna della matrice X.

- **Diagonale principale**. Gli elementi di una matrice quadrata che hanno l'indice di riga uguale all'indice di colonna appartengono alla **diagonale principale**. La diagonale principale della matrice $\mathbf{X}$ è costituita dagli elementi 1, 2, 1.

- **Traccia di una matrice**. La traccia di una matrice quadrata è la somma degli elementi della diagonale principale. La traccia della matrice $\mathbf{X}$ è data da:

$$Tr(\mathbf{X}) = 1 + 2 + 1 = 4$$

- **Matrice trasposta**. La matrice che si ottiene da una matrice data scambiandone le righe con le colonne si chiama matrice trasposta e si indica con il simbolo $'$ (oppure con una piccola t) applicato/a al nome della matrice originaria.

$$\mathbf{A'_{4,3}} = \mathbf{A_{4,3}}^t = \begin{pmatrix} 12 & 2 & 23 \\ 34 & 5 & 1 \\ 5 & 7 & 21 \\ 6 & 4 & 3 \end{pmatrix}$$

  La matrice $\mathbf{A'_{4,3}}$ è la matrice trasposta o semplicemente la trasposta della matrice $\mathbf{A_{3,4}}$.

4.1.2 Algebra matriciale

Somma di due matrici

La somma di matrici è definita solo per matrici simili, cioè dello stesso ordine. Date due matrici di ordine $n \times p$:

$$\mathbf{A_{n,p}} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2p} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{ip} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{np} \end{pmatrix}$$

$$\mathbf{B_{n,p}} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2p} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ b_{i1} & b_{i2} & \cdots & b_{ij} & \cdots & b_{ip} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nj} & \cdots & b_{np} \end{pmatrix}$$

la matrice somma è definita come:

$$\mathbf{A_{n,p}} + \mathbf{B_{n,p}} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1j}+b_{1j} & \cdots & a_{1p}+b_{1p} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2j}+b_{2j} & \cdots & a_{2p}+b_{2p} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1}+b_{i1} & a_{i2}+b_{i2} & \cdots & a_{ij}+b_{ij} & \cdots & a_{ip}+b_{ip} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \cdots & a_{nj}+b_{nj} & \cdots & a_{np}+b_{np} \end{pmatrix}$$

La somma di matrici è **commutativa**.

Prodotto di una matrice per uno scalare

Data una matrice $\mathbf{A_{n,p}}$ ed uno scalare k, il prodotto di $\mathbf{A_{n,p}}$ per k è dato dalla matrice che si ottiene moltiplicando per k ogni elemento della matrice $\mathbf{A_{n,p}}$:

$$k\mathbf{A_{n,p}} = \begin{pmatrix} ka_{11} & ka_{12} & \cdots & ka_{1j} & \cdots & ka_{1p} \\ ka_{21} & ka_{22} & \cdots & ka_{2j} & \cdots & ka_{2p} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ ka_{i1} & ka_{i2} & \cdots & ka_{ij} & \cdots & ka_{ip} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ ka_{n1} & ka_{n2} & \cdots & ka_{nj} & \cdots & ka_{np} \end{pmatrix}$$

Prodotto di due matrici

Data una matrice $\mathbf{A_{n,p}}$ (di ordine $n \times p$) ed una matrice $\mathbf{B_{p,m}}$ (di ordine $p \times m$), il prodotto (detto **prodotto righe per colonne**) di $\mathbf{A_{n,p}}$ per $\mathbf{B_{p,m}}$ è una matrice $\mathbf{C_{n,m}} = \mathbf{A_{n,p}} \cdot \mathbf{B_{p,m}}$ (di ordine $n \times m$) il cui elemento generico c_{ij} si ottiene sommando i prodotti di ciascun elemento della riga i-esima della matrice $\mathbf{A_{n,p}}$ per il corrispondente elemento della colonna j-esima della matrice $\mathbf{B_{p,m}}$

$$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \cdots + a_{ij} \cdot b_{ij} + \cdots + a_{ip} \cdot b_{pj} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

- La matrice prodotto ha il numero di righe del primo fattore e il numero di colonne del secondo fattore.
- Il prodotto di matrici è possibile solo se il numero di colonne della prima matrice è uguale al numero di righe della seconda matrice.
- Il prodotto di matrici **non è commutativo**.

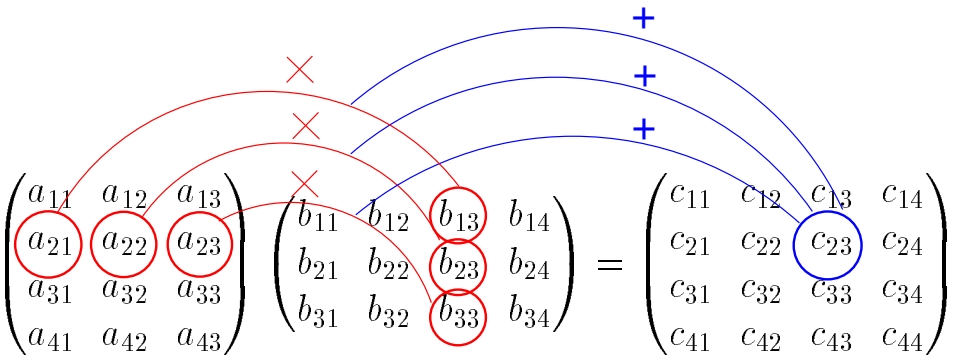


Figura 4.1: Esempio del calcolo di un elemento del prodotto di due matrici.

Esempio numerico.

$$\begin{pmatrix} 2 & 1 \\ 4 & 3 \\ 5 & 0 \\ 6 & 4 \end{pmatrix} \times \begin{pmatrix} 8 & 2 & 9 \\ 0 & 7 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} (2 \times 8 + 1 \times 0) & (2 \times 2 + 1 \times 7) & (2 \times 9 + 1 \times 1) \\ (4 \times 8 + 3 \times 0) & (4 \times 2 + 3 \times 7) & (4 \times 9 + 3 \times 1) \\ (5 \times 8 + 0 \times 0) & (5 \times 2 + 0 \times 7) & (5 \times 9 + 0 \times 1) \\ (6 \times 8 + 4 \times 0) & (6 \times 2 + 4 \times 7) & (6 \times 9 + 4 \times 1) \end{pmatrix} = \begin{pmatrix} 16 & 11 & 19 \\ 32 & 29 & 39 \\ 40 & 10 & 45 \\ 48 & 40 & 58 \end{pmatrix}$$

4.1.3 Alcuni tipi di matrice

- **Matrice simmetrica**. Una matrice quadrata si dice simmetrica quando vale la relazione:

$$\mathbf{A} = \mathbf{A}'$$

  Una tipica matrice simmetrica è la **matrice di correlazione**.

- **Matrice diagonale**. Una matrice simmetrica i cui unici elementi diversi da zero sono quelli della diagonale principale si chiama matrice diagonale.

$$\mathbf{D} = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Matrice identità**. Una matrice diagonale i cui elementi non nulli sono tutti uguali a 1. Si indica generalmente con **I**. Vedremo più avanti come la matrice identità giochi nell'algebra delle matrici lo stesso ruolo del numero 1 nell'algebra dei numeri.

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

  La matrice dell'esempio è una matrice identità di ordine 3×3.

4.1.4 Determinanti

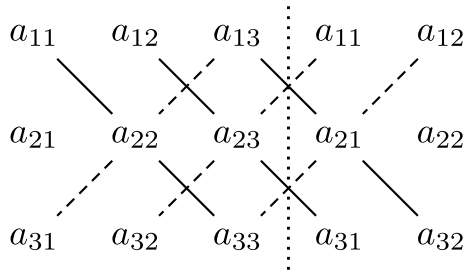
Ad ogni **matrice quadrata** si può associare un numero chiamato **determinante** .
Data una matrice quadrata:

$$\mathbf{A_{n,n}} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn} \end{pmatrix}$$

Il suo determinate si indica con

$$det(\mathbf{A_{n,n}}) = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn} \end{vmatrix}$$

- $n = 1$
  Il determinante di $\mathbf{A_{1,1}}$ coincide con l'unico elemento della matrice
  $$det(\mathbf{A_{1,1}}) = \left|a_{11}\right| = a_{11}$$
- $n = 2$
  $$det(\mathbf{A_{2,2}}) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \times a_{22} - a_{21} \times a_{12}$$
- $n = 3$
  Si può utilizzare la regola di Sarrus. Ovvero:
  $$A_{3,3} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$
  $det(A_{3,3}) = a_{11}a_{22}a_{33}+a_{12}a_{23}a_{31}+a_{13}a_{21}a_{32}-a_{31}a_{22}a_{13}-a_{32}a_{23}a_{11}-a_{33}a_{21}a_{12}$
  Per ricordarla mnemonicamente conviene riferirsi alla seguente figura:



- $n > 3$
  In questo caso dobbiamo introdurre la nozione di **Complemento algebrico** di un elemento di matrice e di **Minore complementare**

  Considerando la solita matrice quadrata $\mathbf{A_{n,n}}$ si definisce **Minore complementare** di un generico elemento a_{ij}, indicato con A_{ij}, il determinante della matrice di ordine $n-1$ che si ottiene eliminando la riga di

posto i e la colonna di posto j dalla matrice data.
Il Minore complementare dell'elemento a_{11}, ad esempio, è dato da

$$A_{11} = det \begin{pmatrix} - & - & - & - & - & - \\ - & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ - & \vdots & \cdots & \vdots & \cdots & \vdots \\ - & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ - & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn} \end{pmatrix}$$

dove i trattini stanno al posto degli elementi eliminati.

Il **Complemento algebrico**, indicato con $\mathcal{A}_{ij}$, di un generico elemento a_{ij} della matrice $\mathbf{A}$ è dato da

$$\mathcal{A}_{ij} = (-1)^{i+j} \times A_{ij}$$

A questo punto siamo in grado di enunciare il Primo Teorema di Laplace che permette di calcolare il determinante di una matrice di ordine qualunque: *Un determinante è sempre uguale alla somma dei prodotti degli elementi di una riga (o colonna) qualunque, per i rispettivi Complementi algebrici.* Il determinante sviluppato rispetto agli elementi della riga di posto i è dato da

$$det(\mathbf{A_{n,n}}) = \sum_{j=1}^{n} a_{ij}\mathcal{A}_{ij}$$

Matrice inversa

Data un matrice quadrata $\mathbf{A}$ la **Matrice inversa** di $\mathbf{A}$ è un'altra matrice quadrata, indicata con $\mathbf{A}^{-1}$ che gode delle proprietà:

$$\mathbf{A} \times \mathbf{A}^{-1} = \mathbf{A}^{-1} \times \mathbf{A} = \mathbf{I}$$

La matrice inversa gioca pertanto nell'algebra delle matrici lo stesso ruolo del reciproco (inverso) di un numero nell'algebra dei numeri.

- Nell'algebra delle matrici non esiste il rapporto (la divisione) di matrici,al suo posto si utilizza il prodotto con l'inversa.
- La matrice inversa non esiste sempre: si può dimostrare che **condizione necessaria e sufficiente affinché una matrice quadrata abbia l'inversa è che il suo determinante si diverso da zero**

Consideriamo la matrice quadrata:

$$\mathbf{A_{n,n}} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn} \end{pmatrix}$$

Costruiamo una matrice in cui ogni elemento di **A** è sostituito dal suo **Complemento algebrico**

$$\begin{pmatrix} \mathcal{A}_{11} & \mathcal{A}_{12} & \cdots & \mathcal{A}_{1j} & \cdots & \mathcal{A}_{1n} \\ \mathcal{A}_{21} & \mathcal{A}_{22} & \cdots & \mathcal{A}_{2j} & \cdots & \mathcal{A}_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathcal{A}_{i1} & \mathcal{A}_{i2} & \cdots & \mathcal{A}_{ij} & \cdots & \mathcal{A}_{in} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathcal{A}_{n1} & \mathcal{A}_{n2} & \cdots & \mathcal{A}_{nj} & \cdots & \mathcal{A}_{nn} \end{pmatrix}$$

La sua trasposta, che indicheremo con $\mathcal{A}^*$, è la **Matrice aggiunta** di **A**

$$\mathcal{A}^* = \begin{pmatrix} \mathcal{A}_{11} & \mathcal{A}_{21} & \cdots & \mathcal{A}_{i1} & \cdots & \mathcal{A}_{n1} \\ \mathcal{A}_{12} & \mathcal{A}_{22} & \cdots & \mathcal{A}_{i2} & \cdots & \mathcal{A}_{n2} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathcal{A}_{1j} & \mathcal{A}_{2j} & \cdots & \mathcal{A}_{ij} & \cdots & \mathcal{A}_{nj} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathcal{A}_{1n} & \mathcal{A}_{2n} & \cdots & \mathcal{A}_{in} & \cdots & \mathcal{A}_{nn} \end{pmatrix}$$

A questo punto siamo in grado di calcolare la matrice inversa che sarà data da:

$$\mathbf{A^{-1}} = \frac{\mathcal{A}^*}{det(\mathbf{A})} = \frac{1}{det(\mathbf{A})} \begin{pmatrix} \mathcal{A}_{11} & \mathcal{A}_{21} & \cdots & \mathcal{A}_{i1} & \cdots & \mathcal{A}_{n1} \\ \mathcal{A}_{12} & \mathcal{A}_{22} & \cdots & \mathcal{A}_{i2} & \cdots & \mathcal{A}_{n2} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathcal{A}_{1j} & \mathcal{A}_{2j} & \cdots & \mathcal{A}_{ji} & \cdots & \mathcal{A}_{nj} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathcal{A}_{1n} & \mathcal{A}_{2n} & \cdots & \mathcal{A}_{in} & \cdots & \mathcal{A}_{nn} \end{pmatrix} =$$

$$= \begin{pmatrix} \frac{\mathcal{A}_{11}}{det(\mathbf{A})} & \frac{\mathcal{A}_{21}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{i1}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{n1}}{det(\mathbf{A})} \\ \frac{\mathcal{A}_{12}}{det(\mathbf{A})} & \frac{\mathcal{A}_{22}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{i2}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{n2}}{det(\mathbf{A})} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \frac{\mathcal{A}_{1j}}{det(\mathbf{A})} & \frac{\mathcal{A}_{2j}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{ij}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{nj}}{det(\mathbf{A})} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \frac{\mathcal{A}_{1n}}{det(\mathbf{A})} & \frac{\mathcal{A}_{2n}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{in}}{det(\mathbf{A})} & \cdots & \frac{\mathcal{A}_{nn}}{det(\mathbf{A})} \end{pmatrix}$$

Passiamo ad un esempio pratico.

- Consideriamo la matrice

$$\mathbf{C} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 1 & 0 & 0 \end{pmatrix}$$

- calcoliamo il suo determinante con la regola di Laplace considerando ovviamente la terza riga. L'unico elemento diverso da zero è $c_{31} = 1$; il suo complemento algebrico è

$$\mathcal{C}_{31} = (-1)^{(3+1)} \times det \begin{pmatrix} 2 & 1 \\ 2 & -1 \end{pmatrix} = (-1)^4 \times (-4) = -4$$

  per cui

$$det(\mathbf{C}) = c_{31} \times \mathcal{C}_{31} = 1 \times (-4) = -4$$

  Il determinante di $\mathbf{C}$ è diverso da zero, quindi l'inversa esiste.

- per costruire la matrice aggiunta abbiamo bisogno anche degli altri complementi algebrici. Mostriamo ancora lo sviluppo del cal calcolo di $\mathcal{C}_{13}$ poi passiamo all-aggiunta lasciando al lettrore la verifica dei calcoli

$$\mathcal{C}_{13} = (-1)^{(1+3)} \times det \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} = (-1)^4 = 1 \times (-2) = -2$$

- La matrice aggiunta di $\mathbf{C}$ è

$$\mathcal{C}^* = \begin{pmatrix} \mathcal{C}_{11} & \mathcal{C}_{21} & \mathcal{C}_{31} \\ \mathcal{C}_{12} & \mathcal{C}_{22} & \mathcal{C}_{32} \\ \mathcal{C}_{13} & \mathcal{C}_{23} & \mathcal{C}_{33} \end{pmatrix} =$$

$$= \begin{pmatrix} 0 & 0 & -4 \\ -1 & -1 & 1 \\ -2 & 2 & 2 \end{pmatrix}$$

- L'inversa finalmente è data da

$$\mathbf{C^{-1}} = \frac{\mathcal{C}^*}{det(\mathbf{C})} = \begin{pmatrix} \frac{\mathcal{C}_{11}}{det(\mathbf{C})} & \frac{\mathcal{C}_{21}}{det(\mathbf{C})} & \frac{\mathcal{C}_{31}}{det(\mathbf{C})} \\ \frac{\mathcal{C}_{12}}{det(\mathbf{C})} & \frac{\mathcal{C}_{22}}{det(\mathbf{C})} & \frac{\mathcal{C}_{32}}{det(\mathbf{C})} \\ \frac{\mathcal{C}_{13}}{det(\mathbf{C})} & \frac{\mathcal{C}_{23}}{det(\mathbf{C})} & \frac{\mathcal{C}_{33}}{det(\mathbf{C})} \end{pmatrix} =$$

$$= \begin{pmatrix} \frac{0}{-4} & \frac{0}{-4} & \frac{-4}{-4} \\ \frac{-1}{-4} & \frac{-1}{-4} & \frac{1}{-4} \\ \frac{-2}{-4} & \frac{2}{-4} & \frac{2}{-4} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0.25 & 0.25 & -0.25 \\ 0.5 & -0.5 & -0.5 \end{pmatrix}$$

4.1.5 Vettori linearmente dipendenti

Possiamo intendere una matrice come composta da una serie di vettori colonna o riga. I vettori (riga o colonna) di una matrice $\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_j}, \cdots, \mathbf{v_n}$ sono detti **linearmente dipendenti** se esistono dei numeri $a_1, a_2, a_3, \cdots a_j \cdots a_n$ (scalari) non tutti uguali a zero, tali che:

$$a_1\mathbf{v_1} + a_2\mathbf{v_2} + a_3\mathbf{v_3} + \cdots + a_j\mathbf{v_j} + \cdots + a_n\mathbf{v_n} = \mathbf{0} \tag{4.1}$$

Uno dei casi frequenti di vettori linearmente indipendenti si ha quando ci sono almeno due vettori esattamente proporzionali l'uno all'altro. o quando la somma (o la differenza) di uno o più vettori, è esattamente proporzionale a un altro vettore (o somma o differenza di vettori). In sostanza è come se ci fossero dei vettori che non portano nessuna nuova informazione rispetto ad uno o più vettori precedenti.

Per esempio si verifica che la seguente matrice ha vettori linearmente dipendenti

$$\mathbf{C} = \begin{pmatrix} 1 & 2 & -1 & -4 \\ -2 & 5 & 2 & -10 \\ 3 & 8 & 2 & -16 \\ -2 & -3 & -1 & 6 \end{pmatrix}$$

Il quarto vettore colonna è uguale al secondo vettore colonna moltiplicato per 2 e cambiato di segno. Quindi, ponendo $a_2 = 2$, $a_4 = 1$ e tutti gli altri a_i

=0, verifico che i vettori della matrice $\mathbf{C}$ sono linearmente dipendenti ovvero ci sono almeno due valori di a che non sono zero e il risultato dell'equazione 4.1 è comunque un vettore di zeri.

Una matrice con vettori linearmente dipendenti **ha il determinate uguale a 0** e come tale è una matrice non invertibile.

A questo punto viene facile introdurre il concetto di **Rango** di una matrice come il numero di colonne (o di righe) linearmente indipendenti esistenti nella matrice. Se il rango di una matrice quadrata è minore del numero di colonne (o di righe) della matrice, il suo determinante è uguale a zero e la matrice non può essere invertita. In R il rango di una matrice è disponibile dalla funzione `rankMatrix()` dentro al pacchetto `Matrix`.

In statistica si può affermare che quando una colonna è correlata perfettamente ($R^2 = 1$) con un altra colonna (o con la somma di due o più colonne o con una combinazione lineare di precedenti colonne) allora la colonna non "porta nessuna nuova informazione" che non fosse già presente nella matrice e la matrice non si può invertire. In genere è sufficente rimuovere la/e colonna/e di dati "ridondante/i", per avere di nuovo una matrice invertibile.

È utile anche sapere che il prodotto di una matrice $\mathbf{A}$ per la sua trasposta $\mathbf{A^T}$, cioè$\mathbf{A^T}\,\mathbf{A}$ ha lo stesso rango di $\mathbf{A}$ e il determinante è uguale al prodotto dei determinanti delle due matrici (che sono a loro volta uguali $det(\mathbf{A}) = \det(\mathbf{A^T})$), cioè

$$det(\mathbf{A^T}\,\mathbf{A}) = det(\mathbf{A^T})\,det(\mathbf{A}).$$

Per cui, se la matrice $\mathbf{A}$ ha colonne linearmente dipendenti e quindi $det(\mathbf{A}) = 0$, allora anche $det(\mathbf{A^T}\,\mathbf{A}) = 0$ e $\mathbf{A^T}\,\mathbf{A}$ sarà non invertibile.

La matrice dei dati

Immaginiamo, a titolo, d'esempio, di aver effettuato su una data unità di campionamento una serie di misure ognuna relativa ad un diverso parametro chimico fisico. Se chiamiamo $\mathbf{P_1}$ l'unità di campionamento e $\mathbf{X_1}, \mathbf{X_2}, \cdots, \mathbf{X_j}, \cdots, \mathbf{X_{p-1}}$ i $p-1$ parametri misurati e $x_{11}, x_{12}, \cdots, x_{1j}, \cdots, x_{1p-1}$ le misure effettuate, l'unità di campionamento sarà individuata dal vettore riga

$$\mathbf{P_1} = \left(\; x_{11}, \;\; x_{12}, \;\; \cdots, \;\; x_{1j}, \;\; \cdots, \;\; x_{1p-1} \; \right)$$

Questo vettore riga può essere rappresentato come un punto (o un vettore) in un riferimento cartesiano $p-1$-dimensionale i cui assi siano individuati dai $p-1$ parametri cioè $\mathbf{X_1}, \mathbf{X_2}, \cdots, \mathbf{X_j}, \cdots, \mathbf{X_{p-1}}$. Quando $p-1$ non è maggiore di 3 se ne può dare anche una rappresentazione grafica. Misurando gli stessi parametri anche in (su) tutte le altre unità di campionamento (supponiamo che siano n) relative al nostro problema, avremo n vettori riga che potremo

disporre in una matrice rettangolare di ordine $n \times (p-1)$, la **matrice dei dati**.

$$\mathbf{X}_{n,p-1} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1p-1} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2p-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{ip-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{np-1} \end{pmatrix}$$

Se leggiamo la matrice dei dati per colonne possiamo dire che essa è composta da $p-1$ vettori colonna, ognuno dei quali, essendo costituito dai risultati delle n misure di un parametro, può essere considerato una variabile stocastica. Esempio. Misuriamo in 9 laghi appenninici due parametri, ad esempio la conducibilità a 20 gradi centigradi (**K20C**) e la concentrazione di ione solfato e (**SO4**) in nove unità di campionamento. La matrice

$$\mathbf{X} = \begin{pmatrix} 202 & 0.30 \\ 175 & 0.23 \\ 204 & 0.29 \\ 96 & 0.21 \\ 159 & 0.29 \\ 37 & 0.09 \\ 128 & 0.08 \\ 222 & 0.37 \\ 202 & 0.28 \end{pmatrix}$$

che contiene i risultati delle misure può essere vista come l'insieme di nove vettori riga

$$\mathbf{X} = \begin{pmatrix} \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \\ \vdots \\ \mathbf{P_9} \end{pmatrix}$$

ciascuno dei quali individua una unità di campionamento e può essere rappresentato come un punto in un uno spazio bidimensionale

$$\begin{aligned} \mathbf{P_1} &= (202, 0.30) \\ \mathbf{P_2} &= (175, 0.23) \\ \mathbf{P_3} &= (204, 0.29) \\ \vdots &= \quad \vdots \\ \mathbf{P_9} &= (202, 0.28) \end{aligned}$$

e come l'insieme di due vettori colonna contenenti i valori delle variabili **K20C** e **SO4**

$$\mathbf{X} = \begin{pmatrix} \mathbf{K20C} & \mathbf{SO4} \end{pmatrix}$$

$$\mathbf{K20C} = \begin{pmatrix} 202 \\ 175 \\ 204 \\ \vdots \\ 202 \end{pmatrix}$$

$$\mathbf{SO4} = \begin{pmatrix} 0.30 \\ 0.23 \\ 0.29 \\ \vdots \\ 0.28 \end{pmatrix}$$

La rappresentazione grafica della matrice $\mathbf{X}$ è mostrata nella figura 4.2.

La matrice di Devianza-Codevianza

La Devianza di una variabile $\mathbf{X}$ con n osservazioni di media $\bar{x}$ è data da

$$Dev(\mathbf{X}) = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Vediamo come può essere calcolata con l'ausilio dell'algebra matriciale: Consideriamo il vettore colonna degli scarti dalla media di ciascun valore di $\mathbf{X}$

$$\mathbf{S} = \begin{pmatrix} x_1 - \bar{x} \\ x_2 - \bar{x} \\ \vdots \\ x_i - \bar{x} \\ \vdots \\ x_n - \bar{x} \end{pmatrix}$$

ed il vettore riga ottenuto trasponendolo

$$\mathbf{S}^t = \begin{pmatrix} x_1 - \bar{x} & x_2 - \bar{x} & \cdots & x_i - \bar{x} & \cdots & x_n - \bar{x} \end{pmatrix}$$

Per verificare che

$$\mathbf{S}^t \times \mathbf{S} = Dev(\mathbf{X})$$

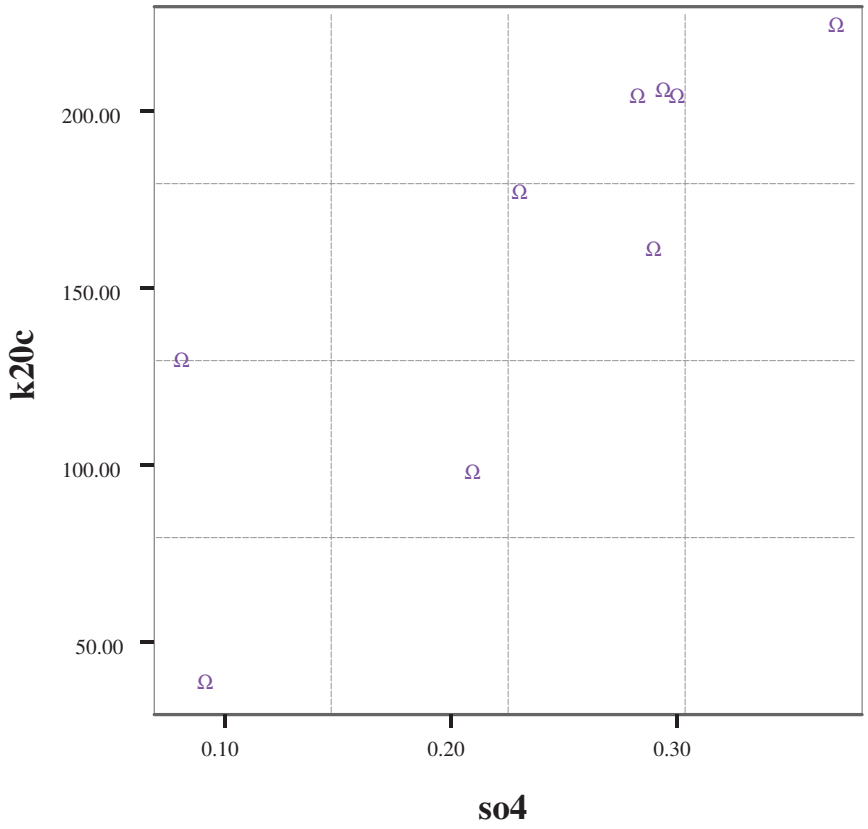


Figura 4.2: Grafico

sviluppiamo il prodotto di matrici

$$\begin{pmatrix} x_1-\bar{x} & x_2-\bar{x} & \cdots & x_i-\bar{x} & \cdots & x_n-\bar{x} \end{pmatrix} \times \begin{pmatrix} x_1-\bar{x} \\ x_2-\bar{x} \\ \vdots \\ x_i-\bar{x} \\ \vdots \\ x_n-\bar{x} \end{pmatrix} =$$

$$= ((x_1-\bar{x})\times(x_1-\bar{x}) + (x_2-\bar{x})\times(x_2-\bar{x}) + \cdots + (x_i-\bar{x})\times(x_i-\bar{x}) +$$

$$+ \cdots + (x_n-\bar{x})\times(x_i-\bar{x})) = \sum_{i=1}^{n}(x_i-\bar{x})^2$$

La Devianza può anche essere espressa nella cosiddetta *forma computazionale* che si ricava dalla precedente con un minimo di algebra

$$Dev(\mathbf{X}) = \sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}$$

Proviamo a ricavare anche questa formula con l'algebra delle matrici.
A questo scopo definiamo un vettore colonna di n elementi (tanti quante sono le osservazioni di $\mathbf{X}$ tutti ugali alla media di $\mathbf{X}$

$$\bar{\mathbf{X}} = \begin{pmatrix} \bar{x} \\ \bar{x} \\ \vdots \\ \bar{x} \\ \vdots \\ \bar{x} \end{pmatrix}$$

Vogliamo verficare che la Devianza di $\mathbf{X}$ è data anche da

$$Dev(\mathbf{X}) = \sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n} = \mathbf{X}^t \times \mathbf{X} - \bar{\mathbf{X}}^t \times \bar{\mathbf{X}}$$

Considerando separatamente i due prodotti matriciali si ha

$$\mathbf{X}^t \times \mathbf{X} = \begin{pmatrix} x_1 & x_2 & \cdots & x_i & \cdots & x_n \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{pmatrix} =$$

$$= (x_1 \times x_1 + x_2 \times x_2 + \cdots + x_i \times x_i + \cdots + x_n \times x_n) = \sum_{i=1}^{n} x_i^2$$

$$\bar{\mathbf{X}}^t \times \bar{\mathbf{X}} = \begin{pmatrix} \bar{x} & \bar{x} & \cdots & \bar{x} & \cdots & \bar{x} \end{pmatrix} \times \begin{pmatrix} \bar{x} \\ \bar{x} \\ \vdots \\ \bar{x} \\ \vdots \\ \bar{x} \end{pmatrix} =$$

$$= (\bar{x} \times \bar{x} + \bar{x} \times \bar{x} + \cdots + \bar{x} \times \bar{x} + \cdots + \bar{x} \times \bar{x})) = n \times (\bar{x})^2 =$$

$$= n \times \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n^2} = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}$$

La Codevianza di due variabili $\mathbf{X}$ e $\mathbf{Y}$ è data da

$$Codev(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

In analogia con quanto già visto per la devianza, se definiamo due vettori colonna $\mathbf{S_X}$ e $\mathbf{S_Y}$ contenenti gli scarti delle variabili dalle loro medie, la Codevianza nelle forma precedente è data da

$$Codev(\mathbf{X}, \mathbf{Y}) = \mathbf{S_X}^t \times \mathbf{S_Y}$$

La *formula computazionale* è

$$Codev(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}$$

In forma matriciale:

$$= \mathbf{X}^t \times \mathbf{Y} - \bar{\mathbf{X}}^t \times \bar{\mathbf{Y}}$$

E' possibile disporre le Devianze e le Codevianze in una matrice simmetrica lui diagonale principale è costituita dalle Devianze che, nel caso delle due variabili precedenti assume la forma

$$\mathbf{C_{XY}} = \begin{pmatrix} Dev(\mathbf{X}) & Codev(\mathbf{X}, \mathbf{Y}) \\ Codev(\mathbf{Y}, \mathbf{X}) & Dev(\mathbf{Y}) \end{pmatrix}$$

chiamata **matrice di Devianza-Codevianza**.
Ovviamente $Codev(\mathbf{X}, \mathbf{Y}) = Codev(\mathbf{Y}, \mathbf{X})$
Se

$$\mathbf{X} = (\mathbf{X}_1 \mathbf{X}_2 \cdots \mathbf{X}_j \cdots \mathbf{X}_k)$$

è una matrice dei dati costituta da k variabili la matrice di Devianza-Codevianza ad essa associata é data da

$$\mathbf{C} = \begin{pmatrix} Dev(\mathbf{X_1}) & Codev(\mathbf{X_1}, \mathbf{X_2}) & \cdots & Codev(\mathbf{X_1}, \mathbf{X_k}) \\ Codev(\mathbf{X_2}, \mathbf{X_1}) & Dev(\mathbf{X_2}) & \cdots & Codev(\mathbf{X_2}, \mathbf{X_k}) \\ \vdots & \vdots & \vdots & \vdots \\ Codev(\mathbf{X_k}, \mathbf{X_1}) & Codev(\mathbf{X_k}, \mathbf{X_2}) & \cdots & Dev(\mathbf{X_k}) \end{pmatrix}$$

Se $\mathbf{S_X}$ è la matrice le cui colonne sono gli scarti di ogni variabile di $\mathbf{X}$ dalla rispettiva media si può facilmente verificare che la matrice di Devianza-Codevianza, è data da

$$\mathbf{C} = \mathbf{S_X}^t \times \mathbf{S_X}$$

Se $\mathbf{X}$ contiene una sola variabile la relazione precedente ci dà la sua Devianza (come abbiamo già visto).
Se vogliamo utilizzare la forma computazionale, possiamo, anche in questo caso, estendere quanto già verificato per una sola variabile

$$\mathbf{C} = \mathbf{X}^t \times \mathbf{X} - \bar{\mathbf{X}}^t \times \bar{\mathbf{X}}$$

In cui $\bar{\mathbf{X}}$ è una matrice ogni colonna della quale è un vettore di costanti tutte uguali date dalla media della colonna corrispondente di $\mathbf{X}$.

Ecco un esempio di come si creano in R le matrici dei dati, la matrice di devianza-codevianza, la matrice di varianza-covarianza e la matrice di correlazione. L'esempio è con una matrice di dati di sole due colonne, ma il computo non cambia con matrici più grandi.

```
> ##Matrice dei dati
> y <-c(202, 175, 204, 96, 159, 37, 128, 222, 202)
> x<- c(0.30, 0.23, 0.29, 0.21, 0.29, 0.09, 0.08, 0.37, 0.28)
> X <- cbind(y,x)
> X
        y    x
 [1,] 202 0.30
 [2,] 175 0.23
 [3,] 204 0.29
 [4,]  96 0.21
 [5,] 159 0.29
 [6,]  37 0.09
 [7,] 128 0.08
 [8,] 222 0.37
 [9,] 202 0.28

> ##Vettore colonna di tanti '1' quanti sono i dati
> uno <-cbind(rep(1,length(y)))
> uno
      [,1]
 [1,]    1
 [2,]    1
 [3,]    1
 [4,]    1
 [5,]    1
 [6,]    1
 [7,]    1
 [8,]    1
 [9,]    1
>
> ##Computo di n
> n <- as.vector( t(uno) %*% uno)
> n
[1] 9
>
> ##Vettore riga delle medie per colonna
```

```
> m <- colMeans(X)
> m
          y           x
158.3333333   0.2377778
>
> Xm <- (uno %*% m)
> Xm
          [,1]      [,2]
 [1,] 158.3333 0.2377778
 [2,] 158.3333 0.2377778
 [3,] 158.3333 0.2377778
 [4,] 158.3333 0.2377778
 [5,] 158.3333 0.2377778
 [6,] 158.3333 0.2377778
 [7,] 158.3333 0.2377778
 [8,] 158.3333 0.2377778
 [9,] 158.3333 0.2377778
>
> ## oppure  anche
> Xm <- cbind(rep(mean(y),n),rep(mean(x),n))
> Xm
          [,1]      [,2]
 [1,] 158.3333 0.2377778
 [2,] 158.3333 0.2377778
 [3,] 158.3333 0.2377778
 [4,] 158.3333 0.2377778
 [5,] 158.3333 0.2377778
 [6,] 158.3333 0.2377778
 [7,] 158.3333 0.2377778
 [8,] 158.3333 0.2377778
 [9,] 158.3333 0.2377778
>
> ##Matrice degli scarti dalla media
> S <- X - Xm
> S
                 y            x
 [1,]   43.6666667  0.062222222
 [2,]   16.6666667 -0.007777778
 [3,]   45.6666667  0.052222222
 [4,]  -62.3333333 -0.027777778
 [5,]    0.6666667  0.052222222
```

```
 [6,] -121.3333333 -0.147777778
 [7,]  -30.3333333 -0.157777778
 [8,]   63.6666667  0.132222222
 [9,]   43.6666667  0.042222222
>
> ##Matrice di devianza - codevianza
>
> C <-t(S) %*% S
> C
            y           x
y 29758.00000 39.71666667
x    39.71667  0.07615556
>
> ##oppure anche
> (t(X) %*% X) - (t(Xm) %*% Xm)
            y           x
y 29758.00000 39.71666667
x    39.71667  0.07615556
>
>
> ## Matrice di varianza-covarianza
>
> V<-(t(S) %*% S)/(n - 1)
> V
            y           x
y 3719.750000 4.964583333
x    4.964583 0.009519444
>
> #controllo che sia uguale a
> cov(X)
            y           x
y 3719.750000 4.964583333
x    4.964583 0.009519444
> var(y)
[1] 3719.75
> var(x)
[1] 0.009519444
> cov(x,y)
[1] 4.964583
>
> ##vettore delle deviazioni standard
```

```
> d <-apply(X,2,sd)
> d
          y           x
60.98975324  0.09756764
>
> ##Matrice diagonale con l'inverso delle deviazioni standard
> D <-diag(1/d)
> D
          [,1]    [,2]
[1,] 0.0163962  0.0000
[2,] 0.0000000 10.2493
>
> ## Matrice di correlazione
> (t(S %*% D) %*%  (S %*% D)) /(n-1)
          [,1]      [,2]
[1,] 1.0000000 0.8342959
[2,] 0.8342959 1.0000000
>
> D %*% V %*% D
          [,1]      [,2]
[1,] 1.0000000 0.8342959
[2,] 0.8342959 1.0000000
>
> cor(X)
          y         x
y 1.0000000 0.8342959
x 0.8342959 1.0000000
```

4.1.6 Esempi ed esercizi per voi

1. Eseguire manualmente i prodotti di matrici AB e BA dove

$$\mathbf{A} = \begin{pmatrix} 2 & -1 \\ 1 & 0 \\ -3 & 4 \end{pmatrix}$$

e

$$\mathbf{B} = \begin{pmatrix} 1 & -2 & 5 \\ 3 & 4 & 0 \end{pmatrix}$$

Riportare tutti i passaggi e infine verificare la correttezza del risultato con R .

2. Verificare se la seguente matrice ha vettori linearmente dipendenti e dire perchè, specificando quali vettori sono dipendenti.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

4.2 Il modello lineare generale

La regressione lineare

Per poter seguire quanto esposto di seguito è consigliabile rileggere i capitoli sulla regressione e sull'analisi della varianza (almeno ad un criterio di classificazione) di un buon libro di statistica generale

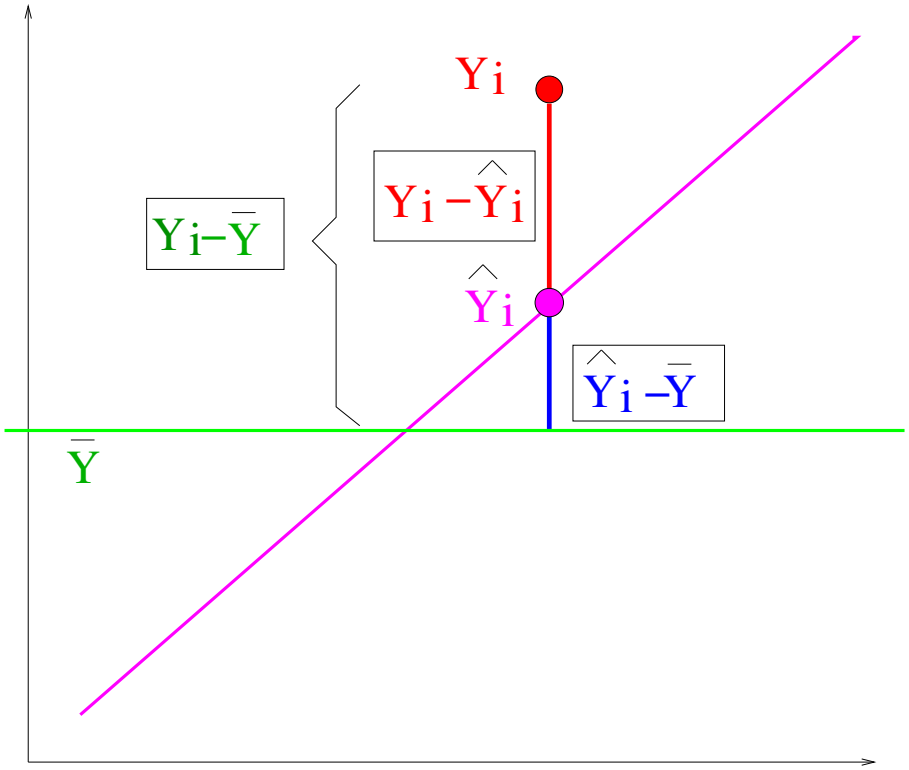


Grafico: scomposizione della varianza nella regressione

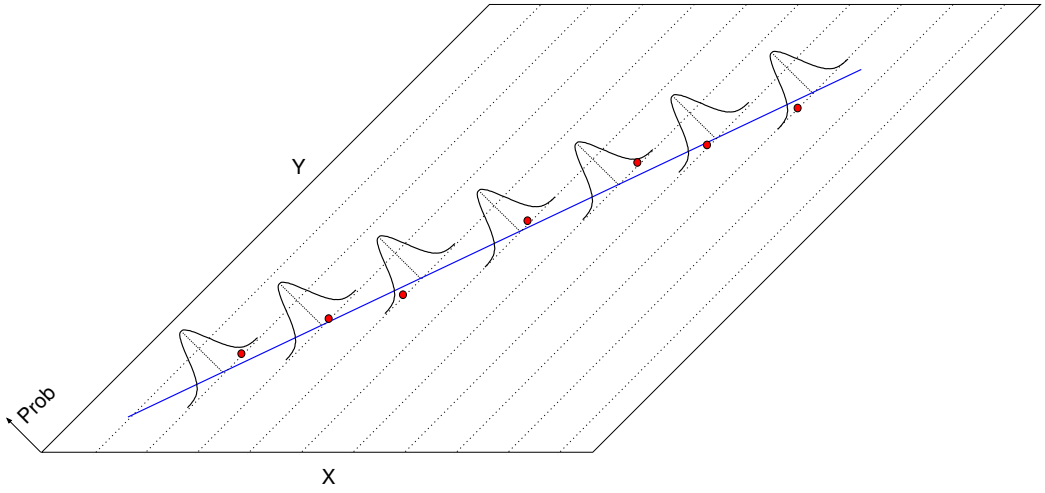


Grafico del modello statistico sottostante ad una regressione lineare

I minimi quadrati

Consideriamo una regressione lineare bivariata. Data una varibile dipendente (stocastica, normalmente distribuita, contenente n misure)

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix}$$

ed una varibile indipendente

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{pmatrix}$$

la variabile

$$\hat{\mathbf{Y}} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_i \\ \vdots \\ \hat{y}_n \end{pmatrix}$$

conterrà i valori attesi di $\mathbf{Y}$ calcolati tramite la regressione

$$\hat{\mathbf{Y}} = \hat{\beta}_0 + \hat{\beta}_1 \mathbf{X}$$

il cui modello é

$$\mathbf{Y} = \beta_0 + \beta_1 \mathbf{X} + \epsilon$$

che corrisponde a:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \vdots + \vdots \\ \beta_0 + \beta_1 x_i \\ \vdots + \vdots \\ \beta_0 + \beta_1 x_n \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_i \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{4.2}$$

ovvero a

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_i \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \times \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_i \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{4.3}$$

o, per la singola osservazione

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

ed il *valore vero (o valore dell'universo)* di y_i è dato da

$$\mu_i = \beta_0 + \beta_1 x_i$$

$\hat{\beta}_0$ e $\hat{\beta}_1$ sono rispettivamente i valori stimati dell'intercetta ed del coefficiente angolare della retta cui appartengono gli $\hat{y}_i$ e sono le **stime** di β_0 e β_1 *i valori veri (o valori dell'universo che non sono noti)* dei parametri ottenute con il metodo dei minimi quadrati che consiste nell'imporre che sia minima la somma dei quadrati degli errori ϵ_i, scarti tra i *valori veri* μ_i (μ_i é la media della distribuzione cui appartiene y_i) e i valori osservati y_i.

Partendo dal modello appena visto scriviamo le relazioni

$$\begin{gathered} \epsilon_1 = \beta_0 + \beta_1 x_1 - y_1 \\ \epsilon_2 = \beta_0 + \beta_1 x_2 - y_2 \\ \vdots \\ \epsilon_i = \beta_0 + \beta_1 x_i - y_i \\ \vdots \\ \epsilon_n = \beta_0 + \beta_1 x_n - y_n \end{gathered}$$

Eleviamo al quadrato e sommiamo membro a membro

$$\begin{gathered} \epsilon_1^2 = (\beta_0 + \beta_1 x_1 - y_1)^2 \\ \epsilon_2^2 = (\beta_0 + \beta_1 x_2 - y_2)^2 \\ \vdots \\ \epsilon_i^2 = (\beta_0 + \beta_1 x_i - y_i)^2 \\ \vdots \\ \epsilon_n^2 = (\beta_0 + \beta_1 x_n - y_n)^2 \end{gathered}$$

$$\sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)^2$$

Imponiamo che le derivate parziali rispetto a β_0 ed β_1 siano nulle [26]

$$\frac{\partial \sum_{i=1}^{n} \epsilon_i^2}{\partial \beta_0} = 2\sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i) = 2\left(n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i\right) = 0$$

$$\frac{\partial \sum_{i=1}^{n} \epsilon_i^2}{\partial \beta_1} = 2\sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)\, x_i = 2\left(\beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i y_i\right) = 0$$

Risolviamo il sistema di due equazioni algebriche lineari in β_0 e β_1 così ottenuto.

$$\begin{cases} n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i & = & \sum_{i=1}^{n} y_i \\ \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 & = & \sum_{i=1}^{n} x_i y_i \end{cases}$$

Procediamo per sostituzione ricavando β_0 dalla prima equazione

$$\begin{cases} \beta_0 = \dfrac{\sum_{i=1}^{n} y_i}{n} - \beta_1 \dfrac{\sum_{i=1}^{n} x_i}{n} \\ \left(\dfrac{\sum_{i=1}^{n} y_i}{n} - \beta_1 \dfrac{\sum_{i=1}^{n} x_i}{n}\right) \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases}$$

$$\begin{cases} \beta_0 = \dfrac{\sum_{i=1}^{n} y_i}{n} - \beta_1 \dfrac{\sum_{i=1}^{n} x_i}{n} \\ \dfrac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n} - \beta_1 \dfrac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i}{n} + \beta_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases}$$

[26] Ricordo che la derivata di $(a+bx+y)^2$ rispetto ad a è uguale a $2(a+bx+y)$ e rispetto a b è uguale a $2x(a+bx+y)$

Ora ricaviamo β_1:

$$\begin{cases} \beta_0 = \dfrac{\sum_{i=1}^{n} y_i}{n} - \beta_1 \dfrac{\sum_{i=1}^{n} x_i}{n} \\ \beta_1 \left(\sum_{i=1}^{n} x_i^2 - \dfrac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i}{n} \right) = \sum_{i=1}^{n} x_i y_i - \dfrac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n} \end{cases}$$

$$\begin{cases} \hat{\beta}_0 = \dfrac{\sum_{i=1}^{n} y_i}{n} - \hat{\beta}_1 \dfrac{\sum_{i=1}^{n} x_i}{n} \\ \hat{\beta}_1 = \dfrac{\sum_{i=1}^{n} x_i y_i - \dfrac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}}{\sum_{i=1}^{n} x_i^2 - \dfrac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i}{n}} \end{cases}$$

Ricordando che

$$\sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n} = \mathbf{Codev}(\mathbf{X}, \mathbf{Y})$$

$$\sum_{i=1}^{n} x_i^2 - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i}{n} = \mathbf{Dev}(\mathbf{X})$$

abbiamo

$$\hat{\beta}_1 = \frac{\mathbf{Codev}(\mathbf{X}, \mathbf{Y})}{\mathbf{Dev}(\mathbf{X})} = \frac{\frac{\mathbf{Codev}(\mathbf{X}, \mathbf{Y})}{n}}{\frac{\mathbf{Dev}(\mathbf{X}, \mathbf{Y})}{n}} = \frac{\mathbf{Covar}(\mathbf{X}, \mathbf{Y})}{\mathbf{Var}(\mathbf{X})} \tag{4.4}$$

È immediato anche notare che $\hat{\beta}_0 = \bar{\mathbf{Y}} - \hat{\beta}_1 \bar{\mathbf{X}}$. Ricordiamo che con β_0, β_1 abbiamo indicato i *valori veri* dei parametri e le incognite del sistema di equazioni dei minimi quadrati, mentre con $\hat{\beta}_0, \hat{\beta}_1$ abbiamo indicato i **valori stimati** dei parametri.

Equazioni normali

Minimi quadrati e algebra delle matrici

Il sistema di equazioni ricavato nel paragrafo precedente è chiamato sistema di **equazioni normali** . Riscriviamole qui:

$$\begin{cases} n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i & = & \sum_{i=1}^{n} y_i \\ \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 & = & \sum_{i=1}^{n} x_i y_i \end{cases}$$

e riscriviamolo anche in forma matriciale

$$\begin{pmatrix} n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i \\ \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

Cambiamo (per economia di scrittura) il nome delle incognite: al posto di β_0 avremo b_0 e al posto di β_1, b_1 Costruiamo ora **il vettore colonna delle incognite**

$$\mathbf{b_{2,1}} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$$

e la **matrice aumentatata della variabile indipendente** aggiungendo una colonna di 1 al vettore che contiene i valori della variabile indipendente

$$\mathbf{X_{n,2}} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_i \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

La sua trasposta è

$$\mathbf{X^t_{2,n}} = \begin{pmatrix} 1 & 1 & \cdots & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_i & \cdots & x_n \end{pmatrix}$$

Eseguiamo ora il prodotto

$$\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}} =$$

$$= \begin{pmatrix} 1 \times 1 + 1 \times 1 + \cdots + 1 \times 1 & 1 \times x_1 + 1 \times x_2 + \cdots + 1 \times x_i + \cdots + 1 \times x_n \\ x_1 \times 1 + x_2 \times 1 + \cdots + x_i \times 1 + \cdots + x_n \times 1 & x_1 \times x_1 + x_2 \times x_2 + \cdots + x_i \times x_i + \cdots + x_n \times x_n \end{pmatrix} =$$

$$= \begin{pmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{pmatrix}$$

Se moltiplichiamo il risultato ottenuto per $\mathbf{b_{n,1}}$

$$\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}} \times \mathbf{b_{2,1}} =$$

$$= \begin{pmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{pmatrix} \times \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} b_0 n + b_1 \sum_{i=1}^{n} x_i \\ b_0 \sum_{i=1}^{n} x_i + b_1 \sum_{i=1}^{n} x_i^2 \end{pmatrix}$$

Abbiamo cosí espresso come prodotto di matrici la "parte sinistra" del nostro sistema di equazioni lineari. Vediamo ora di fare lo stesso con la "parte destra". Eseguiamo il prodotto

$$\mathbf{X^t_{2,n}} \times \mathbf{Y_{n,1}} = \begin{pmatrix} 1 & 1 & \cdots & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_i & \cdots & x_n \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} =$$

$$= \begin{pmatrix} 1 \times y_1 + 1 \times y_2 + \cdots + 1 \times y_i + \cdots + 1 \times y_n \\ x_1 \times y_1 + x_2 \times y_2 + \cdots + x_i \times y_i + \cdots + x_n \times y_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{pmatrix}$$

Abbiamo ottenuto anche la "parte destra" del nostro sistema. Ora possiamo scriverlo come prodotto di matrici

$$\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}} \times \mathbf{b_{2,1}} = \mathbf{X^t_{2,n}} \times \mathbf{Y_{n,1}}$$

******* Supponiamo che il sistema ammetta soluzione, che la soluzione sia unica e che pertanto esista la matrice

$$\left(\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}}\right)^{-1}$$

cioè che esista la **matrice inversa** del **prodotto** tra la **trasposta della matrice aumentata dei dati** e la **matrice aumentata dei dati**.

Moltiplichiamo a destra entrambi i termini del nostro sistema per questa matrice.

$$\left(\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}}\right)^{-1} \times \left(\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}}\right) \times \mathbf{b_{2,1}} = \left(\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}}\right)^{-1} \times \mathbf{X^t_{2,n}} \times \mathbf{Y_{n,1}}$$

Ricordando le proprietà della matrice inversa si ha

$$\mathbf{b_{2,1}} = \left(\mathbf{X^t_{2,n}} \times \mathbf{X_{n,2}}\right)^{-1} \times \mathbf{X^t_{2,n}} \times \mathbf{Y_{n,1}}$$

Il sistema è stato risolto, determinando il vettore $\mathbf{b_{2,1}}$ che come sappiamo contiene l'**intercetta** b_0 (in molti testi chiamata a) ed il **coefficiente angolare** della regressione b_1 (in molti testi chiamato b). Abbiamo ricavato i coefficienti di una regressione lineare semplice (un sola variabile indipendente).

Regressione Multipla

I calcoli con le matrici ricavati nella sezione precedenti hanno alcuni vantaggi rispetto ai calcoli effettuati con le cosiddette **equazioni normali** (pag: 141):

- permettono una notazione più concisa e semplice
- sono simili a quelli effettivamente usati. I principali software statistici usano infatti un approccio matriciale ottimizzato per velocità ed efficenza per effettuare i calcoli della regressione e dell'ANOVA
- il modello matriciale è più generale e generalizzabile rispetto alle equazioni normali

Infatti i può facilmente verificare che tutti i ragionamenti fatti sono ugualmente validi per un qualunque numero p di varibili indipendenti. Se scriviamo la **matrice aumentata dei dati** ed il **vettore dei coefficienti** nella loro forma più generale

$$\mathbf{X_{n,p}} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1p-1} \\ 1 & x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{ip-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{np-1} \end{pmatrix} =$$

$$= \begin{pmatrix} \mathbf{U} & \mathbf{X_1} & \mathbf{X_2} & \cdots & \mathbf{X_j} & \cdots \mathbf{X_p - 1} \end{pmatrix}$$

dove $\mathbf{U}$ è un vettore colonna di tutti 1.

$$\mathbf{b_{p,1}} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_j \\ \vdots \\ b_p \end{pmatrix}$$

La relazione che esprime $\mathbf{b_{p,1}}$ diventa

$$\mathbf{b_{p,1}} = \left(\mathbf{X^t_{p+1,n}} \times \mathbf{X_{n,p}}\right)^{-1} \times \mathbf{X^t_{p,n}} \times \mathbf{Y_{n,1}}$$

che contiene (finalmente) le stime dei parametri della regressione lineare multipla. Abbiamo indicato esplicitamente l'ordine delle matrici per rendere più comprensibili i prodotti.
Possiamo osservare che, utilizzando la notazione matriciale introdotta in questo paragrafo (e tralasciando l'ordine delle matrici) il modello della regressione lineare può essere scritto nella forma

$$\mathbf{Y} = \mathbf{Xb} + \epsilon \tag{4.5}$$

Analogamente per i valori attesi avremo:

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{b}} \tag{4.6}$$

che è il modo usuale in cui si scrive un sistema di equazioni lineari utilizzando le matrici.

Un programma in R per la regressione lineare

```
## I dati vengono inseriti
y <-c(202, 175, 204, 96, 159, 37, 128, 222, 202)
x<- c(0.30, 0.23, 0.29, 0.21, 0.29, 0.09, 0.08, 0.37, 0.28)

## stampo i dati
print(cbind(x,y))

## Il numero dei casi
n <- length(x)
n
```

```
## Costruzione della  matrice aumentata
X <- cbind(1, x)
X

## Il numero dei parametri da stimare (compresa l'intercetta)
p <- ncol(X)
p

## Calcolo dei coefficienti della regressione (vettore  b )
b <- solve(t(X) %*% X) %*% t(X) %*% y
b

## Controllo che siano uguali a quelli di lm
r.lm <- lm(y ~ x)
coef(r.lm)

## ..........................................................
## ANOVA sulla regressione

## Calcolo dei valori attesi di y, della media di y e dei residui
y.att <- X %*% b
y.att

y.m <- mean(y)
y.m

Res <- y - y.att
Res

## Calcolo della devianza dovuta alla (spiegata dalla) regressione
Dev.Reg <- t(y.att - y.m) %*% (y.att - y.m)
Dev.Reg

## Calcolo della varianza dovuta alla (spiegata dalla) regressione
Var.Reg <- Dev.Reg/(p-1)
Var.Reg

## Calcolo della devianza d'errore
Dev.Err <- t(y - y.att) %*% (y - y.att)
Dev.Err
```

```
## Calcolo della varianza d'errore
Var.Err <- Dev.Err/(n - p)
Var.Err

## Calcolo della devianza TOTALE
Dev.Tot <- t(y - y.m) %*% (y - y.m)
Dev.Tot

## Calcolo dell'R-quadrato
R2 <- Dev.Reg/Dev.Tot
R2

## Calcolo dell'F di Fisher
F <- Var.Reg/Var.Err
F

## Calcolo del P dell'anova sulla regressione
PF <-1- pf(F, p-1, n-p)
PF

## Controllo con l'anova di R
anova(r.lm)

## ..........................................................

## Calcolo dell'Errore standard dei coefficienti
Es <- sqrt(diag(solve(t(X) %*% X)) * Var.Err)
Es

## Calcolo delle t di Student relative all' ipotesi nulla H0: b=0
t <- b/Es
t

## Calcolo della "Pr(>|t|)"
Pt <-2 *pt(abs(t),n-p, lower.tail=FALSE)
Pt

##controllo con summary.lm di R
summary(r.lm)

## ..........................................................
```

```
## GRAFICI
par(mfrow=c(2,1))
plot(x,y, pch=19, col= "blue")
abline(a=b[1],b=b[2], col="green")
points(x,y.att, pch = 21, bg = "aquamarine")

hist(Res, col = "aliceblue")
## ..........................................................
```

La regressione con variabili dummy

Il modello della regressione, nella sua forma generale (regressione multipla), espresso come prodotto di matrici, è

$$\mathbf{Y}_{n,1} = \mathbf{X}_{n,p} \times \boldsymbol{\beta}_{p,1} + \boldsymbol{\epsilon}_{n,1}$$

cioè

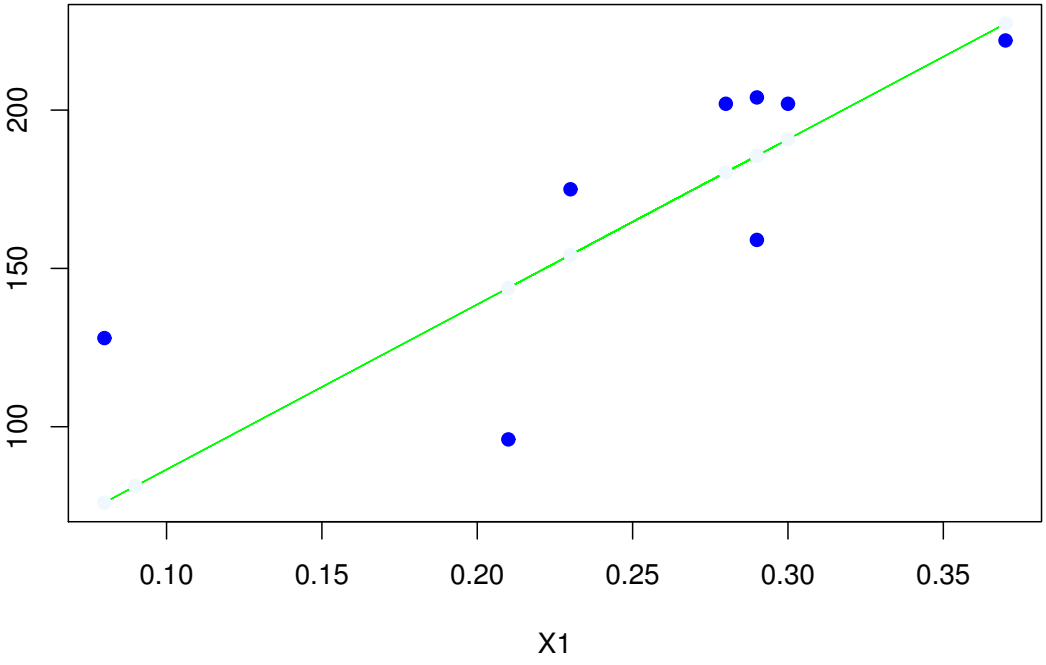
$$\mathbf{Y}_{n,1} = \beta_0 + \sum_{j=1}^{p-1} \beta_j \mathbf{X}_j + \boldsymbol{\epsilon}_{n,1}$$

dove p è il numero di parametri da stimare.

Per quanto riguarda la stima dei parametri le variabili indipendenti non sono soggette ad alcuna restrizione. Devono solo, ovviamente, essere numeriche e su scala almeno ordinale. Nulla vieta di utlizzare variabili binarie.

Esempio di un'ANOVA con le cosiddette *dummy variables*

```
## Anova con le variabili dummy
rm(list=ls())
##carico il file dei dati
e.df <- read.table("../exp1.txt", h=T)
e.df

##Ctrl sarà un vettore con 1 nelle righe dove ci sono i dati relativi
##al gruppo di controllo e 0 nelle altre righe
Ctrl <- cbind(as.numeric(as.character(e.df$group) == "Ctrl"))
cbind(as.character(e.df$group),Ctrl)

##Trt1 sarà un vettore con 1 nelle righe dove ci sono i dati relativi
##al gruppo Trt1 e 0 nelle altre righe
```



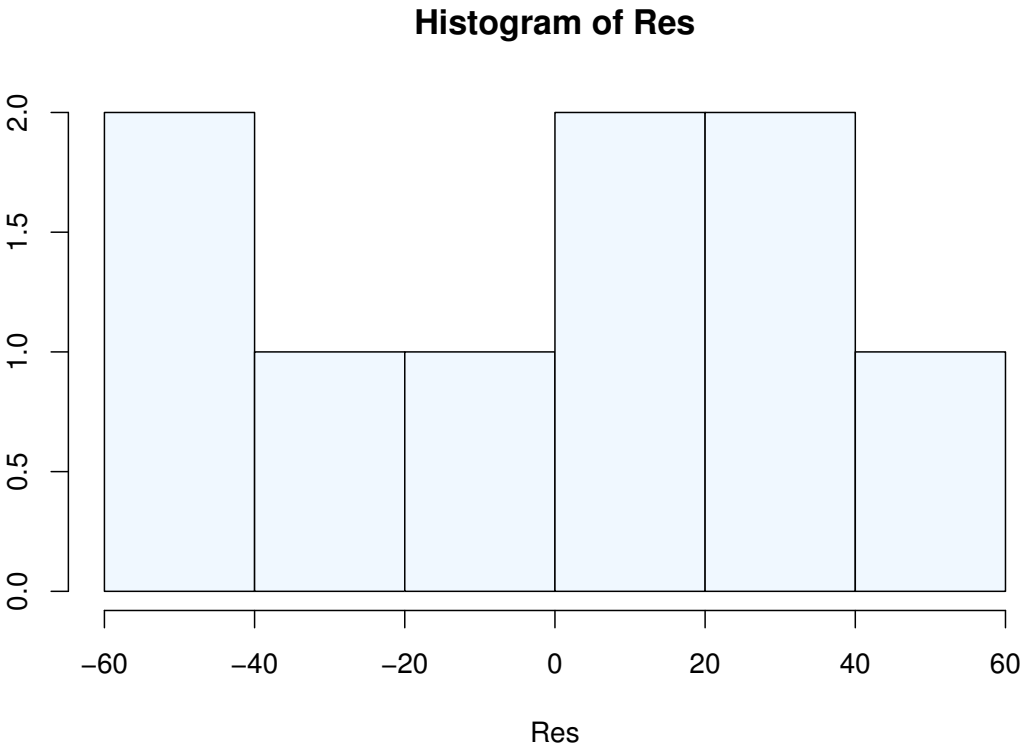


Figura 4.3: Grafico

```
Trt1 <- cbind(as.numeric(as.character(e.df$group) == "Trt1"))
Trt1

##Trt2 sarà un vettore con 1 nelle righe dove ci sono i dati relativi
##al gruppo Trt2 e 0 nelle altre righe
Trt2 <- cbind(as.numeric(as.character(e.df$group) == "Trt2"))
Trt2
X <- cbind(1, Ctrl, Trt1, Trt2)
X
t(X) %*% X
det(t(X) %*% X)
##la matrice è singolare e non si può invertire
##è sufficiente eliminare una colonna. R elimina la prima in ordine alfabetico
## cioè  Ctrl

X <- cbind(1, Trt1, Trt2)
det(t(X) %*% X)
b <-solve(t(X) %*% X) %*% t(X) %*% e.df$weight
b

## usando lm
m.lm <-lm(weight ~ group, data=e.df)
coef(m.lm)
##sono uguali a b

## la matrice aumentata usata da lm si ottiene con il comando model.matrix
model.matrix(m.lm)

## ..........................................................
## ANOVA

##per brevità assegno e.df$weight a y
y <- e.df$weight

##Quanti dati?
n <- length(y)
##Quanti parametri da stimare?
p <- ncol(X)

## Calcolo dei valori attesi di y, della media di y e dei residui
y.att <- X %*% b
```

```
y.att

y.m <- mean(y)
y.m

Res <- y - y.att
Res

## Calcolo della devianza tra
Dev.Reg <- t(y.att - y.m) %*% (y.att - y.m)
Dev.Reg

## Calcolo della varianza tra
Var.Reg <- Dev.Reg/(p-1)
Var.Reg

## Calcolo della devianza d'errore (entro)
Dev.Err <- t(y - y.att) %*% (y - y.att)
Dev.Err

## Calcolo della varianza d'errore (entro)
Var.Err <- Dev.Err/(n - p)
Var.Err

## Calcolo della devianza TOTALE
Dev.Tot <- t(y - y.m) %*% (y - y.m)
Dev.Tot

## Calcolo dell'R-quadrato
R2 <- Dev.Reg/Dev.Tot
R2

## Calcolo dell'F di Fisher
F <- Var.Reg/Var.Err
F

## Calcolo del P dell'anova sulla regressione
PF <-1- pf(F, p-1, n-p)
PF

## Controllo con l'anova di R
```

```
anova(m.lm)

## ..........................................................

## Calcolo dell'Errore standard dei coefficienti
Es <- sqrt(diag(solve(t(X) %*% X)) * Var.Err)
Es

## Calcolo delle t di Student relative all' ipotesi nulla H0: b=0
t <- b/Es
t

## Calcolo della "Pr(>|t|)")
Pt <-2 *pt(abs(t),n-p, lower.tail=FALSE)
Pt

##controllo con summary.lm di R
summary(m.lm)
```

Il modello che abbiamo ottenuto partendo dalla regressione lineare multipla è molto versatile e permette di trattare un grande numero di problemi statistici. È per questo che lo si chiama modello lineare, modello statistico lineare, modello lineare generale (general linear model), da non confondere con modello lineare generalizzato (generalized linear model). Ricordiamo che il modello che stiamo considerando è lineare sia nei parametri che nelle variabili.

I principali tipi di analisi che si possono condurre (i diversi problemi di inferenza che si possono risolvere) stimando i parametri di un modello lineare sono:

- Analisi della Regressione lineare multipla:

  Le p variabili indipendenti sono di tipo razionale, intervallare o ordinale (non binario).

- Analisi della Varianza:

  Le p variabili indipendenti sono tutte dummy (binarie). L'argomento è complesso e verra' trattato separatamente.

- Analisi della Covarianza:

Le p variabili indipendenti sono in parte dummy (binarie e in parte di tipo razionale, intervallare o ordinale (non binario).

4.2.1 Esempio di semplifiazione di un modello di regressione multipla

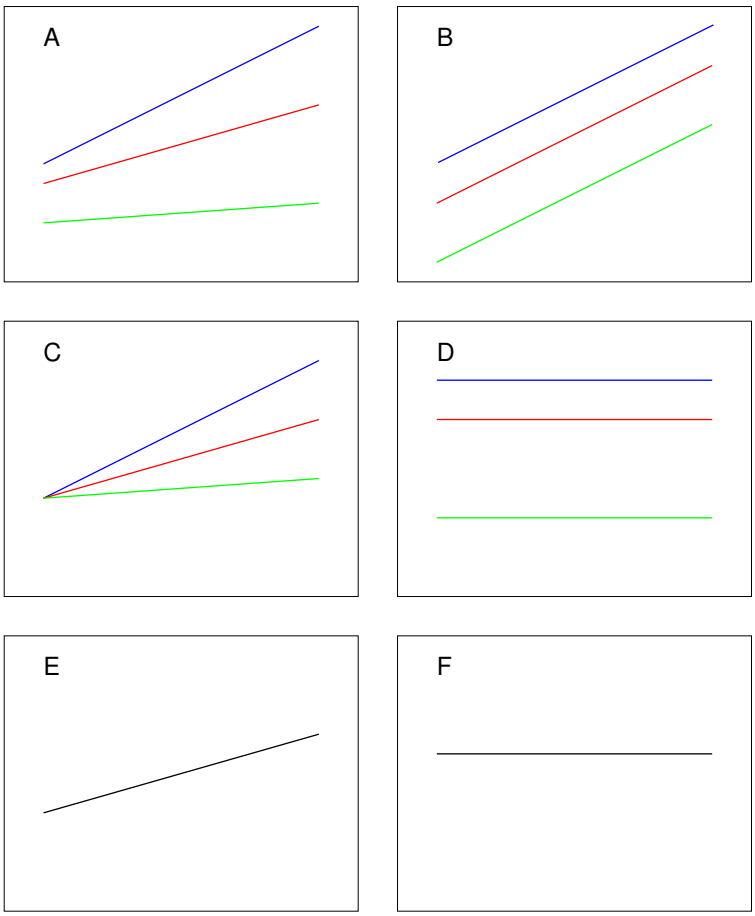
```
p.df <- read.table("../crawley/Pollute.txt", h=T)
## dapprima provo un modello senza interazioni

k.lm <- lm(Pollution ~ Temp + Industry + Population + Wind + Rain + Wet.days,
data=p.df)
summary(k.lm)
k1.lm <-update(k.lm,~. - Wet.days)
summary(k1.lm)
k2.lm <-update(k.lm,~. - Rain)
summary(k2.lm)
k3.lm <-update(k2.lm,~. - Wet.days)
summary(k3.lm)
k4.lm <-update(k3.lm,~. - Wind)
summary(k4.lm)
k5.lm <-update(k4.lm,~. - Temp)
summary(k5.lm)
anova(k.lm,k5.lm)
anova(k.lm,k1.lm,k2.lm,k3.lm,k4.lm,k5.lm)
drop1(k5.lm,test="F")
drop1(k3.lm,test="F")
drop1(k4.lm,test="F")
drop1(k.lm,test="F")
drop1(k5.lm,test="F")
summary(k5.lm)

## provo con tutte le interazioni
m.lm <- lm(Pollution ~ Temp*Industry*Population* Wind *Rain *Wet.days, data=p.df
summary(m.lm)
anova(m.lm)
## troppi parametri da stimare rispetto ai dati che abbiamo
## proviamo ad usare al massimo fino alle interazioni triple
m.lm <- lm(Pollution ~ (Temp+ Industry + Population + Wind + Rain + Wet.days)^3,
data=p.df)
```

```
summary(m.lm)
## troppi parametri da stimare rispetto ai dati che abbiamo
## limitiamoci a usare al massimo doppie interazioni
m.lm <- lm(Pollution ~ (Temp+ Industry + Population + Wind + Rain + Wet.days)^2,
data=p.df)

summary(m.lm)
m1.lm <- update(m.lm, ~. - Rain:Wet.days)
summary(m1.lm)
anova(m1.lm, m.lm)
m2.lm <- update(m1.lm, ~. - Wind:Wet.days)
summary(m2.lm)
m3.lm <- update(m2.lm, ~. - Population:Wet.days)
summary(m3.lm)
m4.lm <- update(m3.lm, ~. - Population:Rain)
summary(m4.lm)
anova(m4.lm,m3.lm)
m5.lm <- update(m4.lm, ~. - Industry:Wet.days)
summary(m5.lm)
m6.lm <- update(m5.lm, ~. - Industry:Rain)
summary(m6.lm)
anova(m5.lm,m6.lm)
m7.lm <- update(m6.lm, ~. - Industry:Population)
summary(m7.lm)
m8.lm <- update(m7.lm, ~. - Temp:Wet.days)
summary(m8.lm)
anova(m5.lm,m6.lm,m7.lm,m8.lm)
summary(m8.lm)
m9.lm <- update(m8.lm, ~. - Temp:Rain)
summary(m9.lm)
anova(m8.lm,m9.lm)
m10.lm <- update(m9.lm, ~. - Temp:Population)
summary(m10.lm)
m11.lm <- update(m10.lm, ~. - Temp:Industry)
summary(m11.lm)
m12.lm <- update(m11.lm, ~. - Temp:Wet.days)
summary(m12.lm)
m13.lm <- update(m11.lm, ~. - Wet.days)
summary(m13.lm)

#provo un analisi stepwise
```

```
step(m.lm)
```

4.2.2 Analsi della Covarianza

Nell'analisi della Covarianza (ANCOVA) si utilizza un modello dove si hanno contemporaneamente sia variabili indipendenti continue (del tipo di quelle della regressione), sia variabili classificatorie (fattori) o variabili dummy. Gia con solo una variabile continua e un fattore i modelli testabili in realtà diventano sei (Figura 4.2.2). Possiamo testare se la relazione lineare (regressione) che lega la y alla x è significativa e unica (cioè la stessa) per tutti i fattori (nella figura rappresentati dai diversi colori).



A La relazione fra x e y è significativa e i gruppi (rosso, blu e verde) hanno sia la pendenza sia l'intercetta che differiscono tra loro. La relazione fra x e y è quindi diversa per ogni gruppo. L'interazione fattore $\times$ variabile continua è significativa.

B (*Common slope*) La relazione fra x e y è significativa, ma i gruppi (rosso, blu e verde) hanno solo l'intercetta diversa, mantre la pendenza è uguale per tutti i gruppi. L'effetto di x è costante per tutti i gruppi, ma i gruppi differiscono uno dall'altro. La differenza fra i gruppi è costante per qualsiasi valore di x. L'interazione fattore $\times$ variabile continua non è significativa e viene tolta dal modello.

C La relazione fra x e y è significativa e i gruppi hanno solo la pendenza diversa, mantre l'intercetta è uguale per tutti i gruppi. Notare che questo modello viola il **Principio di Marginalità** in quanto toglie la variabile singola (gruppi) che però fa parte della interazione (gruppi $\times x$). Si usa in rari casi.

D La relazione fra x e y non è significativa, ma i gruppi hanno medie di y diverse uno dall'altro.

E La relazione fra x e y è significativa, ma non ci sono differenze fra i gruppi né nell'intercetta, né nella pendenza.

F La relazione fra x e y non è significativa e i gruppi non differiscono uno dall'altro.

Il caso **C** viene raramente usato. Il caso **A** viene testato introducendo nel modello l'interazione fra fattore e variabile continua.

Per esempio in R se chiamiamo `y` il vettore della variabile dipendente, `x` quello della variabile continua indipendente e `g` il fattore, i sei modelli possono essere testati partendo da più complicato e semplificando:

```
#A
lm(y  ~ g * x)

#B
lm(y  ~ g + x)

#C questo però viola il principio di Marginalità
lm(y  ~ x + g:x)

#D
lm(y  ~ g)

#E
lm(y  ~ x)

#F
lm(y ~ 1)
```

i risultati possono essere esaminati con `summary` o con `anova` o, meglio, con `anova(modello.complicato, modello.semplice)`.

Ecco un esempio per una analisi della covarianza nel caso del dataset `whiteside` in cui si esamina il consumo di gasolio prima e dopo in intervento di isolamento dell'edificio e in fuzione della temperatura media giornaliera.

Notare che importando questo dataset con `data()` il livello di riferimento per il fattore `Insul` non è come al solito il primo livello in ordine alfabetico (che in questo caso sarebbe `After` bensì il livello `Before`. Se vogliamo possiamo cambiare il livello di riferimento per i contrasti di default (`contr.treatment`) con il comando `relevel`

```
library(MASS)
data(whiteside)
summary(whiteside)
levels(whiteside$Insul)
## il primo livello è "Before". Sarà il livello di riferimento.
```

```
## se voglio cambio il livello di riferimento ad "After"
# whiteside$Insul <-relevel(whiteside$Insul,"After")
# levels(whiteside$Insul)
## ora il primo livello è "After" e sarà il riferimento

ma.lm <- lm(Gas ~ Insul * Temp, data=whiteside)
summary(ma.lm)
mb.lm <- lm(Gas ~ Insul + Temp, data=whiteside)
summary(mb.lm)
anova(ma.lm, mb.lm)
mc.lm <- lm(Gas ~ Temp + Insul:Temp, data=whiteside)
summary(mc.lm)
#possibile plot del modello "C"
#plot(Gas ~ Temp, pch=16, col=whiteside$Insul, data=whiteside)
#abline(5.63978, -0.21557)
#abline(5.63978, -0.21557-0.21640, col="red")
#abline(v=0, lty=3)

md.lm <- lm(Gas ~ Insul, data=whiteside)
summary(md.lm)
me.lm <- lm(Gas ~ Temp, data=whiteside)
summary(me.lm)
mf.lm <- lm(Gas ~ 1, data=whiteside)
summary(mf.lm)
################ Ora esaminiamo il modello corretto per questo caso
################ ovvero il modello A
################ N.B. non posso usare modelli più semplici perchè
################ l'interazione è significativa

summary(ma.lm)
anova(ma.lm)
model.matrix(ma.lm)

###interpretiamo i coefficienti

coef(ma.lm)
colori<-c("red", "blue")
names(colori)<-c("After", "Before")
colori
plot(Gas ~ Temp, pch=16, col=colori[as.character(whiteside$Insul)], data=whitesi
legend("topright",pch=16, lty=1, col=colori, legend=paste(names(colori), "insula
```

```
abline(coef(ma.lm)[c(1,3)],col=colori["Before"] )
##equivale a
##abline( 6.8538, -0.3932, col=colori["Before"])
abline(c(coef(ma.lm)[1]+coef(ma.lm)[2],coef(ma.lm)[3]+coef(ma.lm)[4]), col=color
##equivale a
##abline( 6.8538-2.1300, -0.3932+0.1153, col=colori["After"])

coef(ma.lm)

####Regressioni separate per After e Before
After.df <- subset(whiteside, as.character(Insul) == "After")
Before.df <- subset(whiteside, as.character(Insul) == "Before")

cat("--------------Before---------------\n")
b.lm <- lm(Gas ~ Temp, data=Before.df)
coef(b.lm)
##controllo che i coefficienti siano uguali al modello A precedente di
##ANCOVA
coef(ma.lm)[c(1,3)]


cat("--------------After----------------\n")
a.lm <- lm(Gas ~ Temp, data=After.df)
coef(a.lm)
c(coef(ma.lm)[1]+coef(ma.lm)[2],coef(ma.lm)[3]+coef(ma.lm)[4])
```

La funzione per il modello lineare generale in R

Finora abbiamo usato programmi e/o funzioni scritti ad hoc, ma R dispone di una funzione per il modello lineare generale molto potente e versatile: **lm**, la cui sintassi è:

```
lm(formula, data, subset, weights, na.action, method = "qr",
    model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
    singular.ok = TRUE, contrasts = NULL, offset = NULL, ...)
```

Proviamo ad applicarla ai modelli del paragrafo precedente.

```
> Dati <- read.table("regcherry.dat")
> Dati
     V1   V2   V3 V4   V5
1  14.0 34.5 >300  0  0.0
2  14.2 31.7 >300  0  0.0
3  14.5 36.3 >300  0  0.0
4  16.0 38.3 >300  0  0.0
5  16.3 42.6 >300  0  0.0
6  17.3 55.4 >300  0  0.0
7  17.5 55.7 >300  0  0.0
8  17.9 58.3 >300  0  0.0
9  18.0 51.5 >300  0  0.0
10 18.0 51.0 >300  0  0.0
11 11.3 24.2 <300  1 11.3
12 11.4 21.0 <300  1 11.4
13 11.4 21.4 <300  1 11.4
14 11.7 21.3 <300  1 11.7
15 12.0 19.1 <300  1 12.0
16 12.9 22.2 <300  1 12.9
17 12.9 33.8 <300  1 12.9
18 13.3 27.4 <300  1 13.3
19 13.7 25.7 <300  1 13.7
20 13.8 24.9 <300  1 13.8
> names(Dati) <- c("X", "Y", "A", "D", "XD")
> Dati
      X    Y    A D   XD
1  14.0 34.5 >300 0  0.0
2  14.2 31.7 >300 0  0.0
3  14.5 36.3 >300 0  0.0
4  16.0 38.3 >300 0  0.0
```

```
5  16.3 42.6 >300 0  0.0
6  17.3 55.4 >300 0  0.0
7  17.5 55.7 >300 0  0.0
8  17.9 58.3 >300 0  0.0
9  18.0 51.5 >300 0  0.0
10 18.0 51.0 >300 0  0.0
11 11.3 24.2 <300 1 11.3
12 11.4 21.0 <300 1 11.4
13 11.4 21.4 <300 1 11.4
14 11.7 21.3 <300 1 11.7
15 12.0 19.1 <300 1 12.0
16 12.9 22.2 <300 1 12.9
17 12.9 33.8 <300 1 12.9
18 13.3 27.4 <300 1 13.3
19 13.7 25.7 <300 1 13.7
20 13.8 24.9 <300 1 13.8
```

1.

$$\mathbf{Y}_{n,1} = \beta_0 + \beta_1 \mathbf{X} + \beta_2 \mathbf{D} + \boldsymbol{\epsilon}_{n,1}$$

```
> lm(Y~X+D, data=Dati)

Call:
lm(formula = Y ~ X + D, data = Dati)

Coefficients:
(Intercept)            X            D
    -32.321        4.756       -2.740
```

Un output un po' più ricco:

```
> summary(lm(Y~X+D, data=Dati))

Call:
lm(formula = Y ~ X + D, data = Dati)

Residuals:
    Min      1Q  Median      3Q     Max
-5.6677 -3.0582 -0.5634  2.8835  7.5124
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -32.3206    12.3932  -2.608   0.0184 *
X             4.7557     0.7525   6.320 7.69e-06 ***
D            -2.7401     3.5262  -0.777   0.4478
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.294 on 17 degrees of freedom
Multiple R-Squared: 0.9063,     Adjusted R-squared: 0.8953
F-statistic: 82.23 on 2 and 17 DF,  p-value: 1.816e-09
```

Le due regressioni separatamente

- $\mathbf{D} = 0$:

```
> summary(lm(Y[D==0]~X[D==0], data=Dati))

Call:
lm(formula = Y[D == 0] ~ X[D == 0], data = Dati)

Residuals:
     Min       1Q   Median       3Q      Max
-5.12550 -3.10886 -0.04054  3.41960  4.58030

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -47.5801    13.0507  -3.646  0.00653 **
X[D == 0]     5.6878     0.7937   7.166 9.56e-05 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.869 on 8 degrees of freedom
Multiple R-Squared: 0.8652,     Adjusted R-squared: 0.8484
F-statistic: 51.35 on 1 and 8 DF,  p-value: 9.557e-05
```

- $\mathbf{D} = 1$:

```
> summary(lm(Y[D==1]~X[D==1], data=Dati))
```

```
Call:
lm(formula = Y[D == 1] ~ X[D == 1], data = Dati)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0144 -1.9903 -0.9565  0.9376  8.6697

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -3.764     16.104  -0.234    0.821
X[D == 1]      2.240      1.291   1.735    0.121

Residual standard error: 3.83 on 8 degrees of freedom
Multiple R-Squared: 0.2734,     Adjusted R-squared: 0.1826
F-statistic: 3.011 on 1 and 8 DF,  p-value: 0.1209
```

2.

$$\mathbf{Y}_{n,1} = \beta_0 + \beta_1\mathbf{X} + \beta_2\mathbf{D} + \beta_3\mathbf{XD} + \boldsymbol{\epsilon}_{n,1}$$

```
> summary(lm(Y~X+D+XD, data=Dati))

Call:
lm(formula = Y ~ X + D + XD, data = Dati)

Residuals:
    Min      1Q  Median      3Q     Max
-5.1255 -2.6315 -0.9565  2.5010  8.6697

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -47.5801    12.9855  -3.664  0.00210 **
X             5.6878     0.7898   7.202 2.11e-06 ***
D            43.8159    20.7512   2.111  0.05081 .
XD           -3.4480     1.5189  -2.270  0.03738 *
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.85 on 16 degrees of freedom
Multiple R-Squared: 0.9291,     Adjusted R-squared: 0.9159
```

```
F-statistic: 69.93 on 3 and 16 DF,  p-value: 2.055e-09
```

3.

$$\mathbf{Y}_{n,1} = \beta_0 + \beta_1 \mathbf{D} + \boldsymbol{\epsilon}_{n,1}$$

```
> summary(lm(Y~D, data=Dati))

Call:
lm(formula = Y ~ D, data = Dati)

Residuals:
    Min      1Q  Median      3Q     Max
-13.830  -3.575  -0.900   5.595  12.770

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   45.530      2.415  18.851 2.67e-13 ***
D            -21.430      3.416  -6.274 6.46e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.638 on 18 degrees of freedom
Multiple R-Squared: 0.6862,     Adjusted R-squared: 0.6688
F-statistic: 39.36 on 1 and 18 DF,  p-value: 6.461e-06
```

L'analisi della varianza della regressione:

```
> anova(lm(Y~D, data=Dati))
Analysis of Variance Table

Response: Y
          Df  Sum Sq Mean Sq F value    Pr(>F)
D          1 2296.22 2296.22  39.364 6.461e-06 ***
Residuals 18 1050.00   58.33
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Calcoliamo le medie di $\mathbf{Y}_{n,1}$ nei due gruppi individuati da $\mathbf{D}$

```
> mean(Dati$Y[Dati$D==0])
[1] 45.53
> mean(Dati$Y[Dati$D==1])
[1] 24.1
```

Si osservino gli indici del DataFrame.

Capitolo 5

Modelli lineari generalizzati

Il modello lineare generale che abbiamo utilizzato fino ad ora ha la seguente forma

$$\mathbf{Y} = \mathbf{Xb} + \epsilon \tag{5.1}$$

dove $\mathbf{X}$ è la matrice aggiunta delle variabili indipendenti (continue o dummy variables per i fattori), $\mathbf{Y}$ è il vettore della variabili risposta (dipendenti), $\mathbf{b}$ è il vettore delle incognite contenente i coefficienti (pendenze) che devono essere stimati e ϵ è il vettore degli errori o residui.

Le assunzioni generali del modello sono riassunte nella seguente scrittura

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

che sostanzialmente ci dicono le gli errori (residui) sono attesi seguire una distribuzione normale con varianza constante (omoschedasticità).

La figura di riferimento era quella che abbiamo già visto per la regressione lineare.

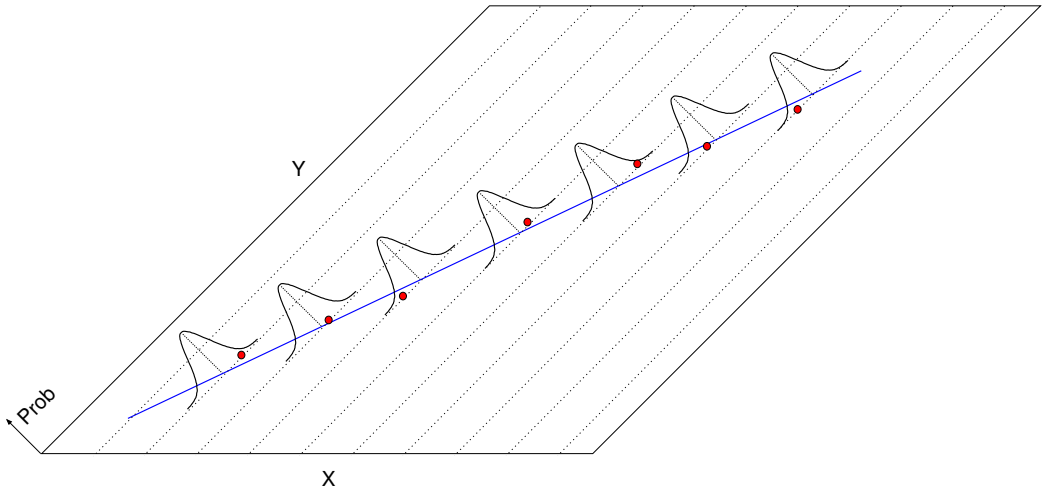


Grafico del modello statistico sottostante ad una regressione lineare

In sostanza si assume che l'ampiezza delle curve normali disegnate in corrispondenza di ogni punto sulla retta sia la stessa.

In questo capitolo affronteremo alcuni tipi di analisi statistiche che sono particolarmente utili in Ecologia e che permettono di rilassare le assunzioni di normalità dei residui e di omoschedasticità della varianza. Ricordiamo che l'assunzione di normalità dei residui richiede che i residui, oltre ad avere una distribuzione normale, abbiano il dominio su tutta la scala dei numeri reali da $-\infty$ a $+\infty$.

I casi più frequentemente usati riguardano

- quando la variabile dipendente (la y) rappresenta delle percentuali o delle proporzioni, cioè quando la variabile dipendente è costretta fra lo 0 e l'1 (o fra lo 0 e il 100%)
- quando la variabile dipendente è binaria del tipo successo/insuccesso (0/1; morto/vivo; sano/malato; germinato/non germinato; ecc)
- quando la variabile dipendente rappresenta dei conteggi interi (*counts*), cioè 0, 1, 2, 3, 4 ecc.

In tutti questi casi, che in realtà sono piuttosto frequenti, si possono utilizzare i **modelli lineari generalizzati** *Genaralized Linear Models* o *glm* [1] Per i glm il caso di una distribuzione normale dei residui è solo un caso particolare Questo tipo di però può essere esteso anche a tipi di variabili diverse.

Un modello lineare generalizzato ha tre componenti:

1) la struttura dell'errore

2) i predittori lineari

3) la funzione di link

[1] Da non confondere con il modello lineare generale usato nel gergo del SAS, che non è altro che una normale regressione semplice o multipla o un analisi della covarianza del tipo $\mathbf{Y} = \mathbf{Xb} + \epsilon$ e che in R è effetturata con la funzione `lm`. L'inventore dei modelli lineari generalizzati stesso ha espresso rincrescimento per non avere adottato inizialmente un nome diverso e più riconoscibile.

Esaminiamo uno alla volta i tre elementi.

Struttura d'errore Fino ad oggi abbiamo considerato solo casi in cui la distribuzione dell'errore era di tipo normale o gaussiana. Inoltre la varianza era assunta costante.

Nei glm gli errori possono avere una distribuzione molto asimmetrica, stretta o larga rispetto ad una normale (plati- o lepto-curtica), essere confinata fra certi valori (come nelle proporzioni) o errori che non possono condurre a valori predetti non negativi (come nei conteggi). In passato per poter analizzare dati con queste strutture di errore le alternative erano affidarsi alla statistisca non parametrica o trasformare la variabile dipendente al fine di rendere normale l'errore e omoschedastica la varianza, ma spesso era difficile ottenere risultati soddisfacenti.

Con i glm possiamo rilassare queste assunzioni e permettono di specificare modelli con una varietà di distribuzione di errore:

Binomiale utile per varibili relative a proporzioni o 0 e 1 (vivo/morto, successo/insuccesso)

Poisson utile per conteggi (*counts*, interi positivi)

Normale o Gaussiana per dati continuui che possono assumere valori sia negativi, sia positivi

Gamma dati continui positivi con varianza che cresce proporzionalmente alla media

Esponenziali utili per dati come il tempo alla morte (analisi della sopravvivenza)

I primi due casi sono quelli più usati. Tutte queste distribuzioni fanno parte di una famiglia di distribuzioni esponenziali.

E' quind possibile immaginare delle curve diverse da quella normale costruite attorno ai valori predetti dalla regressione lineare nella figura precedente.

Predittori lineari E' la parte del modello che coinvolge le varabili indipendenti (x) chiamati anche predittori o variabili esplicative e consiste nella somma di uno o più variabili esplicative ciascuna con moltiplicato per il proprio coefficiente (pendenza) che deve essere stimato dai dati

$$\eta_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_n x_{i,n} \tag{5.2}$$

Le variabili x possono essere sia fattori, sia variabili continue e, come si può notare, questa parte del modello è molto simile ai modelli che abbiamo visto fino ad ora. Nel caso una x fosse un fattore verrebbe automaticamente trasfrmata in *dummy variable* come abbiamo visto in precedenza. Sulle variabili esplicative non c'è nessuna assunzione.

Una differenza fondamentale rispetto alla regressione è che in questo caso non viene predetta direttamente la variabile esplicativa (y), ma viene predetta una variabile y **trasformata** dalla link function.

La funzione di link In questa componente dei glm sta una delle differenze importanti rispetto alla regressione. La funzione di link **associa** il valore atteso di y cioè **la media** ai predittori lineari

$$\mathbf{E}(y) = \mu = g^{-1}(\eta) \tag{5.3}$$

La funzione di link è la funzione $g()$ di cui si può calcolare l'inversa $g^{-1}()$

$$\eta = g(\mu) \tag{5.4}$$

Ogni distribuzione d'errore ha la sua particolare funzione di link canonica.

Errore	**Link canonica**	**Formula**
Normale	identità	$=$
Poisson	log	$\log_e$
Binomiale	logit	$log_e\left(\frac{\mu}{1-\mu}\right)$
Gamma	reciproco	$-x^{-1}$

Alcuni esempi

Per il caso **gaussiano-normale** la funzione di link è semplicemente la funzione identità (che è uguale alla sua inversa), quindi nel caso normale la componente lineare prevede direttamente la media di y. Quello del distribuzione d'errore gaussiana (normale) è il default di R per `glm` e quindi non c'è bisogo di specificare la la distribuzione dell'errore con il parametro `family`. I risultati, cioè i parametri stimati (pendenze e intercetta) in questo caso sono uguali a quelli ottenibili con la funzione `lm`, solo che sono ottenuti con un metodo iterarivo di massima verosimiglianza (*maxium likelihood* e non con il metodo dei minimi quadrati.

Negli altri casi occorre speficare la distribuzione d'errore con il parametro `family` Quindi, per esempio, nel caso di un **modello binomiale** in cui si cerca di modellizzare la proporzione di successi $p_i = \frac{n_i}{N}$ come rapporto fra

il numero successi (n_i) sul totale di casi possibili (N), il modello che viene fittato è il seguente:

$$\log_e \left(\frac{p_i}{1 - p_i} \right) = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_n x_{i,n} \tag{5.5}$$

e questo modello viene fittato in R con il seguente comando

```
glm(y ~ x, family = "binomial")
```

La variabile dipendente y viene però fornita in termini di matrice con due colonne con, nella prima, il numero di successi e, nella seconda, il numero di insuccessi. Anche quando si fittano sui variabili percentuali o poporzioni in realtà bisogna sempre usare i numeri grezzi su cui si calcola la percentuale: in termini statistici è molto diverso dire che ci sono 4 successi su 10, rispetto a 400 su 1000. La percentuale è la stessa, ma nel secondo caso il campione è molto più grande e la stima della percentuale è molto più affidabile. Per esempio:

```
glm(cbind(ni, N-ni) ~ x, family = "binomial")
```

Nel caso della binomiale con un semplice predittore lineare x continuo si può scrivere:

$$\log_e \left(\frac{p}{1 - p} \right) = \beta_0 + \beta_1 x$$

passando all'anti-logaritmo

$$\frac{p}{1 - p} = e^{\beta_0 + \beta_1 x}$$

$$p = \left(e^{\beta_0 + \beta_1 x} \right) (1 - p)$$

$$p = \left(e^{\beta_0 + \beta_1 x} \right) - p \left(e^{\beta_0 + \beta_1 x} \right)$$

$$p + p \left(e^{\beta_0 + \beta_1 x} \right) = \left(e^{\beta_0 + \beta_1 x} \right)$$

$$p \left(1 + e^{\beta_0 + \beta_1 x} \right) = \left(e^{\beta_0 + \beta_1 x} \right)$$

$$p = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

$$p = \frac{1}{e^{-(\beta_0 + \beta_1 x)}}$$

Quindi fittando un modello lineare dove la y è il logaritmo di un rapporto di probabilità (successi diviso insuccessi), chiamato anche *logit*, in realtà si esegue un fitting di un modello esponenziale curvilineo. Questo modello viene chiamato anche **regressione logistica**.

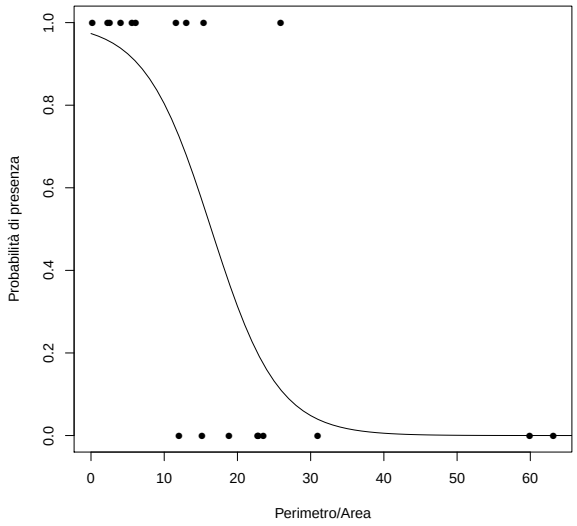
Il processo di fitting è un procedimento iterativo di massima verosimiglianza simile a quello visto per il fitting non lineare (`nls`), quindi molto diverso da una operazione matriciale o da un fitting lineare basato sul criterio dei minimi quadrati. Il processo iterativo però in questo caso è altamente ottimizzato e normalmente non occorre fornire alla funzione `glm` i parametri iniziali.

La curva logit ha il andamento curvilineo, come nella figura seguente che rappresenta una x che influenza negativamente la y.

Facciamo un esempio tratto da Polis *et al.* 1998 dove la presenza di una certa specie di lucertola (*Uta*) su 19 delle isole della California viene predetta da una variabile x continua data dal rapporto perimetro/area delle isole stesse.

```
PerAreaR<-c(15.41,5.63,25.92,15.17,13.04,18.85,30.95,22.87,12.01,
            11.60,6.09,2.28,4.05,59.94,63.16,22.76,23.54,0.21,2.55)
UTA <- c("Present","Present","Present","Absent","Present","Absent","Absent",
         "Absent","Absent","Present","Present","Present","Present","Absent",
         "Absent","Absent","Absent","Present","Present")
uta.df <- data.frame(PerAreaR, UTA)
rm(PerAreaR, UTA)
uta.df

##    PerAreaR     UTA
## 1     15.41 Present
## 2      5.63 Present
## 3     25.92 Present
## 4     15.17  Absent
## 5     13.04 Present
## 6     18.85  Absent
## 7     30.95  Absent
## 8     22.87  Absent
## 9     12.01  Absent
## 10    11.60 Present
## 11     6.09 Present
## 12     2.28 Present
## 13     4.05 Present
## 14    59.94  Absent
```

```
## 15    63.16  Absent
## 16    22.76  Absent
## 17    23.54  Absent
## 18     0.21 Present
## 19     2.55 Present

uta.df$AP <- 0
uta.df$AP[uta.df$UTA=="Present"]  <- 1
uta.glm <- glm(cbind(AP, 1-AP) ~ PerAreaR, data=uta.df, family=binomial)
plot(AP ~ PerAreaR, data=uta.df, xlab="Perimetro/Area", ylab=
     "Probabilità di presenza", pch=16)
```




```
summary(uta.glm)

##
## Call:
## glm(formula = cbind(AP, 1 - AP) ~ PerAreaR, family = binomial,
##     data = uta.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6067  -0.6382   0.2368   0.4332   2.0986
##
## Coefficients:
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.6061     1.6953   2.127   0.0334 *
## PerAreaR     -0.2196     0.1005  -2.184   0.0289 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26.287  on 18  degrees of freedom
## Residual deviance: 14.221  on 17  degrees of freedom
## AIC: 18.221
##
## Number of Fisher Scoring iterations: 6
```

Come si vede in questo caso la probabilità di trovare la lucertola su un'isola cala significativamente al crescere del rapporto fra perimetro e area dell'isola stessa. L'interpretazione del `summary` è analoga a quella della della regressione lineare ordinaria. In questo caso i test sui parametri eseguiti dal `summary` vengono però chiamati Wald's test e sono decisamente meno affidabili dei test sui parametri della regressione lineare.

È molto importante guardare alla riga corrispondente alla `Residual deviance` e il rapporto fra due numeri riportati `14.221` e `17` non devo differire molto da 1. Se la `Residual deviance` è molto più grande dei corrispondenti gradi di libertà, quindi il rapporto fra le due quantità è molto maggiore di 1, allora siamo probabilmente di fronte ad un caso di sovradispersione (**overdipsersion**) di cui bisogna tenere conto. Impareremo come farlo fra breve. Nel caso della lucertola il problema non esiste.

Esiste anche un analogo dell'analisi della varianza con i glm e in questo caso è chiamata **Analisi della devianza**. Il termine devianza che abbiamo visto per i modelli anova ordinari (somma degli scarti dalla medie) vale solo per il caso normale o gaussian. In generale per i glm la devianza è una quantità definita come *-2 volte la differenza fra la log-likelihood del modello proposto e quella del modello saturato*. Quest'ultimo è un modello ipotetico che ha tanti parametri quanti sono i dati e quindi fitterebbe i dati in modo perfetto. E' un modello del tutto inutile ma che serve solo come riferimento in quanto è il modello con la likelihood massima possibile. All'opposto esiste anche la likelihood del *modello nullo* che prevede un solo parametro pari alla media delle y e si assume che sia un modello dove la y non dipende alla x e corrisponde ad una linea orizzontale a pendenza uguale a zero posta ad altezza $\overline{y}$. Questo modello ha un solo parametro ed è il modello con la

likelihood minima possibile.

Il modello proposto avrà una likelihood intermedia tra quella del modello saturato e quella del modello nullo. Il modello saturato viene preso come riferimento per calcolare la devianza degli altri due. Si vuole testare se il modello proposto ha devianza minore (o una likelihood significativamente maggiore) del modello nullo.

L' implementazione e anche l'interpretazione è simile a quella dell'`anova` dei modelli di regressione ordinari. Di solito però non si usa il solito test F ma si sfrutta il fatto che i rapporti fra likelihood, sotto ipotesi nulla, si distribuiscono come un χ^2.

```
anova(uta.glm, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: cbind(AP, 1 - AP)
##
## Terms added sequentially (first to last)
##
##
##          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         18     26.287
## PerAreaR  1   12.066        17     14.221 0.0005134 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Questo test viene effettuato utilizzando un rapporto fra le devianze del modello *NULLO*, in cui la y viene assunta non dipendere dalla x, che quindi avrebbe un andamento piatto rispetto alla x (pendenza zero) ad un altezza corrispondente all'unico parametro stimato (la media delle y), e il modello che abbiamo effettivamente usato, in cui la y dipende dalla x con pendenza diversa da zero.

Si può sempre fare il confronto fra le likelihood di due modelli diversi, come nella regressione ordinaria, confrontando il modello semplice rispetto ad un modello più complesso:

```
anova(modello.semplice.glm, modello.complicato.glm, test="Chisq")
```

e procedere alle semplificazioni (es: togliere le interazioni fra le x che non sono significative).

Nel caso di conteggi si effettua una **Regressione di Poisson** che utilizza una distribuzione d'errore Poissoniana. La y in questo caso è rappresentata solo da numeri interi (zero compreso). Questo tipo di regressione si usa molto spesso in ecologia quando viene valutato, per esempio, il numero di individui di una specie dentro un plot.

Nel caso di una distribuzione d'errore di **Poisson**, usato quando le varibile y è un conteggio di interi positivi, le modalità di esecuzione del test sono del tutto simili a quello per la distribuzione binomiale, ma stavolta va indicato `family="poisson"`.

La funzione di link utilizzata normalmente è il logaritmo, che non può essere calcolato con valori della y negativi. Fittare un modello di tipo Poisson in realtà vuole dire fittare un modello del tipo:

$$log(y_i) = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_n x_{i,n} \tag{5.6}$$

il che equivale a:

$$y_i = e^{\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_n x_{i,n}} \tag{5.7}$$

il cui errore però è distribuito secondo una distribuzione di Poisson, che, ricordiamo, è una distribuzione per soli interi positivi (zero compreso) e che è caratterizzata dall'avere un solo parametro (μ) che ne determina sia la media, sia la varianza.

Usiamo un esempio tratto da Crawley in cui la variabile dipendente è il numero di specie presenti in 90 plot in funzione del fattore pH del suolo e della variabile continua che misura la biomassa totale presente (un indice di fertilità del plot).

```
s.df <- read.table("species.txt", h=T)
s.df
```

```
##      pH    Biomass Species
## 1  high 0.46929722      30
## 2  high 1.73087043      39
## 3  high 2.08977848      44
## 4  high 3.92578714      35
## 5  high 4.36679265      25
## 6  high 5.48197468      29
## 7  high 6.68468591      23
## 8  high 7.51165063      18
## 9  high 8.13220251      19
```

```
## 10 high 9.57212864      12
## 11 high 0.08665367      39
## 12 high 1.23697390      35
## 13 high 2.53204324      30
## 14 high 3.40794153      30
## 15 high 4.60504596      33
## 16 high 5.36771709      20
## 17 high 6.56084215      26
## 18 high 7.24206214      36
## 19 high 8.50363299      18
## 20 high 9.39095342       7
## 21 high 0.76488801      39
## 22 high 1.17647020      39
## 23 high 2.32512082      34
## 24 high 3.22288207      31
## 25 high 4.13612930      24
## 26 high 5.13717652      25
## 27 high 6.42193811      20
## 28 high 7.06552638      21
## 29 high 8.74592918      12
## 30 high 9.98177013      11
## 31  mid 0.17576270      29
## 32  mid 1.37677830      30
## 33  mid 2.55104256      21
## 34  mid 3.00027434      18
## 35  mid 4.90562386      13
## 36  mid 5.34330542      13
## 37  mid 7.70000000       9
## 38  mid 0.55368893      24
## 39  mid 1.99029644      26
## 40  mid 2.91263671      26
## 41  mid 3.21645133      20
## 42  mid 4.97988468      21
## 43  mid 5.65872290      15
## 44  mid 8.10000000       8
## 45  mid 0.73956986      31
## 46  mid 1.52693420      28
## 47  mid 2.23212239      18
## 48  mid 3.88528818      16
## 49  mid 4.62650541      19
```

```
## 50  mid 5.12096844      20
## 51  mid 8.30000000       6
## 52  mid 0.51127858      25
## 53  mid 1.47823269      23
## 54  mid 2.93455800      25
## 55  mid 3.50548891      22
## 56  mid 4.61790914      15
## 57  mid 5.69696382      11
## 58  mid 6.09301688      17
## 59  mid 0.73006280      24
## 60  mid 1.15806838      27
## 61  low 0.10084790      18
## 62  low 0.13859609      19
## 63  low 0.86351508      15
## 64  low 1.29291903      19
## 65  low 2.46916355      12
## 66  low 2.36655309      11
## 67  low 2.62921708      15
## 68  low 3.25228652       9
## 69  low 4.41727619       3
## 70  low 4.78081039       2
## 71  low 0.05017529      18
## 72  low 0.48283691      19
## 73  low 0.65266714      13
## 74  low 1.55533656       9
## 75  low 1.67163820       8
## 76  low 2.87005390      14
## 77  low 2.51072052      13
## 78  low 3.49760385       4
## 79  low 3.67876186       8
## 80  low 4.83154245       2
## 81  low 0.28972266      17
## 82  low 0.07756009      14
## 83  low 1.42902041      15
## 84  low 1.12074092      17
## 85  low 1.50795384       9
## 86  low 2.32596318       8
## 87  low 2.99570582      12
## 88  low 3.53819909      14
## 89  low 4.36454121       7
```

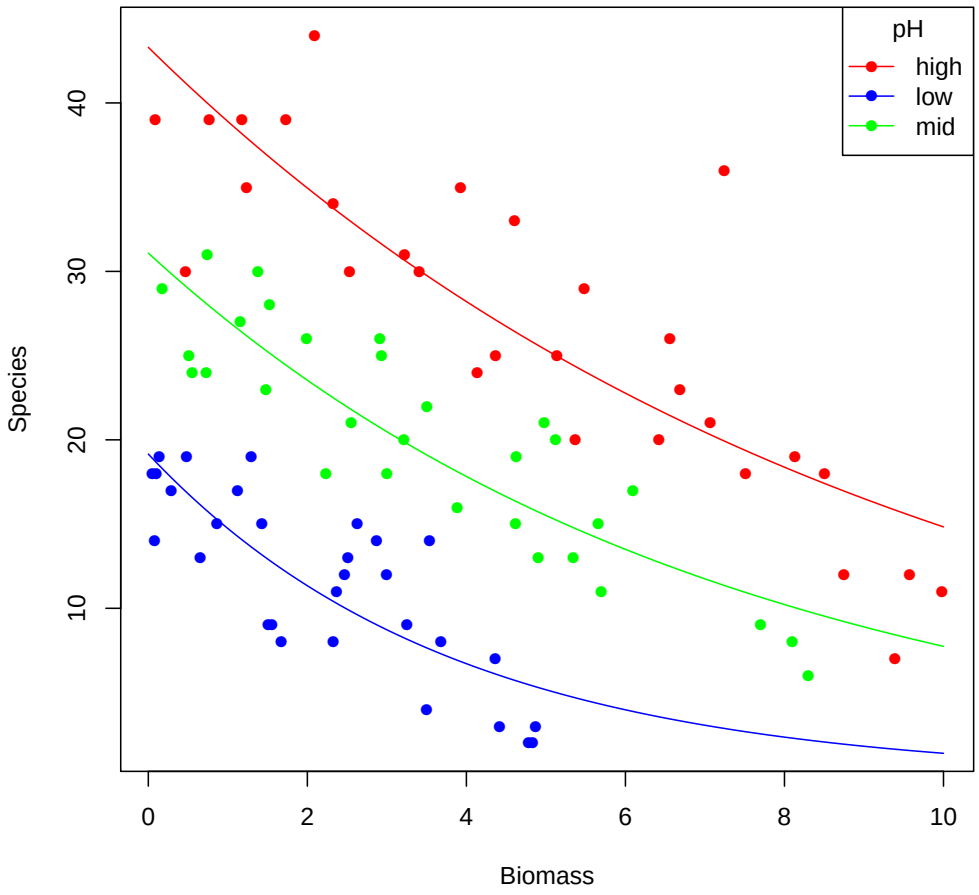
```
## 90  low 4.87050789       3

## utilizziamo prima un modello con l'interazione pH:Biomass
s.glm <- glm(Species ~ Biomass * pH, data=s.df, family="poisson")
summary(s.glm)

##
## Call:
## glm(formula = Species ~ Biomass * pH, family = "poisson", data = s.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4978  -0.7485  -0.0402   0.5575   3.2297
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    3.76812    0.06153  61.240  < 2e-16 ***
## Biomass       -0.10713    0.01249  -8.577  < 2e-16 ***
## pHlow         -0.81557    0.10284  -7.931 2.18e-15 ***
## pHmid         -0.33146    0.09217  -3.596 0.000323 ***
## Biomass:pHlow -0.15503    0.04003  -3.873 0.000108 ***
## Biomass:pHmid -0.03189    0.02308  -1.382 0.166954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 452.346  on 89  degrees of freedom
## Residual deviance:  83.201  on 84  degrees of freedom
## AIC: 514.39
##
## Number of Fisher Scoring iterations: 4

## poi utilizziamo un modello senza interazione
s2.glm <- glm(Species ~ Biomass + pH, data=s.df, family="poisson")
summary(s2.glm)

##
## Call:
## glm(formula = Species ~ Biomass + pH, family = "poisson", data = s.df)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5959  -0.6989  -0.0737   0.6647   3.5604
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.84894    0.05281  72.885  < 2e-16 ***
## Biomass     -0.12756    0.01014 -12.579  < 2e-16 ***
## pHlow       -1.13639    0.06720 -16.910  < 2e-16 ***
## pHmid       -0.44516    0.05486  -8.114 4.88e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 452.346  on 89  degrees of freedom
## Residual deviance:  99.242  on 86  degrees of freedom
## AIC: 526.43
##
## Number of Fisher Scoring iterations: 4

##Effettuiamo il test per vedere se l'interazione sia significativa
anova(s2.glm,s.glm, test="Chisq")

## Analysis of Deviance Table
##
## Model 1: Species ~ Biomass + pH
## Model 2: Species ~ Biomass * pH
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        86     99.242
## 2        84     83.201  2    16.04 0.0003288 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

L'interazione è altamente significativa. Le pendenze della regressione del numero di specie con la biomassa sono diverse per terreni con pH diverso.

La presentazione del grafico è leggermente più complicata del solito per via del fatto che la y è su scala logaritmica e l'andamento non è lineare.

```
colori <- c("red","blue","green")
names(colori) <- levels(s.df$pH)
plot(Species ~ Biomass, data=s.df, pch=16, col=colori[as.character(s.df$pH)])
## creo vettori con una x fittizia e un pH fittizio di 101 elementi
xv <- seq(0,10,0.1)
phv <- factor(rep("high",101))
## calcolo la y per la x fittizia usando predict del modello
yv <- predict(s.glm, new=list(pH=phv,Biomass=xv), type="response")
## aggiungo la linea calcolata per ph="high"
lines(yv ~ xv, col=colori["high"])

phv <- factor(rep("mid",101))
yv <- predict(s.glm, new=list(pH=phv,Biomass=xv), type="response")
lines(yv ~ xv, col=colori["mid"])

phv <- factor(rep("low",101))
yv <- predict(s.glm, new=list(pH=phv,Biomass=xv), type="response")
lines(yv ~ xv, col=colori["low"])

legend("topright", lty=1,pch=16, col=colori, legend=names(colori), title="pH")
```



Cosa fare in caso di over-dispersion

Nel caso di sovra-dispersione, rilevabile dal `summary` nella riga `Residual deviance` dove il rapporto fra devianza residua e i rispettivi gradi di libertà è molto maggiore di 1, si può procedere in diversi modi. E' importante notare che spesso i risultati riguanti la significatività di una certa x nel determinare la y, sono molto sensibili alla presenza di *over-dispersion* e quindi è meglio controllare sempre se il fatto di tenere conto o meno di una leggera sovra-dispersione possa influenzare i nostri risultati.

Uno dei modi in cui si genera la sovradispersione in natura è quando non si tiene conto di un fattore o di una variabile che determina dei raggruppamenti. Per esempio, nell'analisi della presenza di individui di una specie vegetale, se la dispersione dei semi è limitata (i semi tendono a cascare vicini alla

pianta madre), può facilmente succedere che ci sia la maggioranza dei plot con assenza di individui di questa specie e qualche plot con una presenza invece molto abbondante. Quindi il considerare solo i fattori ambientali che caratterizzano il plot non permette di modellizzare con la dovuta correttezza la presenza degli individui della specie nei plot. Un caso frequente di sovradispersione è la presenza dei parassiti o di malattie contagiose. Di per sè il fenomeno può essere raro con pochi ospiti infettati, ma quando un ospite è infettato si rilevano molti parassiti o germi. Nel dataset avrò quindi righe con pochi o zero individui e qualche riga con dei numeri molto alti, ho quindi un eccesso di varianza rispetto all'atteso.

Uno dei modi più semplici per tener conto della sovradispersione è utilizzare le distribuzioni di errore *quasi*, cioè `quasibinomial` o `quasipoisson`. Queste distribuzioni hanno un parametro in più che permette di tenere conto di una varianza dei dati maggiore di quella assunta dal modello. Quindi, nel caso di una distribuzione di errore secondo Poisson, posso fittare un modello *quasi*-poisson cosi:

```
glm(y ~ x, family = "quasipoisson")
```

Quindi con questo tipo di modelli i risultati sono validi anche nel caso di sovra-dispersione e sono spesso diversi dai risultati ottenibili dalle distribuzioni non "*quasi*".

Lo svantaggio nell'usare le distribuzioni "*quasi-*" è che alterano la struttura del modello e non è più possibile calcolare correttamente la likelihood e quindi anche l'AIC. Quindi non è possibile confrontare i modelli *quasi-* con i modelli non *quasi-* utilizzando la likelihood.

Un secondo metodo per ovviare alla sovradipsersione nel caso di una distribuzione d'errore di tipo Poisson è cambiare distribuzione d'errore ed affidarsi, per esempio, ad una distribuzione **Binomiale negativa**. Per utilizzare quest'ultima distribuzione in R è necessario caricare il pacchetto `MASS` e utilizzare la funzione `glm.nb`

```
library(MASS)
m.glmnb <- glm.nb(y ~ x)
summary(m.glmnb)
```

Con quest'ultima distribuzione si possono confrontare modelli diversi con il metodo dell'AIC.

Un caso speciale nell'arco dei possibili risultati, putroppo molto frequente in Ecologia, è il caso in cui si verifichino degli eccessi di zeri nella y. Ci

possono essere cioè dei casi in cui il numero di zeri sia eccessivo rispetto a quanto atteso dal modello. Un esempio di questo genere di risultato è quando codifico come zero un dato mancante che in realtà andrebbe utilizzato `NA`. Per esempio se un giorno non vado a pescare e codifico il risultato come zero pesci presi. Ovviamente è un caso diverso da quello in cui sia andato a pescare e abbia preso zero pesci. In questo caso è meglio utilizzare i cosiddetti *zero inflated models* che tengono conto che i valori di 0 possono essere determinati da due processi diversi. In letteratura è però riportato come questo tipo di modelli siano gestiti in modo più corretto e naturale dai metodi bayesiani che affronteremo nel corso di Modellistica Ambientale nel secondo semestre.

Capitolo 6

La scelta del test statistico giusto

La scelta della analisi statistica appropriata per i propri dati è spesso una delle più difficili. La scelta dipende dalla **natura dei dati** e dalla **domanda** alla quale si vuole rispondere. La chiave è capire bene il tipo di **variabile dipendente** che si ha a disposizione e la natura delle **variabili indipendenti**. La variabile dipendente, che normalmente si riporta sull'asse y, è quella sulla quale si sta lavorando e quella di cui cerchiamo di spiegare la variabilità. Di questa variabile (o meglio dei sui residui) è spesso necessario fare assunzioni la sua distribuzione. È necessario conoscere anche la natura delle variabili indipendenti, quelle riportate sull'asse x, ma le assunzioni sulla distribuzione di queste ultime sono solitamente non rilevanti.

È essenziale sapere rispondere alle seguenti domande:

- Quale delle nostre variabili è la variabile dipendente?
- Quale o quali sono le variabili indipendenti?
- Le variabili indipendenti sono continue, categoriche (fattori) o un misto delle due?
- Le variabili dipendenti sono di tipo continuo, un conteggio (intero positivo), una proporzione, un'età alla morte (durata di un periodo) o delle categorie (fattori)?

La risposta a queste domande conduce facilmente al test appropriato.

Variabile indipendente

Tutte le variabili continue	**Regressione**
Tutte le variabili categoriche	**Analsi della Varianza (ANOVA)**
Misto di variabili continue e categoriche	**Analisi della Covarianza (ANCOVA)**

Variabile dipendente

Continua	**Regressione, ANOVA o ANCOVA**
Proporzione	**Regressione logistica** es: `glm(...,family=binomial)`
Binario (1/0)	**Regressione logistica binaria** es: `glm(...,family=binomial)`
Conteggio	**Modello Log-Lineare** es:`glm(...,family=poisson)`
Età alla morte	**Analisi della Sopravvivenza**

La seguente tabella tratta da Crawley (The R Book) riporta le formule dei più comuni modelli in R.

Table 9.3. Examples of R model formulae. In a model formula, the function I case i) stands for 'as is' and is used for generating sequences I(1:10) or calculating quadratic terms I(x^2).

Model	Model formula	Comments
Null	y ~ 1	1 is the intercept in regression models, but here it is the overall mean *y*
Regression	y ~ x	*x* is a continuous explanatory variable
Regression through origin	y ~ x-1	Do not fit an intercept
One-way ANOVA	y ~ sex	sex is a two-level categorical variable
One-way ANOVA	y ~ sex-1	as above, but do not fit an intercept (gives two means rather than a mean and a difference)
Two-way ANOVA	y ~ sex + genotype	genotype is a four-level categorical variable
Factorial ANOVA	y ~ N * P * K	*N, P* and *K* are two-level factors to be fitted along with all their interactions
Three-way ANOVA	y ~ N*P*K – N:P:K	As above, but don't fit the three-way interaction
Analysis of covariance	y ~ x + sex	A common slope for *y* against *x* but with two intercepts, one for each sex
Analysis of covariance	y ~ x * sex	Two slopes and two intercepts
Nested ANOVA	y ~ a/b/c	Factor *c* nested within factor *b* within factor *a*
Split-plot ANOVA	y ~ a*b*c+Error(a/b/c)	A factorial experiment but with three plot sizes and three different error variances, one for each plot size
Multiple regression	y ~ x + z	Two continuous explanatory variables, flat surface fit
Multiple regression	y ~ x * z	Fit an interaction term as well (x + z + x:z)
Multiple regression	y ~ x + I(x^2) + z + I(z^2)	Fit a quadratic term for both *x* and *z*
Multiple regression	y <- poly(x,2) + z	Fit a quadratic polynomial for *x* and linear *z*
Multiple regression	y ~ (x + z + w)^2	Fit three variables plus all their interactions up to two-way
Non-parametric model	y ~ s(x) +s(z)	*y* is a function of smoothed *x* and *z* in a generalized additive model
Transformed response and explanatory variables	log(y) ~ I(1/x) + sqrt(z)	All three variables are transformed in the model

Capitolo 7

La Statistica Multivariata

La statistica multivariata si occupa di insieme di dati in cui le unità sperimentali (es: i nostri campioni) sono caratterizzate da **molte variabili** che non presentano la distinzione fra variabili dipendenti e indipendenti (possiamo quindi considerle tutte dipendenti o tutte indipendenti). A priori quindi non ci sono variabili più o meno importanti di altre.

Il dataset può essere quindi pensato come una matrice con n righe rappresentanti le unità sperimentali e p colonne rappresentanti le variabili.

Esistono due grandi categorie in cui sono classificati metodi multivariati:

Metodi senza struttura cioè metodi in cui si cerca di individuare un'eventuale "struttura" nei dei dati (es: PCA)

Metodi in cui si assume che ci sia una struttura per esempio si assume che i dati siano già suddivisi in gruppi (es: Analisi discriminante).

Alcune osservazioni:

- Le variabili possono essere solo continue, solo discrete, continue e discrete.
- Tipicamente le variabili sono correlate tra di loro ma le unità sono indipendenti tra loro
- Vi possono essere più variabili che osservazioni ($p > n$) e viceversa ($p < n$).

La statistica multivariata si usa per risolvere problemi come:

- Riduzione della dimensione (ovvero di p) al fine di condurre un analisi esplorativa;

- semplificare la rappresentazione (ad esempio con una o due variabili)
- poter ottenere maggior stabilità nelle procedure statistiche (rimuovendo, ad esempio, le variabili che sono maggiormente correlate) (es: Analisi delle componenti principali, analisi fattoriale, ...)
- Ricerca di regole di separazione/discriminazione per assegnare un individuo a due o più gruppi prestabiliti sulla base di p informazioni aggiuntive. Ad esempio stabilire se un individuo appartiene alla specie A o alla specie B.

Analisi delle Componenti Principali

Il metodo chiamato **Analisi delle Componenti Principali** o PCA riassumono i dati delle variabili originarie in due o tre variabili "principali" che, appunto, sintetizzano i dati. Le nuove variabili sintetiche sono calcolate sempre partendo dalle variabili originarie, e tramite algoritmi dell'algebra lineare si calcolano delle nuove variabili che sono una *combinazione lineare* delle variabili originali (p_i)

$$Y_1 = a_1 p_1 + a_2 p_2 + a_3 p_3 + a_4 p_4 + \dots \tag{7.1}$$

Le nuove variabili sintetiche sono quindi una "somma" delle variabili originali, ma ciascuna variabile avrà un peso che dipende dal coefficiente a_i, che può assumere valori positivi o negativi.

Una volta trovati valori i dei coefficienti a_i si possono sostituire per ogni stazione le variabili originarie nella formula precedente e si ricava il valore di Y_1 (score) per ogni campione. Si può fare lo stesso per una seconda variabile sintetica Y_2, che avrà i coefficienti a_i diversi dalla prima, e quindi fare un grafico con le due Y come ordinata e come ascissa.

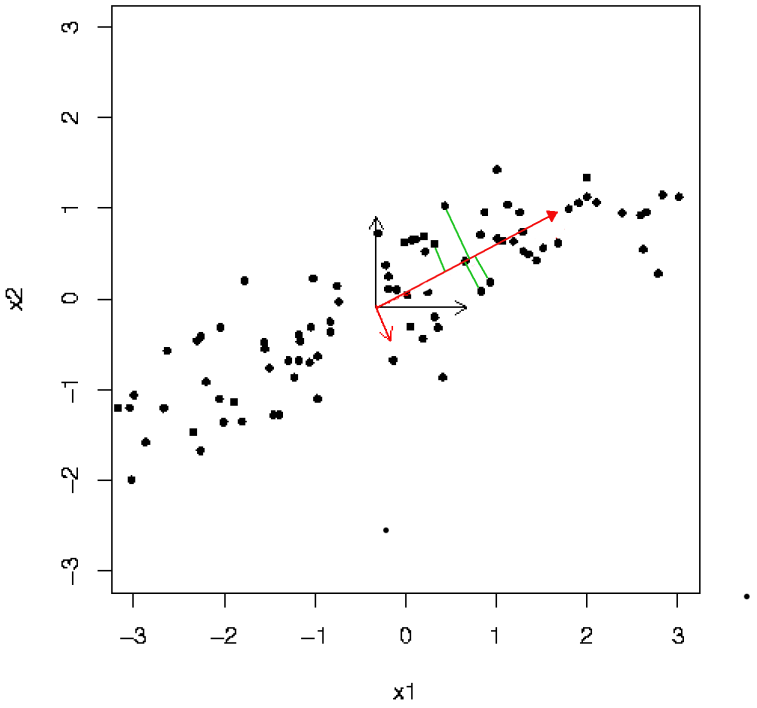


Figura 7.1

La chiave è quindi trovare i coefficienti a_i. Questo lo fanno i software, ma può essere utile sapere che in generale viene minimizzata (un po' lo stesso procedimento dei minimi quadrati) la somma quadratica della distanza perpedicolare da ogni punto alla retta che individua la nuova variabile (nell'esempio riportato in figura si tratterebbe di minimizzare la somma dei quadrati dei segmenti in verde). Da notare che in questo modo si trova la nuova variabile che massimizza la varianza dei punti (nel esempio l'asse rosso più lungo). La seconda variabile (asse rosso più corto) si trova imponendo la perpendicoltarità rispetto alla prima e massimizzando di nuovo la varianza residua. Il procedimento potrebbe proseguire e creare tante nuove variabili quante erano le variabili originare, ma ovviamente di solito ci si limita a guardare le prime due o tre in quanto riassumo la maggior parte dell'informazione (varianza).

I risultati del nuovo plot si interpretano di solito in maniera semplice: punti vicini sui grafici indicano una similarità anche per le variabili originarie dei relativi campioni.

Il tutto viene computato partendo da una matrice di varianza-covarianza o da una matrice di correlazione delle variabili originarie. Altri metodi

multivariati utilizzano matrici di distanze o similarità diverse

Esempio in R:

```
data(iris)
colori <- c("red","blue","green")
names(colori)<- unique(as.character(iris$Species))
i.df <- iris[,1:4]
i.pca <-princomp(i.df)
summary(i.pca)
loadings(i.pca)
plot(i.pca)
plot(i.pca$scores[,1:2], col=colori[as.character(iris$Species)],pch=16)
```