Materiale didattico per il corso di Inferenza Statistica I

a cura di:
Alessandra Dalla Valle e Carlo Gaetan [1]

Anno Accademico 2001-2002

[1]Si ringrazia A. Brazzale, M. Chiogna, S.M. Iacus, N. Sartori per aver messo a disposizione il loro materiale didattico. Queste note sono fortemente influenzate dal loro lavoro.

Sommario

Laboratorio 1

Introduzione a R

1.1 Avvertenza

Queste note contengono errori e inesattezze sicuramente non voluti ma comunque presenti. Fate sempre riferimento alla documentazione che accompagna il programma e alla guida in linea del programma.

1.2 Iniziare e chiudere una sessione di R

Per iniziare una sessione R fare un doppio click di mouse sulla icona di R.
Per uscire da R, usa `q()`. Per salvare i dati rispondere "Si", altrimenti rispondere "No".
Per controllare cosa c'e' disponibile nella directory dei dati:

```
> ls()
character(0)
```

Per eliminare un oggetto, usa `rm()`.

```
> rm(thing)
> thing
Error: Object "thing" not found
```

Se si vogliono eliminare più oggetti, bisogna elencarli separati da virgole.

```
> rm(thing1,thing2)
```

Quando si inizia una nuova sessione di lavoro, è opportuno rimuovere tutti i vecchi oggetti che non servono. Un comando utile è:

```
> rm(list=ls())      oppure     rm(list=objects())
```

1.3 Semplice aritmetica

In R, qualunque cosa venga scritta al prompt viene valutata:

```
> 1+2+3
[1] 6

> 2+3*4
[1] 14

> 3/2+1
[1] 2.5

> 2+(3*4)
[1] 14

> (2 + 3) * 4
[1] 20

> 4*3**3         Usa ** o ^ per calcolare un elevamento a potenza.
[1] 108
```

R fornisce anche tutte le funzioni che si trovano su un calcolatore tascabile:

```
> sqrt(2)
[1] 1.414214

> sin(3.14159)          sin(Pi greco) e' zero
[1] 2.65359e-06         e questo e' vicino...
```

Fornisce anche il valore di π

```
> sin(pi)
[1] 1.224606e-16    ancora piu' vicino a  zero...
```

Ecco una breve lista

```
  Nome                  Operazione

  sqrt                                  radice quadrata
  abs                                   valore assoluto
  sin   cos   tan               funzioni trigonometriche
  asin  acos  atan      funzioni trigonometriche inverse
  exp   log             exponenziale e logaritmo naturale
```

Le funzioni possono essere annidate:

```
> sqrt(sin(45*pi/180))
[1] 0.8408964
```

In realtà R possiede un gran numero di funzioni, anzi proprio la ricchezza di funzioni e la possibilitaà di incrementarne il numero costituiscono uno dei punti di forza del linguaggio. Per chiedere aiuto su di una funzione si può digitare

```
> help(sqrt)
```

ma se non si sa se esiste una funzione particolare è possibile ricorrere ad un aiuto più generale

```
> help.start()
```

Esercizio: Provate a cercare se esistono, delle funzioni iperboliche o per i numeri complessi.

1.4 Assegnazioni di valori

Si può salvare un valore assegnandolo ad un oggetto mediante il simbolo `<-` oppure il simbolo _ :

```
> x <- sqrt(2)      salva in x la radice quadrata di 2
> x
[1] 1.414214
> x**3
[1] 2.828427
```

Valori logici
R permette di gestire operazioni e variabili logiche:

```
> x <- 10         fissa x uguale a 10
> x > 10          x e' piu' grande di 10?
[1] FALSE
> x <= 10
[1] TRUE
> tf <- x > 10
> tf
[1] FALSE
```

1.5 Gestione di vettori

1.5.1 Creazione di vettori

Per creare un vettore, si usa la funzione `c()`:

```
> x <- c(2,3,5,7,11)
> x
[1]  2  3  5  7 11
```

Se si hanno tanti dati da scrivere, puo' essere più conveniente usare `scan()`:

```
> x <- scan()
1: 1
2: 6
3: 3
4: 4
5:
> x
[1] 1 6 3 4
>

> x <- scan()
1: 23 34 32
4: 33 88 44
7:
```

Esercizio: `scan()` può anche servire per leggere un vettore da un file. Con un editor, prova a creare il file data1.dat contenente i seguenti dati:

```
243 251 275 291 347 354 380 392
206 210 226 249 255 273 289 295 309
241 258 270 293
```

Puoi leggere il vettore con il comando:

```
> redcell <- scan("data1.dat")
```

Successioni
Si può usare la notazione a:b per creare vettori che sono sequenze di numeri:

```
> xx <- 1:10
> xx
 [1]  1  2  3  4  5  6  7  8  9 10

> xx <- 100:1
> xx
  [1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84 83
 [19]  82  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66 65
 [37]  64  63  62  61  60  59  58  57  56  55  54  53  52  51  50  49  48 47
 [55]  46  45  44  43  42  41  40  39  38  37  36  35  34  33  32  31  30 29
 [73]  28  27  26  25  24  23  22  21  20  19  18  17  16  15  14  13  12 11
 [91]  10   9   8   7   6   5   4   3   2   1
```

La stessa operazione può essere fatta con:

```
> xx<-seq(from=100, to=1)
> xx
```

Possono anche essere creati dei vettori che contengono elementi ripetuti

```
> rep(2,times=3)
[1] 2 2 2
> rep(2,3)
[1] 2 2 2
> a_c(rep(2,3),4,5,rep(1,5))
> a
 [1] 2 2 2 4 5 1 1 1 1 1
```

Ai vettori puoò essere applicata la stessa aritmetica di base che è stata applicata ai valori scalari:

```
> x_1:10
> x*2
 [1] 2 4 6 8 10 12 14 16 18 20
> x * x
 [1] 1 4 9 16 25 36 49 64 81 100
```

Possono essere eseguite operazioni logiche anche sui vettori

```
> x > 5
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

1.5.2 Estrazione degli elementi da un vettore

Gli elementi di un vettore possono essere estratti usando le parentesi quadre []:

```
> xx[7]
[1] 94
```

Si possono estrarre anche sottoinsiemi di elementi:

```
> xx[c(2,3,5,7,11)]
[1] 99 98 96 94 90
> xx[85:91]
[1] 16 15 14 13 12 11 10
> xx[91:85]
[1] 10 11 12 13 14 15 16
> xx[c(1:5,8:10)]
[1] 100  99  98  97  96  93  92  91
> xx[c(1,1,1,1,2,2,2,2)]
[1] 100 100 100 100  99  99  99  99
```

Ovviamente, sottoinsiemi di elementi possono essere salvati in nuovi vettori:

```
> yy <- xx[c(1,2,4,8,16,32,64)]
> yy
[1] 100  99  97  93  85  69  37
```

Se le parentesi quadre racchiudono un numero negativo, l'elemento corrispondente viene omesso dal vettore risultante:

```
> x <- c(1,2,4,8,16,32)
> x
[1]  1  2  4  8 16 32
> x[-4]
[1]  1  2  4 16 32
```

Alcune funzioni utili per la manipolazione di vettori

```
> x <- 26:3
> length(x)
[1] 24                  ...il numero di elementi
> max(x)
[1] 26                  ...il massimo
> min(x)
[1] 3                   ...il minimo
> sum(x)
[1] 348                 ...la somma dei valori in x
> prod(x)
[1] 2.016457e+26         ...il prodotto dei valori in x
> sort(x)
 [1]  3  4  ....   26    ...ordina il vettore x
```

1.6 Elementi di programmazione in R

Abbiamo già sottolineato come sia possibile aumentare il numero delle funzioni in R. Vediamo ora alcuni semplici esempi. Una funzione in R deve sempre iniziare con:

```
 >nomefunzione<-function(a,b,c,...){ }
```

in cui a, b, c, ... sono gli input che vengono passati alla funzione stessa. Se scriviamo:

```
>nomefunzione<-function(a=1,b=5,c=-6){ }
```

i parametri vengono automaticamente inizializzati con i valori che abbiamo scritto. Vediamo un primo esempio di come programmare in R

```
> cubo<-function(x){
y<-x^3
return(y)
}
```

o più semplicemente

```
> cubo<-function(x){
return(x^3)
}
```

e ora proviamola

```
> cubo(3)
[1] 27
```

sembra funzionare. La funzione `cubo` resterà in memoria fino alla chiusura del programma. Ora un esempio un po' più complicato

```
> media<-function(x){
y<-0
for (i in 1:length(x)) {
y<-y+x[i]
}
y<-y/length(x)
return(y)
}
```

Si noti la presenza di un ciclo `for` la cui sintassi è

```
> for (name in expr1) expr2
```

dove `name` è una variabile per il ciclo (in questo caso `i`), `expr1` è un vettore (in questo caso `1:length(x)`) di valori per `i` e `expr2` è un'espresssione che viene ripetuta tante volte quanti sono gli elementi di `expr1`.

```
> media(x)
[1] 14.5
```

Forse sarete un po' frustrati nel saper che si poteva semplicemente digitare

```
> mean(x)
[1] 14.5
```

Introdurremo altri elementi di programmazione piú avanti.
Esercizio: Scrivete una funzione che calcola la varianza di n osservazioni $X = X_1, \ldots, X_n$ utilizzando una delle due espressioni,

$$V(X) = \frac{\sum_{i=1}^{n}(X_i - M(X))^2}{n} = M(X^2) - [M(X)]^2,$$

dove $M(Y) = \sum_{i=1}^{n} Y_i/n$ e confrontate il risultato che ottenete con quello della funzione in R `var`.

1.7 Matrici

R consente anche di usare le matrici:

```
> x <- matrix(c(2,3,5,7,11,13),ncol=2)
> x
     [,1] [,2]
[1,]    2    7
[2,]    3   11
[3,]    5   13
```

NB: Bisogna specificare nrow o ncol per comunicare a R la dimensione della matrice. Se gli elementi di una matrice sono contenuti in un file, possiamo usare ancora `scan()`

```
1,24,32,36,33
2,16,44,34,33
3,20,31,43,32
4,23,35,37,35
5,27,40,40,31
6,19,43,32,37
```

Se questi elementi sono contenuti nel file matdata, li possiamo mettere in una matrice 6X5 con il comando:

```
> x2 <- scan('matdata',sep=',')
> mx <- matrix(x2,ncol=5, byrow=T)
> mx
     [,1] [,2] [,3] [,4] [,5]
[1,]    1   24   32   36   33
[2,]    2   16   44   34   33
[3,]    3   20   31   43   32
[4,]    4   23   35   37   35
[5,]    5   27   40   40   31
[6,]    6   19   43   32   37
```

Per estrarre da una matrice un elemento, bisogna specificarne le due coordinate:

```
> x[2,1]
[1] 3
> x[2,2]
[1] 11
```

Se non si mette una delle coordinate, si ottiene una intera riga/colonna:

```
> x[,1]
[1] 2 3 5
> x[3,]
[1]  5 13
```

Possono essere estratti sottoinsiemi di righe e/o colonne:

```
> x <- matrix(1:16,ncol=4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
```

```
[4,]    4    8   12   16
> x[c(1,4),c(3,4)]          Riga 1 e 4,
     [,1] [,2]              Col  3 e 4
[1,]    9   13
[2,]   12   16
```

La funzione dim indica la dimensione (numero di righe e numero di colonne) della matrice

```
> dim(mx)
[1] 6 5
```

1.8 Data-frames

Un data frame è un oggetto simile ad una matrice, ma usato per rappresentare dati sperimentali. Ogni riga rappresenta una unità statistica, ogni colonna rappresenta una variabile misurata sulle unità statistiche. Le colonne possono contenere variabili numeriche o categoriali.
Per leggere un insieme di dati di questo tipo si usa la funzione `read.table()`, che automaticamente controlla se le variabili sono numeriche o qualitative, se le righe e/o le colonne hanno etichette. Supponi che il file `Cherry.dat` sia così costituito:

```
8.3     70      10.3
8.6     65      10.3
8.8     63      10.2
10.5    72      16.4
10.7    81      18.8
10.8    83      19.7
...
...
```

Possiamo acquisirlo con il comando:

```
> Ciliegi <- read.table("I:/inferenza/Cherry.dat")
> Ciliegi                        (nota che Ciliegi e' diverso da ciliegi)
```

Il data frame è anche una matrice

```
> dim(Ciliegi)
[1] 31  3           (31 osservazioni e 3 variabili)
```

Se non specificati, i nomi delle tre variabili sono V1 V2 e V3:

```
> names(Ciliegi)
[1] "V1" "V2" "V3"
```

Si possono cambiare le etichette con il comando:

```
> names(Ciliegi) <- c('diametro','altezza','volume')
```

Alternativamente, potevano assegnare questi nomi direttamente in fase di lettura da file:

```
> Ciliegi<-read.table("I:/inferenza/Cherry.dat",
+ col.names=c("diametro","altezza","volume"))
```

Essendo il data frame una matrice, possiamo considerare, ad esempio, la terza variabile con:

```
Ciliegi[,3]
 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Tuttavia, la struttura di data frame permette un metodo migliore per indicare le variabili:

```
> Ciliegi$volume
 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Utilizziamo il comando `attach()` per comunicare ad R che le operazioni che faremo si riferiscono al dataframe `Ciliegi`:

```
> attach(Ciliegi)
> volume
 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Per avere delle statistiche di base sulle variabili contenute in `Ciliegi` possiamo usare la funzione `summary()`:

```
> summary(Ciliegi)
    diametro        altezza       volume
 Min.   : 8.30   Min.   :63   Min.   :10.20
 1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40
 Median :12.90   Median :76   Median :24.20
 Mean   :13.25   Mean   :76   Mean   :30.17
 3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30
 Max.   :20.60   Max.   :87   Max.   :77.00
```

Possiamo anche rappresentare graficiamente la distribuzione di una variabile ad esempio `diametro`, mediante un istogramma

```
hist(diametro)
```

oppure un diagramma a scatola (*box–plot*)

```
boxplot(diametro)
```

Per estrarre elementi da un data frame valgono le stesse regole valide per le matrici.

```
> altezza
 [1] 70 65 63 72 81 83 66 75 80 75 79 76 76 69 75 74 85 86 71 64 78 80 74 72 77
[26] 81 82 80 80 80 87

> Ciliegi[altezza > 80,]                    estrae un dataframe...
   diametro altezza volume                 ...di alberi che..
5      10.7      81   18.8                 ...  sono alti piu' di
6      10.8      83   19.7                 ... 70 piedi
17     12.9      85   33.8
18     13.3      86   27.4
26     17.3      81   55.4
27     17.5      82   55.7
31     20.6      87   77.0
```

Laboratorio 2

Statistica descrittiva I

2.1 Avvertenza sempre valida

Queste note contengono errori e inesattezze sicuramente non voluti ma comunque presenti. Fate sempre riferimento alla documentazione che accompagna il programma e alla guida in linea del programma.

2.2 Tabelle di frequenza

Consideriamo i dati relativi alle altezze per 65 persone di sesso maschile

```
> maschi<-scan('maschi.dat')
```

Proviamo a costruire una tabella di frequenza con il comando `table`

```
> table(maschi)
maschi
165 166 170 171 172 173 174 175 176 178 179 180 181 183 184 185 186
  1   2   5   1   3   3   3   7   1   8   1   8   3   3   1   6   2
187 188 190 192 193
  2   1   1   2   1
```

Se vogliamo avere una tabella di frequenza più significativa dovremo raccogliere in classi i dati. Prima formiamo le classi

```
> classi <-150+5*(0:10)
```

e poi assegnamo i maschi ad ogni classe con il comando

```
cut(maschi,breaks=classi)
```

e quindi creiamo la tabella di frequenza

```
> table(cut(maschi,breaks=classi))

(150,155] (155,160] (160,165] (165,170] (170,175] (175,180]
        0         0         1         7        17        18
(180,185] (185,190] (190,195] (195,200]
       13         6         3         0
```

Se vogliamo lasciare ad R l'onere di costruire le classi, c'è la possibilità di scegliere solo il numero di classi in cui vogliamo suddividere il nostro insieme di dati

```
> table(cut(maschi,breaks=10))

(165,168] (168,171] (171,173] (173,176] (176,179] (179,182]
        3         5         7        11         9        11
(182,185] (185,187] (187,190] (190,193]
        4        10         2         3
```

In questo caso abbiamo semplicemente una suddivisione in intervalli di uguale ampiezza del campo di variazione dei nostri dati. Dalla tabella delle frequenze si può ricavare quella delle frequenze cumulate tramite la funzione `cumsum`. Tale funzione calcola una somma cumulata degli elementi di un vettore creando un vettore di dimensione uguale al vettore cui viene applicata e i cui elementi contengono le somme cumulate parziali. Se vogliamo quindi ottenere le frequenze cumulate relative

```
> freqcum <- cumsum(table(cut(maschi,breaks=classi))/length(maschi))
> freqcum
 [1] 0.00000000 0.00000000 0.01538462 0.12307692 0.38461538
 [6] 0.66153846 0.86153846 0.95384615 1.00000000 1.00000000
```

2.3 Grafici

Costruiamo dapprima un istogramma di frequenza. Il comando più semplice è

```
> hist(maschi)
```

Come per `table` anche `hist` anche permette di stabilire il numero di classi in cui vogliamo rappresentare i nostri dati ad esempio

```
> hist(maschi,breaks=classi)
> hist(maschi,breaks=10)
```

Esercizio: Provate a far variare il numero di classi e osservate come varia l'istogramma corrispondente.
Se all'interno del comando `hist` aggiungiamo l'opzione `freq=F` che vuol dire di non usare le frequenze assolute otteniamo un grafico analogo ma con l'asse delle ordinate rappresentato dalle frequenze relative.

```
> hist(maschi,freq=FALSE)
```

Possiamo anche ottenere un grafico della funzione di ripartizione come segue. Dapprima aggiungiamo un limite inferiore alle classi

```
> freqcum <- c(0,freqcum)
```

Quindi con il comando `plot` rappresentiamo la funzione di ripartizione

```
> plot(classi,freqcum,type='s')
```

Il comando `plot` è molto versatile e permette di rappresentare nei modi più vari i dati. Nel nostro caso abbiamo rappresentato le coppie di coordinate del tipo (x,y) dove le ascisse x sono gli estremi delle classi e le ordinate y sono i valori delle frequenze cumulate. Si noti l'opzione `type='s'` che permette di costruire il grafico a 'gradini'. Infine consideriamo i dati, contenuti nel file `gelati.dat` provenienti da un indagine svolta su 40 ragazzi circa la loro preferenza per 5 tipi di gelato. Trattandosi di un vettore di stringhe leggiamolo con il comando `read.table`

```
> gelati<-read.table('gelati.dat')
> names(gelati)<-'tipo'
```

Ora il carattere è di tipo qualitativo e giustamente R non è in grado di produrre un istogramma

```
> hist(gelati)
Error in hist.default(gelati) : 'x' must be numeric
```

Proviamo a costruire un diagramma a torte

```
> piechart(table(gelati),names(table(gelati)),
      col=c("brown","pink","yellow","white","green"))
```

2.4 Indici di posizione e variabilità

R dispone di un'ampio insieme di funzioni che permettono di calcolare gli indici statistici più comunemente usati. Vediamo brevemente quali sono:

- `min` fornisce il valore della più piccola osservazione campionaria;
- `max` fornisce il valore della più grande osservazione campionaria;
- `range` fornisce i valori del minimo e del massimo dei dati campionari, cioè gli estremi del campo di variazione;
- `mean` calcola la media;
- `median` calcola la mediana;
- `var` calcola la varianza campionaria (fate molta attenzione a questo);
- `quantile` calcola i quantili, di qualsiasi ordine, di una distribuzione di dati;
- `summary` fornisce una tabella che riassume la maggior parte dei valori sopra esposti.

Vediamo solo due esempi: le funzioni `quantile` e `summary`. Nel suo uso più semplice si ha

```
> quantile(maschi)
  0%  25%  50%  75% 100%
 165  174  178  183  193
```

Se vogliamo avere solo, ad esempio, il 15-esimo e il 47-esimo percentile scriveremo

```
> quantile(maschi,probs=c(.15,.47))
15% 47%
172 178
```

La funzione `summary` ha come risultato

```
> summary(maschi)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  165.0   174.0   178.0   178.6   183.0   193.0
```

ovvero per quanto riguarda i quartili

```
> quantile(maschi,probs=c(0,.25,.50,.75,1))
  0%  25%  50%  75% 100%
 165  174  178  183  193
```

Le informazioni del comando `summary` possono essere rappresentate nel diagramma a scatola

```
> boxplot(maschi)
```

Esercizio: Provate a costruire una funzione che calcoli questi due indici di simmetria e curtosi

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - M(X)}{\mathrm{sqm}(X)}\right)^3, \quad \frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - M(X)}{\mathrm{sqm}(X)}\right)^4.$$

Esercizio: Provate a costruire una funzione che calcoli il quantile (almeno approssimativamente) di un vettore di osservazioni. Come suggerimento provate ad utilizzare la funzione `sort`.

2.5 Diagrammi di dispersione

Riprendiamo l'insieme di dati `Ciliegi` della precedente lezione. Se non era stato salvato, bisogna rileggerlo da file:

```
> Ciliegi <- read.table("Cherry.dat",
      col.names=c('diametro','altezza','volume'))
```

Possiamo comunicare a R che le operazioni che faremo d'ora in avanti si riferiscono al data frame `Ciliegi`:

```
> attach(Ciliegi)
```

I dati contengono, per 31 alberi di ciliego abbattuti, la misura del volume di legno ricavato dall'albero (`volume`), il diametro del tronco misurato a circa un metro dal suolo (`diametro`) e l'altezza dell'albero (`altezza`). Vogliamo indagare la relazione tra il volume di legno e il diametro. Possiamo disegnare il diagramma di dispersione di `diametro` e `volume` con il comando:

```
> plot(diametro,volume)
```

Con la funzione `points` possiamo anche evidenziare, magari con un colore a piacimento, alcuni punti ad esempio, le unità che presentano un diametro superiore a 15

```
> points(diametro[diametro > 15],volume[diametro > 15],
    col='blue',pch=20)
```

Con il comando

```
> identify(diametro,volume)
```

possiamo identificare a quale unità statistitiche appartengono i punti evidenziati.

2.6 Confronto tra popolazioni

I dati contenuti nel file `penicillin.dat`, si riferiscono ad un esperimento di produzione di penicillina tendente a valutare gli effetti di 4 modi differenti di produzione (A, B, C, D). Si è osservato precedentemente come la miscela adottata nella produzione sia piuttosto variabile e questo possa in qualche maniera influire sulla produzione. Quindi si è deciso di controllare anche l'effetto della miscela considerando 5 miscele (I, II, II, IV, V) e impiegando ognuna di queste nei quattro processi produttivi. Si noti come in questo caso l'interesse è rivolto a verificare se esiste una diversità d'effetto nei modi di produzione e non tanto nel tipo di miscela impiegata.

```
> pen <- read.table("penicillin.dat", header=TRUE)
> pen
   miscela modo penicillina
1        I    A          89
2        I    B          88
3        I    C          97
4        I    D          94
5       II    A          84
6       II    B          77
7       II    C          92
8       II    D          79
9      III    A          81
10     III    B          87
11     III    C          87
12     III    D          85
13      IV    A          87
14      IV    B          92
15      IV    C          89
16      IV    D          84
17       V    A          79
18       V    B          81
19       V    C          80
20       V    D          88
```

Questo è un insieme di dati un po' particolare in quanto abbiamo due variabili di tipo categoriale `miscela, modo` e una di tipo continuo `penicillina`. Notate cosa accade se tentiamo di usare il comando `summary`

```
> summary(pen)
 miscela modo   penicillina
 I  :4   A:5   Min.   :77
 II :4   B:5   1st Qu.:81
 III:4   C:5   Median :87
 IV :4   D:5   Mean   :86
 V  :4         3rd Qu.:89
               Max.   :97
```

Infatti la variabile, ad esempio, `modo`, è una variabile di tipo `factor`

```
> is.factor(pen$modo)
[1] TRUE
```

mentre la variabile `penicillina` è una variabile di tipo numerico

```
> is.numeric(pen$penicillina)
[1] TRUE
```

Comunichiamo a R che le operazioni che faremo d'ora in avanti si riferiscono al data frame `pen`:

```
> attach(pen)
```

Osserviamo ora i risultati della diversa applicazione del comando `plot`

```
> plot(modo,penicillina)
```

Confrontiamo ora i dati relativi alle altezze di un gruppo di femmine con il gruppo di maschi

```
> femmine<-scan('femmine.dat')
> boxplot(maschi,femmine)
```

Esercizio: Provate a costruire a partire da `maschi.dat` e `femmine.dat` un unico file contenente due colonne: `sesso` e `altezza`. Caricate il file in un unico `dataframe` di nome `stature` e disegnate il diagramma a scatola con il comando `boxplot(altezza,statura)`.

Laboratorio 3

Statistica descrittiva II

3.1 La regressione semplice

Riprendiamo in esame l'insieme di dati `Ciliegi` delle precedenti lezioni. Se non era stato salvato, bisogna rileggerlo da file:

```
> Ciliegi <- read.table("Cherry.dat",
+ col.names=c('diametro','altezza','volume'))
```

Possiamo comunicare a R che le operazioni che faremo d'ora in avanti si riferiscono al data frame `Ciliegi`:

```
> attach(Ciliegi)
```

I dati contengono, per 31 alberi di ciliego abbattuti, la misura del volume di legno ricavato dall'albero (`volume`), il diametro del tronco misurato a circa un metro dal suolo (`diametro`) e l'altezza dell'albero (`altezza`). Vogliamo indagare la relazione tra il volume di legno e il diametro. Possiamo fare un grafico di `diametro` e `volume` con il comando:

```
> plot(diametro,volume)
```

Il numero di osservazioni è:

```
> n<-dim(Ciliegi)[1]
```

Calcoliamo i coefficienti della regressione

$$\texttt{volume} = \alpha + \beta * \texttt{diametro} + \varepsilon$$

mediante il metodo dei minimi quadrati.

```
> beta<-(sum(diametro*volume)/n-mean(volume)*mean(diametro))/
+ (mean(diametro^2)-mean(diametro)^2)
> beta
[1] 5.065856
```

Questo è equivalente a

```
>beta<-cov(diametro,volume)/var(diametro)
>beta
[1] 5.065856
```

La stima di α è quindi pari a

```
>alpha<-mean(volume)-beta*mean(diametro)
>alpha
[1] -36.94346
```

Rappresentiamo la retta calcolata nel grafico, con il comando

```
> abline(alpha,beta,lty="dashed")
```

Il comando `abline(a,b)` traccia una retta nel grafico corrente con intercetta `a` e coefficiente angolare `b`. L'opzione `lty` è un'opzione grafica generale (valida ad esempio anche per il comando `plot`) e definisce il tipo di linea. Assume valori: "blank", "solid" (default), "dashed", "dotted", "dotdash", "longdash" o "twodash", oppure rispettivamente i numeri da 0 a 7. L'opzione "blank" (o 0) traccia una linea invisibile.

I valori predetti dal modello sono:

```
> valori.predetti <- alpha+beta*diametro
> valori.predetti
 [1]  5.103149  6.622906  7.636077 16.248033 17.261205 17.767790 18.780962
 [8] 18.780962 19.287547 19.794133 20.300718 20.807304 20.807304 22.327061
[15] 23.846818 28.406089 28.406089 30.432431 32.458774 32.965360 33.978531
[22] 34.991702 36.511459 44.110244 45.630001 50.695857 51.709028 53.735371
[29] 54.241956 54.241956 67.413183
```

Ovviamente, i valori predetti dal modello sono quelli che stanno sulla retta di regressione. Possiamo aggiungere questi punti nel grafico precedente con il comando

```
> points(diametro,valori.predetti,pch="X")
```

Il comando `points` aggiunge i punti in un plot esistente. L'opzione `pch` permette di scegliere il tipo di carattere da utilizzare nel grafico per identificare un punto (in questo caso si è scelto X).
I residui sono dati dalla differenza tra i valori osservati e quelli stimati

```
> residui <- volume - valori.predetti
> residui
 [1]  5.1968508  3.6770939  2.5639226  0.1519667  1.5387954  1.9322098
 [7] -3.1809615 -0.5809615  3.3124528  0.1058672  3.8992815  0.1926959
[13]  0.5926959 -1.0270610 -4.7468179 -6.2060887  5.3939113 -3.0324313
[19] -6.7587739 -8.0653595  0.5214692 -3.2917021 -0.2114590 -5.8102436
[25] -3.0300006  4.7041430  3.9909717  4.5646292 -2.7419565 -3.2419565
[31]  9.5868168
```

Il coefficiente di determinazione è dato da

```
> R2<- 1-var(residui)/var(volume)
> R2
[1] 0.9353199
```

Consideriamo ora un altro diagramma di dispersione, utilizzando i logaritmi delle variabili `volume` e `diametro`.

```
> plot(log(diametro),log(volume))
```

E cerchiamo di calcolare la retta dei minimi quadrati. Notiamo che, nella regressione precedente, abbiamo utilizzato la formula

$$\log(\texttt{volume}) = \alpha_1 + \beta_1 \log(\texttt{diametro})\,.$$

Questo, per le proprietà dei logaritmi, significa che

$$\log(\texttt{volume}) = \log(e^{\alpha_1}\texttt{diametro}^{\beta_1})\,.$$

e quindi che

$$\texttt{volume} = e^{\alpha_1}\texttt{diametro}^{\beta_1}\,.$$

Ovvero abbiamo 'linearizzato' il modello.
Quindi, supponendo di aver salvato negli oggetti `alpha1` e `beta1` i coefficienti α_1 e β_1:

```
> beta1<-cov(log(volume),log(diametro))/var(log(diametro))
> alpha1<-mean(log(volume))-beta1*mean(log(diametro))
```

possiamo ottenere i valori predetti secondo questo modello nella scala originale delle variabili:

```
> valori.predetti1 <- exp(alpha1)*diametro^beta1
```

e poi confrontare i risultati graficamente con quelli precedenti:

```
> attach(Ciliegi)
> plot(diametro,volume)
> abline(alpha,beta,lty="dashed")
> lines(diametro,valori.predetti1)
> detach(Ciliegi)
```

La linea continua sembra adattare meglio le osservazioni negli estremi. La funzione `lines` serve per aggiungere linee su un grafico esistente (vedi `help(lines)` per ulteriori dettagli).

Se definiamo i residui del modello in scala logaritmica, nella scala originaria delle variabili, come

```
> residui1<-volume-valori.predetti1
```

vediamo che la varianza di questi residui è minore rispetto a quella del primo modello utilizzato:

```
> var(residui1)
[1] 10.56006
> var(residui)
[1] 17.47675
```

3.2 La funzione `lm`

I dati riportati nel file `cement.dat` si riferiscono ad uno studio sulla resistenza del cemento alla tensione. La resistenza dipende, tra le altre cose, dal tempo di essicazione. Nello studio si è misurata la resistenza alla tensione di lotti di cemento sottoposti a diversi tempi di essicazione. Si vuole studiare la relazione tra la resistenza alla tensione e il tempo di essicazione.
In questo caso il tempo è la variabile indipendente e la resistenza è la variabile dipendente

```
> cement <- read.table("cement.dat",col.names=c("tempo", "resist"))
> attach(cement)
```

Proviamo a disegnare un diagramma di dispersione:

```
> plot(tempo,resist)
```

oppure mediante

```
> plot(resist ~ tempo)
```

Si noti che la sintassi utilizzata è differente e sottolinea, per così dire, come la variabilità di `resist` dipenda (`~`) da `tempo`.
Il grafico indica chiaramente una relazione non lineare. Un modello del tipo $\texttt{resist} = \alpha + \beta\texttt{tempo} + \varepsilon$ non parrebbe appropriato.
Tuttavia proviamo a considerarlo e utilizziamo una nuova funzione `lm`. Questa calcola i coefficienti secondo il metodo dei minimi quadrati [1].

```
> fit <- lm( resist ~ tempo )
```

Notate la sintassi di `resist~tempo` . A sinistra di `~` vi è il regressore a destra la variabile esplicativa. La costante α è automaticamente inclusa. Abbiamo creato un oggetto, che abbiamo chiamato `fit`, di tipo `lm`. Un oggetto è qualcosa di più complicato di un vettore o di una matrice. È una lista di elementi su cui si può applicare una serie di funzioni. Ad esempio con il comando seguente vediamo i risultati dell'adattamento.

```
> summary(fit)

Call:
lm(formula = resist ~ tempo)

Residuals:
     Min       1Q   Median       3Q      Max
-11.4452  -2.0034   0.7848   3.4385   8.5059
```

[1] In realtà questa funzione fornisce le stime di massima verosimiglianza per un modello lineare quando gli errori si distribuiscono come una variabile casuale normale, ma questo lo si vedrà nei corsi successivi

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  22.5871     1.6831  13.420 3.84e-11 ***
tempo         0.6581     0.1187   5.546 2.38e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.739 on 19 degrees of freedom
Multiple R-Squared: 0.6182,     Adjusted R-squared: 0.5981
F-statistic: 30.76 on 1 and 19 DF,  p-value: 2.382e-05
```

Come si può osservare, otteniamo diverse statistiche e più in generale quantità. Di queste ci interessano i coefficienti ottenuti con i minimi quadrati riportati nella colonna `Estimate`. **Esercizio:** Si provi a calcolare quanto sopra non utilizzando la funzione `lm`.

L'oggetto `fit` contiene più quantità:

```
> names(fit)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

Proviamo a considerare i residui e i valori previsti dal modello.

```
> res <- resid(fit)
> fit.val <- fitted(fit)
```

I residui contenuti in `fit` e che abbiamo appena salvato nel vettore `res` sono le quantità $e_i = y_i - \hat{y}_i$.

Un grafico che possiamo fare per verificare la linearità della relazione, è il grafico

```
> plot(res~tempo)
```

Dal grafico vediamo che il modello non è soddisfacente. Possiamo allora cercare qualche trasformazione delle variabili che ci riporti ad una relazione più lineare. Generalmente, si preferisce trasformare le variabili esplicative. Proviamo allora a trasformare la variabile `tempo`. Si noti l'utilizzo della funzione `par` con l'opzione `mfrow`. Questo permette di visualizzare in un un'unica finestra $2 \times 2 = 4$ grafici.

```
> par(mfrow=c(2,2))
> plot(log(tempo), resist)
> plot(1/(tempo), resist)
> plot(1/sqrt(tempo), resist)
> plot(sqrt(tempo), resist)
> par(mfrow=c(1,1))
```

Le prime tre trasformazioni pare linearizzino in maniera soddisfacente la relazione, in particolare la terza. Adottiamo quindi la trasformazione

```
> x <- 1/sqrt(tempo)
```

e procediamo specificando il modello di regressione

$$\texttt{resist} = \alpha + \frac{\beta}{\texttt{tempo}} + \varepsilon$$

```
> fit2 <- lm( resist ~ x )

> summary(fit2)

Call:
lm(formula = resist ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-3.79469 -1.25666 -0.05666  1.89544  3.10531

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   45.655      1.023   44.63  < 2e-16 ***
x            -32.599      1.764  -18.48 1.34e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.133 on 19 degrees of freedom
Multiple R-Squared: 0.9473,     Adjusted R-squared: 0.9445
F-statistic: 341.4 on 1 and 19 DF,  p-value: 1.337e-13
```

Proviamo a considerare di nuovo i residui.

```
> res <- resid(fit2)

> plot(res~tempo)
```

Dal grafico osserviamo un miglioramento sensibile. Per terminare digitiamo

```
> detach(cement)
```

Laboratorio 4

Probabilità

Lasciata la parte dedicata alla Statistica descrittiva, da qui in avanti ci occuperemo di Probabilità e, in conclusione dei nostri incontri, di Inferenza: in particolare in questa quarta sessione tratteremo le distribuzioni di probabilità più comuni, il loro utilizzo e la simulazione di variabili (pseudo) casuali. Per iniziare però, è forse utile illustrare alcune funzioni di R per effettuare alcuni conteggi di calcolo combinatorio.

4.1 Calcolo combinatorio

Il fattoriale di un numero n, indicato con $n!$ cioè il prodotto di un intero positivo n per tutti gli interi positivi più piccoli di questo fino ad arrivare all'1, ovvero

$$n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 1$$

si ottiene semplicemente utilizzando la funzione `prod` $(1:n)$. Ad esempio, 5!=120 si ottiene col comando

```
> prod(1:5)
[1] 120
```

Allo stesso modo si possono calcolare le Disposizioni semplici di n oggetti a gruppi di k, $D_{n,k}$, ovvero il prodotto di un intero positivo n per i primi $(k-1)$ interi positivi piŏiccoli di questo, che come sappiamo fornisce tutti i gruppi che si possono formare prendendo k tra n oggetti distinti, in modo che ogni gruppo differisca dai restanti o per almeno un elemento o per l'ordine con cui gli oggetti sono disposti; è sufficiente calcolare `prod(n:(n-k+1))`. Ad esempio $D_{6,3}$ è

```
> prod(6:(6-3+1))
[1] 120
```

Esercizio: Provare a scrivere una funzione per calcolare le combinazioni di n oggetti a gruppi di k, indicate anche col simbolo $\binom{n}{k}$ che prende il nome di coefficiente binomiale.

$$\begin{aligned} C_{n,k} &= \frac{D_{n,k}}{k!} = \frac{n \cdot (n-1) \ldots (n-k+1)}{k(k-1) \ldots 1} \\ &= \frac{n!}{k!(n-k)!} = \binom{n}{k}. \end{aligned}$$

gruppo In R esistono due funzioni per calcolare il coefficiente binomiale $\binom{n}{k}$ e il suo logaritmo che si chiamano rispettivamente `choose` e `lchoose`. Il coefficiente binomiale $\binom{4}{2}$ è quindi

```
> choose(4,2)
[1] 6
```

e il suo logaritmo

```
> lchoose(4,2)
[1] 1.791759
```

4.2 Distribuzioni di probabilità

R consente di gestire tutte le distribuzioni di probabilità più comuni di cui fornisce automaticamente la densità, la funzione di ripartizione, quantili e generazione di numeri casuali. Alcune delle distribuzioni disponibili sono:

R	Distribuzione	Parametri	Defaults
norm	normale	mean, sd	0, 1
lnorm	log-normal	mean, sd	0, 1
chisq	chi-square	df	0, 1
f	F	df1, df2	- -
gamma	Gamma	shape	-
t	Student's t	df	-
unif	Uniform	min, max	0, 1
binom	Binomiale	n, p	- -
pois	Poisson	λ	-

Prendiamo ad esempio la distribuzione Binomiale: se $X \sim \text{Bin}(n = 10, p = 0.2)$ la probabilità che X assuma il valore $x = 2$ è data dalla funzione

```
> dbinom(2,10,0.2)
[1] 0.3019899
```

Come si vede per ottenere la densità della variabile casuale in $x = 3$ è stato semplicemente aggiunto il prefisso `"d"` al nome della distribuzione `binom`; la sintassi è quindi `dbinom(x,n,p)`. Per calcolare la funzione di ripartizione ossia $P(X \leq x)$ il prefisso da utilizzare è `"p"`, con sintassi del tutto simile; per ottenere invece la probabilità dell'evento complementare $P(X > x)$ si deve aggiungere l'opzione `lower.tail=F`.

```
> pbinom(2,10,0.2)
[1] 0.6777995
> pbinom(2,10,0.2,lower.tail=F)
[1] 0.3222005
```

Allo stesso modo per ottenere i quantili delle distribuzioni il prefisso da utilizzare è `"q"`. Sempre per fare un esempio nell'ambito della Binomiale la sintassi da usare è `qbinom`$(\alpha, \texttt{n}, \texttt{p})$ dove con α si è indicata la probabilità in corrispondenza della quale si vuole ottenere il quantile.

```
> qbinom(0.45,3,1/3)
[1] 1
```

che è corretto. Infatti, il quantile in corrispondenza di 0.45 è il valore 1 poiché cumulando fino a 1 la densità di probabilità di una binomiale $n = 3$ e $p = 1/3$ si ha

```
> dbinom(1,3,1/3)
[1] 0.4444444
```

4.3 Simulazione di variabili casuali

Per ottenere la generazione di una serie di numeri casuali provenienti dalle variabili casuali disponibili in R si deve anteporre il prefisso "r" al nome che identifica la distribuzione. Se vogliamo dunque simulare 1000 valori da una distribuzione normale standard, la sintassi è

```
> x_rnorm(1000)
```

A questo punto possiamo pensare di produrre una stima della densità del vettore x, attraverso la funzione `density()`

```
> plot(density(x))
```

e di metterla a confronto con la densità vera di una variabile casuale normale standard utilizzando il comando `curve()`, che serve a tracciare il grafico di una qualunque funzione,

```
> par(mfrow=c(1,2))
> plot(density(x))
> curve(dnorm(x))
> par(mfrow=c(1,1))
```

4.4 Confronto tra distribuzioni

I dati contenuti nel file `babyboom.dat` si riferiscono ad alcune variabili registrate su 44 bambini nati in un periodo di tempo di 24 ore all'ospedale di Brisbane in Australia il 18 dicembre 1997. Le quattro variabili presenti nel data-set sono: `tempo`, il momento preciso del giorno in cui ogni singola nascita è avvenuta; `sesso`, sesso del bambino nato (1 per femmine e 2 per i maschi); `peso`, peso alla nascita in grammi; `minuti`, numero di minuti dopo la mezzanotte in cui è avvenuta la nascita. Questo insieme di dati è utile per dimostrare come alcune distribuzioni note come la normale, la poisson, la binomiale e la geometrica possano essere utilizzate per "modellare" situazioni tratte dalla vita reale.
Consideriamo anzitutto la variabile `peso` che potrebbe essere modellata da una distribuzione normale e proviamo preliminarmente a studiare graficamente la distribuzione dei dati. Vediamo per incominciare l'istogramma e il boxplot.

```
> hist(peso)
> hist(peso,nclass=10)
> hist(peso,nclass=10,prob=T)
```

È evidente una netta asimmetria a sinistra, che possiamo confermare o meno costruendo un boxplot.

```
> boxplot(peso)
```

In effetti, l'impressione che esista una rilevante asimmetria rimane confermata anche osservando il boxplot e ciò è probabilmente imputabile a un gruppo di bambini nati prematuramente e quindi sottopeso oppure aventi malformazioni gravi o malattie. Questi sembrano i fattori che principalmente contribuiscono alla non-normalità. Verifichiamo comunque con gli strumenti di cui disponiamo se utilizzare per la modellazione dei dati una distribuzione normale è una buona idea o meno.

```
> qqnorm(peso)
> qqline(peso)
```

Come si vede osservando il grafico c'è un netto scostamento dalla retta a conferma della asimmetria rilevata. Proviamo a distinguere tra pesi dei maschi e delle femmine, che notoriamente alla nascita mostrano differenze anche rilevanti, per evidenziare eventuali diversità.

```
> peso.m_peso[sesso==2]
> peso.f_peso[sesso==1]
> peso.m
 [1] 3554 3838 3625 2846 3166 3520 3380 3294 3521 2902 2635 3920 3690 3783 3345
[16] 3034 3300 3428 4162 3630 3406 3402 3736 3370 2121 3150
> peso.f
 [1] 3837 3334 2208 1745 2576 3208 3746 3523 3430 3480 3116 3428 2184 2383 3500
[16] 3866 3542 3278
> par(mfrow=c(1,2))
> hist(peso[sesso==1])
> hist(peso[sesso==2])
> boxplot(peso[sesso==1])
> boxplot(peso[sesso==2])
```

In effetti la distribuzione dei pesi dei maschi appare meno asimmetrica rispetto a quella delle femmine che mostra una decisa asimmetria a sinistra. Proviamo ora a verificare separatamente per i due gruppi distinti secondo il sesso l'adattamento normale.

```
> qqnorm(peso.m)
> qqline(peso.m)
> qqnorm(peso.f)
> qqline(peso.f)
```

La differenza tra i due gruppi è rilevante, anche se l'adattamento normale in complesso non appare soddisfacente.

Laboratorio 5

La stima puntuale e la stima intervallare

In questa sezione ci occuperemo della stima puntuale e della stima intervallare. In particolare, verificheremo empiricamente due importanti teoremi del Calcolo delle porbabilità la legge dei grandi numeri e il teorema limite centrale che sono degli strumenti base per valutare le proprietà degli stimatori

5.1 La legge dei grandi numeri

Ricordiamo ora la legge forte (debole) dei grandi numeri.
Se X_i $i = 1, \ldots$ è una successione di variabile casuali indipendenti e identicamente distribuite (i.i.d.) di media μ e varianza σ^2 finita allora la media campionaria

$$\overline{X}_n = \frac{\sum_{i=1}^{n} X_i}{n}$$

converge, quasi certamente (in probabilità) al valore μ.
Questo teorema può essere verificato operativamente utilizzando R : fissata la variabile casuale di riferimento, si possono generare n valori casuali, calcolarne la media e iterare il procedimento aumentando n di volta in volta. Supponiamo perciò di iniziare con $n = 10$ replicazioni da $X \sim Poisson(5)$ e calcoliamo la media.

```
> set.seed(30)
> x<-rpois(10,5)
> mean(x)
[1] 4.7
```

Raddoppiamo ora le replicazioni

```
> x<-c(x,rpois(10,5))
> mean(x)
[1] 4.6
```

e raddoppiamo ancora

```
> x<-c(x,rpois(20,5))
> mean(x)
[1] 4.95
```

Come possiamo vedere, la media campionaria oscilla intorno al vero valore della media. Proviamo ad aumentare ulteriormente la sequenza di mille replicazioni

```
> x<-c(x,rpois(1000,5))
> mean(x)
[1]  4.970192
```

e di altre 10000 replicazioni

```
> x<-c(x,rpois(10000,5))
> mean(x)
[1]   5.008514
```

I risultati ottenuti confermano il fatto che la media campionaria si avvicini al vero valore della media della distribuzione campionaria di riferimento, al crescere del numero di replicazioni. È comunque chiaro che vi possono essere delle oscillazioni e velocità di convergenza diverse a seconda del tipo di generatore di numeri casuali che si utilizza. Rappresentiamo in un grafico l'andamento della successione $\overline{X}_n$, $n = 1, \ldots$.

```
> nobs<-(1:length(x))
> m<-cumsum(x)/nobs
> plot(nobs,m,type='l')
```

Cosa potete notare ?

5.2 Il Teorema Limite Centrale

Ricordiamo ora l'enunciato del Teorema Limite Centrale.
Se X_i $i = 1, \ldots$ è una successione di variabile casuali indipendenti e identicamente distribuite (i.i.d.) di media μ e varianza σ^2 finita allora

$$\overline{Z}_n = \frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}}$$

converge in distribuzione ad una v.c. $\mathcal{N}(0, 1)$.
Anche in questo caso possiamo provare a verificare empiricamente questo teorema mediante R : il procedimento è leggermente diverso rispetto a prima, nel senso che ora dobbiamo verificare che la distribuzione empirica di una variabile casuale (la media campionaria) tenda ad avere una determinata distribuzione (la $\mathcal{N}(\mu, \sigma^2/n)$). Per fare questo è necessario produrre una serie di M campioni di dimensione n fissata su ciascuno dei quali calcolare la media $\overline{X}_n$. Più elevato è il numero M di campioni generati, più valori della media campionaria avremo a disposizione per costruire la distribuzione empirica, più accurata sarà quest'ultima. Viceversa, più elevata è la dimensione del singolo campione, maggiormente accurata sarà la stima della media campionaria per singolo campione generato.
Generiamo un campione di $n = 10$ variabili casuali $X \sim Poisson(5)$ di cui calcoliamo la media

```
> x<-rpois(10,5)
> mean(x)
[1] 5
```

Per generare altri campioni di questo tipo (diciamo in numero di M=1000) aventi dimensione $n = 10$, conviene predisporre uno schema iterativo che prevede l'utilizzo della funzione `for`,

```
> set.seed(15)
> m<-mean(rpois(10,5))
> for(i in 1:999) m<-c(m,mean(rpois(10,5)))
```

In questo caso, si è calcolata una media campionaria su un generico campione, si è inserita nell'oggetto m e si è poi lanciato un ciclo di 999 iterazioni accodando alla prima stima ottenuta tutte le altre generate in sequenza, ottenendo quindi un vettore avente 1000 valori della media per i 1000 campioni generati.
Confrontiamo graficamente le due distribuzioni, quella empirica e quella teorica (si tenga presente che nel caso della v.c. Poisson media e varianza coincidono),

```
> par(mfrow=c(1,2))
> hist(m,freq=FALSE,ylim=c(0,1))
> curve(dnorm(x,mean=5,sd=sqrt(5/10)),add=TRUE)
```

Come si può vedere queste distribuzioni differiscono ancora tra loro e non di poco. A questo punto, effettuiamo sempre $M = 1000$ iterazioni, ma aumentiamo la dimensione dei campioni generati passando da 10 a 100.

```
> m<-mean(rpois(100,5))
> for(i in 1:999) m<-c(m,mean(rpois(100,5)))
```

Vediamo il confronto grafico, fra le due distribuzioni,

```
> hist(m,freq=FALSE,ylim=c(0,2))
> curve(dnorm(x,mean=5,sd=sqrt(5/100)),add=TRUE)
```

Il confronto grafico conferma quanto previsto dal teorema limite centrale evidenziando che in effetti la distribuzione empirica della media campionaria tende effettivamente ad avere distribuzione normale.

Esercizio 1: Verificare il teorema del limite centrale utilizzando anziché la distribuzione di Poisson, la Binomiale e la Normale. Che cosa vi attendete a priori?

Esercizio 2: Verificare il teorema limite centrale aumentando il numero dei campioni generati anziché la loro dimensione e confrontare con i risultati ottenuti nell'altro modo.

5.3 Intervalli di confidenza

Vediamo ora come si possono costruire gli intervalli di confidenza, per la media e la varianza.
Riprendiamo in esame l'insieme di dati `babyboom` della precedente lezione. Se non era stato salvato, bisogna rileggerlo da file:

```
> babyboom <-
read.table("babyboom.dat",col.names=c("tempo","sesso","peso","minuti"))
> attach(babyboom)
```

Ci interessa la variabile peso. Supponiamo che sia distribuita secondo una variabile casuale di tipo normale avente parametri μ e σ^2 e costruiamo un intervallo di confidenza per la media μ al livello 0.95, supponendo dapprima che la varianza σ^2 sia nota e successivamente che sia incognita. Come sappiamo, nel primo caso in cui la varianza è nota la variabile casuale

$$\frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}}$$

si distribuisce come una normale di media 0 e varianza 1, nel secondo caso, in cui l'ignota varianza è stimata da

$$S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2,$$

la variabile casuale

$$\frac{\overline{X}_n - \mu}{S_n/\sqrt{n}}$$

è una t di Student con $n-1$ gradi di libertà. In questi due casi alternativi gli intervalli di confidenza al livello $(1-\alpha)$ sono rispettivamente:

1. $x_n - \frac{\sigma}{\sqrt{n}} z_{1-\alpha/2}, \quad x_n + \frac{\sigma}{\sqrt{n}} z_{1-\alpha/2}$
2. $x_n - \frac{s_n}{\sqrt{n}} t_{n-1,1-\alpha/2}, \quad x_n + \frac{s_n}{\sqrt{n}} t_{n-1,1-\alpha/2}$

dove $z_{1-\alpha/2}$ e $t_{n-1,1-\alpha/2}$ sono i quantili di livello $1-\alpha/2$ della normale e della t di Student con $n-1$ gradi di libertà.
Supponiamo prima che la varianza sia nota (pari a 278818.3) e che α sia 0.05: calcoliamo dunque il quantile della normale e estremo inferiore e superiore dell'intervallo di confidenza,

```
> mean(peso)
[1] 3275.955
> qnorm(0.975)
[1] 1.959964
> mean(peso)-qnorm(0.975)*sqrt(278818.3/44)
[1] 3119.934
> mean(peso)+qnorm(0.975)*sqrt(278818.3/44)
[1] 3431.975
```

Esercizio: Costruire una funzione che, a partire da un vettore di dati, calcoli intervalli di confidenza per la media a qualunque livello, per un valore noto di σ.

Nel caso in cui la varianza sia ignota, si procede calcolando il quantile della t con (44-1) gradi di libertà e stimando la varianza dai dati,

```
> qt(0.975,43)
[1] 2.016692
> mean(peso)-qt(0.975,df=43)*sqrt(var(peso)/44)
[1] 190333.7
> mean(peso)+qt(0.975,df=43)*sqrt(var(peso)/44)
[1] 447602
```

Per esercizio, provare a verificare che l'intervallo di confidenza si allarga all'aumentare del livello di confidenza. Costruiamo ora un intervallo di confidenza per la varianza. Si ricorda che la variabile casuale

$$\frac{(n-1)S_n^2}{\sigma^2}$$

si distribuisce come un χ^2_{n-1} gradi di libertà. Pertanto gli estremi dell'intervallo di confidenza assumono questo aspetto:

$$\frac{(n-1)S_n^2}{\chi^2_{(n-1),1-\alpha/2}}, \quad \frac{(n-1)S_n^2}{\chi^2_{(n-1),\alpha/2}}$$

Per quanto concerne dunque l'intervallo di confidenza per la varianza della variabile peso del `data-frame` babyboom, si devono calcolare i quantili $(1-\alpha/2)$ e $\alpha/2$ di una v.c. χ^2 avente (44-1) gradi di libertà. La sintassi è quindi la seguente

```
> qchisq(0.975,43)
[1] 62.99036
> qchisq(0.025,43)
[1] 26.78537
> 43*var(peso)/qchisq(0.975,43)
[1] 194.5991
> 43*var(peso)/qchisq(0.025,43)
[1] 457.6328
```

Concludiamo osservando che nel linguaggio R è presente una collezione di funzioni, ovvero una `library`, grazie alle quali è possibile ridurre il numero di istruzioni per calcolare intervalli di confidenza e per condurre, come vedremo, verifiche d'ipotesi. Per utilizzare questa `library` è sufficiente scrivere

```
library(ctest)
```

e quindi

```
> t.test(peso,alternative = "two.sided", conf.level = 0.95)
        One Sample t-test
```

```
data:  peso
t = 41.1532, df = 43, p-value = < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 3115.418 3436.491
sample estimates:
mean of x
 3275.955
```

Laboratorio 6

La Verifica d'ipotesi

6.1 Il test t ad un campione

Si supponga di aver determinato il peso, espresso in grammi, di alcuni granelli di polvere rilevati su una piastra di silicio. Si suppone che il peso sia distribuito secondo una variabile casuale normale di parametri μ e σ^2 . I dati sono contenuti nel file `polveri.dat`

```
>polveri<-scan("polveri.dat")
```

Vogliamo effettuare un test sulla media della popolazione per verificare l'ipotesi che la media in incognita sia pari a 1. 07. Scegliamo un'alternativa bilaterale

$$\begin{cases} H_0: & \mu = 1.07 \\ H_1: & \mu \neq 1.07 \end{cases}$$

Supporremo di non conoscere il valore della varianza σ^2 e quindi il test sarà basato sulla statistica

$$t = \frac{\overline{X} - \mu}{s/\sqrt{n}}$$

che sotto l'ipotesi nulla si distribuisce come una v.c. t di Student con 11 gradi di libertà . Determiniamo la regione di rifiuto per tale test usando i quantili della distribuzione e calcoliamo il valore della statistica test.

```
> test<-(mean(polveri)-1.07)/sqrt(var(polveri)/12)
> test
[1] 2.310533
```

Il quantile 0.975 della distribuzione è dato da

```
> qt(0.975,11)
[1] 2.200985
```

e pertanto rifiutiamo l'ipotesi nulla ad un livello di significatività prefissato di 0.05. Il livello di significatività prefissato è dato da

```
> 2*(1-pt(test,11))
[1] 0.04125998
```

La medesima verifica d'ipotesi può essere condotta utilizzando una funzione del *package* `ctest`

```
> library(ctest)
>  t.test(polveri,mu = 1.07,conf.level=0.95)

        One Sample t-test

data:  polveri
t = 2.3105, df = 11, p-value = 0.04126
alternative hypothesis: true mean is not equal to 1.07
95 percent confidence interval:
 1.099159 2.270841
sample estimates:
mean of x
    1.685
```

R permette di e ettuare anche test ad una coda del tipo

$$\begin{cases} H_0: & \mu = 1.07 \\ H_1: & \mu > 1.07 \end{cases}$$

e

$$\begin{cases} H_0: & \mu = 1.07 \\ H_1: & \mu < 1.07 \end{cases}$$

ed i comandi relativi sono

```
> t.test(polveri,mu = 1.07, alternative="greater", conf.level=0.95)
> t.test(polveri,mu = 1.07, alternative="less",conf.level=0.95)
```

Quale sarà la vostra decisione in questo caso ?

6.2 L'analisi della varianza ad un fattore

I dati riportati nel file `sturdy.dat` si riferiscono ad un esperimento effettuato per studiare la resistenza allo strappo di diverse marche di impermeabili. Capi di cinque diverse marche sono stati sottoposti alla stessa sollecitazione. La resistenza allo strappo è stata misurata con il tempo (in minuti e frazioni decimali di minuti) intercorso tra la sollecitazione e lo strappo.

1. Esistono differenze nella resistenza tra le varie marche?
2. È possibile dire se le marche A, B, C sono più resistenti, in media, delle marche D, E?

```
Marca A: 2,34  2,46  2,83  2,04  2,69
Marca B: 2,64  3,00  3,19  3,83
Marca C: 2,61  2,07  2,80  2,58  2,98  2,30
Marca D: 1,32  1,62  1,92  0,88  1,50  1,30
Marca E: 0,41  0,83  0,58  0,32  1,62
```

Acquisiamo i dati.

```
> dati <- read.table("sturdy.dat")
> names(dati) <- "time"
```

Dobbiamo costruire il vettore indicatore delle cinque marche ed unirlo al dataframe.

```
> group <- c(rep("A",5), rep("B",4), rep("C",6), rep("D",6), rep("E",5))
> group <- factor(group)
> sturdy <- data.frame(dati, group)
```

Possiamo provare a fare un po' di analisi esplorativa, anche se abbiamo pochissime osservazioni per gruppo.

```
> attach(sturdy)
> plot(time~group)
```

Il grafico mostra una evidente differenza tra i gruppi in termini di mediane. (Il gruppo B ha la mediana più elevata).
Anche se a livello assolutamente indicativo, la variabilità entro i gruppi parrebbe comparabile.
Considerato il numero ridotto di osservazioni per gruppo, le distribuzioni dentro i gruppi appaiono sufficientemente simmetriche (con qualche eccezione per il gruppo E). Questo, unito alle osservazioni circa la diversità delle mediane, ci rende inclini a pensare che esista una differenza anche in termini di medie. Quindi ci si fa l'idea che il test si concluda con il rifiuto dell'ipotesi nulla.
La sola simmetria non garantisce comunque la normalità dei dati. Per completare l'analisi della normalità, possiamo utilizzare anche i `qqnorm` (vedi comunque quanto detto nel laboratorio 5 circa l'efficacia dei `qqnorm` quando si hanno poche osservazioni).

```
> par(mfrow=c(2,3), pty="s")
> qqnorm(time[group=="A"])
> qqline(time[group=="A"])
> qqnorm(time[group=="B"])
> qqline(time[group=="B"])
> qqnorm(time[group=="C"])
> qqline(time[group=="C"])
> qqnorm(time[group=="D"])
> qqline(time[group=="D"])
> qqnorm(time[group=="E"])
> qqline(time[group=="E"])
```

Come da attendersi, il gruppo che pare più anomalo è il gruppo E.
In conclusione però, parrebbe che normalità e omoschedasticità possano essere assunzioni applicabili in questo esempio.
Procediamo quindi con l'analisi della varianza.

```
> oneway.test(time~group,var.equal=TRUE)

        One-way analysis of means

data:  time and group
F = 28.2605, num df = 4, denom df = 21, p-value = 3.475e-08
```

La quantità indicata con `p-value` indica il livello di significatività osservato. Questo è ottenibile come:

```
>   1-pf(28.2605, 4, 21)
[1] 3.474797e-08
```

Per avere invece la soglia della regione di rifiuto, calcoliamo il quantile di livello 0.95 per una distribuzione $F_{4,21}$.

```
> qf(0.95, 4, 21)
[1] 2.8401
```

Conclusioni: I risultati delle analisi effettuate portano ad un rifiuto dell'ipotesi nulla. Per rispondere ora al quesito 2, possiamo accorpare i gruppi A, B, C ed i gruppi D, E per verificare poi la significatività della differenza tra le medie delle due nuove popolazioni (quella composta dalle marche A, B, C e quella composta dalle marche D, E).
Per fare ciò possiamo creare un nuovo vettore indicatore dei due nuovi gruppi ed aggiungerlo al dataframe.

```
> detach()
> gnew <- c(rep("G1", 15), rep("G2", 11))
> gnew <- factor(gnew)
> sturdy1 <- data.frame(sturdy, gnew)
> attach(sturdy1)
```

Per valutare la significatività della differenza tra le due medie, possiamo applicare il test t di Student. Quindi:

```
> t.test(time[gnew=="G1"], time[gnew=="G2"], alternative="greater",
+         var.equal=TRUE)

        Two Sample t-test

data:  time[gnew == "G1"] and time[gnew == "G2"]
t = 8.0229, df = 24, p-value = 1.500e-08
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 1.237152      Inf
sample estimates:
mean of x mean of y
 2.690667  1.118182
```

Conclusioni: L'ipotesi nulla è decisamente rifiutata a favore della alternativa, che prevede che la resistenza media del gruppo di marche G1 sia maggiore della resistenza media del gruppo G2.

```
> detach()
```