Progetto Lauree Scientifiche
Laboratorio Statistico - Informatico
La Statistica con R

Claudio Agostinelli[1]
Dipartimento di Scienze Ambientali, Informatica e Statistica
Università di Ca' Foscari
Venezia

Anno Scolastico 2011-2012

[1]Materiale a cura di: S. Bozza, C. Gaetan, F. Giummolè, N. Sartori. Si ringrazia inoltre A. Brazzale, M. Chiogna, S.M. Iacus per aver messo a disposizione il loro materiale didattico.

Indice

0.1 Obiettivi e Descrizione del Progetto

Il principale obiettivo del laboratorio è quello di avvicinare gli studenti ai metodi di analisi dei dati. A questo scopo si utilizzarà un programma informatico di pubblico dominio considerato attualmente lo standard di riferimento per le analisi statistiche: R (www.r-project.org). Dopo una introduzione al linguaggio di programmazione dell'ambiente R , si coinvolgeranno gli studenti nell'analisi di dati reali, enfatizzando l'uso degli strumenti matematici, statistici e informatici appresi durante il corso di studi.

Capitolo 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 `y`, altrimenti rispondere `n`.
Per controllare cosa c'è disponibile nella directory di lavoro, chiamata anche *workspace*, si usa il comando:

```
> ls()
```

```
 [1] "beta"      "fit"       "laureati"  "p.value"
 [5] "polveri"   "sigma2"    "stat.test" "studenti"
 [9] "tab.cont"  "test"      "v.l"       "v.m"
[13] "var.beta"  "voto.f"    "voto.m"
```

Supponendo che sia presente un oggetto di nome `thing`, è possibile eliminarlo con il comando `rm()`

```
> rm(thing)
```

A questo punto, l'oggetto di nome `thing` non sarà più presente nel *workspace*

```
> 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())
```

o, in alternativa, `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

[1] 108
```

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

```
> sqrt(2)

[1] 1.414214

> sin(3.14159)

[1] 2.65359e-06
```

Il seno di π è zero e questo è vicino. R fornisce anche il valore di π:

```
> sin(pi)

[1] 1.224647e-16
```

che è molto più 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 possibilità 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()
```

Altre funzioni utili sono `apropos()` e `help.search()`. Provate a vedere il relativo help per capire cosa fanno...

1.4 Assegnazioni di valori

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

```
> x <- sqrt(2)
> x

[1] 1.414214

> x^3

[1] 2.828427

> y <- log(x)
> y

[1] 0.3465736

> z = x + y
> z

[1] 1.760787
```

1.5 Valori logici

R permette di gestire operazioni e variabili logiche:

```
> x <- 10
> x > 10

[1] FALSE

> x <= 10

[1] TRUE

> tf <- x > 10
> tf

[1] FALSE

> FALSE

[1] FALSE
```

Gli operatori logici sono `<`, `<=`, `>`, `>=`, `==` per l'uguale e `!=` per il diverso. Inoltre, se `a1` e `a2` sono espressioni logiche, allora `a1 & a2` rappresenta l'intersezione, `a1 | a2` rappresenta l'unione e `!a1` è la negazione di `a1`. Per i dettagli, vedere ad esempio `help(&)`.

1.6 Vettori

1.6.1 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, può essere più conveniente usare `scan()`:

```
> x <- scan()

1: 2
2: 3
3: 5
4: 7
5: 11
6:
Read 5 items
```

```
> x

[1]  2  3  5  7 11

> 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")
```

1.6.2 Successioni

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

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

La stessa operazione può essere fatta con:

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

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

Per creare sequenze di numeri equispaziati si può utilizzare l'opzione `by`:

```
> seq(0, 1, by = 0.1)

 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

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 può 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
[10]  TRUE
```

1.6.3 Estrazione di 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 sono le seguenti:

```
> x <- 26:3
> x

 [1] 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9
[19]  8  7  6  5  4  3

> length(x)

[1] 24

> max(x)

[1] 26

> min(x)

[1] 3

> sum(x)

[1] 348

> prod(x)

[1] 2.016457e+26

> sort(x)

 [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[19] 21 22 23 24 25 26

> sort.list(x)

 [1] 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7
[19]  6  5  4  3  2  1
```

Esercizio: Porre n = 5 + il proprio giorno di nascita. Creare un vettore dei multipli di n compresi fra 1253 e 2037.

1.6.4 Gli indicatori di categoria o fattori

Non sempre però avremo a che fare con vettori numerici. Supponiamo di avere un esperimento su 6 soggetti, due dei quali ricevono il trattamento 'a', tre quello 'b' ed uno il trattamento 'c'. Per salvare questa informazione possiamo creare un vettore di stringhe o fattore, usando il comando `factor()`:

```
> tratta <- factor(c("a", "b", "b", "c", "a", "b"))
> tratta
```

```
[1] a b b c a b
Levels: a b c
```

Per vedere il nome delle categorie contenute nel fattore `tratta` usiamo la funzione `levels()`:

```
> levels(tratta)

[1] "a" "b" "c"
```

Supponiamo ora che gli esiti dell'esperimento sui 6 individui siano i seguenti:

```
> risposta <- c(10, 3, 7, 6, 4, 5)
```

Allora possiamo trovare gli esiti per un particolare trattamento con il comando

```
> risposta[tratta == "a"]

[1] 10  4

> risposta[tratta == "b"]

[1] 3 7 5
```

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()`. Ad esempio, il file matdata contiene i seguenti elementi:

```
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
```

Possiamo metterli in una matrice 6×5 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 un elemento da una matrice, 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)]
```

```
     [,1] [,2]
[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

Una matrice di dati o *data-frame* è un oggetto simile ad una matrice, usato per rappresentare dati. Ogni riga rappresenta un'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. Supponiamo 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("Cherry.dat")

> Ciliegi

     V1 V2   V3
1   8.2 70 10.3
2   8.6 65 10.3
3   8.8 63 10.2
4  10.5 72 16.4
5  10.7 81 18.8
6  10.8 83 19.7
7  11.0 66 15.6
8  11.0 75 18.2
9  11.1 80 22.6
10 11.2 75 19.9
11 11.3 79 24.2
12 11.4 76 21.0
13 11.4 76 21.4
14 11.7 69 21.3
15 12.0 75 19.1
16 12.9 74 22.2
17 12.9 85 33.8
18 13.3 86 27.4
19 13.7 71 25.7
20 13.8 64 24.9
21 14.0 78 34.5
22 14.2 80 31.7
```

```
23 14.5 74 36.3
24 16.0 72 38.3
25 16.3 77 42.6
26 17.3 81 55.4
27 17.5 82 55.7
28 17.9 80 58.3
29 18.0 80 51.5
30 18.0 80 51.0
31 20.6 87 77.0
```

Il data-frame è anche una matrice

```
> dim(Ciliegi)

[1] 31  3
```

Però in realtà è un oggetto con una struttura più complessa. Possiamo vedere la struttura dell'oggetto con il comando

```
> str(Ciliegi)

'data.frame':        31 obs. of  3 variables:
 $ V1: num  8.2 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
 $ V2: int  70 65 63 72 81 83 66 75 80 75 ...
 $ V3: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

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("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
[12] 21.0 21.4 21.3 19.1 22.2 33.8 27.4 25.7 24.9 34.5 31.7
[23] 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0 77.0
```

Tuttavia, la struttura di data-frame fornisce 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
[12] 21.0 21.4 21.3 19.1 22.2 33.8 27.4 25.7 24.9 34.5 31.7
[23] 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0 77.0
```

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

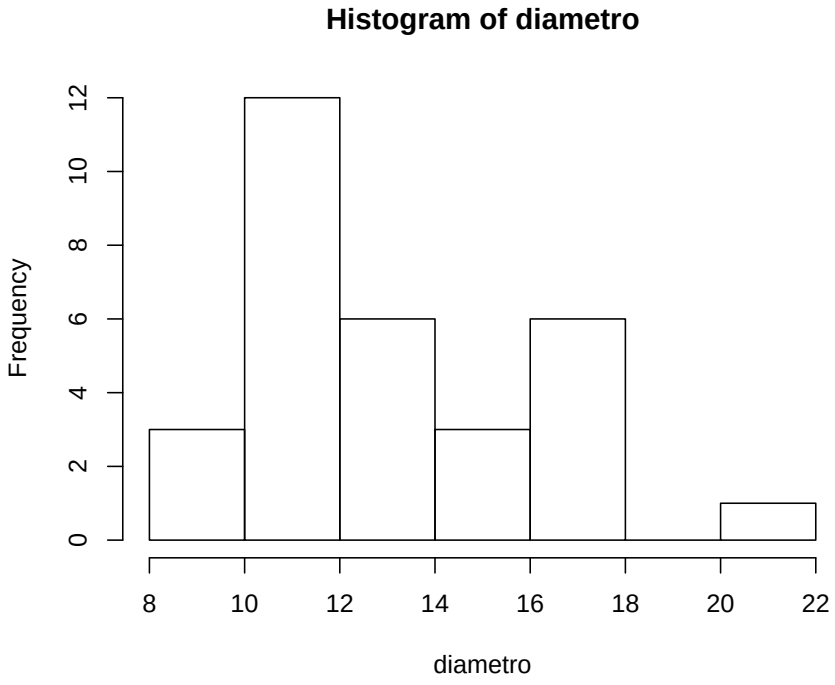
```
> attach(Ciliegi)
> volume

 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2
[12] 21.0 21.4 21.3 19.1 22.2 33.8 27.4 25.7 24.9 34.5 31.7
[23] 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0 77.0
```

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

```
> summary(Ciliegi)

    diametro        altezza       volume
 Min.   : 8.20   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
```

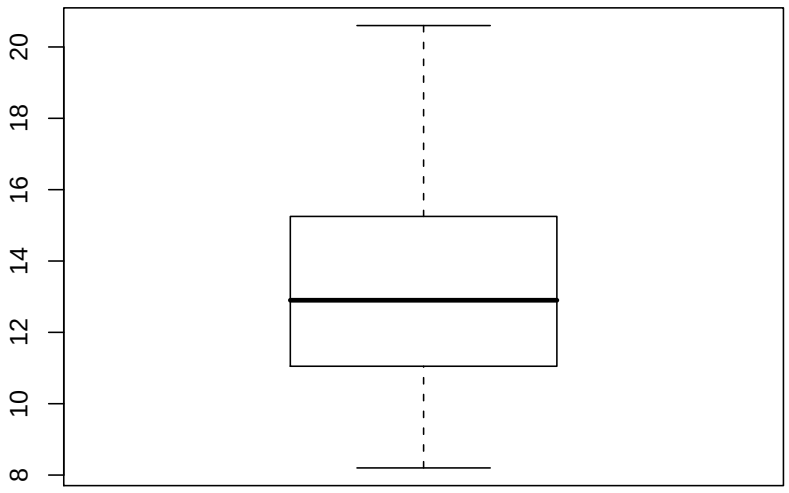
Possiamo anche rappresentare graficamente la distribuzione di una variabile ad esempio `diametro`, mediante un istogramma

```
> hist(diametro)
```



oppure un diagramma a scatola (*boxplot*)

```
> 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
[19] 71 64 78 80 74 72 77 81 82 80 80 80 87
```

```
> Ciliegi[altezza > 80, ]

   diametro altezza volume
5      10.7      81   18.8
6      10.8      83   19.7
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
```

Esercizio: L'insieme di dati nel file laureati.dat riguarda 467 laureati triennali in Economia nel 2003. Le variabili considerate sono corso di laurea, matricola (dato modificato per ragioni di *privacy*), sesso, sigla provincia di residenza, anno prima immatricolazione a Venezia, tipo immatricolazione, diploma maturità, voto maturità, base voto maturità (60 o 100), voto laurea, lode (L, si; NL, no). Separare in due vettori distinti i dati corrispondenti al voto di maturità di maschi e femmine. Calcolare media e varianza dei voti di maturità relativi ai due gruppi, espressi in centesimi. Calcolare la media totale a partire dalle medie dei due gruppi. Se i voti si riportano in sessantesimi, come cambia la loro media? E la loro varianza?

1.9 Elementi di programmazione in R

Abbiamo già sottolineato come sia possibile aumentare il numero delle funzioni di 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, se non forniti come input, 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 proviamo la nostra funzione:

```
> cubo(3)

[1] 27
```

La funzione `cubo()` resterà definita 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'espressione che viene ripetuta tante volte quanti sono gli elementi di `expr1`.

```
> media(x)

[1] 8.5
```

In realtà questa funzione si poteva implementare più semplicemente nel modo seguente:

```
> media <- function(x) {
+     sum(x)/length(x)
+ }
```

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

```
> mean(x)

[1] 8.5
```

Introdurremo altri elementi di programmazione più 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 $M(Y) = \sum_{i=1}^{n} Y_i/n$. Confrontate il risultato che ottenete con quello della funzione `var()` di R.

Capitolo 2

Statistica descrittiva I

2.1 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
  1   2   5   1   3   3   3   7   1   8   1   8   3   3   1
185 186 187 188 190 192 193
  6   2   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)
> classi
```

```
 [1] 150 155 160 165 170 175 180 185 190 195 200
```

e poi assegnamo i maschi ad ogni classe con il comando

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

```
 [1] (185,190] (180,185] (180,185] (175,180] (165,170]
 [6] (170,175] (170,175] (180,185] (175,180] (185,190]
[11] (165,170] (180,185] (175,180] (180,185] (170,175]
[16] (175,180] (175,180] (170,175] (185,190] (170,175]
[21] (170,175] (175,180] (175,180] (175,180] (165,170]
[26] (180,185] (170,175] (190,195] (170,175] (180,185]
[31] (175,180] (175,180] (180,185] (170,175] (170,175]
[36] (175,180] (165,170] (185,190] (190,195] (180,185]
```

```
[41] (180,185] (185,190] (175,180] (170,175] (175,180]
[46] (180,185] (170,175] (170,175] (170,175] (165,170]
[51] (190,195] (175,180] (175,180] (160,165] (185,190]
[56] (165,170] (180,185] (165,170] (180,185] (170,175]
[61] (170,175] (170,175] (175,180] (175,180] (175,180]
10 Levels: (150,155] (155,160] (160,165] ... (195,200]
```

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 opportuna in 10 intervalli 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

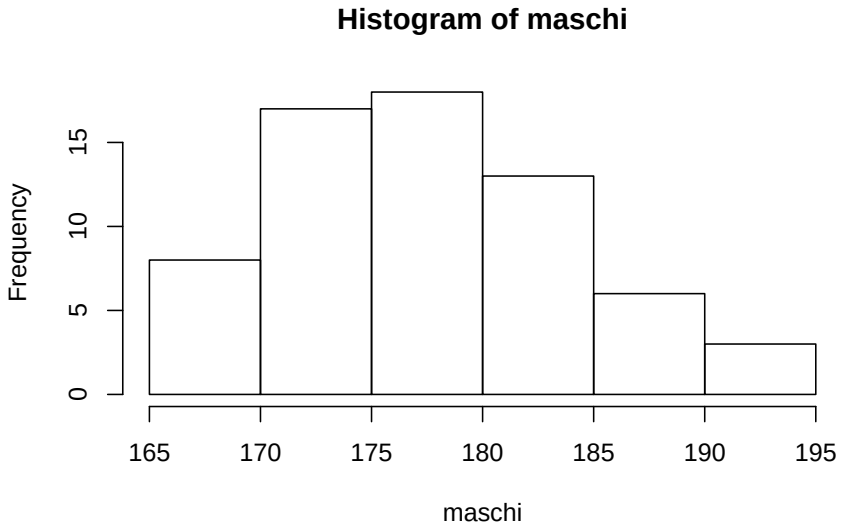
```
> freqcum <- cumsum(table(cut(maschi, breaks = classi))/length(maschi))
> freqcum

 (150,155]  (155,160]  (160,165]  (165,170]  (170,175]
0.00000000 0.00000000 0.01538462 0.12307692 0.38461538
 (175,180]  (180,185]  (185,190]  (190,195]  (195,200]
0.66153846 0.86153846 0.95384615 1.00000000 1.00000000
```

Esercizio: Per i dati contenuti nel file femmine.dat costruire la tabella delle frequenze relative e relative cumulate, scegliendo un'opportuna suddivisione in classi.

2.2 Grafici

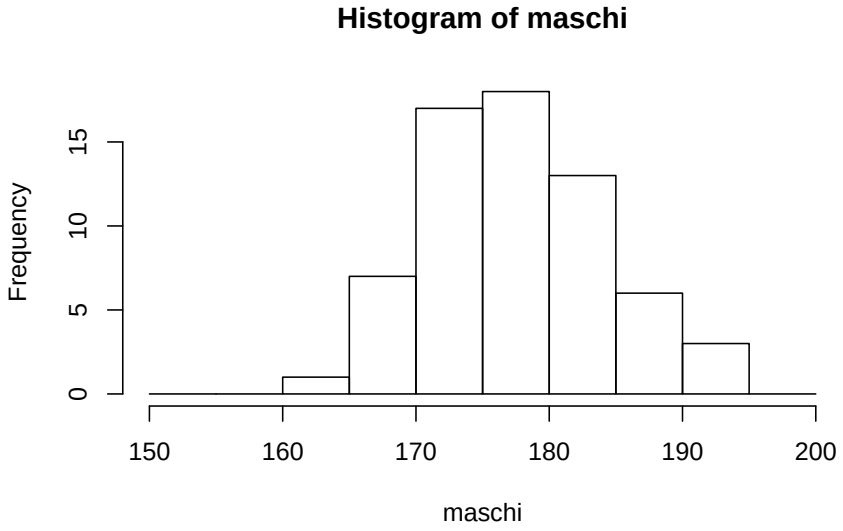
Costruiamo dapprima un istogramma di frequenza. Il comando più semplice è

```
> hist(maschi)
```

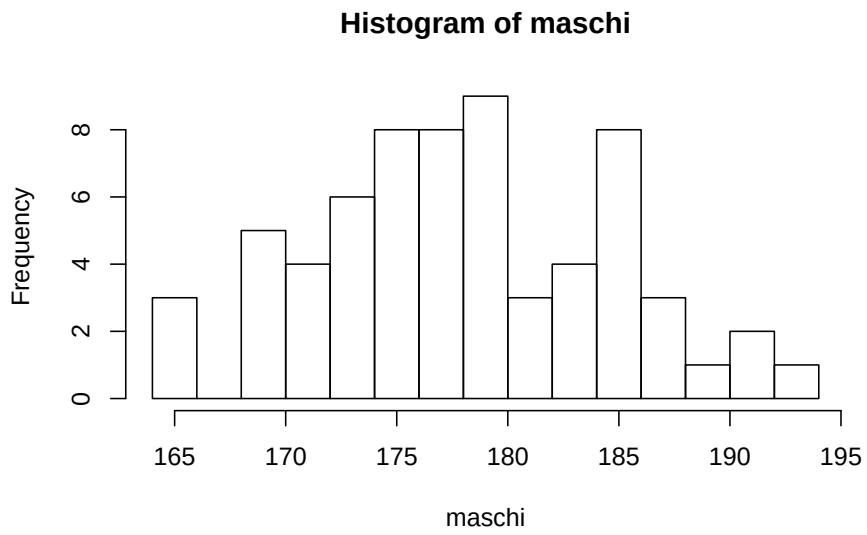


Come per `table()` anche `hist()` permette di stabilire il numero di classi in cui vogliamo rappresentare i nostri dati. Ad esempio

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



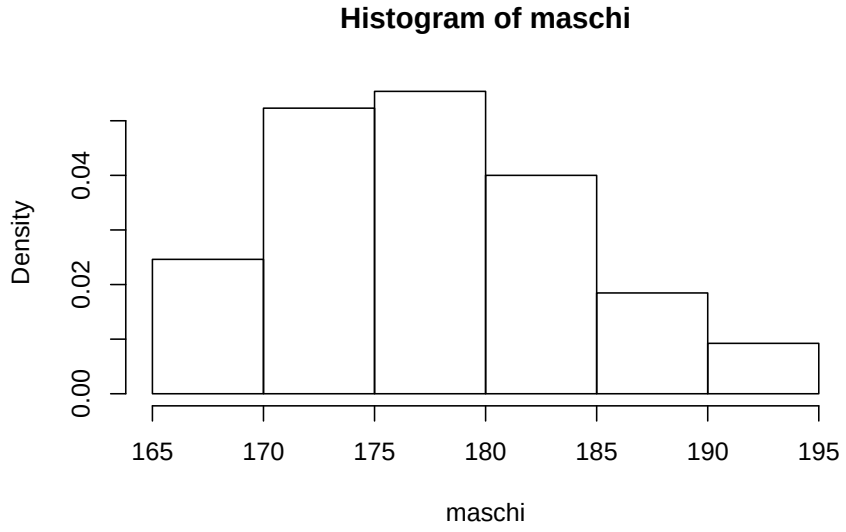

```
> hist(maschi, breaks = 10)
```



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 le frequenze relative sull'asse delle ordinate.

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

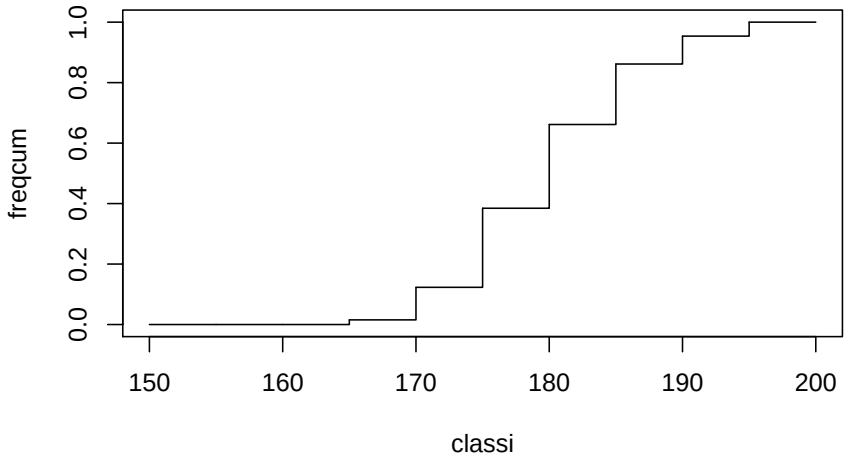


Possiamo anche ottenere un grafico della funzione di ripartizione come segue. Dapprima aggiungiamo un limite inferiore alle classi

```
> 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'.
Consideriamo ora 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()`

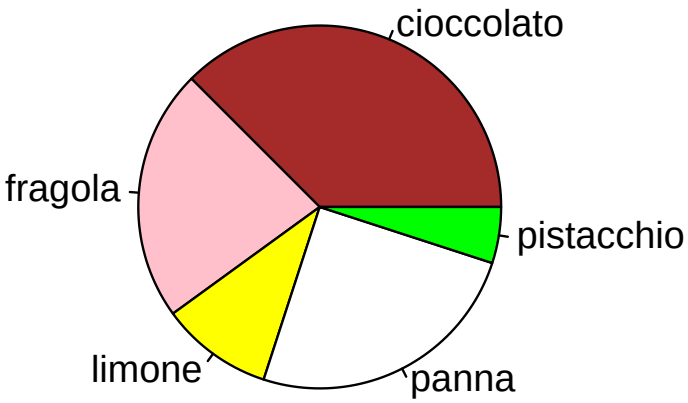
```
> gelati <- read.table("gelati.dat")

> names(gelati) <- "tipo"
```

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
```

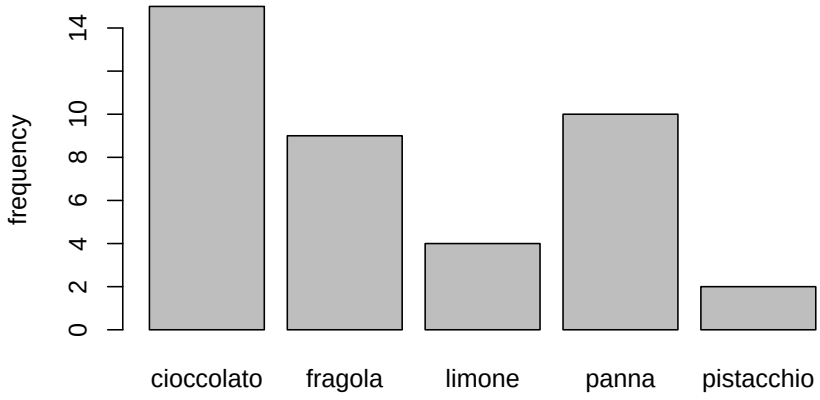
Proviamo a costruire un diagramma a torta

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



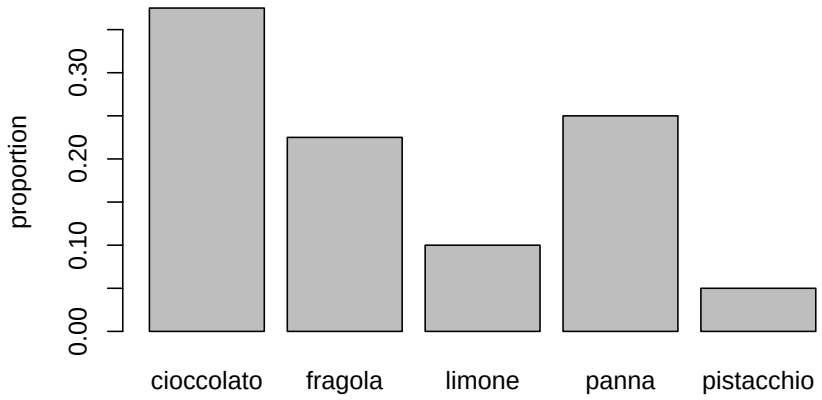
Un altro modo per rappresentare dati qualitativi è dato dal diagramma a barre o *barplot*. Possiamo costruire un barplot delle frequenze assolute

```
> barplot(table(gelati), xlab = "", ylab = "frequency")
```



o delle frequenze relative

```
> barplot(table(gelati)/dim(gelati)[1], xlab = "",
+     ylab = "proportion")
```



Esercizio: Di seguito sono indicate le percentuali di diffusione negli Stati Uniti di diversi browser:

$InternetExplorer$	86%
$Gecko-based(Netscape, Mozilla)$	4%
$NetscapeNavigator4$	5%
$Opera$	1%
$unidentified$	4%

Costruire un diagramma a barre e a torta di questi dati.

2.3 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()`. L'uso più semplice della funzione `quantile()` è

```
> 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(0.15, 0.47))

15% 47%
172 178
```

La funzione `summary()` ha come risultato

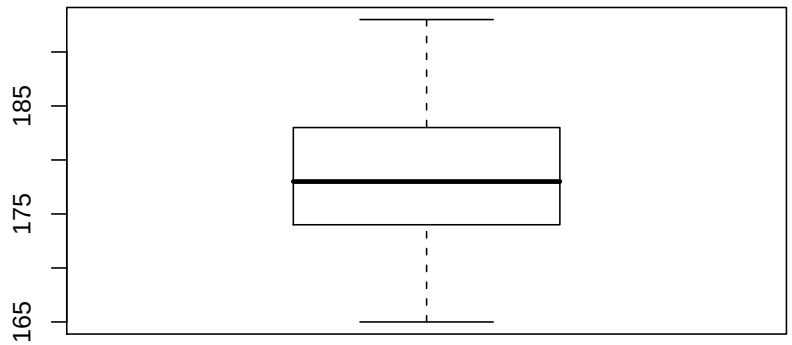
```
> summary(maschi)

   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, 0.25, 0.5, 0.75,
+     1))

  0%  25%  50%  75% 100%
 165  174  178  183  193
```

Le informazioni del comando `summary()` possono essere rappresentate nel diagramma a scatola (con baffi)

```
> boxplot(maschi)
```



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

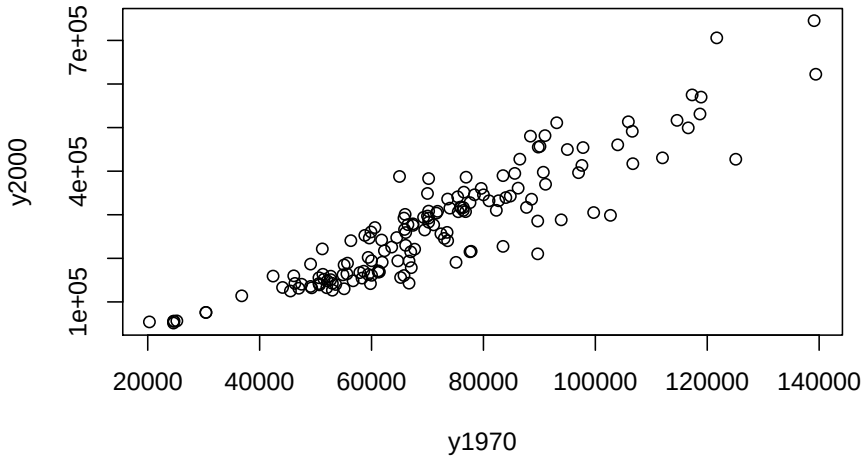
$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - M(X)}{\mathrm{sd}(X)}\right)^3, \quad \frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - M(X)}{\mathrm{sd}(X)}\right)^4.$$

Esercizio: Provate a costruire una funzione che calcoli il quantile (almeno approssimativamente) di un vettore di osservazioni. Suggerimento: utilizzate la funzione `sort()`.

2.4 Dipendenza tra variabili

Un buon punto di partenza per investigare la relazione esistente tra due variabili quantitative è dato dal diagramma di dispersione o *scatterplot*. Anche in questo caso può essere utilizzata la funzione `plot()` con diverse opzioni. Si possono ottenere tutte le informazioni del caso consultando l'`help` in linea.
Consideriamo un esempio. Il data set homedata.dat contiene il valore in $ di 150 abitazioni del New Jersey, valutato a distanza di 30 anni (nel 1970 e nel 2000). Può essere interessante valutare se nell'arco dei 30 anni si sia verificato un cambiamento nel valore delle case. Dopo aver caricato l'insieme di dati, è possibile ottenere un diagramma di dispersione.

```
> homedata <- read.table("homedata.dat")

> attach(homedata)
> plot(y1970, y2000)
```



L'osservazione del grafico ottenuto suggerisce una relazione tra le variabili: le case che erano costose nel 1970 lo sono anche nel 2000, e viceversa. L'intensità del legame lineare fra le due variabili può essere misurata tramite il coefficiente di correlazione:

```
> cor(y1970, y2000)
```

```
[1] 0.9111092
```

Con il comando `summary()` è anche possibile ottenere alcune informazioni sulla distribuzione di ciascuna variabile.

```
> summary(homedata)
```

```
     y1970            y2000
 Min.   : 20300   Min.   : 51600
 1st Qu.: 57000   1st Qu.:163175
 Median : 68500   Median :260100
 Mean   : 71277   Mean   :273824
 3rd Qu.: 83500   3rd Qu.:342475
 Max.   :139400   Max.   :745400
```

```
> summary(y2000/y1970)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.098   2.893   3.797   3.678   4.296   5.968
```

Alcune case hanno aumentato il loro valore di due volte, altre di quasi 6, con una media di 3.68.

2.5 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 metodi 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, III, 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 numerico `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, ad esempio, la variabile `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)
```

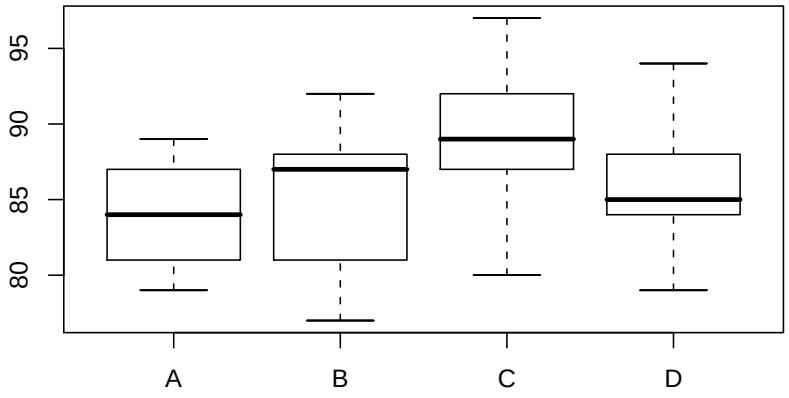
```
[1] TRUE
```

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

```
> attach(pen)
```

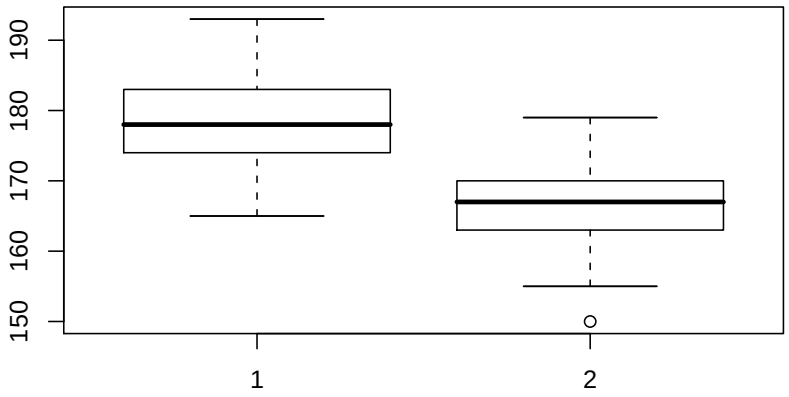
Osserviamo ora i risultati della diversa applicazione del comando `plot()`

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



Confrontiamo ora le altezze di un gruppo di femmine con quelle di un gruppo di maschi

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

> femmine

 [1] 162 165 173 170 165 165 170 178 173 170 168 165 165 157
[15] 168 157 169 174 167 168 160 175 174 170 160 160 168 176
[29] 173 162 175 165 160 164 163 173 163 162 168 163 160 170
[43] 155 172 160 162 167 174 165 163 172 158 174 155 174 160
[57] 160 167 164 166 170 150 165 179 168 165 168 168 166 167
[71] 174 165 167 170

> boxplot(maschi, femmine)
```



Esercizio: Provate a costruire a partire da `maschi` e `femmine` un unico data-frame di nome `stature`, contenente due colonne: `sesso` e `statura`. In particolare, `sesso` deve essere una variabile di tipo `factor` con due modalità: `M` e `F`. Disegnate il diagramma a scatola con il comando `boxplot(statura~sesso)` oppure con il comando `plot(sesso,statura)`.

Capitolo 3

Statistica descrittiva II

3.1 Diagrammi di dispersione

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

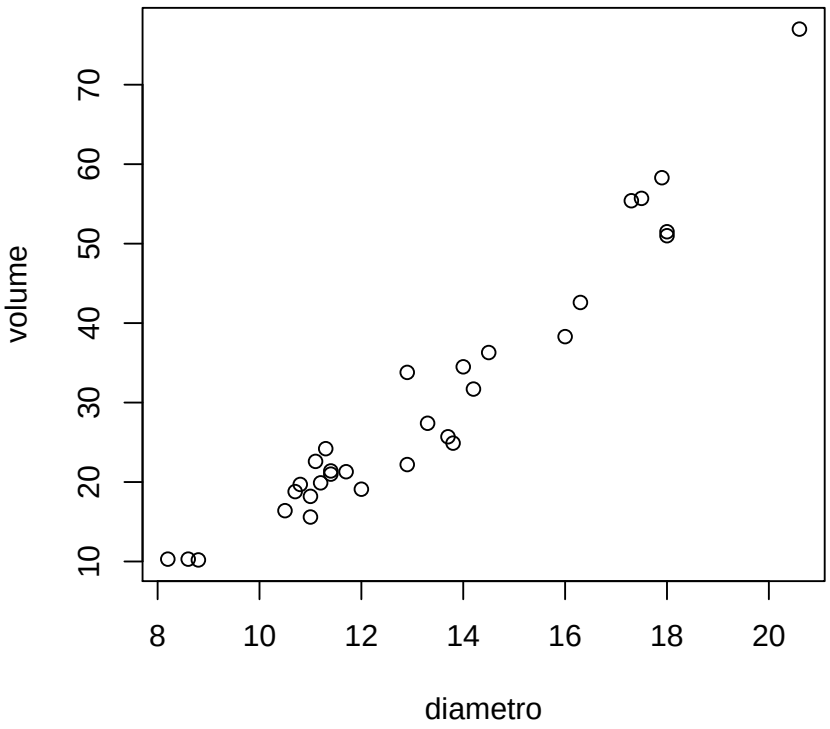
```
> Ciliegi <- read.table("Cherry.dat", col.names = c("diametro",
+     "altezza", "volume"))
```

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

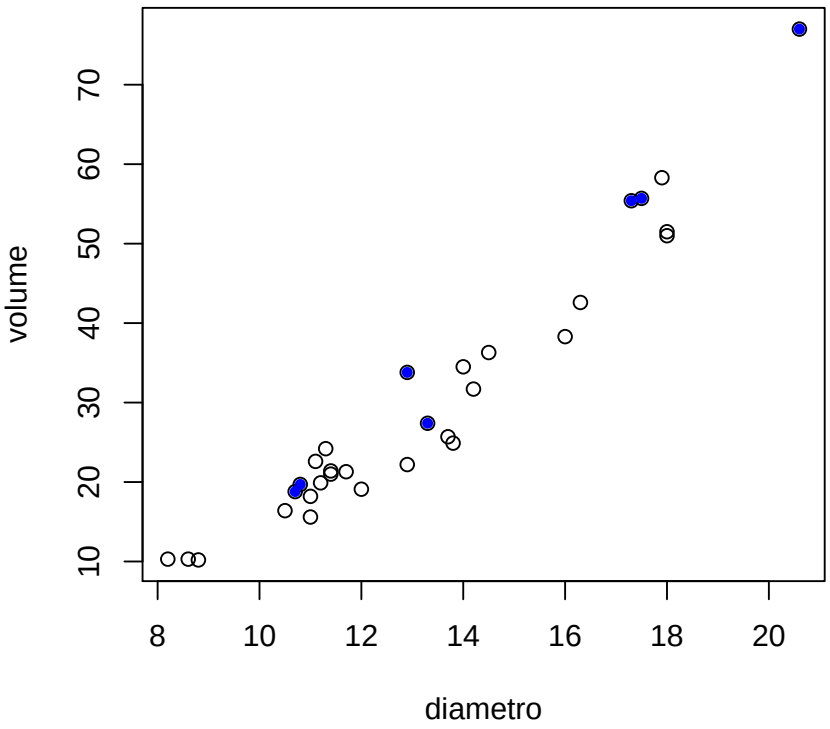
```
> attach(Ciliegi)
```

I dati contengono, per 31 alberi di ciliegio 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'altezza superiore a 80




```
> points(diametro[altezza > 80], volume[altezza >
+     80], col = "blue", pch = 20)
```

Con il comando

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

possiamo identificare a quali unità statistitiche appartengono i punti evidenziati.

3.2 La regressione semplice

Continuiamo con i dati sui ciliegi. Il numero di osservazioni è:

```
> n <- nrow(Ciliegi)
> n
```

```
[1] 31
```

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] 5.055482
```

Questo è equivalente a

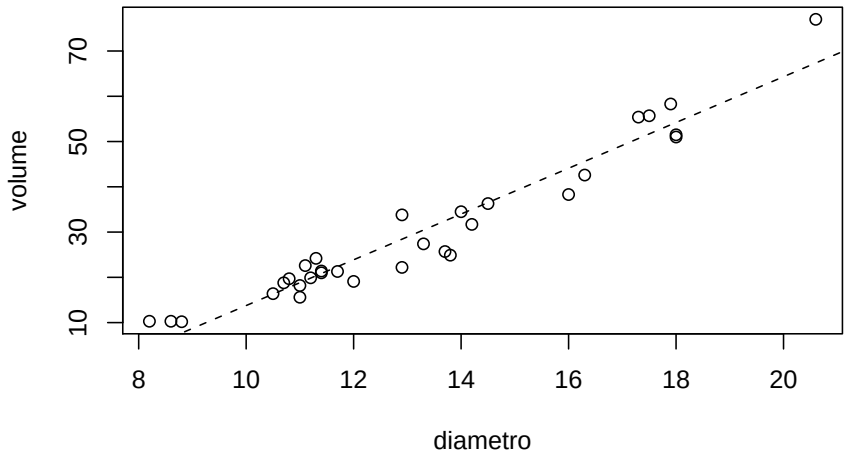
```
> beta <- cov(diametro, volume)/var(diametro)
> beta

[1] 5.055482
```

La stima di α è quindi pari a

```
> alpha <- mean(volume) - beta * mean(diametro)
> alpha

[1] -36.7897
```

Rappresentiamo la retta calcolata nel grafico, con il comando

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



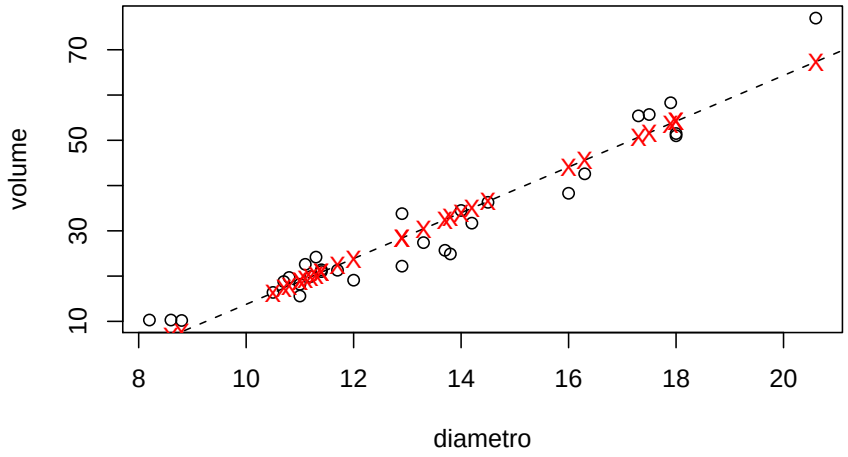
Il comando `abline(a,b)` traccia una retta nel grafico corrente con intercetta `a` e coefficiente angolare `b`. L'opzione `lty` è un'opzione grafica generale (valida ad esempio anche per il comando `plot()`) e definisce il tipo di linea. Assume valori: "blank", "solid" (default), "dashed", "dotted", "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]  4.665248  6.687441  7.698537 16.292855 17.303952
 [6] 17.809500 18.820596 18.820596 19.326144 19.831693
[11] 20.337241 20.842789 20.842789 22.359433 23.876078
[16] 28.426011 28.426011 30.448204 32.470396 32.975945
[21] 33.987041 34.998137 36.514782 44.098004 45.614649
[26] 50.670130 51.681226 53.703419 54.208967 54.208967
[31] 67.353219
```

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", col = 2)
```



Il comando `points()` aggiunge i punti in un grafico 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]  5.63475212  3.61255950  2.50146319  0.10714454
 [5]  1.49604822  1.89050007 -3.22059624 -0.62059624
 [9]  3.27385560  0.06830745  3.86275929  0.15721113
[13]  0.55721113 -1.05943333 -4.77607780 -6.22601120
[17]  5.37398880 -3.04820383 -6.77039645 -8.07594461
[21]  0.51295908 -3.29813723 -0.21478170 -5.79800404
[25] -3.01464850  4.72986994  4.01877363  4.59658100
[29] -2.70896715 -3.20896715  9.64678080
```

Il coefficiente di determinazione è dato da

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

```
[1] 0.9346436
```

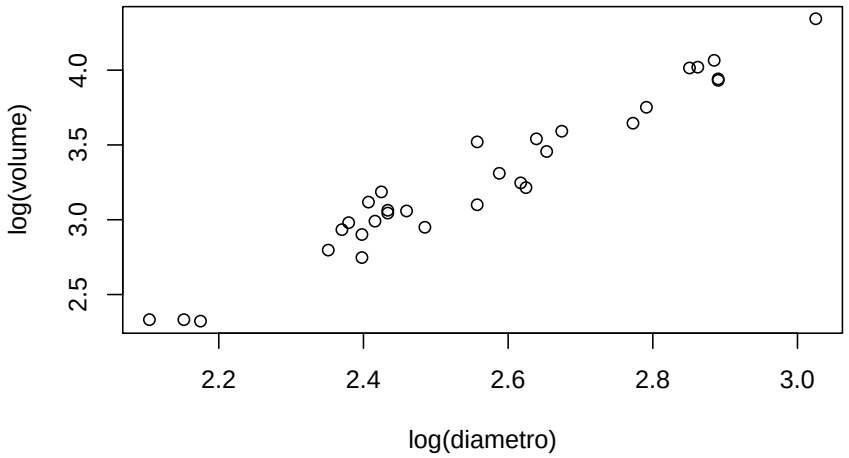
oppure, nel caso di regressione semplice,

```
> cor(diametro, volume)^2
```

```
[1] 0.9346436
```

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 questa volta 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))
> beta1
```

```
[1] 2.192426
```

```
> alpha1 <- mean(log(volume)) - beta1 * mean(log(diametro))
> alpha1
```

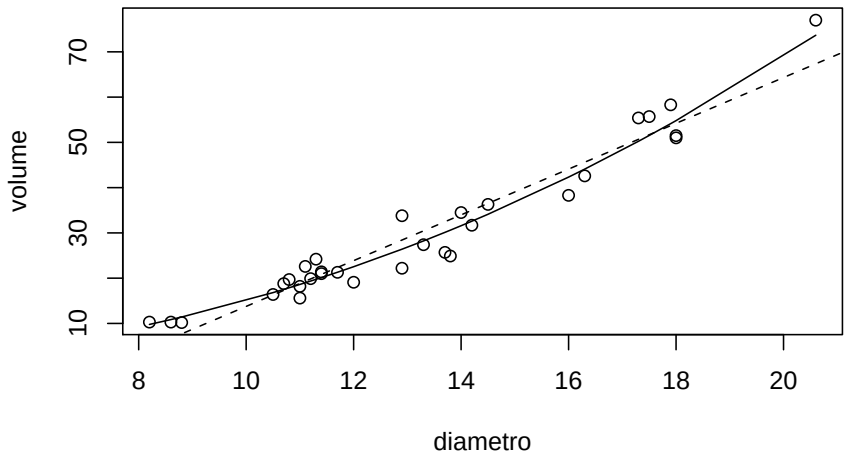
```
[1] -2.333174
```

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

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

e poi confrontare graficamente i risultati con quelli precedenti:

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



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,

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

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

```
> var(residui1)
```

```
[1] 10.60626
```

```
> var(residui)
```

```
[1] 17.65948
```

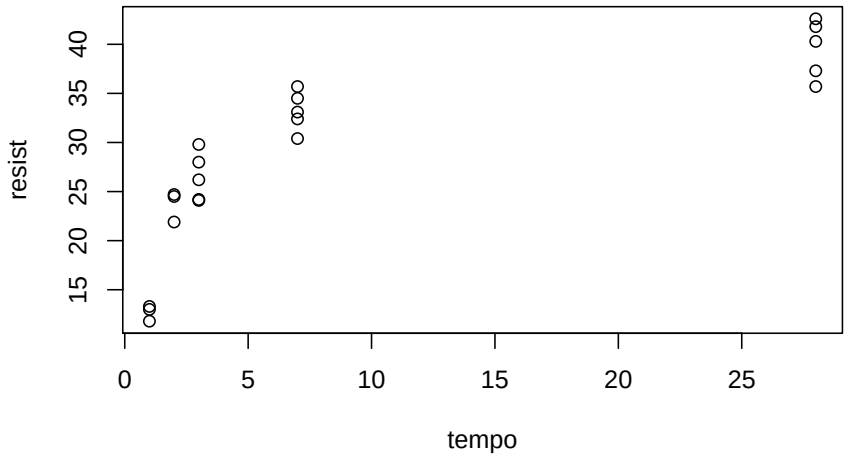
3.3 La funzione `lm()`

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 `resist` $= \alpha + \beta$`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.

```
> cement.lm <- lm(resist ~ tempo)
```

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

```
> summary(cement.lm)

Call:
lm(formula = resist ~ tempo)

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

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 senza utilizzare la funzione `lm()`.

L'oggetto `cement.lm` contiene più quantità:

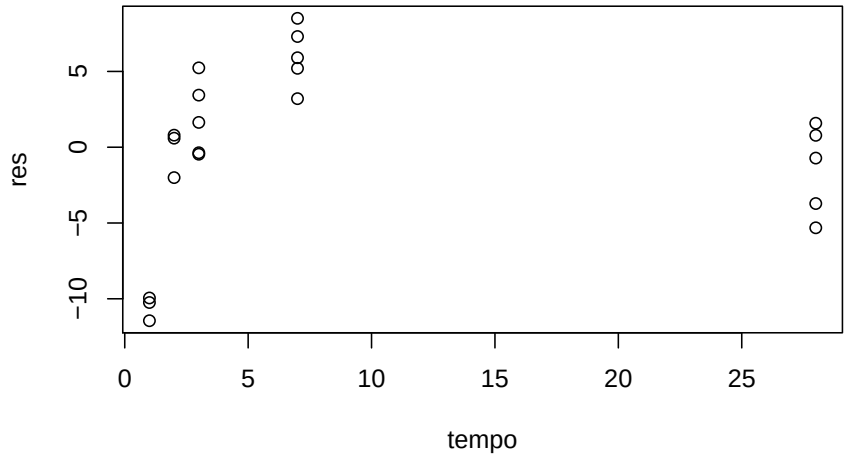
```
> names(cement.lm)

 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
```

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

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

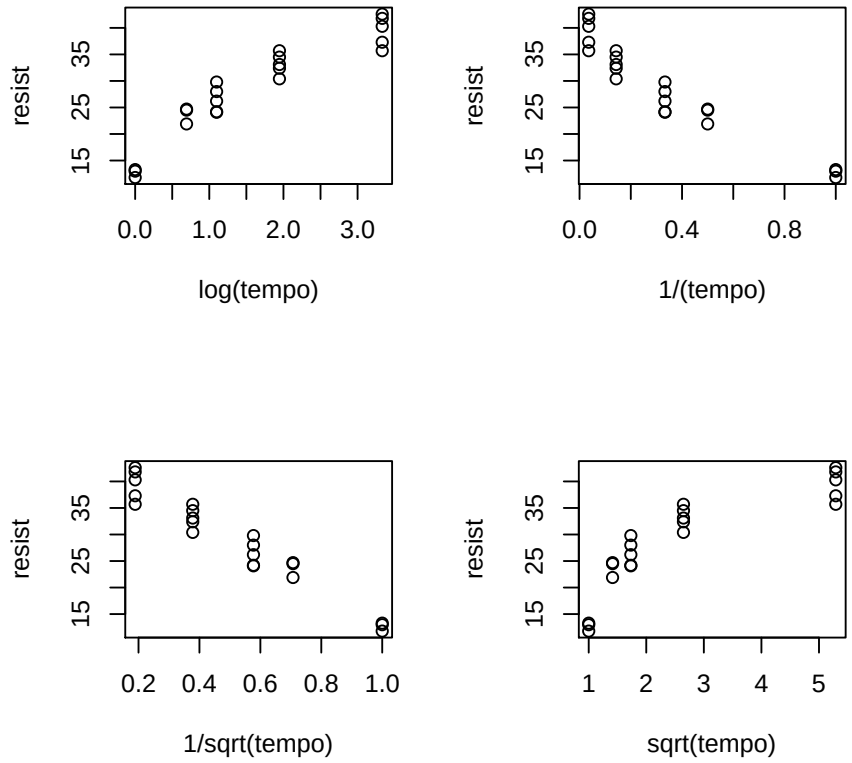
I residui contenuti in `cement.lm` e che abbiamo appena salvato nel vettore `res` sono le quantità $e_i = y_i - \hat{y}_i$.
Un grafico che possiamo fare per verificare la linearità della relazione, è il grafico

```
> 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`, che 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)
```



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

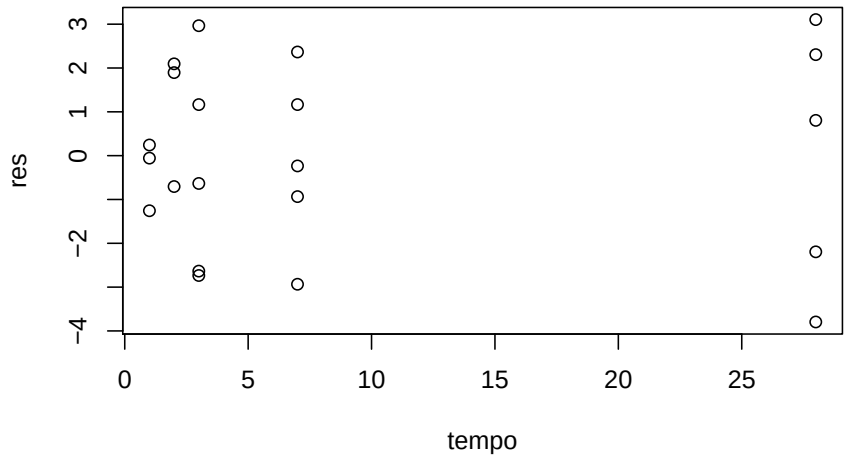
$$\texttt{resist} = \alpha + \beta \frac{1}{\sqrt{\texttt{tempo}}} + \varepsilon$$

```
> cement.lm2 <- lm(resist ~ x)
> summary(cement.lm2)

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(cement.lm2)
> plot(res ~ tempo)
```



Dal grafico osserviamo un miglioramento sensibile. Per terminare digitiamo

```
> detach(cement)
```

Esercizio: Un agente immobiliare intende prevedere gli affitti mensili (in dollari) degli appartamenti sulla base della loro dimensione (in piedi al quadrato). Per questo conduce un'indagine e reperisce i dati su 25 appartamenti in una zona residenziale. I dati sono raccolti nel file rent.dat. Si svolga un'analisi statistica tesa a costruire un modello di previsione lineare.

Esercizio: Costruire la retta di regressione per i dati `cars` in R e sovrapporla al corrispondente diagramma di dispersione. Secondo il modello ottenuto, dopo quanti metri si arresta una macchina che corre a 150 km/h?

Esercizio: L'insieme di dati `Animals` della libreria MASS contiene il peso corporeo e del cervello di diversi tipi di animali. Cosa suggerisce lo scatterplot? Provare a fare una trasformazione logaritmica di entrambe le variabili, rappresentare nuovamente il diagramma di dispersione e stimare la retta di regressione.

Capitolo 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 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 più piccoli di questo. Come sappiamo $D_{n,k}$ 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:

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

Quindi calcolare il numero di sottoinsiemi di 3 elementi presi da un insieme di 50.

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 densità, 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 $x = 2$ 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

```
> pbinom(1, 3, 1/3)

[1] 0.7407407
```

mentre

```
> pbinom(0, 3, 1/3)

[1] 0.2962963
```

Esercizio: Utilizzando la distribuzione normale standard Z, trovare la probabilità che siano soddisfatte le seguenti condizioni:

$$Z < -2.85,\ Z > 2.85,\ Z < 2.85,\ Z > -1.66,\ -1.66 < Z < 2.85$$

Trovare i quantili corrispondenti alle seguenti percentuali:

`25%, 40%, 50%, 60%, 95%`

Esercizio: Per una variabile aleatoria X di Poisson di parametro $\lambda = 10$, calcolare le seguenti probabilità: $P(2 \leq X \leq 7), P(2 \leq X < 7), P(X \geq 9), P(X > 4), P(X < 3 \cap X > 8)$.

Esercizio: Si suppone che il tempo di vita delle lampadine prodotte da una ditta segua una distribuzione esponenziale di parametro $\lambda = 0.005$ (1/giorni). Qual è la probabilità che una lampadina duri almeno 300 giorni? Se dalla produzione si scelgono 50 lampadine, qual è la probabilità che solo 3 di queste durino più di 300 giorni?
(0.2231302, 0.001528616)

4.3 Simulazione di variabili casuali

Il comando `sample()` permette di estrarre (con o senza reinserimento) un certo numero di valori da un insieme prefissato. Per generare i possibili risultati di 10 lanci di un dado equilibrato, possiamo fare così:

```
> sample(1:6, 10, replace = T)

 [1] 4 2 1 4 1 1 3 1 1 6
```

Possiamo anche creare un'urna contenente, ad esempio, quattro palline bianche numerate da 1 a 4 e tre nere numerate da 1 a 3:

```
> urna <- c("b1", "b2", "b3", "b4", "n1", "n2",
+     "n3")
```

ed estrarre due palline prima senza e poi con reinserimento:

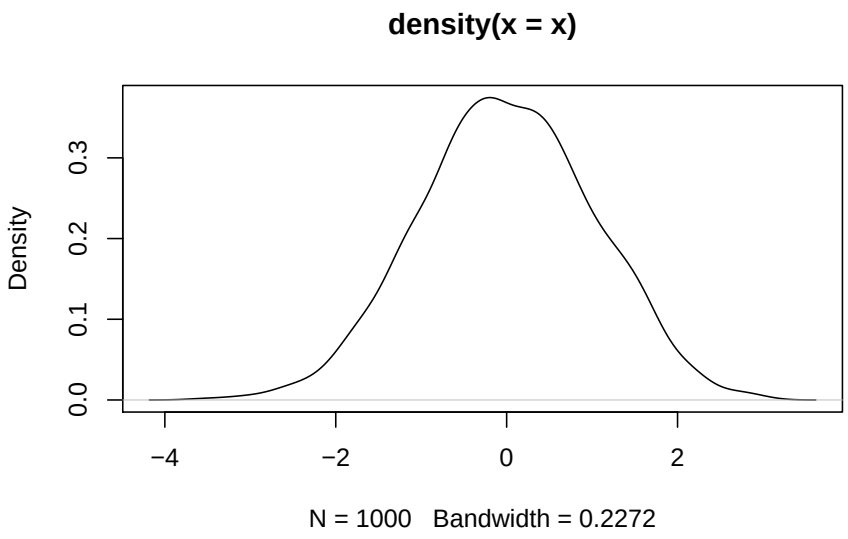
```
> sample(urna, 2)

[1] "b2" "b4"

> sample(urna, 2, replace = T)

[1] "b2" "n3"
```

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)
```

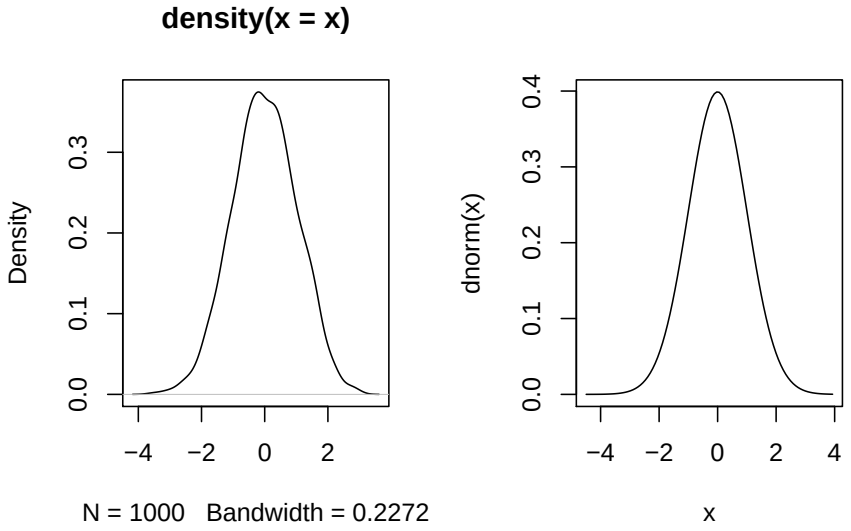
A questo punto possiamo pensare di produrre una stima della densità del vettore x, attraverso la funzione `density()` (sappiamo già come rappresentare graficamente la distribuzione di un insieme di dati attraverso i comandi `boxplot()` e `hist()`)

```
> 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))
```



Esercizio: Generare casualmente, utilizzando in modo opportuno il comando `sample()`, un valore della variabile che conta il numero di teste in 5 lanci di una moneta equilibrata. Che comando conviene usare se la moneta è truccata e la probabilità di ottenere testa è pari a 0.8?

Esercizio: Costruire l'istogramma relativo alla variabile `eruptions` nell'insieme di dati `faithful` di R e sovrapporne una stima della densità ottenuta con il comando `density()`.

4.4 Verifica di normalità

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.

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

Notiamo che la variabile `sesso` è considerata come una variabile di tipo numerico

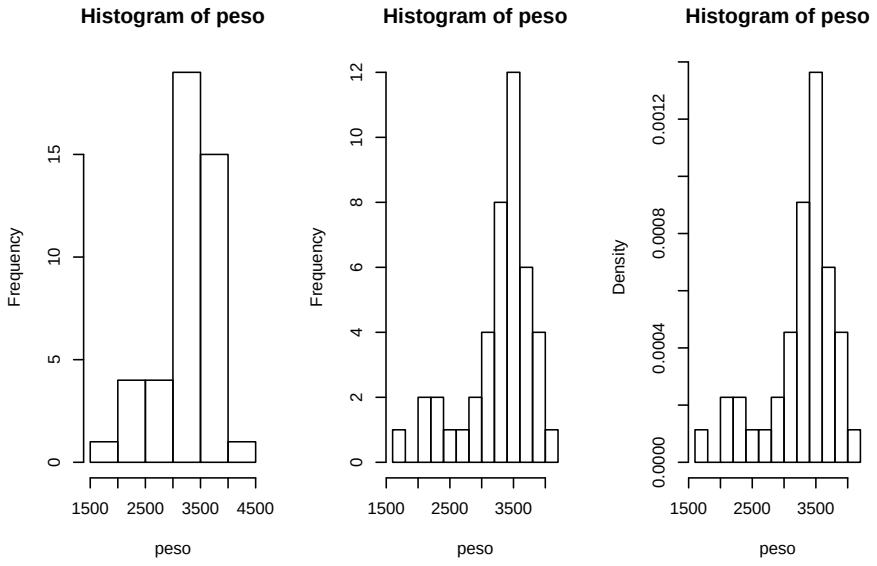
```
> str(babies$sesso)

 int [1:44] 1 1 2 2 2 1 1 2 2 2 ...
```

Dobbiamo trasformare tale variabile in una variabile qualitativa. Ci sono vari modi per farlo; ad esempio,

```
> babies$sesso[babies$sesso == 1] <- "F"
> babies$sesso[babies$sesso == 2] <- "M"
> babies$sesso <- factor(babies$sesso)
> babies$sesso

 [1] F F M M M F F M M M M M F F M F F M M M M F F F F M M M
[29] F M F M M M M M F M M M M F F F
Levels: F M

> attach(babies)
```

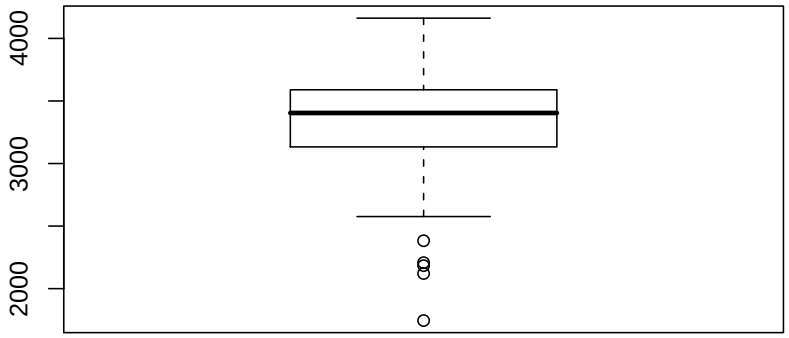
Consideriamo anzitutto la variabile `peso` e vediamo se può essere modellata da una distribuzione normale. Proviamo preliminarmente a studiare graficamente la distribuzione dei dati. Vediamo per cominciare l'istogramma e il boxplot.

```
> par(mfrow = c(1, 3))
> hist(peso)
> hist(peso, nclass = 10)
> hist(peso, nclass = 10, prob = T)
> par(mfrow = c(1, 1))
```



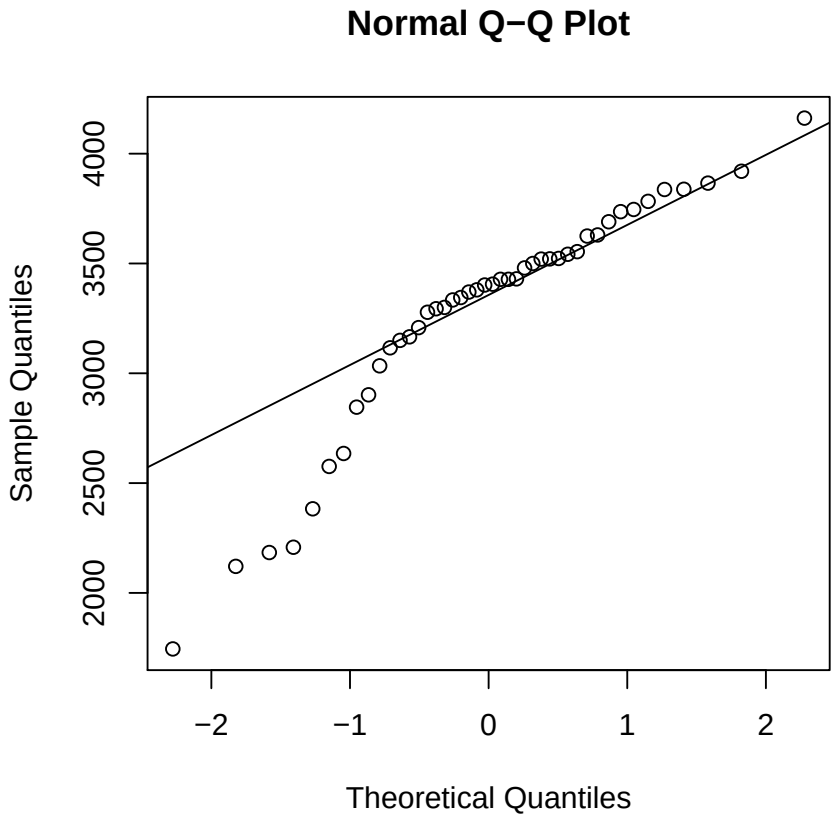
È evidente una netta asimmetria a sinistra, che possiamo confermare o meno costruendo un boxplot.

```
> boxplot(peso)
```



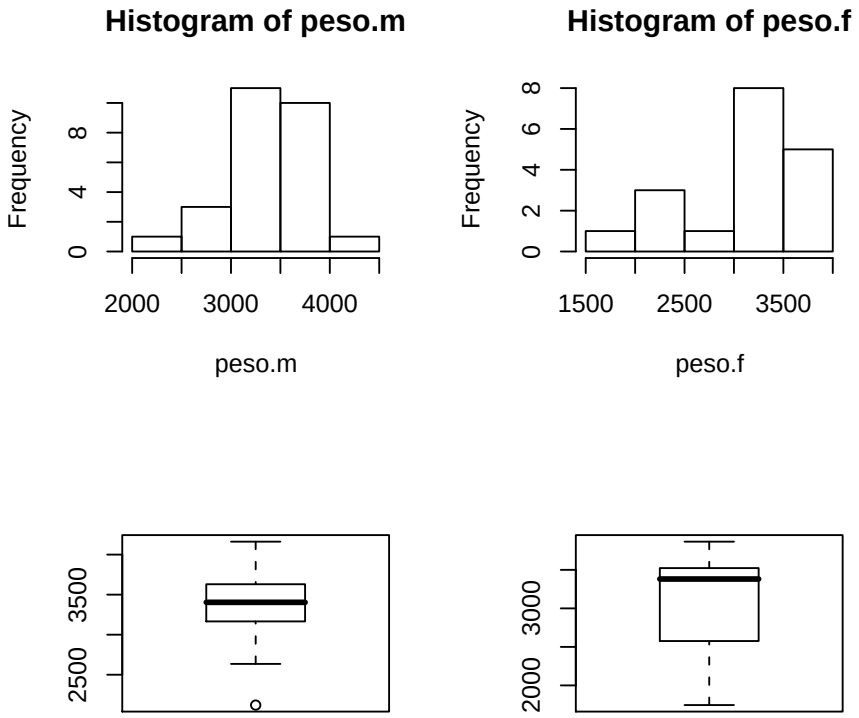
In effetti, l'impressione che esista una rilevante asimmetria rimane confermata anche osservando il boxplot e ciò è probabilmente imputabile a un gruppo di bambini nati prematuramente e quindi sottopeso oppure aventi malformazioni gravi o malattie. Questi sembrano i fattori che principalmente contribuiscono alla non normalità.
Un importante strumento di verifica della normalità di un insieme di dati è il grafico quantile–quantile o *qqplot*. Permette di confrontare i quantili empirici dei dati con quelli teorici di un'opportuna distribuzione normale. Si costruisce così:

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



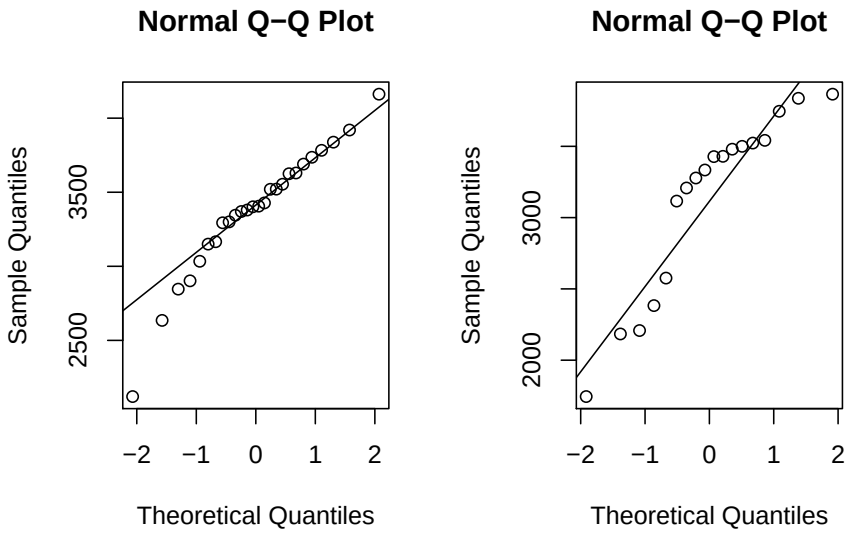
Come si vede osservando il grafico c'è un netto scostamento dalla retta a conferma della asimmetria rilevata. Proviamo a distinguere tra pesi dei maschi e delle femmine, che notoriamente alla nascita mostrano differenze anche rilevanti, per evidenziare eventuali diversità.

```
> peso.m <- peso[sesso == "M"]
> peso.f <- peso[sesso == "F"]
> peso.m

 [1] 3554 3838 3625 2846 3166 3520 3380 3294 3521 2902 2635
[12] 3920 3690 3783 3345 3034 3300 3428 4162 3630 3406 3402
[23] 3736 3370 2121 3150

> peso.f

 [1] 3837 3334 2208 1745 2576 3208 3746 3523 3430 3480 3116
[12] 3428 2184 2383 3500 3866 3542 3278

> par(mfrow = c(2, 2))
> hist(peso.m)
> hist(peso.f)
> boxplot(peso.m)
> boxplot(peso.f)
```



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.

```
> par(mfrow = c(1, 2))
> qqnorm(peso.m)
> qqline(peso.m)
> qqnorm(peso.f)
> qqline(peso.f)
```



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

Esercizio: Verificare la normalità dei dati nei file femmine.dat e maschi.dat.

Capitolo 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 probabilità, 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 variabili 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] 4.5
```

Raddoppiamo ora le replicazioni

```
> x <- c(x, rpois(10, 5))
> mean(x)
```

```
[1] 4.7
```

e raddoppiamo ancora

```
> x <- c(x, rpois(20, 5))
> mean(x)

[1] 4.325
```

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

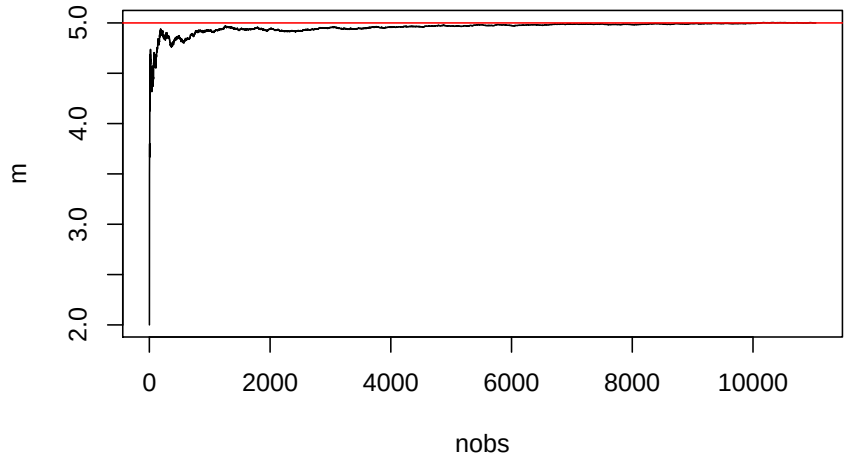
```
> x <- c(x, rpois(1000, 5))
> mean(x)

[1] 4.907692
```

e di altre 10000 replicazioni

```
> x <- c(x, rpois(10000, 5))
> mean(x)

[1] 5.000181
```

I risultati ottenuti confermano il fatto che la media campionaria si avvicina 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")
> abline(h = 5, col = 2)
```



Cosa potete notare?

Esercizio: Dall'urna introdotta nella lezione precedente, estrarre 50 palline con reinserimento. Calcolare la frequenza relativa di 'b1' e confrontarla con la probabilità di estrarre 'b1'. Calcolare la frequenza relativa dell'evento "pallina con numero 2 e confrontarla con la sua probabilità. Ripetere tutto con un campione di numerosità 5000 e commentare.

Esercizio: Generare 50 valori dalla distribuzione binomiale di parametri n=15 e p=0.2 e confrontare le frequenze dei valori simulati con le probabilità della distribuzione teorica. Ripetere il procedimento con 1000 e 5000 valori e commentare.

5.2 Il Teorema Limite Centrale

Ricordiamo ora l'enunciato del Teorema Limite Centrale.
Se X_i $i = 1, \ldots$ è una successione di variabili 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 media campionaria per singolo campione generato.
Generiamo un campione di $n = 10$ variabili casuali $X \sim Poisson(5)$ di cui calcoliamo la media

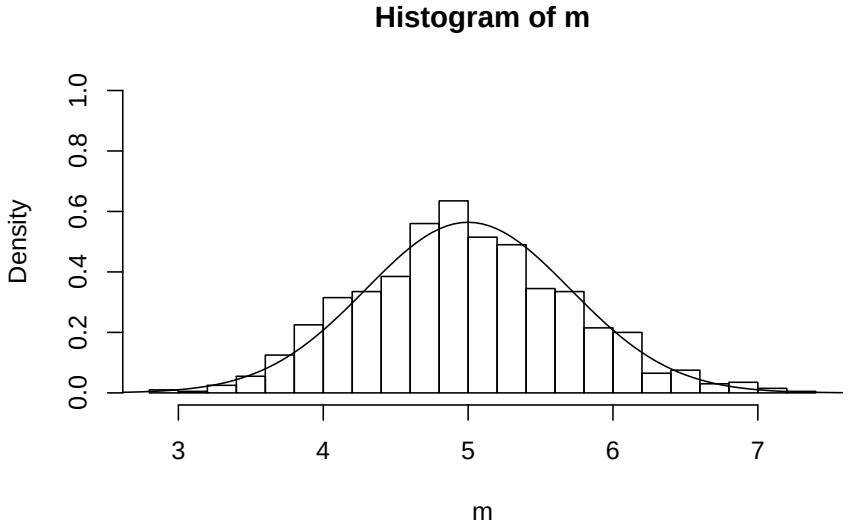
```
> x <- rpois(10, 5)
> mean(x)

[1] 4.5
```

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 di un ciclo `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),

```
> hist(m, freq = FALSE, ylim = c(0, 1), breaks = 20)
> curve(dnorm(x, mean = 5, sd = sqrt(5/10)), add = TRUE)
```

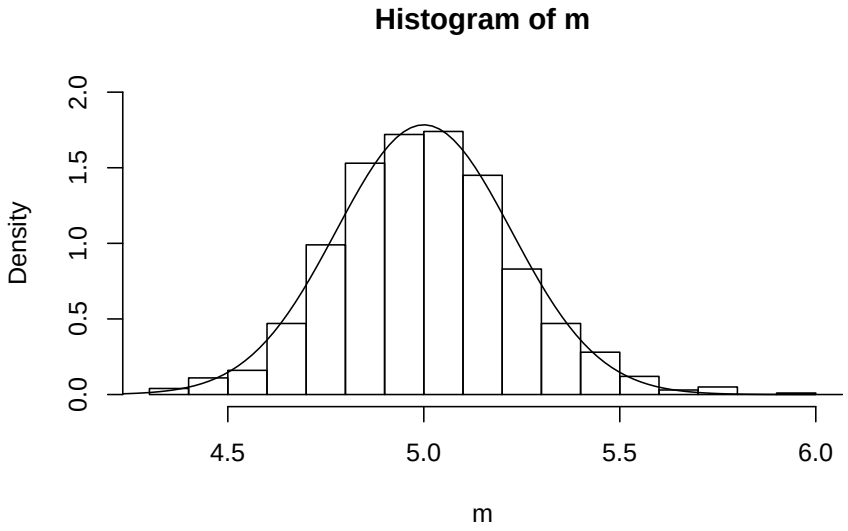


Come si può vedere queste distribuzioni differiscono ancora tra loro e non di poco. A questo punto, effettuiamo sempre $M = 1000$ iterazioni, ma aumentiamo la dimensione dei campioni generati passando da 10 a 100.

```
> 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), breaks = 20)
> curve(dnorm(x, mean = 5, sd = sqrt(5/100)), add = TRUE)
```



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

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

Esercizio: 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 Il campionamento e la stima puntuale

Il file laureati.dat contiene i dati relativi a 467 laureati triennali in Economia nel 2003. Le variabili considerate sono corso di laurea, matricola (dato modificato per ragioni di *privacy*), sesso, sigla provincia di residenza, anno prima immatricolazione a Venezia, tipo immatricolazione, diploma maturità, voto maturità, base voto maturità (60 o 100), voto laurea, lode (L, si; NL, no).

```
> laureati <- read.table(file = "laureati.dat",
+     header = FALSE)
> names(laureati) <- c("corso", "matricola", "sesso",
+     "provincia", "anno", "tipo", "diploma", "votom",
+     "base", "votol", "lode")
```

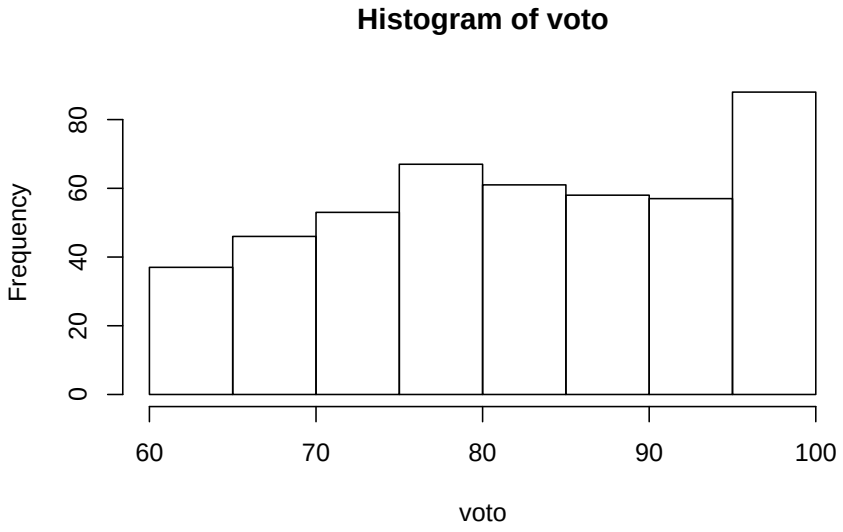
Di questi 467 studenti considereremo il voto di maturità

```
> attach(laureati)
> voto <- votom/base * 100
> hist(voto)
> media <- mean(voto)
> media
```

```
[1] 83.06138
```

```
> varianza <- mean(voto^2) - media^2
> varianza
```

```
[1] 138.1728
```



Poniamoci ora il problema di estrarre un campione di numerosità 30. Utilizzeremo la funzione `sample()`.

```
> nobs <- dim(laureati)[1]
> ind <- sample(1:nobs, 10, replace = TRUE)
> mean(voto[ind])
```

```
[1] 80.76667
```

Il valore 80.767 è una stima della media 83.061 ed è ottenuta mediante un campionamento con reinserimento. Se vogliamo un campionamento senza reinserimento scriveremo

```
> ind <- sample(1:nobs, 10, replace = FALSE)
> mean(voto[ind])
```

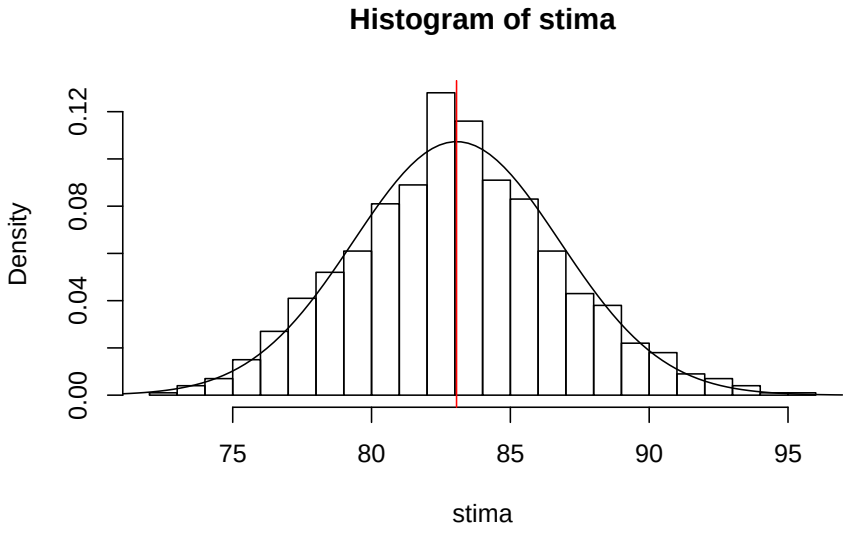
```
[1] 81.63333
```

Vogliamo vedere ora la distribuzione di questo stimatore confrontandola con quella di una v.c. $\mathcal{N}(83.061, \sqrt{138.173/10})$ (perché?).

```
> ncampioni <- 1000
> stima <- numeric(ncampioni)
> n <- 10
> for (i in 1:ncampioni) {
+     ind <- sample(1:nobs, n, replace = TRUE)
+     stima[i] <- mean(voto[ind])
+ }
> summary(stima)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  69.40   80.40   83.00   83.01   85.53   94.40
```

```
> hist(stima, freq = FALSE, nclass = 20)
> curve(dnorm(x, mean = media, sd = sqrt(varianza/n)),
+     add = TRUE)
> abline(v = media, col = 2)
```



Maschi e femmine sembrano ottenere risultati differenti all'esame di maturità

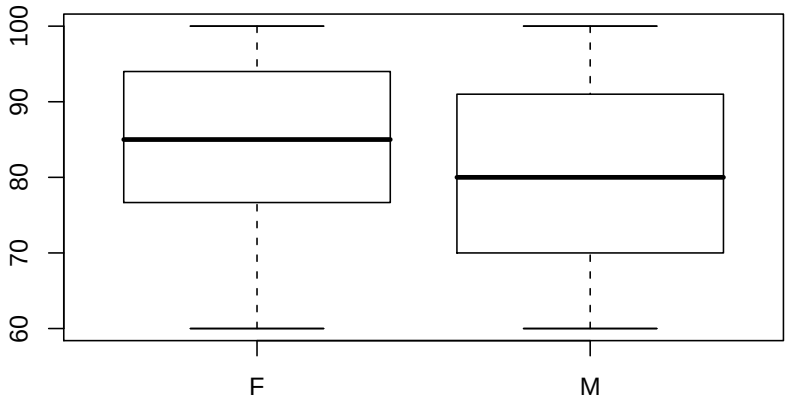
```
> summary(voto[sesso == "F"])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  60.00   76.67   85.00   84.70   94.00  100.00
```

```
> summary(voto[sesso == "M"])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  60.00   70.00   80.00   80.58   91.00  100.00
```

```
> boxplot(voto ~ sesso)
```



Per estrarre un campione (senza reinserimento) di 30 unità dalle femmine e un analogo campione di 30 unità dai maschi scriveremo

```
> indice.femmine <- (1:nobs)[sesso == "F"]
> ind.f <- sample(indice.femmine, 30, replace = FALSE)
> mean(voto[ind.f])

[1] 85.25556

> indice.maschi <- (1:nobs)[sesso == "M"]
> ind.m <- sample(indice.maschi, 30, replace = FALSE)
> mean(voto[ind.m])

[1] 79.63333
```

Queste sono due stime delle medie delle due sottopopolazioni `'M'` e `'F'`.

5.4 Intervalli di confidenza

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

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

> babies$sesso[babies$sesso == 1] <- "F"
> babies$sesso[babies$sesso == 2] <- "M"
> babies$sesso <- factor(babies$sesso)
> babies$sesso
```

```
 [1] F F M M M F F M M M M M F F M F F M M M M F F F F M M M
[29] F M F M M M M M F M M M M F F F
Levels: F M
```

```
> attach(babies)
```

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 gli estremi 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] 3115.418

> mean(peso) + qt(0.975, df = 43) * sqrt(var(peso)/44)

[1] 3436.491
```

Esercizio: Verificare che l'intervallo di confidenza si allarga all'aumentare del livello di confidenza.

Osserviamo che nel linguaggio R è presente un insieme di funzioni grazie alle quali è possibile ridurre il numero di istruzioni per calcolare intervalli di confidenza e per condurre, come vedremo, verifiche d'ipotesi. Quindi, ad esempio, si ha

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

        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

> peso.test$conf.int

[1] 3115.418 3436.491
attr(,"conf.level")
[1] 0.95
```

Per finire, facciamo una simulazione che aiuta a capire l'esatta interpretazione da dare agli intervalli di confidenza, in particolare al livello di confidenza.
A tale scopo, riconsideriamo i dati relativi ai laureati in Economia. Da un certo numero di campioni (`ncampioni`) di numerosità `n` estratti dall'insieme dei 467 laureati, costruiamo intervalli di confidenza per la media

```
> ncampioni <- 100
> stima <- numeric(ncampioni)
> varianza <- numeric(ncampioni)
> n <- 50
> for (i in 1:ncampioni) {
+     ind <- sample(1:nobs, n, replace = TRUE)
+     stima[i] <- mean(voto[ind])
+     varianza[i] <- var(voto[ind])
+ }
> mean(stima)

[1] 83.03253

> mean(voto)

[1] 83.06138
```

Costruiamo adesso una matrice con 2 righe e `ncampioni` colonne, nella j-esima colonna la prima riga rappresenta l'estremo inferiore mentre la seconda riga rappresenta l'estremo superiore dell'intervallo di confidenza relativo al j-esimo campione.

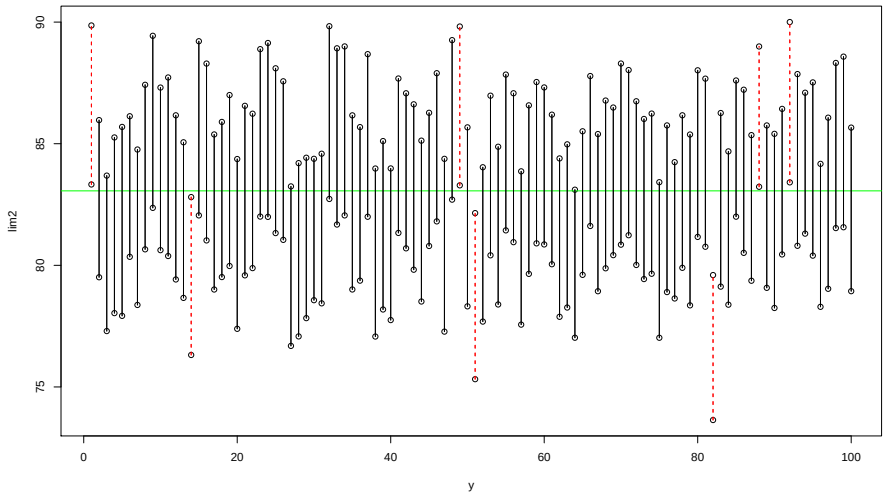
```
> zeta <- qt(0.975, df = n - 1)
> lim <- matrix(0, 2, ncampioni)
> lim[1, ] <- stima - zeta * sqrt(varianza)/sqrt(n)
> lim[2, ] <- stima + zeta * sqrt(varianza)/sqrt(n)
```

Possiamo controllare che percentuale di tali intervalli contiene la (vera) media della popolazione (di 467 unità):

```
> vero <- mean(voto)
> ok <- (lim[1, ] < vero) & (lim[2, ] > vero)
> mean(ok)

[1] 0.93
```

In effetti, il 93% degli intervalli contiene la vera media della popolazione. Questo valore è vicino al valore nominale 95%. Per ottenere una valutazione più accurata avrei dovuto fare una simulazione con più campioni.
Con i comandi seguenti possiamo visualizzare i 100 intervalli di confidenza:

```
> y <- seq(1, ncampioni)
> y <- matrix(y, ncampioni, 2)
> lim2 <- t(lim)
> plot(y, lim2)
> abline(h = vero, col = "green")
> segments(y[ok], lim2[ok, 1], y[ok], lim2[ok, 2])
> segments(y[!ok], lim2[!ok, 1], y[!ok], lim2[!ok,
+     2], col = "red", lty = "dashed")
```



Gli intervalli tratteggiati sono quelli che non contengono il vero valore del parametro. Il comando `segments()` produce segmenti che collegano i vertici indicati (guardate l'help per capire la sintassi).

Esercizio: Per l'insieme di dati nel file homedata.dat trovare un intervallo di confidenza al 90% per la media di entrambe le variabili, `y1970, y2000`. Utilizzare il comando `t.test()` dopo averne discusso l'appropriatezza (normalità dei dati).

Capitolo 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 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 osservato è dato da

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

[1] 0.04125998
```

La medesima verifica d'ipotesi può essere condotta utilizzando il comando `t.test()`

```
> 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 effettuare anche test unilaterali 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)

        One Sample t-test

data:  polveri
t = 2.3105, df = 11, p-value = 0.02063
alternative hypothesis: true mean is greater than 1.07
95 percent confidence interval:
 1.206985      Inf
sample estimates:
mean of x
    1.685

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

        One Sample t-test

data:  polveri
t = 2.3105, df = 11, p-value = 0.9794
```

```
alternative hypothesis: true mean is less than 1.07
95 percent confidence interval:
     -Inf 2.163015
sample estimates:
mean of x
    1.685
```

Quale sarà la vostra decisione in questo caso?

Esercizio: Una libreria universitaria ritiene che in media uno studente paghi 101.75$ per ogni libro di testo. Un gruppo di studenti intende valutare questa affermazione selezionando a caso 10 corsi e valutando il costo dei libri di testo per ognuno di questi. I dati raccolti sono:

```
 140 125 150 124 143 170 125 94 127 53
```

Effettuare un test di significatività per saggiare l'ipotesi nulla $H_0 : \mu = 101.74$ contro l'alternativa $H_1 : \mu > 101.74$.

Esercizio: Il file nortemp.dat contiene le temperature corporee (variabile `temperature`) misurate su 130 individui sani. Si può considerare che la temperatura corporea di un individuo sano sia 98.6 ^{o}F?

6.2 Il test t a due campioni

Riconsideriamo l'insieme di dati relativi a 467 laureati triennali in Economia nel 2003.

```
> laureati <- read.table(file = "laureati.dat",
+     header = FALSE, col.names = c("corso", "matricola",
+         "sesso", "provincia", "anno", "tipo",
+         "diploma", "votom", "base", "votol", "lode"))
```

Ci chiediamo se vi è una differenza nel voto di maturità tra le femmine e i maschi. Se procediamo in termini inferenziali possiamo estrarre un campione all'incirca proporzionale alla numerosità delle due sottopopolazioni

```
> attach(laureati)
> table(sesso)

sesso
  F   M
281 186

> set.seed(191)
> voto.f <- sample(votol[sesso == "F"], 30, replace = TRUE)
> voto.m <- sample(votol[sesso != "F"], 20, replace = TRUE)
```

Assumendo che il voto dei maschi e delle femmine segua una distribuzione normale $\mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = \mu_F, \mu_M$, conduciamo una verifica d'ipotesi

$$\begin{cases} H_0: & \mu_M = \mu_F \\ H_1: & \mu_M \neq \mu_F \end{cases}$$

```
> t.test(voto.m, voto.f, var.equal = TRUE)

        Two Sample t-test

data:  voto.m and voto.f
t = 0.4327, df = 48, p-value = 0.6671
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.646384  5.646384
sample estimates:
mean of x mean of y
    101.2     100.2
```

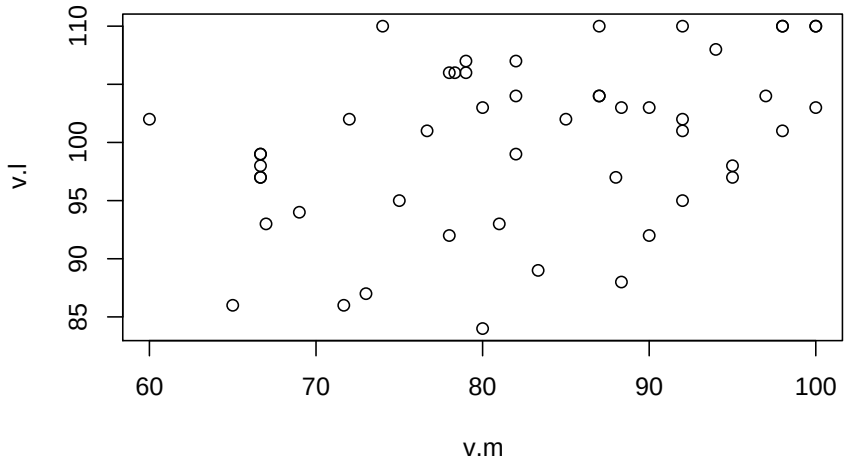
Da cui possiamo concludere che una differenza secondo il sesso del voto di laurea sembra da escludersi.

Esercizio: Il file nortemp.dat contiene le temperature corporee (variabile `temperature`) misurate su 130 individui sani, divisi per sesso (variabile `gender`). Dopo aver verificato la normalità dei dati, controllare graficamente e tramite un test t se la differenza nelle temperature di maschi e femmine si può considerare significativa.

6.3 La regressione lineare

Sempre riguardo allo stesso insieme di dati ci possiamo chiedere se il voto conseguito alla maturità sia un indicatore della successiva riuscita negli studi universitari. Consideriamo un campione casuale semplice di 50 studenti e disegnamo il diagramma di dispersione del voto di laurea rispetto al voto di maturità

```
> studenti <- sample(1:467, 50, replace = TRUE)
> v.m <- votom[studenti]/base[studenti] * 100
> v.l <- votol[studenti]
> plot(v.l ~ v.m)
```

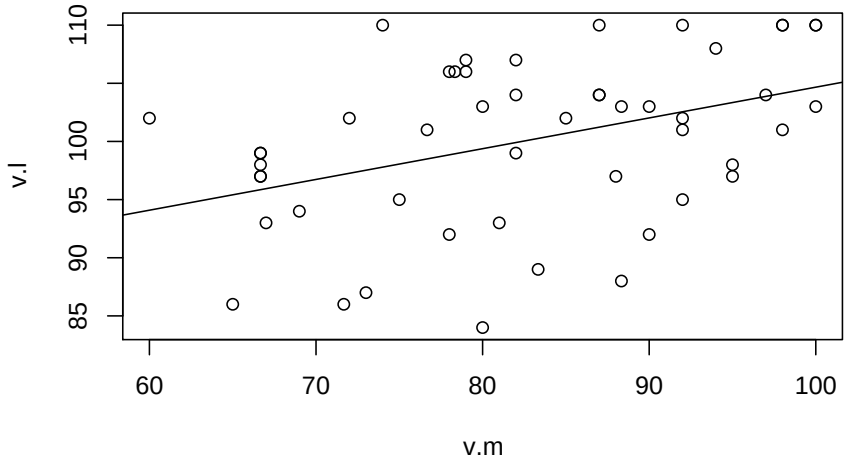


La relazione, se esiste, sembrerebbe essere debole. Consideriamo ora un modello di regressione lineare semplice

$$y_i = \alpha + \beta x_i + \varepsilon_i, \qquad i = 1, \ldots, 50$$

con $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ indipendenti e utilizziamo di nuovo il comando `lm()` per la stima secondo i minini quadrati del modello.

```
> fit <- lm(v.l ~ v.m)
> plot(v.l ~ v.m)
> abline(fit)
```



Il sistema d'ipotesi che adotteremo sarà il seguente

$$\begin{cases} H_0: & \beta = \beta_0 \\ H_1: & \beta \neq \beta_0 \end{cases},$$

dove nel nostro caso $\beta_0 = 0$.

Calcoliamo ora la statistica test

$$t = \frac{\hat{\beta} - \beta_0}{\sqrt{\hat{var}(\hat{\beta})}}, \qquad \text{con} \qquad \sqrt{\hat{var}(\hat{\beta})} = \sqrt{\frac{\hat{\sigma}^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}}$$

e il livello di significatività osservato $p_{oss} = \Pr_{H_0}\{|T| > |t|\}$

```
> sigma2 <- sum(fit$resid^2)/fit$df.residual
> beta <- as.numeric(fit$coefficients[2])
> var.beta <- (sigma2/sum((v.m - mean(v.m))^2))
> stat.test <- beta/sqrt(var.beta)
> stat.test
```

```
[1] 3.088896
```

```
> p.value <- 2 * (1 - pt(abs(stat.test), fit$df.residual))
> p.value
```

```
[1] 0.003336494
```

Si noti che si possono ottenere più semplicemente le quantità richieste considerando le ultime tre colonne prodotte con il comando `summary()` applicato ad un oggetto di tipo `lm()`.

```
> summary(fit)
```

```
Call:
lm(formula = v.l ~ v.m)

Residuals:
    Min      1Q  Median      3Q     Max
-15.377  -4.153   1.225   5.232  12.210

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  78.2232     7.1378  10.959 1.16e-14 ***
v.m           0.2644     0.0856   3.089  0.00334 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.63 on 48 degrees of freedom
Multiple R-Squared: 0.1658,        Adjusted R-squared: 0.1484
F-statistic: 9.541 on 1 and 48 DF,  p-value: 0.003336
```

Il livello di significatività osservato ci dà un'indicazione abbastanza in controtendenza rispetto alla nostra prima intuizione.

Esercizio: Il file diamond.dat contiene i dati relativi a 48 anelli di diamanti. La variabile `prezzo` registra il prezzo, e la variabile `carat` registra la dimensione del diamante. Fare il diagramma di dispersione di `carat` verso `prezzo`. Aggiungere la retta di regressione dopo aver controllato la significatività del coefficiente di regressione della variabile `carat`.

6.4 Tabelle di contingenza

Relativamente al campione di 50 studenti, ci chiediamo ora se vi è una qualche associazione tra il sesso e il voto di lode. Prima di tutto calcoliamo la relativa tabella di contingenza

```
> tab.cont <- table(sesso[studenti], lode[studenti])
> tab.cont

   L NL
F  3 24
M  3 20
```

Le tabelle marginali si ottengono con i seguenti comandi

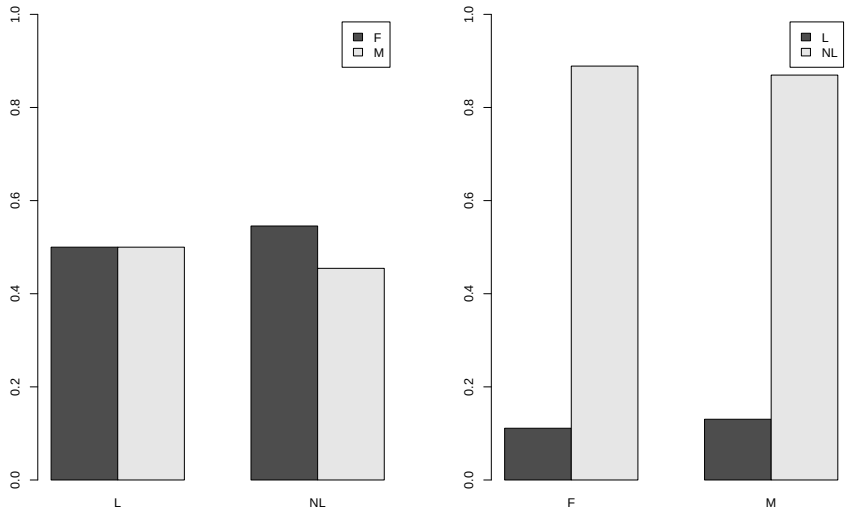
```
> margin.table(tab.cont, 1)

 F  M
27 23

> margin.table(tab.cont, 2)

 L NL
 6 44
```

Il comando `prop.table()` fornisce la tabella con le frequenze relative e, con l'opportuna opzione (1 o 2), le distribuzioni relative condizionate alle modalità delle righe o delle colonne

```
> prop.table(tab.cont)

     L   NL
F 0.06 0.48
M 0.06 0.40

> prop.table(tab.cont, 1)

          L        NL
F 0.1111111 0.8888889
M 0.1304348 0.8695652

> prop.table(tab.cont, 2)

          L        NL
F 0.5000000 0.5454545
M 0.5000000 0.4545455
```

Le distribuzioni condizionate possono essere visualizzate tramite barplot:

```
> par(mfrow = c(1, 2))
> barplot(prop.table(tab.cont, 2), beside = T, legend = T,
+     ylim = c(0, 1))
> barplot(t(prop.table(tab.cont, 1)), beside = T,
+     legend = T, ylim = c(0, 1))
```



Non sembra esserci grande dipendenza tra le due variabili (nel senso che le distribuzioni condizionate non cambiano molto al variare della modalità condizionante). L'associazione tra le due variabili può essere sottoposta a verifica formale mediante un sistema d'ipotesi nella forma

$$\begin{cases} \mathrm{H}_0 : \pi_{ij} = \pi_{i+}\pi_{+j}, \;\; i = 1,2 \;\; j = 1,2 \quad \text{(indipendenza)} \\ \mathrm{H}_1 : \text{le } \pi_{ij} \text{ non rispettano i vincoli previsti da } \mathrm{H}_0 \quad \text{(dipendenza)} \end{cases}$$

dove π_{ij}, π_{i+} e π_{+j} sono, rispettivamente, le probabilità congiunte e marginali, utilizzando il comando

```
> summary(tab.cont)

Number of cases in table: 50
Number of factors: 2
Test for independence of all factors:
        Chisq = 0.04392, df = 1, p-value = 0.834
        Chi-squared approximation may be incorrect
```

Si noti che, quando una frequenza congiunta è "piccola", l'approssimazione asintotica può risultare inadeguata. Comunque il livello di significatività ci indica un'assenza di associazione.

Esercizio: Analizzare graficamente la relazione fra colore degli occhi e dei capelli per i dati relativi alle femmine dell'insieme di dati `HairEyeColor` di R. Effettuare un test per verificare la dipendenza delle due variabili.

Capitolo 7

Mischiare R e LaTeX

7.1 Scrivere rapporti statistici

L'analisi statistica viene usualmente divisa in due parti:

1. L'analisi dei dati, utilizzando un programma statistico che utilizza dei file di dati e dei file per memorizzare le procedure seguite (il codice) e i risultati ottenuti;

2. Si utilizzano i risultati per scrivere un rapporto utilizzando un altro strumento informatico.

Quando il processo di analisi, come spesso accade, non è sequenziale, ma ad esempio si fanno modifiche alle procedure statistiche utilizzate mentre si scrive il rapporto, è facile confondersi e alla fine non ricordare più a quale tipo di analisi statistica corrispondano i risultati presenti nel rapporto. Questo modo di procedere può creare ulteriori problemi se l'analisi deve essere rivista dopo mesi.
È bene allora utilizzare procedure che integrino l'aspetto di analisi e di stesura del rapporto statistico contenendo, i dati, le procedure statistiche e le conclusioni in modo da ottenere un documento che da un punto di vista filosifico possa essere utilizzato come "dimostrazione" del risultato statistico.
È possibile ottenere un risultato di questo tipo utilizzando congiuntamente il programma statistico R e il programma di formattazione (e molto altro) LaTeX.
Attraverso questi due strumenti otterremo un rapporto statistico riproducibile, cioè che porterà agli stessi risultati, mostrando il modo con cui esso sono stati ottenuti, e anche rapporti statistici dinamici, che consentono di essere velocemente riutilizzati per nuovi dati.

7.2 Sweave

`Sweave` è uno strumento che consente di incorporare codice R all'interno dei documenti prodotti con LaTeX; cioè il nostro documento conterrà sia la documentazione (scritta in LaTeX) che il codice R necessario a svolgere tutte le analisi statistiche. Il documento sarà analizzato da `Sweave` che estrarrà il codice R , lo eseguirà e formatterà il risultato in modo che possa essere inserito sostituito al codice all'interno del documento LaTeX. Anche le figure e le tabelle saranno prodotte e inserite

automaticamente nella posizione richiesta. A questo punto avremo un documento LaTeXche può essere compilato per produre, ad esempio, un documento in *Portable Data Format* (pdf).
Da un punto di vista interno, `Sweave` è un insieme di funzioni di R presenti nel pacchetto `utils` che

- Sono in grado di estrarre codice R da un documento LaTeX;
- Restituiscono l'elaborazione ottenuta eseguendo il codice R (se richiesto);
- Creano i grafici e il codice LaTeXnecessario per inserirli nel documento (se richiesto);
- Creano un file con il codice R (se richiesto).

7.3 Come istallare `Sweave`

Se R e LaTeXsono presenti nel vostro sistema, non è necessaria nessuna ulteriore istallazione. Dalla versione 1.5.0 di R , `Sweave` è disponibile nella versione base all'interno del pacchetto `utils` che è caricato automaticamente all'inizio di ogni sessione. Inoltre, per chi conosce già R e LaTeXnon vi è la necessità di imparare nessun nuovo linguaggio.

7.4 Come funziona

Il rapporto statistico sarà scritto in un file in stile LaTeXcon una estensione `.Rnw` (o `.Snw`) invece dell'usuale estensione `.tex`, ad esempio: `esempio.Rnw`. Lo stesso file conterrà anche il codice R opportunamente separato dal contenuto scritto in LaTeX. Dalla console di R sarà sufficiente digitare `Sweave('esempio.Rnw')`, assumendo che il vostro file sia nella cartella di lavoro di R . Questo comando valuterà il codice R e produrrà un file dal nome `esempio.tex` che è pronto per essere compilato con LaTeX.

7.5 La sintassi Noweb

Per separare il codice dalla documentazione si può usare la sintassi chiamata `Noweb`. `Noweb` è un semplice strumento di *literate programming* che consente di mischiare codice sorgente scritto in un qualche linguaggio di programmazione con la documentazione del programma stesso. I paragrafi di codice, detti *chunks* sono individuati dalla struttura

- `<< options >>=` indica l'inizio del *chunk*, dove `options` indica diverse opzioni spiegate in seguito;
- `@` indica la fine del *chunk*.

Sono possibili due operazioni

- *weave*: scrivi la documentazione insieme con il codice eseguito e i suoi risultati;

- *tangle*: estrai il codice.

Le opzioni principali sono

- *label*: consente di dare un nome al *chunk*;
- *echo*: se `TRUE` mostra nel documento finale il codice R presente nel *chunk*;
- *fig*: se `TRUE` crea una immagine.

7.6 Un esempio: *esempio.Rnw*

Di seguito mostriamo il contenuto del file *esempio.Rnw* che contiene la generazione dei dati (codice etichettato con Primo), l'analisi degli stessi con un modello di regressione (Secondo) e la creazione di alcuni grafici (Terzo).

```
\documentclass[a4paper]{article}
\title{Un semplice esempio di uso di Sweave}
\author{Claudio Agostinelli}
\begin{document}
\maketitle
Nel primo \textit{chunk} (\verb+<<>>= @+) simuliamo
$20$ osservazioni per due variabili $x$ e $y$
e stimiamo un modello di regressione lineare semplice:
$y=\alpha+\beta x$ utilizzando il metodo dei minimi quadrati
<<label=Primo>>=
library(xtable)
x <- rnorm(20); e <- rnorm(20, 0, 0.2);
y <- 0.5*x+1+e
ls <- lm(y~x)
@
Di seguito riportiamo le stime dei coefficienti
e l'inferenza sui parametri
<<label=Secondo,echo=FALSE,results=tex>>=
xtable(summary(ls))
@
insieme al grafico dei dati, della retta dei regressione
stimata e del diagramma a scatola e baffi dei residui.
\begin{center}
<<label=Terzo,echo=FALSE,fig=TRUE,width=10,height=5>>=
par(mfcol=c(1,2))
plot(x,y,main='Least Squares',xlab='x',ylab='y')
abline(ls)
boxplot(ls$residuals,main='Boxplot of the residuals')
@
\end{center}
\end{document}
```

Nel Primo *chunck* non abbiamo specificato nessuna opzione (tranne l'etichetta) e quindi il comportamento predefinito è `echo=TRUE` e `fig=FALSE` e quindi verranno mostrati i comandi e non sarà prodotta nessuna immagine. Nel Secondo *chunck* non verrà incluso il codice (`echo=FALSE`) ma solo il risultato; l'opzione `results=tex` indica che il risultato è già scritto in LaTeXe quindi non ha bisogno di ulteriori elaborazioni prima di essere inserito nel documento. Il Terzo *chunck* produrrà una figura in quanto `fig=TRUE`. `include=TRUE` farà in modo che la figura venga inserita automaticamente nel documento, `pdf=TRUE,eps=FALSE` indicano che l'immagine sarà salvata in un file `pdf` e non `eps` e `width=10,height=5` indicano la dimensione della figura in pollici.
Per produrre il file `tex` è sufficiente utilizzare il comando `Sweave(esempio.Rnw)` dalla console di R . Attenzione: successive modifiche al documento andranno fatto sul file originale con estensione `Rnw` e non sul file `tex`.

```
> Sweave("esempio.Rnw")

Writing to file esempio.tex
Processing code chunks with options ...
 1 : echo keep.source term verbatim (label = Primo)
 2 : keep.source term tex (label = Secondo)
 3 : keep.source term verbatim pdf  (label = Terzo)

You can now run (pdf)latex on âesempio.texâ
```

Qui di seguito mostriamo il file ottenuto

```
\documentclass[a4paper]{article}
\title{Un semplice esempio di uso di Sweave}
\author{Claudio Agostinelli}
\usepackage{Sweave}
\begin{document}
\maketitle
Nel primo \textit{chunk} (\verb+<<>>= @+) simuliamo
$20$ osservazioni per due variabili $x$ e $y$
e stimiamo un modello di regressione lineare semplice:
$y=\alpha+\beta x$ utilizzando il metodo dei minimi quadrati
\begin{Schunk}
\begin{Sinput}
> library(xtable)
> x <- rnorm(20); e <- rnorm(20, 0, 0.2);
> y <- 0.5*x+1+e
> ls <- lm(y~x)
\end{Sinput}
\end{Schunk}
Di seguito riportiamo le stime dei coefficienti
e l'inferenza sui parametri
% latex table generated in R 2.14.0 by xtable 1.5-6 package
% Tue Mar 27 17:51:47 2012
```

```
\begin{table}[ht]
\begin{center}
\begin{tabular}{rrrrr}
  \hline
 & Estimate & Std. Error & t value & Pr($>$$|$t$|$) \\
  \hline
(Intercept) & 0.9617 & 0.0448 & 21.46 & 0.0000 \\
  x & 0.4242 & 0.0461 & 9.20 & 0.0000 \\
   \hline
\end{tabular}
\end{center}
\end{table}insieme al grafico dei dati, della retta dei regressione
stimata e del diagramma a scatola e baffi dei residui.
\begin{center}
\includegraphics{esempio-Terzo}
\end{center}
\end{document}
```

e dopo averlo compilato

```
> system("pdflatex esempio.tex")
```

Un semplice esempio di uso di Sweave

Claudio Agostinelli

March 27, 2012

Nel primo *chunk* (`<<>>= @`) simuliamo 20 osservazioni per due variabili x e y e stimiamo un modello di regressione lineare semplice: $y = \alpha + \beta x$ utilizzando il metodo dei minimi quadrati

```
> library(xtable)
> x <- rnorm(20); e <- rnorm(20, 0, 0.2);
> y <- 0.5*x+1+e
> ls <- lm(y~x)
```

Di seguito riportiamo le stime dei coefficienti e l'inferenza sui parametri insieme

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.9617	0.0448	21.46	0.0000
x	0.4242	0.0461	9.20	0.0000

al grafico dei dati, della retta dei regressione stimata e del diagramma a scatola e baffi dei residui.