Introduzione ad R

per il corso di

Probabilità e Statistica (Informatica)

Federica Giummolè
giummole@unive.it

Anno Accademico 2012-2013

Indice

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] "a"        "risposta" "tf"       "tratta"   "x"        "xx"       "y"
[8] "yy"       "z"
```

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 #Usa ** o ^ per calcolare un elevamento a potenza.

[1] 108
```

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

```
> sqrt(2)

[1] 1.414214

> sin(3.14159)  #sin(Pi greco) e' zero

[1] 2.65359e-06
```

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)  #salva in x la radice quadrata di 2
> x

[1] 1.414214

> x**3

[1] 2.828427

> log(x)->y
> 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          #fissa x uguale a 10
> x > 10         #  x e' piu' grande di 10?

[1] FALSE

> x <= 10

[1] TRUE

> tf <- x > 10
> tf

[1] FALSE

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

La stessa operazione può essere fatta con:

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

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

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  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  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 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  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('./dati/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 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Tuttavia, la struttura di data-frame 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 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Utilizziamo il comando `attach()` per comunicare ad R che le operazioni che faremo si riferiscono al 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 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

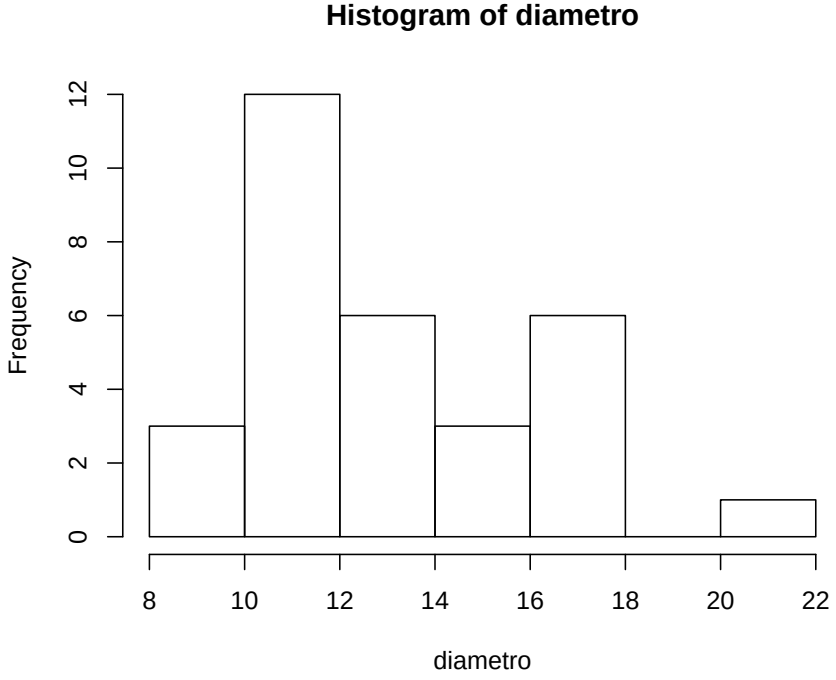
Per avere delle statistiche di base sulle variabili contenute in `Ciliegi` possiamo usare la funzione `summary()`:

```
> summary(Ciliegi)
```

```
    diametro        altezza       volume
 Min.   : 8.20   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
```

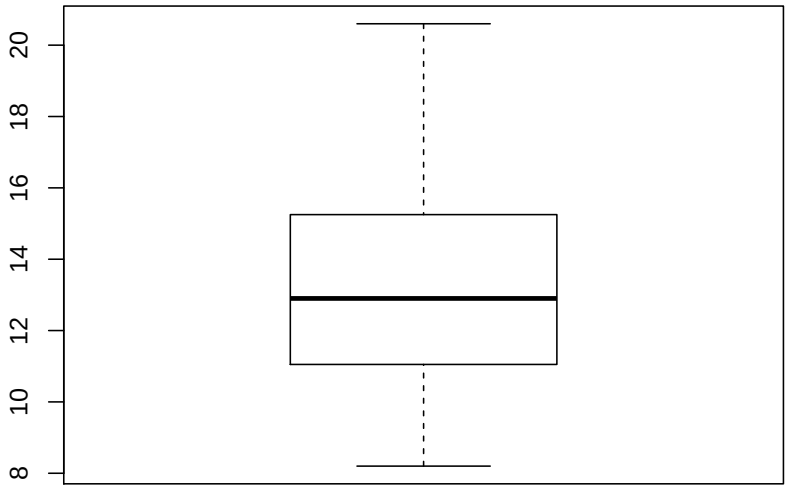
Possiamo anche rappresentare graficamente la distribuzione di una variabile ad esempio `diametro`, mediante un istogramma

```
> 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 71 64 78 80 74 72 77
[26] 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

> detach()
```

Esercizio: L'insieme di dati nel file laureati.txt 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)
+ }
```

O ancora più semplicemente:

```
> 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,

$$V(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} = \bar{x^2} - \bar{x}^2,$$

dove $\bar{x} = \sum_{i=1}^{n} x_i/n$ e $\bar{x^2} = \sum_{i=1}^{n} x_i^2/n$. Confrontate il risultato che ottenete con quello della funzione `var()` di R.

1.9.1 L'algoritmo di Erone per il calcolo della radice quadrata

I matematici della Mesopotamia conoscevano un algoritmo per il calcolo delle radici quadrate, oggi noto con il nome di algoritmo di Erone.
L'idea di base è la seguente: supponiamo che a_1 sia un'approssimazione per eccesso di $\sqrt{x}$, cioè $a_1 > \sqrt{x}$. Moltiplicando ambo i membri per $\sqrt{x}/a_1$ si ottiene $\sqrt{x} > x/a_1$. Quindi x/a_1 è un'approssimazione per difetto di $\sqrt{x}$. Prendendo allora il punto di mezzo fra x/a_1 e a_1 otterremo $a_2 = (a_1 + x/a_1)/2$, un'approssimazione migliore di $\sqrt{x}$. Si dimostri che $\sqrt{x} < a_2$. Proseguendo allo stesso modo con a_2 otterremo approssimazioni successive via via più precise per $\sqrt{x}$. Al passo n-esimo avremo:

$$a_n = \frac{a_{n-1} + \frac{x}{a_{n-1}}}{2}.$$

Costruiamo ora una funzione che utilizzi l'algoritmo di Erone per approssimare una radice quadrata con un errore più piccolo di un certo valore prefissato.

```
> erone<-function(x, a1, maxerr=0.1)
+ {r1 <- a1
+  rad <- a1
+  errore <- maxerr+1
+  while (errore > maxerr)
+        { r2 <- (r1+x/r1)/2
+          errore <-r1-r2
+          r1 <- r2
+          rad <- c(rad,r2)
+         }
+  return(rad)
+ }
```

Si osservi che all'interno della funzione abbiamo utilizzato un ciclo `while` la cui sintassi è

```
> while (condition) expr
```

dove `condition` è una condizione di uscita dal ciclo e `expr` è un'espressione che viene ripetuta ad ogni iterazione, fino a che `condition` non è verificata.
Sapendo che l'algoritmo funziona per $a_1 > \sqrt{x}$, possiamo aggiungere un controllo per verificare questa condizione prima di far girare l'algoritmo.

```
> erone<-function(x, a1, maxerr=0.1)
+ {r1 <- a1
+  rad <- a1
+  errore <- maxerr+1
+  if ( a1<0 | x>a1^2 )
+        { print("errore! a1 non valido")
+          errore<-maxerr-1
+          rad<-NA
+         }
+  while (errore > maxerr)
+        { r2 <- (r1+x/r1)/2
+          errore <-r1-r2
+          r1 <- r2
+          rad <- c(rad,r2)
+         }
+  return(rad)
+ }
```

Si cerchino informazioni sul comando `if`.
Adesso proviamo la nostra funzione:

```
> options(digits=10)
> erone(2,4,0.01)

[1] 4.000000000 2.250000000 1.569444444 1.421890364 1.414234286
```

Capitolo 2

Statistica descrittiva

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
165 166 170 171 172 173 174 175 176 178 179 180 181 183 184 185 186 187 188 190
  1   2   5   1   3   3   3   7   1   8   1   8   3   3   1   6   2   2   1   1
192 193
  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] (170,175] (170,175]
 [8] (180,185] (175,180] (185,190] (165,170] (180,185] (175,180] (180,185]
[15] (170,175] (175,180] (175,180] (170,175] (185,190] (170,175] (170,175]
[22] (175,180] (175,180] (175,180] (165,170] (180,185] (170,175] (190,195]
[29] (170,175] (180,185] (175,180] (175,180] (180,185] (170,175] (170,175]
[36] (175,180] (165,170] (185,190] (190,195] (180,185] (180,185] (185,190]
[43] (175,180] (170,175] (175,180] (180,185] (170,175] (170,175] (170,175]
[50] (165,170] (190,195] (175,180] (175,180] (160,165] (185,190] (165,170]
[57] (180,185] (165,170] (180,185] (170,175] (170,175] (170,175] (175,180]
[64] (175,180] (175,180]
10 Levels: (150,155] (155,160] (160,165] (165,170] (170,175] ... (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] (180,185] (185,190]
        0         0         1         7        17        18        13         6
(190,195] (195,200]
        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] (182,185] (185,187]
        3         5         7        11         9        11         4        10
(187,190] (190,193]
        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
```

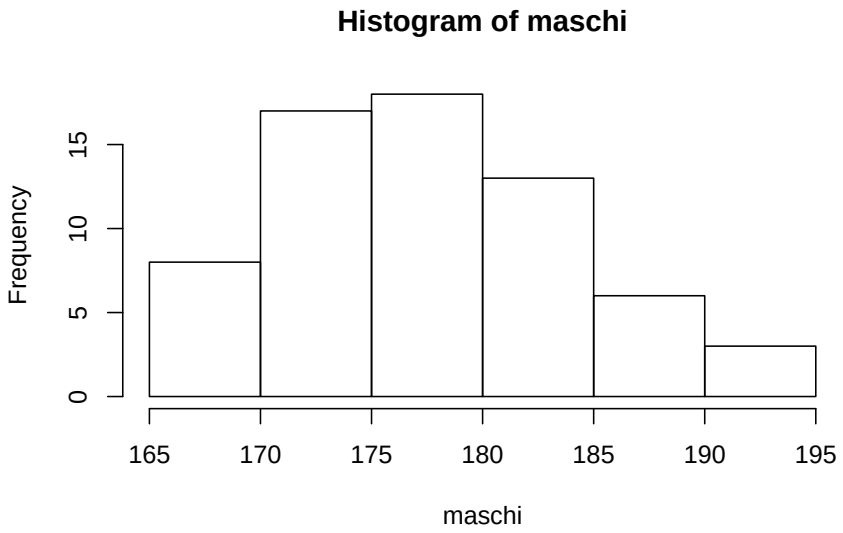
```
 (150,155]  (155,160]  (160,165]  (165,170]  (170,175]  (175,180]  (180,185]
0.00000000 0.00000000 0.01538462 0.12307692 0.38461538 0.66153846 0.86153846
 (185,190]  (190,195]  (195,200]
0.95384615 1.00000000 1.00000000
```

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

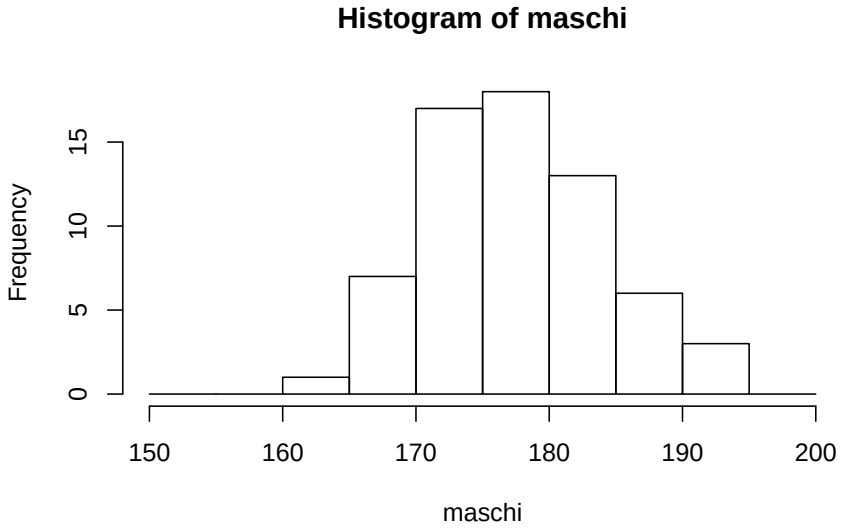
Costruiamo dapprima un istogramma di frequenza. Il comando più semplice è

```
> hist(maschi)
```

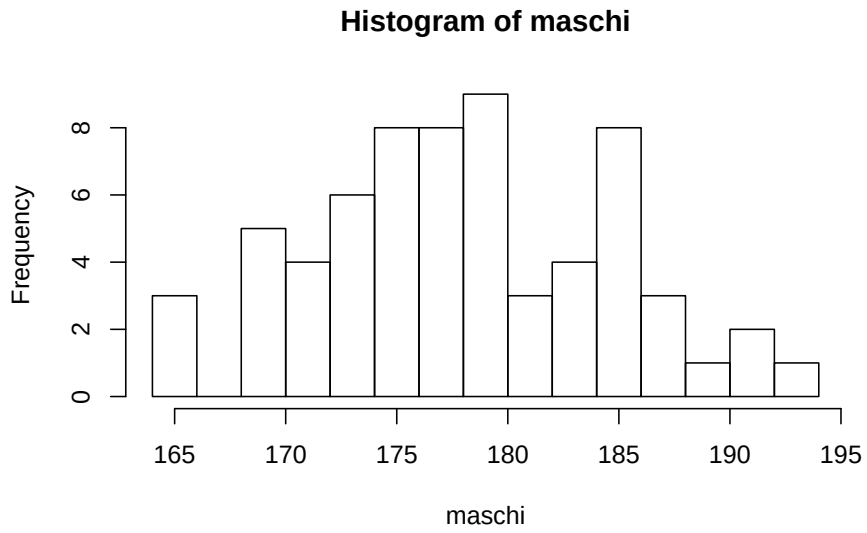


Come per `table()` anche `hist()` permette di stabilire il numero di classi in cui vogliamo rappresentare i nostri dati. Ad esempio

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



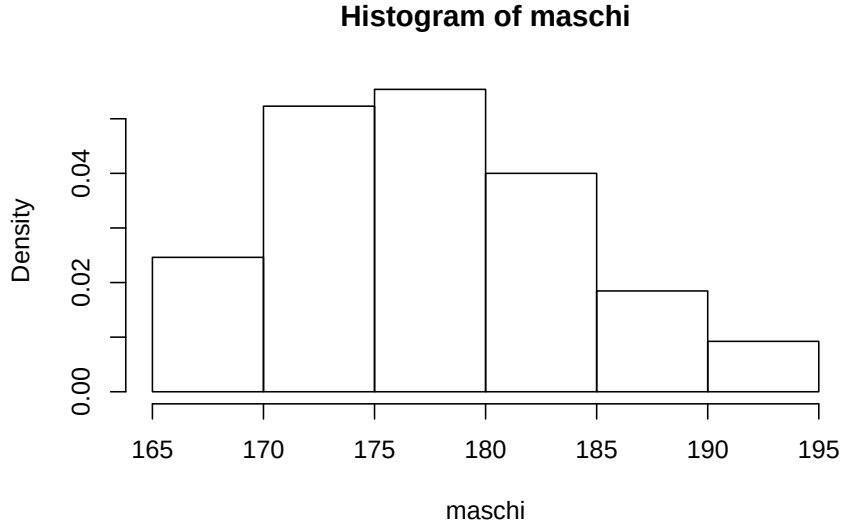

```
> hist(maschi,breaks=10)
```



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`, otteniamo un grafico analogo ma con le densità di frequenza sull'asse delle ordinate. Significa che l'area di ogni rettangolo coincide con la corrispondente frequenza relativa.

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

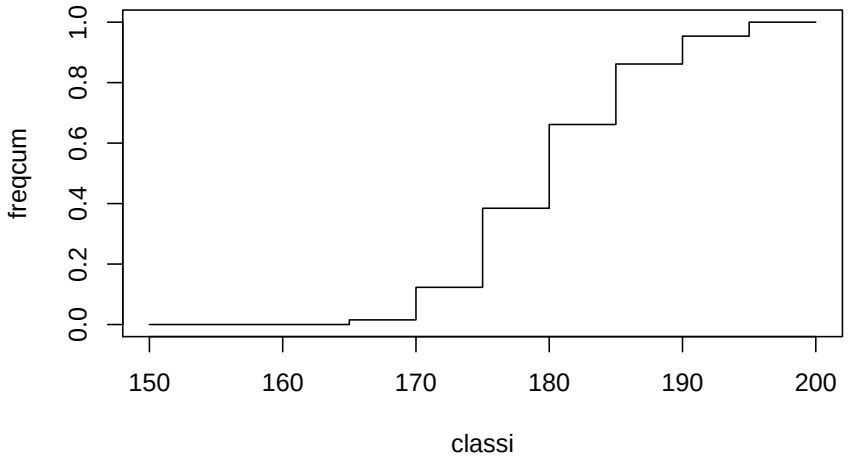


Possiamo anche ottenere un grafico della funzione di ripartizione come segue. Dapprima aggiungiamo un limite inferiore alle classi

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

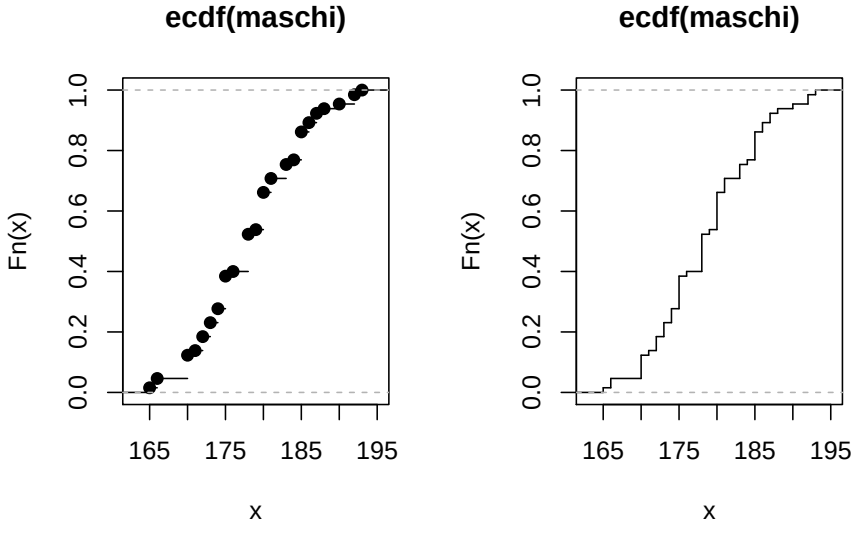
Quindi con il comando `plot()` rappresentiamo la funzione di ripartizione

```
> 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'.
Abbiamo appena tracciato il grafico della funzione di ripartizione empirica per i dati raggruppati in classi. Per i dati originali, possiamo utilizzare il comando `ecdf()`, che sta per *Empirical Cumulative Distribution Function*:

```
> par(mfrow=c(1,2))
> plot(ecdf(maschi))
> plot(ecdf(maschi),verticals= TRUE, do.points = FALSE)
> par(mfrow=c(1,1))
```



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()`

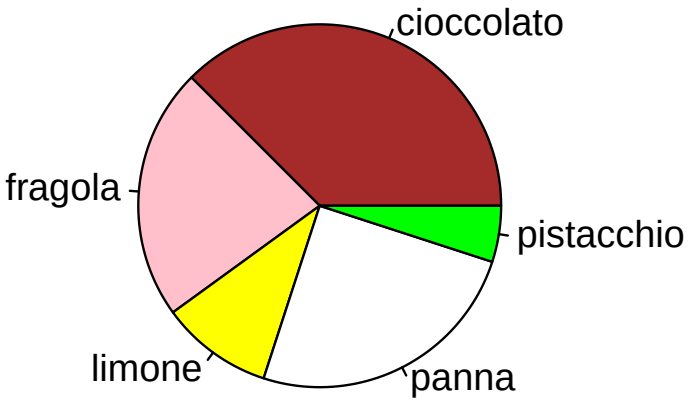
```
> gelati<-read.table('gelati.dat')

> 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
```

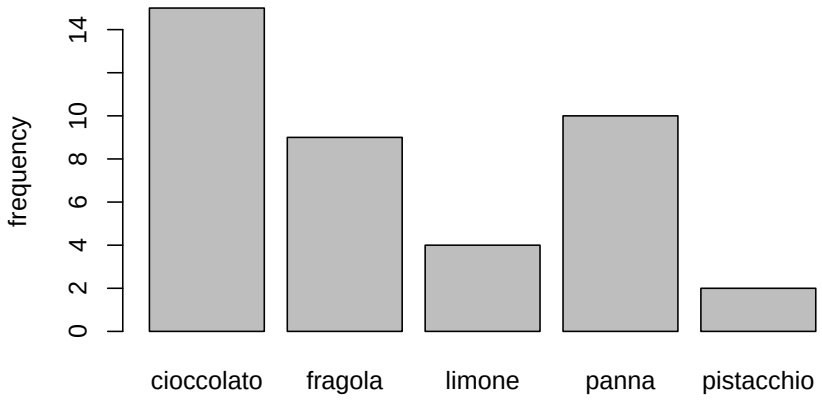
Proviamo a costruire un diagramma a torta

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



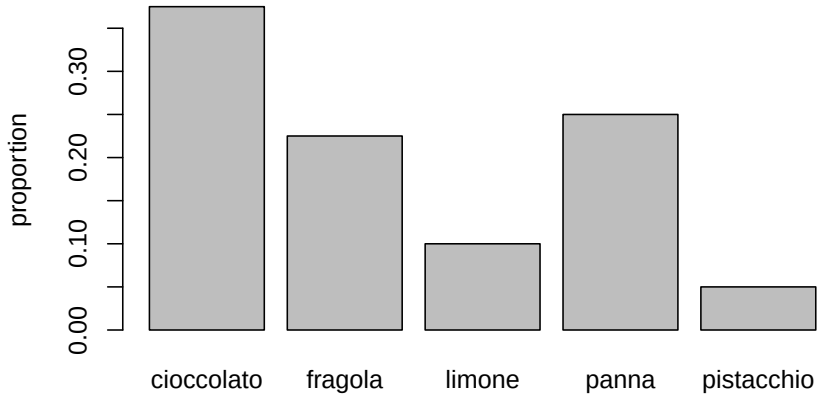
Un altro modo per rappresentare dati qualitativi è dato dal diagramma a barre o *barplot*. Possiamo costruire un barplot delle frequenze assolute

```
> 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(.15,.47))

15% 47%
172 178
```

La funzione `summary()` ha come risultato

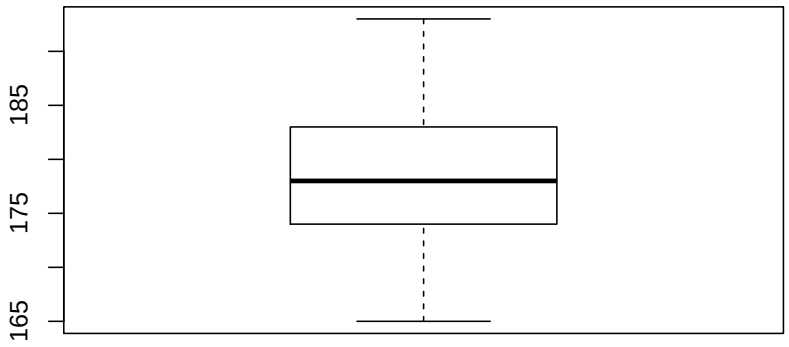
```
> summary(maschi)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  165.0   174.0   178.0   178.6   183.0   193.0
```

ovvero per quanto riguarda i quartili

```
> quantile(maschi,probs=c(0,.25,.50,.75,1))

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

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

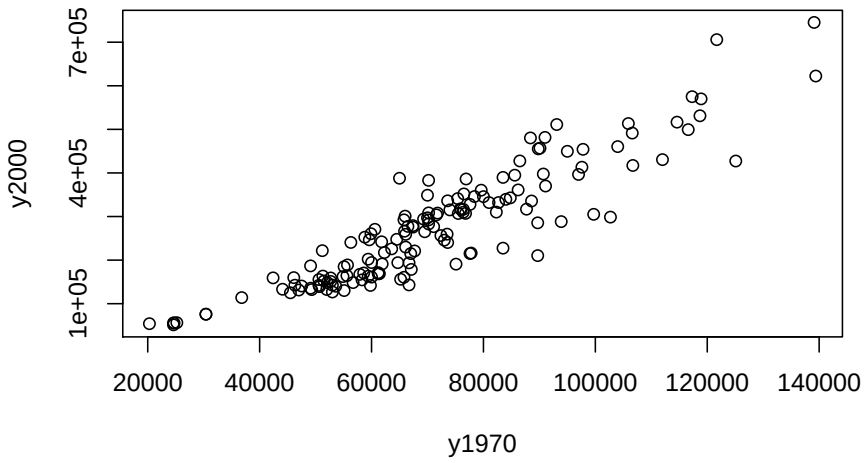
2.4.1 Diagrammi di dispersione

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)
```

```
The following object(s) are masked from 'homedata (position 3)':

    y1970, y2000
```

```
> 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.

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

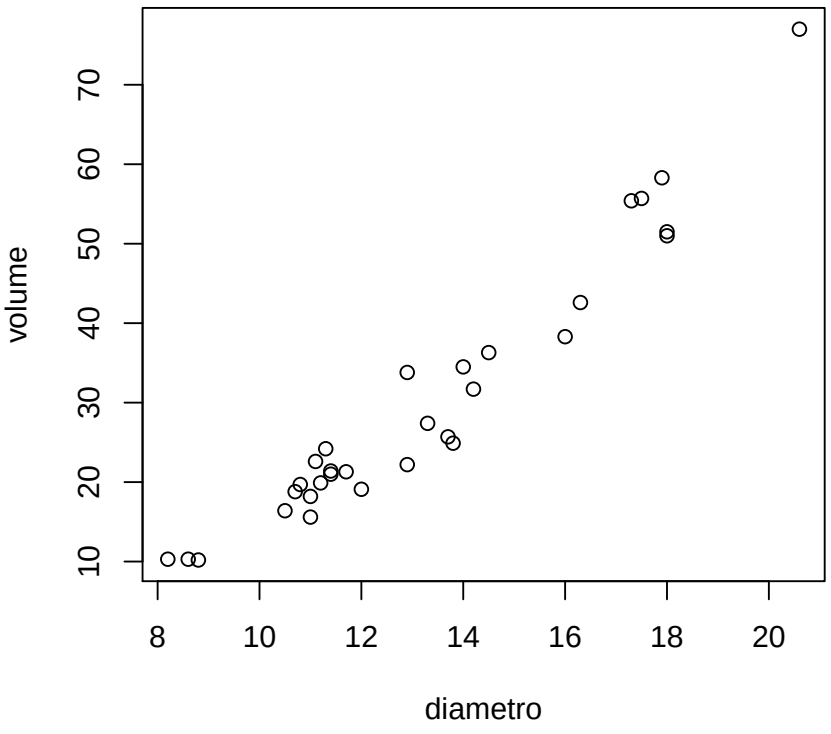
```
> Ciliegi <- read.table("Cherry.dat",
+       col.names=c('diametro','altezza','volume'))
```

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

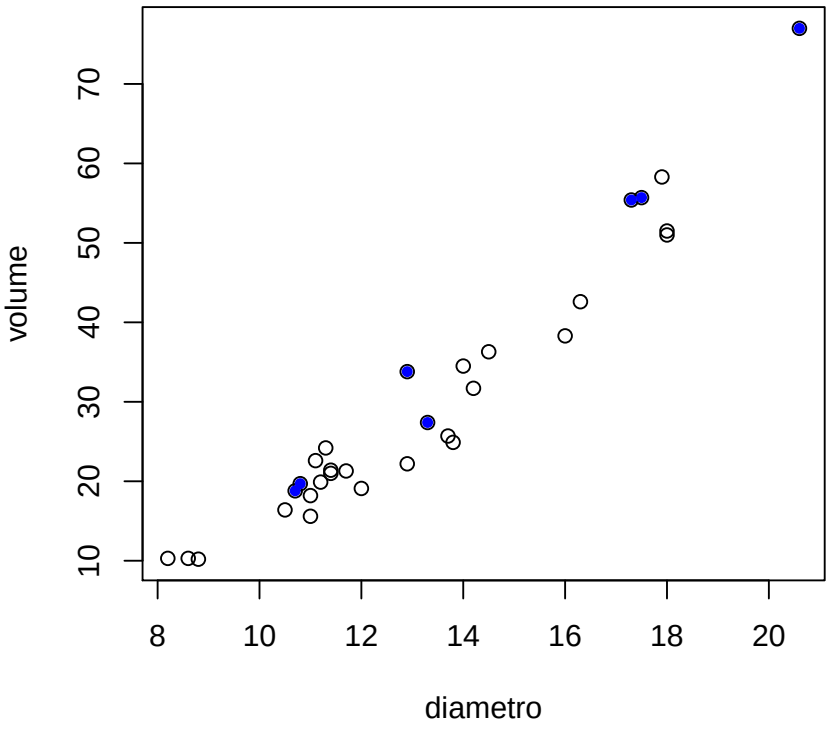
```
> attach(Ciliegi)
```

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.

```
> detach(Ciliegi)
```

2.4.2 Tabelle di frequenza

Consideriamo i dati *laureati.txt* che riguardano 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("laureati.txt",header=TRUE)
```

```
> names(laureati)
```

```
 [1] "corso"     "matricola" "sesso"     "provincia" "anno"      "tipo"
 [7] "diploma"   "votomat"   "base"      "votolau"   "lode"
```

```
> attach(laureati)
```

Consideriamo ora due variabili: il sesso e il voto di lode. Ci chiediamo ora se vi è una qualche associazione tra il sesso e il voto di lode. In altre parole ci chiediamo se i maschi riescono meglio negli studi rispetto alle femmine oppure il contrario. Per questo esaminiamo la seguente tabella di contingenza:

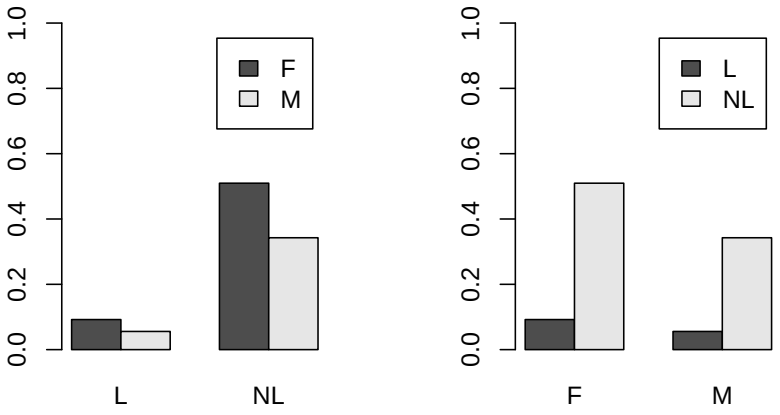
```
> tab.cont<-table(sesso,lode)
> tab.cont

     lode
sesso   L  NL
    F  43 238
    M  26 160
```

dove consideriamo tutti i possibili incroci di modalità delle due variabili ($2 \times 2 = 4$). e le frequenze rappresentano quante unità cadono in ciascun di questi incroci. Ad esempio, 43 è il numero di studenti (unità statistiche) di sesso femminile che hanno conseguito la laurea. Possiamo anche considerare le frequenze relative, ottenute semplicemente dividendo le frequenze assolute per il numero totale $n = 467$ di unità che è equivalente ad usare il comando `prop.table`.

```
> n<-length(sesso)
> tab.cont/n

     lode
sesso          L         NL
    F 0.09207709 0.50963597
    M 0.05567452 0.34261242

> prop.table(tab.cont)

     lode
sesso          L         NL
    F 0.09207709 0.50963597
    M 0.05567452 0.34261242
```

Possiamo rappresentare le frequenze (sia assolute che relative) della tabella attraverso un appropriato diagramma a barre. La stessa informazione può essere rappresentata in due modi diversi ("per riga" o "per colonna"). Stiamo rappresentando graficamente le frequenze condizionate:

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



2.4.3 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)
```

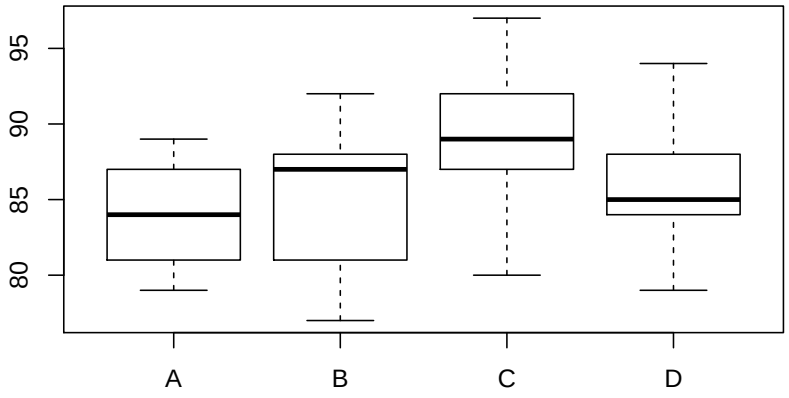
```
[1] TRUE
```

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)
```




```
> detach(pen)
```

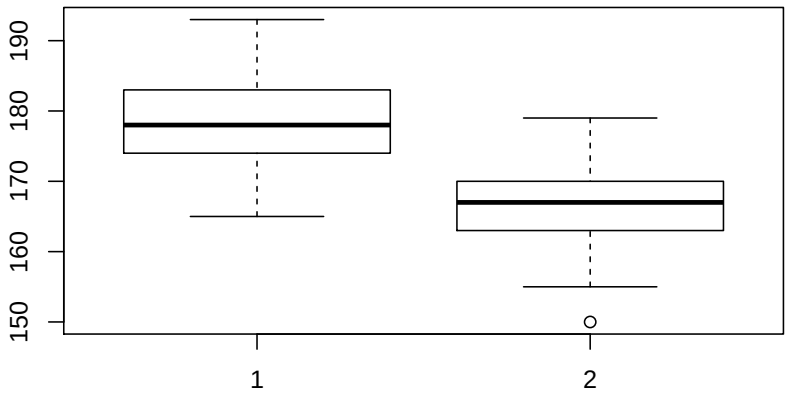
Confrontiamo ora le altezze di un gruppo di femmine con quelle di un gruppo di maschi

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

```
> femmine
```

```
 [1] 162 165 173 170 165 165 170 178 173 170 168 165 165 157 168 157 169 174 167
[20] 168 160 175 174 170 160 160 168 176 173 162 175 165 160 164 163 173 163 162
[39] 168 163 160 170 155 172 160 162 167 174 165 163 172 158 174 155 174 160 160
[58] 167 164 166 170 150 165 179 168 165 168 168 166 167 174 165 167 170
```

```
> boxplot(maschi,femmine)
```



2.5 Esercizi

1. 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)`.

2. Analizzare i dati `org.txt` come visto a lezione, riassumendoli dapprima in tabelle di frequenza per ognuna delle tre organizzazioni, rappresentandoli graficamente tramite istogrammi, boxplot affiancati e grafico sovrapposto delle funzioni di ripartizione empiriche.

3. Simon Newcomb misurò il tempo necessario affiché la luce andasse dal suo laboratorio sul fiume Potomac ad uno specchio alla base del Washington Monument e ritornasse indietro, per una distanza totale di circa 7400 metri. Egli fece 66 misurazioni qui sotto riportate (in milionesimi di secondo).

28	24	27	30	29	24
22	33	32	33	36	25
36	21	34	29	32	27
26	36	30	27	28	24
28	32	25	29	40	16
28	31	26	28	19	29
26	25	26	22	37	20
24	24	25	26	23	28
32	25	-44	27	32	27
30	28	23	16	29	39
27	36	21	31	-2	23

   (a) Si rappresenti la distribuzione dei dati mediante un diagramma a scatola con baffi;

   (b) si calcoli un opportuno indice di posizione per la velocità della luce.

4. Il diametro del fusto di una pianta viene misurato attraverso uno strumento chiamato "cavalletto" e simile ad un grosso calibro. La misura viene effettuata tenendo il cavalletto in posizione perpendicolare al fusto ad una altezza dal terreno di $1.30m$, con una precisione non superiore al cm. Nell'autunno del 1999 sono stati misurati i diametri di 1887 Abeti rossi (*Picea Abiens*) presenti in una zona di bosco a San Vito di Cadore. Le misure sono le seguenti:

D	f	D	f	D	f	D	f	D	f	D	f
18	21	28	43	38	45	48	38	58	17	68	7
19	47	29	48	39	41	49	26	59	23	69	1
20	34	30	51	40	43	50	31	60	12	70	5
21	69	31	65	41	49	51	46	61	14	71	0
22	74	32	76	42	45	52	48	62	10	72	6
23	52	33	64	43	42	53	23	63	11	73	9
24	46	34	72	44	39	54	39	64	4	74	0
25	28	35	33	45	40	55	30	65	3	75	4
26	49	36	32	46	47	56	29	66	0	76	0
27	40	37	59	47	35	57	16	67	4	77	2

D: diametro in cm, f: frequenza assoluta.

Di solito, per dati riferiti al diametro del fusto, l'informazione disponibile è già parzialmente sintetizzata attraverso l'uso di classi (chiamate classi diametriche) di ampiezza $5cm$ centrate nei valori: 20,25,30,$\cdots$,65,70, 75. Si costruisca questo nuovo insieme di dati e si calcolino, per ogni classe, le frequenza relative, assolute e cumulate. Si valuti se sia necessario calcolare le densità delle osservazioni per un eventuale istogramma.
Si calcoli la media con i dati originari e con i dati raggruppati; si commenti l'approssimazione indotta dall'uso delle classi.

5. In un gruppo di 50 soggetti classificati secondo la loro attitudine al fumo, è stata registrata la presenza di un'affezione bronchiale. Nel seguente prospetto sono riportati i risultati dello studio, tenendo presente che la variabile `Affezione bronchiale` presenta modalità 0 o 1 (assenza o presenza di affezione) e la variabile `Attitudine al fumo` ha modalità codificate con 1, 2 e 3, dove 1 sta ad indicare un non fumatore, 2 un ex fumatore e 3 un fumatore.

Affezione bronchiale	0	1	1	0	0	1	1	1	0	0	0	0	0	0
Attitudine fumo	1	3	1	1	3	3	1	2	2	1	3	1	1	3
Affezione bronchiale	0	0	0	1	0	0	0	0	1	1	0	0	0	0
Attitudine fumo	1	3	2	1	1	1	1	3	3	2	1	1	2	2
Affezione bronchiale	0	1	0	1	0	0	1	0	0	0	0	0	0	0
Attitudine fumo	3	1	1	1	1	2	1	1	3	1	1	1	1	3
Affezione bronchiale	0	0	0	0	0	0	0	0						
Attitudine fumo	2	1	1	1	2	3	1	3						

(a) Costruire un'opportuna tabella di contingenza.

(b) Identificare le distribuzioni marginali dell'`Affezione bronchiale` e dell'`Attitudine al fumo` e rappresentarle secondo un diagramma a barre. In entrambi i casi, identificare la moda e calcolare l'indice di Gini normalizzato e l'Entropia di Shannon normalizzata. Commentare.

(c) Calcolare la distribuzione condizionata dell'`Affezione bronchiale` data l'`Attidudine al fumo` e la distribuzione condizionata dell'`Attitudine al fumo` data l'`Affezione bronchiale`. Rappresentarle secondo opportuni diagrammi a barre.

(d) Valutare l'esistenza di un'associazione tra `Affezione bronchiale` e `Attitudine al fumo`.