

ME L'HO IMPARATO

Che cos'è R

- R è un ambiente integrato (ovvero un insieme di librerie, funzioni, oggetti) che permette di elaborare dati, eseguire calcoli ed realizzare rappresentazioni grafiche.
- E' distribuito gratuitamente ed disponibile per diverse architetture hardware e sistemi operativi: Windows, MacOS, Unix, Linux.
- Sul sito <http://www.r-project.org> è possibile scaricare, oltre che il programma base, anche una serie di moduli aggiuntivi (*packages*) e un'ampia manualistica.

All'indirizzo

http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=2

potrete trovare un articolo del NY Times su R

A che serve R?

Fornisce un ricco corredo di programmi per l'analisi e la modellazione statistica dei dati.

Consente di predisporre proprie routine non disponibili sui pacchetti standard per i calcoli più specifici.

E' anche un linguaggio orientato agli oggetti in continua espansione.

Contiene un insieme crescente di pacchetti dedicati a problemi ricorrenti nelle applicazioni.

Gli approcci più moderni alla trattazione dei dati empirici si configurano come nuovi pacchetti di R per arrivare più rapidamente alla comunità dei potenziali utenti e ricercatori.

Perché R?



1) E' gratuito. Questo è un grande vantaggio rispetto ai costi di acquisto (o, peggio, di noleggio) dei programmi concorrenti.



2) Opera allo stesso modo su piattaforme diverse: Mac OS, Windows, Linux liberando gli utenti dalla preoccupazione di come esportare dati e programmi.



3) Si adatta all'esigenza dell'utente: può essere usato in modo molto naive (ad esempio come calcolatrice in linea) come nel modo più sofisticato (ad esempio nel calcolo parallelo)



4) Offre possibilità grafiche straordinarie con le quali è possibile, se si vuole, interagire.

The official R website

<http://www.r-project.org>

Qui si possono trovare

- I pacchetti di base delle tecniche già consolidate
- I pacchetti nuovi o più specializzati
- Manuali in diverse lingue
- Risposte ai quesiti già emersi

Manualistica in italiano

<http://www.dst.unive.it/~claudio/R/index.html#manuale>

<http://cran.r-project.org/doc/contrib/Mineo-dispensaR.pdf>

<http://cran.r-project.org/doc/contrib/nozioniR.pdf>

<http://www.isib.cnr.it/~brazzale/ModStat/>

<http://digilander.libero.it/~robicox/manuali/pdf/mainr.pdf>

http://www.mat.uniroma3.it/didatticacds/corsi/didattica_interattiva/aa_01_02/st1/st1.html

http://venus.unive.it/statcomp/r/man_Parpinel.pdf

<http://www.dip-statistica.uniba.it/html/docenti/pollice/materiale.htm>

<http://www.stat.unipg.it/~luca/R-note.pdf>

<http://www.economia.unimi.it/facus/corsoR/>

R Links

[The R Project for Statistical Computing](#)

[Web site for Fox's An R and S-PLUS Companion to Applied Regression](#)

(code, data, etc. used in the book)

[Summary of R Commands by Category](#)

[Bret Larget's R Help](#)

[A Brief Guide to R for Beginners in Econometrics by Mahmood Arai](#)

[Econometrics in R by Grant V. Farnsworth](#)

[Using R for Data Analysis and Graphics: An Introduction by J.H.](#)

[Maindonald](#)

[Kickstarting R by Jim Lemon](#)

[simpleR: Using R for Introductory Statistics by John Verzani](#)

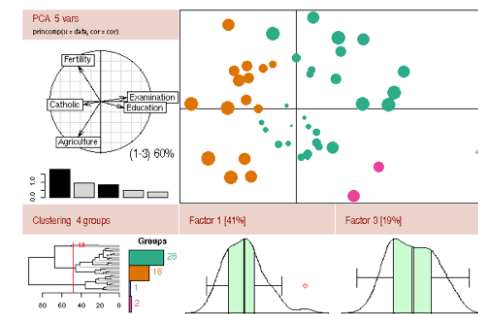
[Some translations of Stata commands into R commands](#)

[tseries: R package for time series analysis and computational finance](#)

[quadprog: R package containing routines and documentation for solving quadratic programming problems \(required by tseries\)](#)

Installazione

The R Project for Statistical Computing



Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).
- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

Scelta dello URL da cui scaricare

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found [here](#).

Argentina	http://cran.patan.com.ar/	Patan.com.ar, Buenos Aires
Australia	http://cran.ms.unimelb.edu.au/	University of Melbourne
Austria	http://cran.at.r-project.org/	Wirtschaftsuniversitaet Wien
Belarus	http://cran.promotionalpro.com/	www.ghost.by
Belgium	http://www.freeststatistics.org/cran/	K.U.Leuven Association
Brazil	http://cran.br.r-project.org/ http://cran.fiocruz.br/ http://www.vps.fmvz.usp.br/CRAN/ http://brieger.esalq.usp.br/CRAN/	Universidade Federal do Parana Oswaldo Cruz Foundation, Rio de Janeiro University of Sao Paulo, Sao Paulo University of Sao Paulo, Piracicaba
Canada	http://cran.stat.sfu.ca/ http://probability.ca/cran/ http://cran.skazkaforyou.com/	Simon Fraser University, Burnaby University of Toronto iWeb, Montreal
Chile	http://dirichlet.mat.puc.cl/	Pontificia Universidad Catolica de Chile, Santiago

URL in Italia da cui avere R

Italy

<http://rm.mirror.garr.it/mirrors/CRAN/>
<http://cran.stat.unipd.it/>
<http://dssm.unipa.it/CRAN/>

Garr Mirror, Milano
University of Padua
Universita degli Studi di Palermo

Si tratta di siti abbastanza sicuri con i quali si può operare con relativa tranquillità

Pagina principale di R



The Comprehensive R Archive Network

Frequently used pages

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Linux](#)
- [MacOS X](#)
- [Windows](#)

Source Code for all Platforms

Windows and Mac users most likely want the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- **The latest release** (2008-10-20): [R-2.8.0.tar.gz](#) (read [what's new](#) in the latest version).
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)

Software
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)
[Newsletter](#)

<http://cran.r-project.org/>

Link alla pagina per la versione Windows



R for Windows

This directory contains binaries for a base distribution and packages to run on i386/x64 Windows.

Note: CRAN does not have Windows systems and cannot check these binaries for viruses. Use the normal precautions with downloaded executables.

Subdirectories:

[base](#) Binaries for base distribution (managed by Duncan Murdoch)
[contrib](#) Binaries of contributed packages (managed by Uwe Ligges)

Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Last modified: April 4, 2004, by Friedrich Leisch

CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)

Software
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)
[Newsletter](#)

Link alla pagina per la versione Mac OS



R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.2 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) [here](#)

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Important note - R 2.8.0 binaries have been updated 2008/10/22

The binaries posted yesterday were (for a yet unknown reason) incomplete. The correct binaries have been posted today. Please check the MD5 sum of the downloaded file (see below)! If it differs from the posted checksum, please download it again - if in doubt from another mirror. If you see errors compiling packages from sources or invalid repository URLs, please make sure you update your binary.

Universal R 2.8.0 for Mac OS X released on 2008/10/22

This binary distribution of R and the GUI supports both PowerPC and Intel based Macs. The corresponding binaries of R packages are available for both architectures as well. Starting with R 2.3.1, CRAN binaries support Mac OS X 10.4 (Tiger) and higher only. It is, however, possible to compile binaries for earlier OS X versions from sources.

R 2.8.0 uses updated optional libraries - Tcl/Tk 8.5.5 and cairo 1.8.0. In addition, better font support is provided for cairo-based devices such as the built-in X11 cairo and the Cairo package. All dependent libraries are fork()-safe and thus can be used in projects such as Rserve or RApache. 64-bit build for Mac OS X 10.5 (Leopard) will be available shortly.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
md5 R-2.8.0.dmg
in the Terminal application to print the MD5 checksum for the R-2.8.0.dmg image.

CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)

Software
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)
[Newsletter](#)

Pagina iniziale di R (MAC)

R version 2.8.0 (2008-10-20)

Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

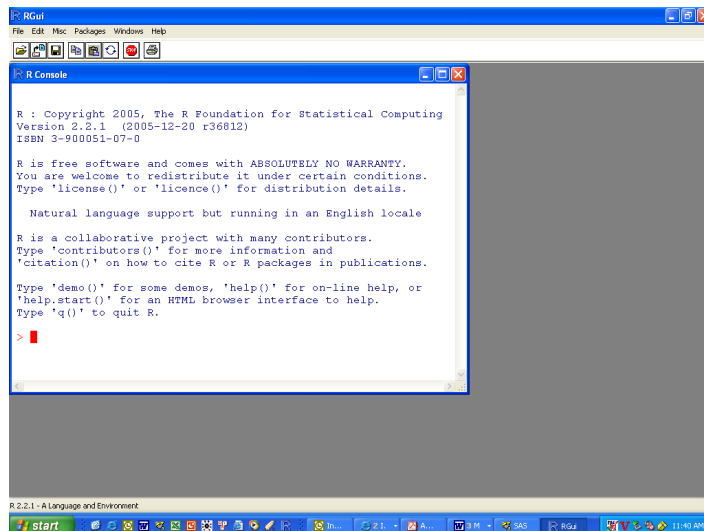
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace restored from /Users/agostinotarsitano/.RData]

Pagina iniziale (WIN)



La console di R

- Si interagisce con R attraverso la R-console digitando dopo il prompt ">" i comandi che si vogliono eseguire.
 - Ad esempio digitando
> demo(graphics)
si ottiene una dimostrazione delle potenzialità grafiche di R.

R distingue tra minuscole e maiuscole

t.test() and T.test() sono comandi diversi

R dispone di un help in linea

Attenzione alle parentesi (aperte e chiuse in numero pari)

Attenzione agli apici

Riga di comando

A partire dal prompt `>` della linea di comando le operazioni sintatticamente legittime di R consistono in

```
x<-c(2,2,3,3,4,5,7,9,9)
```

```
> sqrt(var(x))  
[1] 2.803767
```

Espressioni

```
> summary(x)
```

Assegnazioni.

```
Min. 1st Qu. Median Mean 3rd Qu. Max.  
2.000 3.000 4.000 4.889 7.000 9.000
```

Ogni espressione viene valutata e stampata in output a schermo come nei seguenti esempi

```
> median(x)  
[1] 4
```

```
> range(x)  
[1] 2 9
```

```
> quantile(x,0.25)  
25%  
3
```

Esempi

```
> -27*12/21  
[1] -15.42857
```

```
> sqrt(10)  
[1] 3.162278
```

```
> log(10)  
[1] 2.302585
```

```
> log10(2+3*pi)  
[1] 1.057848
```

```
> exp(2.7689)  
[1] 15.94109
```

```
> (25 - 5)^3  
[1] 8000
```

```
> cos(pi)  
[1] -1
```

```
Permutation: 3!
```

```
> prod(3:1)  
[1] 6
```

```
# 10.9.8.7.6.5.4  
> prod(10:4)  
[1] 604800
```

```
> prod(10:4)/prod(40:36)  
[1] 0.007659481
```

```
> choose(5, 2)  
[1] 10
```

```
> 1/choose(5, 2)  
[1] 0.1
```

Ortografia di R

- Differenza tra minuscole e maiuscole

```
a <- 5  
A <- 7  
B <- a+A
```

- Niente spazi bianchi nei nomi degli oggetti, ma soprattutto tra “<” e “-”

```
var<- 5 (questo si tollera) var< - 5 (questo è un errore)
```

- Si può inserire il “.” ed anche l’underscore “_”

```
var.a <- 5  
var.b <- 10  
var.c <- var.a + var.b  
Var_X<-var(x);mean_x.y<-mean(x*y)
```

- Una singola linea può contenere più comandi separati dal ;
`Nr<-2;Mg<-(-2)`

il simbolo “#” indica l’inizio di una linea di commento.

R ignorerà quanto segue dopo il cancelletto e fino alla successiva riga di comando

Sequenza delle operazioni

Le parentesi stabiliscono la sequenza delle operazioni. Il comando

```
> C<-A+2*sqrt(A)/A+5*sqrt(A)
```

produce un risultato diverso se scritto come

```
> C<-A + 2*(sqrt(A)/A) + 5*sqrt(A)
```

Senza ulteriori specificazioni, la sequenza è

- (1) Esponenziazione
- (2) moltiplicazione e divisione
- (3) addition and subtraction

```
> b <- 12-4/2^3 produce 12 - 4/8 = 12 - 0.5 = 11.5
```

```
> b <- (12-4)/2^3 produce 8/8 = 1
```

```
> b <- -1^2 produce -(1^2) = -1
```

```
> b <- (-1)^2 produce 1
```

Tipologia e rappresentazione

Per default R usa una aritmetica in doppia precisione

```
> x<-c(-3.4, 2.8, 0, 5, -17)
> mode(x)
[1] "numeric"
> storage.mode(x)
[1] "double"
```

```
> is.na(c(Inf, NaN, NA, 1))
[1] FALSE TRUE TRUE FALSE
> is.nan(c(Inf, NaN, NA, 1))
[1] FALSE TRUE FALSE FALSE
> is.finite(c(Inf, NaN, NA, 1))
[1] FALSE FALSE FALSE TRUE
```

Possiamo modificare la rappresentazione interna con la coercizione

```
> x<-as.integer(x)
> x
[1] -3 2 0 5 -17
> mode(x)
[1] "numeric"
> storage.mode(x)
[1] "integer"
```

Se ignoriamo la natura di un dato, possiamo chiedere ad R di indicarlo

```
> a<-"Cielo";is.numeric(a)
[1] FALSE
> b<-21;is.logical(b)
[1] FALSE
> c<-TRUE; is.character(c)
[1] FALSE
```

Coercizione

Esistono diverse funzioni che convertono i dati da una tipologia ad un'altra

```
> as.numeric(c("1", "2", "2a", "b"))
[1] 1 2 NA NA
> as.numeric(c(TRUE, FALSE, NA))
[1] 1 0 NA
> as.character(c(TRUE, FALSE, NA))
[1] "TRUE" "FALSE" NA
```

Alcune sorprese

```
> 1 == TRUE
[1] TRUE
> 1 == "1"
[1] TRUE
> "1" == TRUE
[1] FALSE
```

Alcune procedure si aspettano oggetti aventi natura specifica. Se non si è sicuri, è meglio verificare con il comando `typeof()`

Arrotondamenti

Se per avere gli interi si vuole evitare la coercizione si possono usare altri comandi

```
> x<-c(-3.46, 2.84, 0.50, 5, -17)
> x
[1] -3.46 2.84 0.50 5.00 -17.00
> y<-round(x,digits=2)
> y
[1] -3.46 2.84 0.50 5.00 -17.00
> z<-round(x,digits=1)
> z
[1] -3.5 2.8 0.5 5.0 -17.0
> w<-round(x,digits=0)
> w
[1] -3 3 0 5 -17
> storage.mode(w)
[1] "double"
```

```
> Debito<-1000000000 # Un miliardo
> Tasso<-2.175
> Interesse1<-Debito*2.17/100
> Interesse2<-Debito*2.18/100
> Interesse3<-Debito*2.175/100
> Interessi<-c(Interesse1,Interesse2,Interesse3)
> Interessi
[1] 21 700 000 21 800 000 21 750 000
```

Le differenze sono ben visibili.

round usa la regola del “5”: 0-4=intero inferiore, 5-9=intero superiore

Altri arrotondamenti

ceiling

Restituisce l'intero più piccolo maggiore dell'argomento.

floor

Restituisce l'intero più grande minore dell'argomento.

trunc

Escude la parte decimale di un numero

signif

Arrotonda al numero di cifre significative indicate

Per fissare il numero di cifre da visualizzare si usa il

`option(digits=n)`

in questo modo tutte le cifre successivamente ottenute saranno visualizzate con n cifre decimali.

Altri arrotondamenti/2

```
> x2 <- pi * 100^(-1:3) # questo è un vettore
> x2
[1] 3.141593e-02 3.141593e+00 3.141593e+02 3.141593e+04 3.141593e+06

> ceiling(x2)
[1] 1 4 315 31416 3141593 # intero superiore più piccolo

> floor(x2)
[1] 0 3 314 31415 3141592 # intero inferiore più grande

> trunc(x2, 4)
[1] 0 3 314 31415 3141592 # buttare cifre decimali

> signif(x2, 4)
[1] 3.142e-02 3.142e+00 3.142e+02 3.142e+04 3.142e+06 # esprimere il
numero sempre con le stesse cifre

> round(x2, 4)
[1] 0.0314 3.1416 314.1593 31415.9265 3141592.6536
```

Directory di lavoro

In R ogni operazione su file avviene a partire dalla directory corrente di lavoro, la quale è visibile con il comando:

```
> getwd()
```

Se si vuol leggere un file dati, ad esempio dati.txt con il comando `read.table("dati.txt")`, l'operazione sarà eseguita correttamente solo se il file si trova nella directory corrente.

L'uso delle directory in R è di fondamentale importanza al fine di poter salvare e utilizzare successivamente il lavoro svolto.

Ogni sessione di R crea, nella directory di lavoro, due file `RData` ed `Rhistory` che memorizzano i dati ed i comandi che sono stati utilizzati nella sessione.

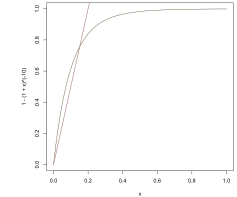
```
> setwd(mydir)
```

sposta la directory di lavoro corrente in `mydir`

Applicazione elementare

Valore attuale del prestito di P euro a tasso di interesse mensile i , restituito in n rate posticipate di importo costante R

$$P = R \left[\frac{1 - (1+i)^{-n}}{i} \right] \Rightarrow R = P \left[\frac{i}{1 - (1+i)^{-n}} \right]$$



```
intRate <- 0.01
n <- 10
principal <- 1500
payment <- principal * intRate / (1 - (1 + intRate)^(-n))
payment # Rata mensile da pagare
#
# Se la rata fosse di importo pari a 300 euro, quale
# sarebbe il tasso risultante?
payment <- 300
curve(1 - (1+x)^(-10), 0, 1, 201, col="gold4")
curve(x*(principal/payment), 0, 1, 201, col="tomato", add=T)
```

Salvataggio della History

L'history di R è dato da tutti i comandi digitati in una sessione.

La history è memorizzata nel file `Rhistory` automaticamente generato o aggiornato uscendo da R nella directory di lavoro.

Possiamo anche salvare la history in uno dei seguenti modi:



savehistory()

salva la history nel file `Rhistory` nella directory corrente.



savehistory (file=myfile)

salva la history nel file `myfile` della directory corrente

Per caricare la history precedentemente salvata si può usare il comando: `loadhistory (file=myfile)` carica la history memorizzata in `myfile`

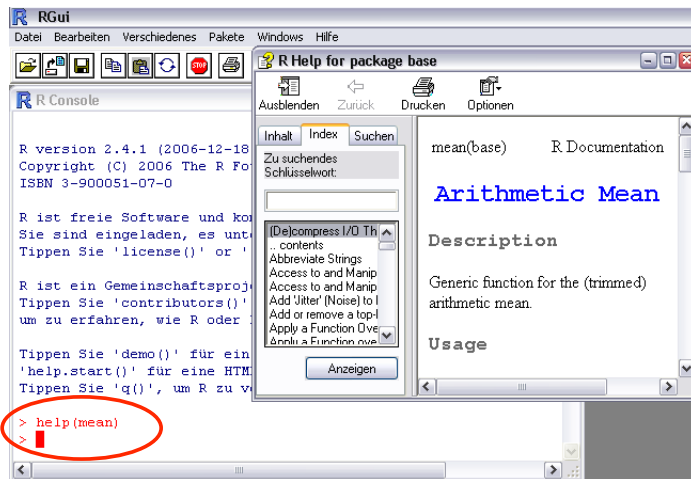
history() Visualizza gli ultimi 25 comandi

history(n) Visualizza gli ultimi n comandi

Aiuti in R



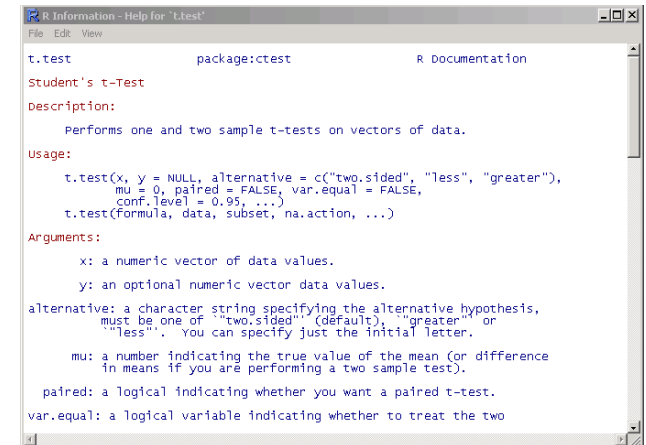
- Help function



Si possono avere dettagli su di un comando specifico di cui si conosce il nome

>? t.test
or
>help(t.test)

In mac basta inserire il nome nello spazio di ricerca indicato con la lente di ingrandimento



Contenuto e manipolazione del workspace

```
ls()           # mostra il contenuto del workspace
rm(i)          # elimina uno o piu' oggetti, in questo caso l'oggetto "i"
ls(pattern="V") # mostra gli oggetti che contengono "V" nel nome
ls(pat="V")    # i nomi dei parametri dei comandi possono essere abbreviati
save(list=ls(),file="prova.rda")
# salviamo il workspace in un file chiamato prova.rda
# rda e' l'estensione standard per indicare un file dati di R
# ma si puo' usare qualsiasi altra estensione oppure ometterla
rm(list=ls(pat="V")) # rimuoviamo tutte le variabili che hanno "V" nel nome
rm(list=ls())       # cancelliamo tutto il workspace
load("prova.rda")   # ricarichiamo i dati salvati in precedenza
save(V1,V2, file="prova2.rda")
# si possono salvare selettivamente solo alcuni oggetti
save.image()        # o salvare tutto il workspace in un file di default di R
# chiamato .RData
load(".RData")      # ricarichiamo tutto
```

Tipologia delle variabili

```
> a <- 49.0
> sqrt(a)
[1] 7
```

Numeric (reali in doppia precisione)

```
> m <- 1:5
> m
[1] 1 2 3 4 5
```

Integer (Interi)

```
> b <- "Una buona attività formativa"
> sub("buona", "discreta", b)
```

[1] "Una discreta attività formativa" Sub è un comando sulle stringhe

Carattere o stringa (tra apici)

```
> c <- (1+1==3) # espressione logica
> c
[1] FALSE
```

Logiche: TRUE/FALSE

```
> as.character(b)
[1] "FALSE"
```

```
x <- 9
y <- x > 10
y
[1] FALSE
```

Tipologia delle variabili/2

La funzione `typeof` consente di accertare la tipologia della variabile

```
typeof(x)           as.integer(2^31 - 1)
[1] "double"        [1] 2147483647
is.double(8.9)      as.integer(2^31)
[1] TRUE            [1] NA
test <- 1223.456     Warning message:
is.double(test)      NAs introduced by coercion
[1] TRUE

nchild <- as.integer(3)
is.integer(nchild)   [1] TRUE
3.0 non è un intero e 3 non lo è
automaticamente

x <- as.integer(7)   x <- c("a","b","c")
y <- 2.0             x
z <- x/y             [1] "a" "b" "c"
                    mychar1 <- "This is a test"
                    mychar2 <- "This is another test"
                    charvector <- c("a", "b", "c", "test")

x <- c(9,166)
y <- (3 < x) & (x <= 10)
[1] TRUE FALSE

x <- 1:15
## somma del numero di elementi
in x che risultano maggiori di 9
sum(x>9)
[1] 6
```

Esempi sugli operatori logici

<code>x == 5</code>	<code>x uguale a 5</code>
<code>x != 5</code>	<code>x diverso da 5</code>
<code>y < x</code>	<code>y minore di x</code>
<code>x > y</code>	<code>x maggiore di y</code>
<code>z <= 7</code>	<code>z minore o uguale 7</code>
<code>p >= 1</code>	<code>p maggiore o uguale ad 1</code>
<code>is.na(x)</code>	<code>x, è un valore mancante?</code>
<code>A & B</code>	<code>A e B</code>
<code>A B</code>	<code>A oppure B</code>
<code>!</code>	<code>non</code>

Operatori di R

Operatori relazioni	Significato
<code>x < y</code>	minore di
<code>x > y</code>	maggiore di
<code>x <= y</code>	minore o uguale di
<code>x >= y</code>	maggiore o uguale di
<code>x == y</code>	uguale a
<code>x != y</code>	non uguale a
Operatori logici	Significato
<code>! x</code>	negazione logica (NOT)
<code>x & y</code>	intersezione logica (AND)
<code>x y</code>	unione logica (OR)
<code>xor(x, y)</code>	OR esclusivo (XOR)

Operatore & (and)

```
# OPERATORE and
#confronti unari
x<-1:10
a<-x<7 & x>2 # assegna ad a un confronto logico
print(a)
```

```
#confronti binari
a<-x<7 && x>2 # notare la differenza
print(a)
```

```
y<-rep(5,10)
#confronti unari
a<-y<7 & y>2
print(a)
```

```
#confronti binari
a<-y<7 && y>2
print(a)
```

Operatore | (or)

```
# OPERATORE or
```

```
x<-1:10  
a<-x<7 | x>2  
print(a)
```

```
# confronti binari
```

```
a<-x<7 || x>2  
print(a)  
y<-rep(5,10)
```

```
#confroni unari
```

```
a<-y<7 | y>2  
print(a)
```

```
#confronti binari
```

```
a<-y<7 || y>2  
print(a)
```

Inserimento dati

Scegliere il nome dell'oggetto in cui registrare i dati

```
name <- c( , , , , )
```

ESEMPIO: Consumo di birra procapite

Paesi:

```
country<-c("Australia", "Canada", "Denmark", "Finland", "England", "Island", "Netherlands",  
"Norway", "Sweden", "Switzerland", "USA")
```

Valori

```
beer<-c(480,500,380,1100,1100,230,490,250,300,510,1300)
```

Etichettatura dei valori: si adopera il comando **names**.

```
names(beer)<-country  
>beer
```

Australia	Canada	Denmark	Finland	England	Island	Netherlands
480	500	380	1100	1100	230	490
Norway	Sweden	Switzerland	USA			
250	300	510	1300			

Data e orario

In questo caso è indicata la tipologia **as.date**

```
temp <- c("12-09-1973", "29-08-1974")
```

```
z <- as.Date(temp, "%d-%m-%Y")
```

```
z
```

```
[1] "1973-09-12" "1974-08-29"
```

```
data.class(z)
```

```
[1] "Date"
```

```
format(z, "%d-%m-%Y")
```

```
[1] "12-09-1973" "29-08-1974"
```

You can add a number to a date object, the number is interpreted as the number of day to add to the date.

```
z + 19
```

```
[1] "1973-10-01" "1974-09-17"
```

You can subtract one date from another, the result is an object of class 'difftime'

```
dz = z[2] - z[1]
```

```
dz
```

```
data.class(dz)
```

```
Time difference of 351 days
```

```
[1] "difftime"
```

Vettori

L'elemento base di tutte le procedure di R è il vettore. Le varietà più frequenti sono **NUMERICI**, **STRINGA** E **LOGICI** (Vero/Falso)

```
> c(1, 2, 3, 4, 5)
```

```
[1] 1 2 3 4 5
```

```
> c("Huey", "Dewey", "Louie")
```

```
[1] "Huey" "Dewey" "Louie"
```

```
> c(T, T, F, T)
```

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

```
> c(1, 2, 3, 4, 5) > 3
```

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

T and F are convenient abbreviations for TRUE and FALSE respectively.

The length of any vector can be determined by the **length** function:

```
> gt.3 <- c(1, 2, 3, 4, 5) > 3
```

```
> gt.3
```

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

```
> length(gt.3)
```

```
[1] 5
```

Costruzione di vettori

```
seq(4,9); seq(4,10,0.5); seq(length=10)
seq(1, 9, by = 2) # estremo incluso
seq(1, 9, by = pi) # prima di raggiungere l'estremo
seq(9,1) # indici decrescenti
```

```
rep(1:4, 2)
rep(1:4, each = 2) # non è la stessa cosa.
rep(1:4, c(2,2,2,2)) # idem come sopra.
rep(1:5,length=12)
rep(c("one","two"),c(6,3))
```

Costruzione di vettori con cut

La funzione cut serve per assegnare i valori di un vettore a classi di ampiezza (dcretizzazione del vettore).

```
set.seed(3141593)
# Fissa l'avvio dei numeri pseudo-casuali
y<-sample(18:30,15,replace=TRUE)
# genera uno pseudo-campione di ampiezza 15 di valori tra 18 e 30
x<-cut(y,4); x
# suddivide y in 4 gruppi
x<-
cut(y,breaks=c(18,21,24,27,30),labels=c("Primo","Secon","Terzo","Quart"),right=FALSE);
x #suddivide i dati nei gruppi 18-21, 21-24, 24-27, 27-30. Le classi sono chiuse a sinistra
e destra
#Se right=TRUE le classi sono aperte a sinistra
```

```
> x<-cut(y,4); x # suddivide y in 4 gruppi
[1] (20.5,23] (25.5,28] (18,20.5] (20.5,23] (20.5,23] (23,25.5] (18,20.5]
[9] (18,20.5] (25.5,28] (20.5,23] (18,20.5] (18,20.5] (25.5,28] (25.5,28]
Levels: (18,20.5] (20.5,23] (23,25.5] (25.5,28]
> x<-cut(y,breaks=c(18,21,24,27,30),labels=c("Primo","Secon","Terzo","Quart"),right=FALSE);x
[1] Secon Terzo Primo Secon Secon Terzo Primo Primo Terzo Secon Primo Primo
[14] Quart Quart
Levels: Primo Secon Terzo Quart
```

Costruzione di vettori/2

Repliche ponderate

```
x<-c(10,40,50,100) # Vettore delle istanze
w<-c(2,1,3,2) # ripetizione previste per ciascuna
rep(x,w)
> x<-c(10,40,50,100) # Vettore delle istanze
> w<-c(2,1,3,2) # ripetizione previste per ciascuna
> rep(x,w)
[1] 10 10 40 50 50 50 100 100
```

Conversioni

```
> v <- c(1, 2, 3)
> is.vector(v)
[1] TRUE
> # Attenzione: se i contenuti non sono dello stesso tipo R li converte tutti i carteri
> gg <- c("1",3,4)
> is.vector(gg)
[1] TRUE
> gg
[1] "1" "3" "4"
> is.character(gg)
[1] TRUE
```

Vettori di caratteri

Sono molto utili i vettori stringa

```
> Paesi<-c("Romania","Bulgaria","Ungheria","Slovenia","Cekia")
> Paesi
[1] "Romania" "Bulgaria" "Ungheria" "Slovenia" "Cekia"
> mode(Paesi)
[1] "character"
> storage.mode(Paesi)
[1] "character"
```

Numeri come lettere

```
> x<-c("1","2","Uno","Due","One","Two")
> x
[1] "1" "2" "Uno" "Due" "One" "Two"
> is.character(x)
[1] TRUE
> x[1]==x[3]
[1] FALSE
> x[3]==x[5]
[1] FALSE
```

> x[1]+x[2]
Error in x[1] + x[2] : non-numeric argument to binary operator

Vettori di caratteri/2

La costruzione di etichette è più semplice grazie ad alcuni comandi sulle stringhe

```
> x<-rep("Indicatore",4);y<-seq(1,8,by=2)
> x
[1] "Indicatore" "Indicatore" "Indicatore" "Indicatore"
> y
[1] 1 3 5 7
> z<-paste(x,y,sep="")
> z
[1] "Indicatore1" "Indicatore3" "Indicatore5" "Indicatore7"
```

Stringhe predefinite

```
> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x"
[25] "y" "z"
> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X"
[25] "Y" "Z"
> month.name
[1] "January" "February" "March" "April" "May" "June" "July" "August"
[9] "September" "October" "November" "December"
> onth.abb
Error: object "onth.abb" not found
> month.abb
[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

Richiamo di elementi

```
> intake.pre
[1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
> intake.pre[5]
[1] 6390
> intake.pre[c(3, 5, 7)]
[1] 5640 6390 6805
> ind <- 1:c(3, 5, 7)
> intake.pre[ind]
[1] 5640 6390 6805
> intake.pre[8:13]
[1] 7515 7515 8230 8770 NA NA
> intake.pre[c(1, 2, 1, 2)]
[1] 5260 5470 5260 5470
```

Se si usa un indice maggiore della lunghezza saranno inseriti dei codici NA

Gli indici possono essere ripetuti richiamando più volte lo stesso elemento

Accesso agli elementi del vettore

Si deve indicare la posizione cercata tra parentesi quadre

Gli indici in indirizzo possono essere anche vettori.

Un indice negativo implica l'esclusione di un elemento

```
> x
[1] 1 3 5 7 9 11 13 15 17 19
> x[7]
[1] 13
> x[1:4]
[1] 1 3 5 7
> x[seq(1,10,by=3)]
[1] 1 7 13 19
> x[-c(2,6)]
[1] 1 5 7 9 13 15 17 19

> x[x<=7 | x>=13]
[1] 1 3 5 7 13 15 17 19
> x[x>=5 & x<=12]
[1] 5 7 9 11
```

Indici negativi (esclusioni)

Se l'indice è preceduto da un segno negativo ciò implica l'esclusione del corrispondente elemento

```
> intake.pre
[1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
> intake.pre[-5]
[1] 5260 5470 5640 6180 6515 6805 7515 7515 8230 8770
> ind <- -c(3, 5, 7)
> ind
[1] -3 -5 -7
> intake.pre[ind]
[1] 5260 5470 6180 6515 7515 7515 8230 8770
```

E' bene non mescolare indici negativi ed indivi positivi

Valori mancanti

Tutte le variabili (numeric, character, logical) possono contenere l'indicazione **NA**: not available.

- NA is not the same as 0
- NA is not the same as "" (stringa vuota)
- NA is not the same as FALSE
- NA is not the same as NULL

Operationi con NA possono achen non produrre un NA:

<pre>> NA==1 [1] NA > 1+NA [1] NA > max(c(NA, 4, 7)) [1] NA > max(c(NA, 4, 7), na.rm=T) [1] 7</pre>	Max richiama il massimo in un oggetto conentente dei valori	<pre>> NA TRUE [1] TRUE > NA & TRUE [1] NA INF</pre>
---	--	---

Esempio

c() crea un insieme-vettore di elementi

```
> genus <- c("Daphnia", "Boletus", "Hippopotamus", "Salmo", "Linaria",
+ "Ixodes", "Apis")

> species <- c("magna", "edulis", "amphibius", "trutta", "alpina",
+ "ricinus", "mellifera")

> weight <- c(0.001, 100, 3200000, 1000, 2.56, 0.001, 0.01)

> legs <- as.integer(c(0, 0, 4, 0, 0, 8, 6)) # vincola i valori ad interi

> animal <- c(TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE)
```

Non dimenticate mai le virgole di separazione all'interno di c

```
> p <- c(.2, .3, .5)
```

Il simbolo "+" indica
che la riga comando
continua su un'altra
riga comando

Infiniti e Non definiti

Gli infiniti sono rappresentati da **Inf** e **-Inf** e lo status può essere controllato con i comandi **is.infinite** ovvero **is.finite**.

```
x <- c(1,3,4)
y <- c(1,0,4)
x/y
[1] 1 Inf 1
z <- log(c(4,0,8))
is.infinite(z)
[1] FALSE TRUE FALSE
```

La notazione **NaN** (Not a Number) indica che il risultato della operazione richiesta non ha un valore conclusivo

```
pi / 0 ## = Inf a non-zero number divided by zero creates
infinity
0 / 0 ## = NaN

1/0 + 1/0 # Inf
1/0 - 1/0 # NaN
```

Fattori (variabili nominali)

Gestione dei dati categoriali con R.

Alcune volte sono espressi con dei numeri, ma occorre avvertire il software che sono solo dei codici

```
> pain <- c(0, 3, 2, 2, 1)
> fpain <- factor(pain, levels = 0:3)
> fpain
[1] 0 3 2 2 1
Levels: 0 1 2 3
> levels(fpain) <- c("none", "mild", "medium", "severe")
> fpain
[1] none severe medium medium mild
Levels: none mild medium severe
> as.numeric(fpain)
[1] 1 4 3 3 2
```

La tipologia di dati "factor" è
utile per gestire i dati
categoriali cioè dati che variano
in un insieme di codici
alfanumerici

```
> text.pain <- c("none", "severe", "medium", "medium",
+ "mild")
> factor(text.pain)
[1] none severe medium medium mild
Levels: medium mild none severe
```

Fattori/2

L'ordinamento dei livelli è alfabetico (per default). In alcuni casi è preferibile specificarlo direttamente

```
X <- c("a","b","c","c","a","a","b","a","a","c","c","b")
```

```
Y <- c(1,3,5,5,4,3,2,1,5,3,1,2,3,4,5,4,1,5)
```

- `>x<-factor(x)`
si crea un factor da caratteri che saranno ordinati secondo ordine alfabetico
- `>y<-factor(y)`
si crea un factor da numeri che saranno ordinati secondo l'ordine naturale
- `>x<-ordered(x),levels=c("c","b","a"))`
si crea un factor da caratteri che saranno ordinati secondo l'ordine dato dall'opzione `levels`
- `>y<-ordered(y),levels=c(5,4,3,2,1))`
si crea un factor da numeri che saranno ordinati secondo l'ordine dato dall'opzione `levels`

Fattori/4

You can transform factor variables to double or integer variables using the `as.double` or `as.integer` function.

```
sex.numeric <- as.double(sex)
sex.numeric
[1] 2 2 1 2 1
```

The 1 is assigned to the female level, only because alphabetically female comes first. If the order of the levels is of importance, you will need to use ordered factors.
Use the function `ordered` and specify the order with the `levels` argument. For example:

```
Income <- c("High","Low","Average","Low","Average","High","Low")
Income <- ordered(Income, levels=c("Low","Average","High"))
Income
[1] High Low Average Low Average High Low
Levels: Low < Average < High
```

The last line indicates the ordering of the levels within the factor variable. When you transform an ordered factor variable, the order is used to assign numbers to the levels.

```
Income.numeric <- as.double(Income)
Income.numeric
[1] 3 1 2 1 2 3 1
```

Fattori/3

```
sex <- c("male","male","female","male","female") # sex è di tipo carattere, va trasformato
sex <- factor(sex)
sex
[1] male male female male female # le modalità si controllano con levels
levels(sex)
[1] "female" "male"
# L'esito è di tipo character. Un'alternativa è la seguente
sex <- c(1,1,2,1,2)
# The object 'sex' is an integer variable, you need to transform it to a factor.
sex <- factor(sex)
sex
[1] 1 1 2 1 2
Levels: 1 2
The object 'sex' looks like, but is not an integer variable. The 1 represents level "1"
here. So arithmetic operations on the sex variable are not possible:
sex + 7
[1] NA NA NA NA NA
Warning message:
+ not meaningful for factors in: Ops.factor(sex, 7)
It is better to rename the levels, so level "1" becomes male and level "2" becomes female:
levels(sex) <- c("male","female")
sex
[1] male male female male female
```

Lo script

Le analisi diventano più gestibili ed efficienti se i comandi sono inseriti in un file testo da richiamare in blocco o a gruppi di righe.

Ogni sistema ha il suo modo di rendere disponibili gli script (win ha il notepad). Anche Word va bene.

```
Sesso = as.factor(c("F","F","M","M","M","M","F","M","F","M","F","M","M",
"F","M","M","M","M","F","M","M","F","M","F"))
Eta = c(25,26,19,20,23, 34, 45, 18, 27, 32, 43, 61, 29, 27, 38, 40, 51,
26, 54,
18,18,19,21,22)
PrimaAuto = factor(c(2,2,1,2,1,0,0,1,1,0,0,0,1,1,0,1,0,1,0,1,1,0,2,2),
labels
= c("No", "Si","UENP"))
table(Sesso); table(Eta); table(PrimaAuto) # tabelle marginali
tab <- table(Sesso) # Cattura dei valori
tab/sum(tab) # tabella frequenze relative
tab<- table(Sesso, PrimaAuto); tab # tabella a doppia entrata
margin.table(tab,1); margin.table(tab,2) # marginali di riga e colonna
```

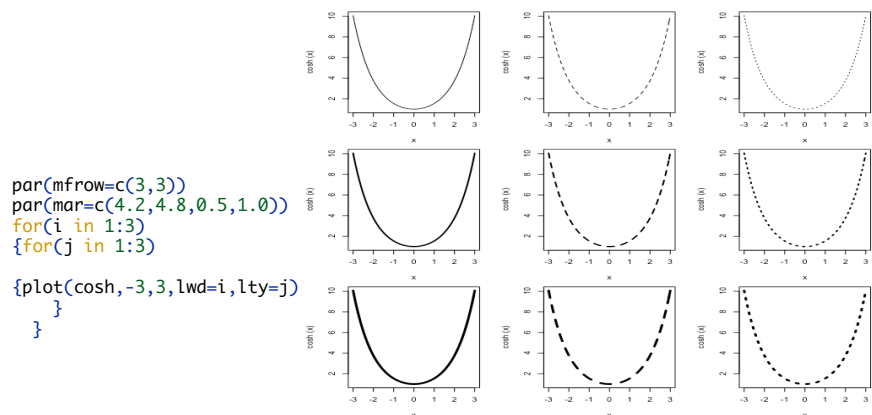
Esempio

```
par(mfrow=c(1,1),mar=c(4,4,2,2))
# Finale          - Voto finale
# Bonus           - Attività corsuali
# Genere          - M o F
# Liceo           - Umanistico o scientifico
# Iscrizione      - Scelta di immatricolazione
# Mese_Abb        - Mese di abbandono studi
Iscri<-read.table("IscriStat1.csv",sep=" ",dec=".",head=T);attach(Iscri)
summary(Iscri)
Voto<-factor(Finale,levels=c("Basso","Medio","Alto"))
Sesso<-factor(Genere,levels=c("F","M"),labels=c("Donna","Uomo"))
ScuolaSec<-factor(Liceo,levels=c("Sc","Um"),labels=c("Scientifico","Altri"))
Mese<-
factor(Mese_Abb,ordered=TRUE,levels=c(1:12),labels=c("Ott","Nov","Dic","Gen","Feb","
Mar","Apr","Mag","Giu","Lug","Ago","Set"))
Iscrizione<-factor(Iscrizione)
Table1<-table(Mese);print(Table1)
Table2<-table(Bonus); print(Table2)# Soluzione inefficace
classi<-c(40,44,48,52,56,60)
Table4<-table(cut(Bonus,br=classi,right=FALSE)); print(Table4)
hist(Bonus,main="Istogramma dei valori osservati",ylab="Frequenze
relative",xlab="Bonus del corso",breaks=classi,freq=FALSE,right=TRUE,col="purple4")
Table4<-table(Voto,cut(Bonus,br=classi,right=FALSE));print(Table4)
Table5<-prop.table(Table4);print(Table5,digits=3)
# Frequenze relative congiunte
# Verifica dell'indipendenza
summary(Table5)
```

Esempio

Il comando plot rappresenta le funzioni intrinseche, cioè in cui la x non è espressa nel comando.

Ecco alcuni esempi con diverse modalità per lo spessore della curva ed il tipo di linea

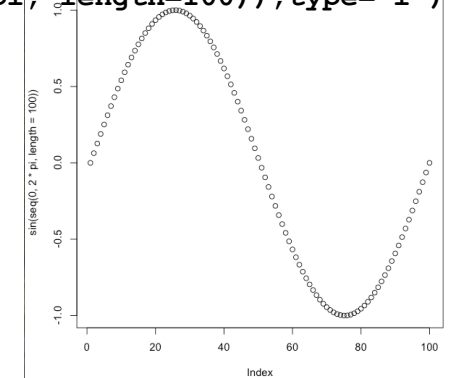
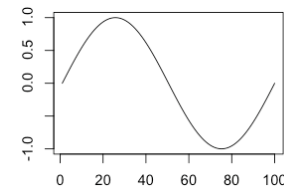


Plot

Il comando plot è uno dei più potenti di R. E' ricco di opzioni e vantaggi per chi vuole sfruttare i grafici statistici per comunicare dei risultati

```
> par(mfrow=c(1,2))
> plot(sin(seq(0, 2*pi, length=100)))
> plot(sin(seq(0, 2*pi, length=100)),type="l")
```

L'opzione linea è il raccordo tra i punti indicata con una l (elle) tra apici

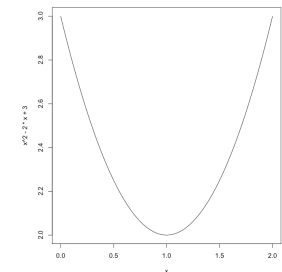


Rappresentazioni di curve analitiche

Una delle caratteristiche più interessanti di R è la semplicità ed efficacia con cui rappresenta le curve

$$f(x) = x^2 - 2x + 3$$

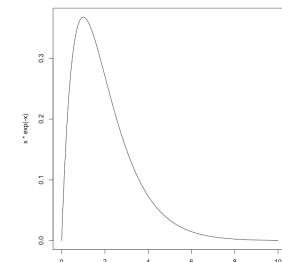
```
par(mfrow=c(1,1))
par(mar=c(4.2,4.8,0.5,1.0))
curve(x^2-2*x+3,0,2,101)
```



Nel comando si inserisce: l'espressione, il limite inferiore, il limite superiore e il numero dei punti di interpolazione

$$f(x) = xe^{-x}$$

```
par(mfrow=c(1,1))
par(mar=c(4.2,4.8,0.5,1.0))
curve(x*exp(-x),0,10,201)
```



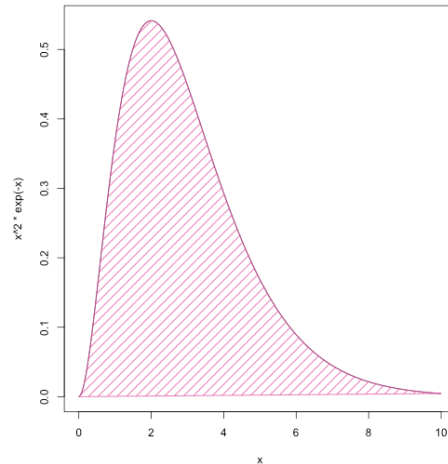
Coloratura dell'area sottesa

Spesso è utile colorare o campire con tessiture diverse l'area sottesa ad una curva

$$f(x) = x^2 e^{-x}$$

```
par(mfrow=c(1,1))
par(mar=c(4.2, 4.8, 0.5, 1.0))
curve(x^2*exp(-x), 0, 10, 201)
asc<-seq(0, 10, length=201)
ord<-asc^2*exp(-asc)
polygon(asc, ord,
10, 45, col="violetred")
```

Il primo numero indica la densità della campitura ed il secondo indica l'inclinazione delle linee



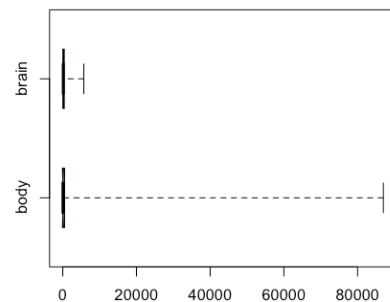
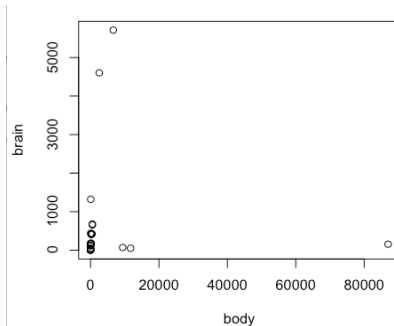
Riassunto dei dati

```
>library(MASS)
>Animals
> summary(Animals)
```

- La presenza di un valore anomalo
- outlier rende poco utile il
- sommario dei dati

```
> summary(Animals)
```

body		brain	
Min. :	0.023	Min. :	0.40
1st Qu.:	3.100	1st Qu.:	22.23
Median :	53.830	Median :	137.00
Mean :	4278.439	Mean :	574.52
3rd Qu.:	479.000	3rd Qu.:	420.00
Max. :	87000.000	Max. :	5712.00



Graphical Summaries

- stem & leaf plots

```
> data(hills)
> dist
```

2.5	6.0	6.0	7.5	8.0	8.0	16.0	6.0	5.0	6.0	28.0
5.0	9.5	6.0	4.5	10.0	14.0	3.0	4.5	5.5	3.5	3.5
6.0	2.0	3.0	4.0	6.0	5.0	6.5	5.0	10.0	6.0	18.0
4.5	20.0									

```
> stem(dist)
```

The decimal point is 1 digit(s) to the right of the |

0	2	3	3	3	4	4
0	5	5	5	5	5	6
1	0	0	0	4		
1	6	8				
2	0					
2	8					

Quick, easy, no data
are lost — actual
values are retained

Grafica esplorativa

```
stem(waiting)
```

The decimal point is 1 digit(s) to the right of the 1

```
library(MASS)
data(geyser);attach(geyser)
summary(geyser)
stem(duration)
stem(waiting)
```

```
> stem(duration)
```

The decimal point is 1 digit(s) to the left of the 1

[illegible]

Il diagramma ramo-foglia contiene tutte le informazioni dei dati originali, ma ha il merito di proporli in una chiave grafica

[illegible]

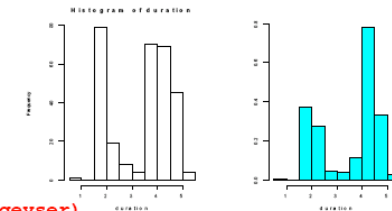
Istogrammi

- > Sintetizzano graficamente un collettivo statistico (data set)
- > Con essi si propone una stima della densità dei valori nella popolazione
- > Per realizzarlo occorre scegliere il numero e le ampiezze delle classi
- > Le classi dovrebbero stare tra 5 e 25. Alcune regole aiutano a decidere (e.g. la regola di Sturges)
- > L'aspetto dell'istogramma si modifica con la tipologia delle classi

```
# Istogramma di un campione di dati
par(mar=c(4.4,4.5,1.0,1.5));par(mfrow=c(1,2))
Bas<-read.table("basesal.csv",sep=";",header=T)
attach(Bas)
Salary<-sort(Salary);n<-length(Salary)
hist(Salary,breaks="FD",col="mistyrose3",freq=T,ylab="Freq.As
s.",main="")
k=8;kp1<-k+1;kp2<-k+2;X0<-Salary[1];Xn<-Salary[n]
R<-Xn-X0;d<-R/k
ci<-matrix(0,kp1);fi<-matrix(0,kp1)
Li<-matrix(0,kp1);Ui<-matrix(0,kp1)
for (i in 1:k) {Li[i+1]<-Li[1]+i*d;Ui[i]<-Li[i+1]}
for (i in 1:k) {ci[i]<-(Ui[i]+Li[i])/2}
for (j in 1:n)
{if (Salary[j]>=Li[i] & Salary[j]<Ui[i]) fi[i]<-fi[i]+1}
fi[i]<-fi[i]/n}
nc<-matrix(0,kp2);fc<-matrix(0,kp2)
for (i in 1:k){nc[i+1]<-ci[i];fc[i+1]<-fi[i]}
nc[1]<-X0;fc[1]<-0;nc[kp2]<-Xn;fc[kp2]<-0
A<-cbind(nc,fc);print(A)
plot(nc,fc,type="l",frame.plot=F,xlab="Salary",ylab=expressio
n(f[i]))
polygon(A, col="orange4", border = "black")
```

Esempio

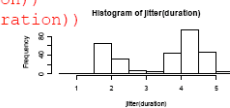
Qual è la scala degli assi?



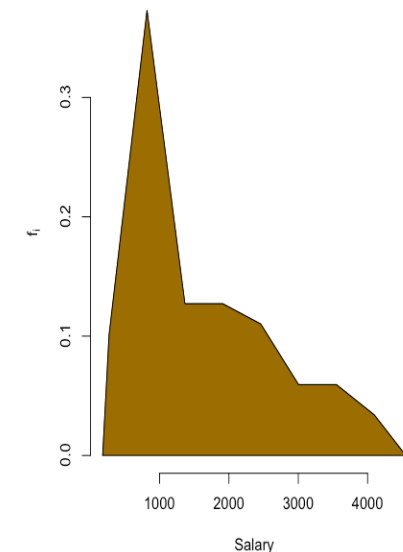
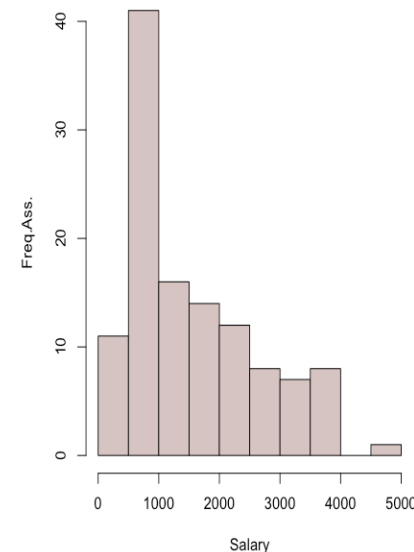
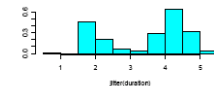
```
> attach(geyser)
> hist(duration)
> truehist(duration)
```

Two histograms of same data with different rules for values on class boundaries

```
> hist(jitter(duration))
> truehist(jitter(duration))
```

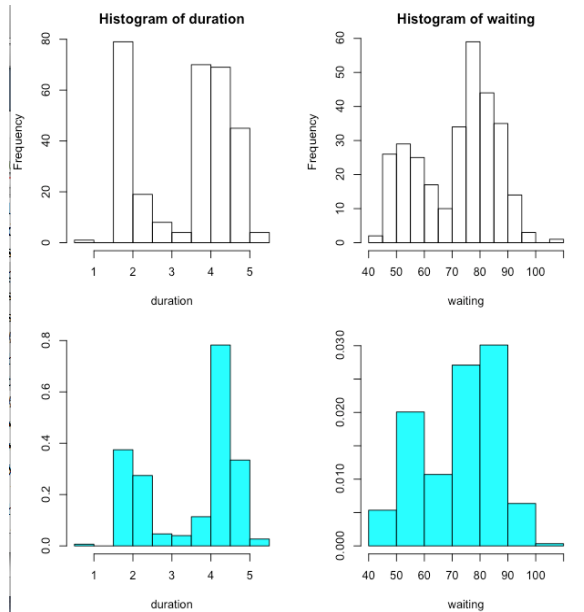


Add random amount to avoid values on class boundary



Grafica esplorativa/2

```
library(MASS)
data(geyser)
attach(geyser)
#####
par(mfrow=c(2,2),
    mar=c(4,4,2,2))
hist(duration)
hist(waiting)
truehist(duration)
truehist(waiting)
```



Esempio

```
29.6 28.2 19.6 13.7 13.0 7.8 3.4 2.0 1.9 1.0 0.7 0.4 0.4 0.3
0.3 0.3 0.3 0.3 0.2 0.2 0.2 0.1 0.1 0.1 0.1 0.1
```

```
> hist(x,breaks=10) # 10 breaks, or just hist(x,10)
> hist(x,breaks=c(0,1,2,3,4,5,10,20,max(x))) # specify break points
```

Altro esempio

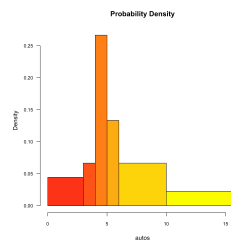
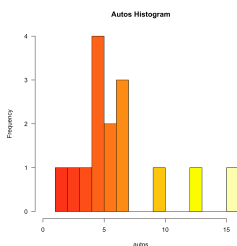
```
cars<-c(1,3,6,4,9)
trucks<-c(2,5,4,5,12)
suvs<-c(4,4,6,6,16)
autos.data<-
data.frame(cbind(cars,trucks,suvs))
```

autos.dat

cars	trucks	suvs
1	2	4
3	5	4
6	4	6
4	5	6
9	12	16

#Read car and truck values from tab-delimited autos.dat

```
autos_data <- read.table(file="autos.dat", header=T, sep=";")
```



Esercizio

```
# Read values from tab-delimited autos.dat
autos_data<- read.table("auto_data.csv", header=T, sep=";", row.names=1)
# Concatenate the three vectors
autos <- c(autos_data$Cars, autos_data$trucks, autos_data$Suvs)
# Compute the largest y value used in the autos
max_num <- max(autos)
# Create uneven breaks
brk <- c(0,3,4,5,6,10,16)
# Create a histogram for autos with fire colors, set uneven
# breaks, make x axis range from 0-max_num, disable right-
# closing of cell intervals, set heading, make y-axis labels
# horizontal, make axis labels smaller, make areas of each
# column proportional to the count
hist(autos, col=heat.colors(length(brk)), breaks=brk,
     xlim=c(0,max_num), right=F, main="Probability Density",
     las=1, cex.axis=0.8, freq=F)
```

Funzioni intrinseche

Sono richiamate a comando, senza ulteriori specificazioni

<code>abs(x)</code>	absolute value
<code>cos(x)</code> , <code>sin(x)</code> , <code>tan(x)</code>	cosine, sine, tangent of angle x in radians
<code>exp(x)</code>	exponential function
<code>log(x)</code>	natural (base-e) logarithm
<code>log10(x)</code>	common (base-10) logarithm
<code>sqrt(x)</code>	square root

```
> A=3; C=(A+2*sqrt(A))/(A+5*sqrt(A)); C
[1] 0.5543706
```

Notare il sengo di “;” che separa le istruzioni di una stessa linea

Superare le sfide: caricamento di dati

Il caricamento dei dati è il primo passo per la loro analisi computerizzata

Il comando `c()` sta per *combine* ed indica la composizione o il contenuto di un oggetto (variabili in questo caso)

```
> age<-c(50,62,60,40,48,47,57,70,48,67)
> insulin<-c(16.5,10.8,32.3,19.3,14.2,11.3,
             15.5,15.8,16.2,11.2)
> ins<- data.frame(age, insulin)# aggrega
> attach(ins)# rende disponibili
> Ins # visualizza
```

Dati originali

age	insulin
50	16.5
62	10.8
60	32.3
40	19.3
48	14.2
47	11.3
57	15.5
70	15.8
48	16.2
67	11.2

age	insulin
1 50	16.5
2 62	10.8
3 60	32.3
4 40	19.3
5 48	14.2
6 47	11.3
7 57	15.5
8 70	15.8
9 48	16.2
10 67	11.2

Vedremo meglio più avanti l'uso di tali comandi

Logaritmi

``log'` naturali (base 'e')
``log10'` decimali (i.e., base 10) ``log2'` binari(i.e., base 2)

The general form ``logb(x, base)'` computes logarithms with base `'base'` (``log10'` e ``log2'` sono casi speciali).

```
Usage:
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)
:
x: un valore singolo o un vettore di valori numrici.
base: numero positivo. Se non indicato si intende 'e'.
```

``log(0)'` produce `'-Inf'`

Superare le sfide: lettura di un file

```
tempmar<-scan("tempsea.dat",sep="")
```

Se i dati da analizzare formano un unico flusso omogeneo (ovvero sono sulla singola colonna di una matrice) possono essere letti con il comando **scan**

Sep indica il carattere che separa i diversi dati

Esempio:
Media mensile della temperatura al livello del mare a Callao Peru (1956-1985)
I dati sono contenuti nel file testo:
`tempsea.dat`

```
> tempmar
[1] 19.4 21.2 22.6 21.5 19.6 18.0 17.8 17.0 17.2 16.8 17.0 17.6 18.5 19.0 18.4
[16] 17.1 16.3 15.4 15.9 14.6 15.0 15.1 15.8 18.2 20.8 19.6 17.8 18.4 17.0 16.8
[31] 16.6 15.9 16.3 15.7 16.1 16.8 18.4 19.8 20.2 18.8 18.2 17.8 17.6 17.0 16.6
[46] 16.2 16.4 16.2 17.8 22.3 22.1 21.8 22.2 21.2 20.3 18.7 17.7 17.9 17.9 20.6
[61] 21.8 22.2 22.0 20.1 18.8 18.2 18.0 17.0 17.0 17.1 17.5 17.0 19.0 21.2 20.6
[76] 19.6 18.7 17.8 16.9 16.6 16.8 17.2 17.6 18.6 19.0 19.5 19.2 18.0 17.2 17.1
[91] 16.6 16.8 16.7 16.6 16.6 17.6 19.0 20.6 19.3 18.7 18.2 17.3 16.7 16.6 16.4
[106] 16.4 16.4 16.8 19.0 19.1 18.1 17.4 17.6 17.0 16.6 16.4 16.6 16.0 16.4 16.5
[121] 17.4 18.8 19.4 18.2 18.4 17.9 17.7 17.4 17.3 17.0 17.0 18.0 19.0 19.5 19.2
[136] 17.8 16.2 15.6 15.3 15.7 15.8 16.0 16.2 16.2 17.6 19.6 20.4 21.3 20.6 19.5
[151] 19.0 18.4 17.6 17.5 17.9 18.6 19.9 20.4 19.2 18.1 17.5 16.8 16.4 16.2 15.6
[166] 16.2 16.5 16.8 18.4 19.6 19.1 17.6 16.9 16.2 16.1 15.4 15.4 15.0 15.1 16.4
[181] 17.6 17.6 18.5 16.6 16.4 15.5 15.8 16.6 16.4 16.1 16.6 17.1 18.7 19.0 20.4
[196] 20.5 21.0 19.5 17.4 17.3 17.2 17.3 17.3 17.8 19.2 19.8 20.0 19.1 18.6 17.8
[211] 16.8 16.9 16.9 17.3 17.0 17.2 18.2 19.1 19.6 19.8 18.7 18.0 18.0 18.0 17.4
[226] 16.8 17.1 17.2 18.6 20.6 21.8 21.4 21.0 21.4 21.1 20.0 18.9 19.0 19.3 21.4
[241] 23.2 23.0 21.3 18.4 17.4 16.6 16.0 15.5 15.7 16.2 17.1 16.4 17.0 18.2 18.6
[256] 18.9 18.6 19.1 17.6 16.8 16.1 15.8 16.5 16.3 16.7 18.1 21.1 19.8 18.6 16.9
[271] 16.7 16.1 16.0 15.9 15.6 16.4 17.2 21.0 21.3 19.6 19.8 19.9 19.4 19.1 17.6
[286] 18.0 18.4 20.2 20.4 20.5 20.6 20.6 19.1 18.2 17.6 17.0 16.6 16.6 17.2 17.8
[301] 18.0 20.0 19.9 19.1 17.6 16.5 16.6 15.9 16.2 16.6 17.1 17.3 18.5 18.5 19.2
[316] 19.0 18.3 17.3 17.4 17.4 17.0 17.2 17.4 18.3 18.6 18.8 19.4 19.1 18.3 18.0
[331] 17.5 16.8 16.6 16.6 16.9 17.6 17.4 18.8 18.5 18.3 18.5 17.6 16.8 16.8 16.2
[346] 17.0 16.9 17.0 17.6 18.8 19.1 18.9 19.3 18.6 18.4 17.6 17.5 19.3 21.9 23.7
```

Soluzione di problemi <http://pj.freefaculty.org/R/Rtips.html>

Funzioni statistiche

Dato un campione A, per calcolarne la media, la varianza, la deviazione standard e la mediana si usano le seguenti funzioni:

```
A<-tempmar
> mean(A)      mad(x, center = median(x), constant = 1.4826, na.rm = FALSE,
> var(A)       low = FALSE, high = FALSE)
> sd(A)
> median(A)
Il numero dei casi di A, il valore massimo e il minimo si ottengono con le
chiamate:
> length(A)
> max(A)
> min(A)
```

La funzione summary genera un riepilogo di sei statistiche calcolate sul campione, ossia il minimo, il primo quartile, la media, la mediana il terzo quartile e il massimo. Ad esempio, sul campione dei primi 20 numeri interi si ottiene il seguente output:

```
> A <- 1:20 # vettore contenente i primi 20 numeri interi positivi
> summary(A)
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.00 5.75 10.50 10.50 15.25 20.00
```

Operatività di R

L'ambiente R è subito operativo

Ghosh studia il dataset relativo al livello delle piene del fiume Iroquois (IN-USA).

```
14.1 15.2 18.0 19.1 20.7 22.6 23.9 27.3 33.3 33.4 33.6 34.3 37.1 41.1 42.7 45.1
45.3 47.4 48.0 51.0 55.1 63.3 70.3 74.0 77.7 77.9 83.0 93.4 106.0 109.0 130.0
146.0
```

Per analizzarlo occorre poterlo nello spazio di lavoro di R

Possibilità 1) Digitazione diretta

```
River<-c(14.1, 15.2, 18.0, 19.1, 20.7, 22.6, 23.9, 27.3, 33.3, 33.4, 33.6, 34.3, 37.1,
41.1, 42.7, 45.3, 47.4, 48.0, 51.0, 55.1, 63.3, 70.3, 74.0, 77.7, 77.9, 83.0,
93.4, 106.0, 109.0, 130.0,146.0)
```

I valori confluiscono nell'oggetto River che si autoconfigura come vettore di valori.

L'oggetto River è immediatamente disponibile per le elaborazioni

```
> River
[1] 14.1 15.2 18.0 19.1 20.7 22.6 23.9 27.3 33.3 33.4 33.6 34.3 37.1 41.1 42.7
[16] 45.1 45.3 47.4 48.0 51.0 55.1 63.3 70.3 74.0 77.7 77.9 83.0 93.4 106.0 109.0
[31] 130.0 146.0
```

Calcolo delle medie

```
Quota<-scan("Cadmium.txt",sep=" ")
N<-length(Quota)
# Medie ferme o analitiche
MQ<-sqrt(sum(Quota^2)/n) # Media quadratica
MA<-mean(Quota) # Media aritmetica
MG<-exp(sum(log(Quota))/n) # Media geometrica
MH<-n/sum(1/Quota) # Media armonica
MB<-sqrt(n/sum(1/Quota^2)) # Media biarmonica
cat(MQ,MA,MG,MH,MB,"\n") # Medie lasche
Mp1<-mean(Quota,trim=0.1) # Media potata del 10% agli estremi
Mo<-sort(Quota)[which.max(table(Quota))] # Moda
Me<-median(Quota) # Mediana
cat(Mp1,Mo,Me,"\n")
```

```
62.0723 57.24419 51.5021 44.82847 37.84315
```

```
56.16571 11.9 56.7
```

Uso dei dati: distribuzione di frequenza

```
> table(River)
River
14.1 15.2 18 19.1 20.7 22.6 23.9 27.3 33.3 33.4 33.6 34.3 37.1 41.1 42.7 45.1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
45.3 47.4 48 51 55.1 63.3 70.3 74 77.7 77.9 83 93.4 106 109 130 146
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Così non è molto utile: ogni dato rappresenta se stesso

```
table(cut(River,breaks=c(10,20,40,80,160)))
```

```
> table(cut(River,breaks=c(10,20,40,80,160)))
```

```
(10,20] (20,40] (40,80] (80,160]
4 9 13 6
```

Il raggruppamento in classi aiuta a percepire l'ordine generale del fenomeno

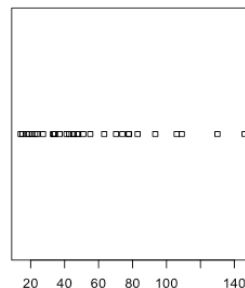
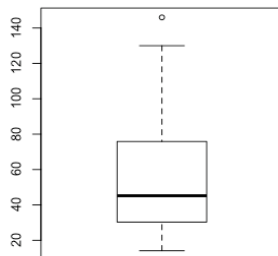
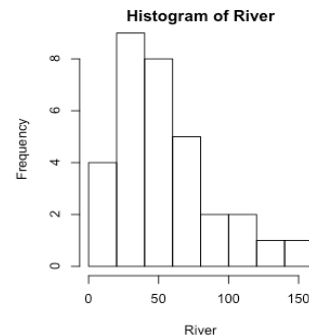
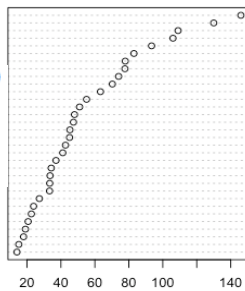
Esempio

```
> par(mfrow=c(2,2),mar=c(4.0,4.5,1,2))
> dotchart(River)
> hist(River)
> boxplot(River)
> stripchart(River)
```

I quattro comandi richiamano altrettanti grafici standard per la descrizione dei dati.

In ogni comando sono previste opzioni che ne migliorano l'estetica ed agevolano la comunicazione dei messaggi.

Maggiori notizie con l'help di linea o sui manuali



Operatività di R/2

Possibilità 2) Importazione di dati come flusso da un file testo

Glucosio nel sangue di 46 soggetti anziani

```
3.52, 3.905, 4.07, 4.07, 4.29, 4.345, 4.4, 4.455, 4.565, 4.62, 4.62, 4.675, 4.840,
4.840, 4.895, 4.895, 4.95, 4.95, 5.115, 5.115, 5.225, 5.225, 5.225, 5.335, 5.335,
5.39, 5.39, 5.39, 5.455, 5.555, 5.61, 5.665, 5.72, 5.775, 5.83, 5.83, 5.885, 5.885,
6.215, 7.095, 7.205, 8.14, 9.9, 10.89, 11.605, 12.045
```

Ipotizziamo che i dati siano in un file testo: Glucosio.txt, separati da virgole

```
> Glu<-scan("Glucosio.txt",sep=",") # il separatore può essere anche ";" o " "
```

I valori confluiscono nell'oggetto Glu che si autoconfigura come vettore di valori.

L'oggetto Glu è immediatamente disponibile per le elaborazioni

```
> Glu
[1] 3.520 3.905 4.070 4.070 4.290 4.345 4.400 4.455 4.565 4.620 4.620 4.675 4.840
[14] 4.840 4.895 4.895 4.950 4.950 5.115 5.115 5.225 5.225 5.225 5.335 5.335 5.390
[27] 5.390 5.390 5.455 5.555 5.610 5.665 5.720 5.775 5.830 5.830 5.885 5.885 6.215
[40] 7.095 7.205 8.140 9.900 10.890 11.605 12.045
```

Questa lettura non va bene perché i nomi sulle righe sono ripetuti.

```
Demog<-read.table("demoitalia.txt",sep=" ",dec=".",header=TRUE,row.names=1)
```

#

```
Demog<-read.table("demoitalia.txt",sep=" ",dec=".",header=TRUE)
```

Questa lettura va bene, ma ora la chiave del record diventa una variabile nominale

#

```
Natal<-Demog[,6];Natal
```

Recupero di una variabile ed assegnazione ad un oggetto poi visualizzato

```
is.vector(Natal);is.matrix(Natal)
```

Controllo della tipologia

```
Natal.mat<-as.matrix(Natal)
```

Trasformazione in matrice per usarlo in operazioni matriciali

```
is.vector(Natal);is.matrix(Natal)
```

#

```
n<-length(Natal) # ampiezza del campione
```

```
Mu<-mean(Natal) # media aritmetica
```

```
Me<-median(Natal) # calcolo della mediana
```

```
Sqm<-sd(Natal) # scarto quadratico medio
```

```
M3<-(n*sum((Natal.mat-Mu)^3))/((n-1)*(n-2)) # Momento terzo
```

```
G1 = M3/(Sqm^3) # Indice di asimmetria (uno dei tanti)
```

```
M4<-(n*(n+1)*sum((Natal.mat-Mu)^4)-3*(n-1)*(sum((Natal.mat-Mu)^2))^2)/((n-1)*(n-2)*(n-3))
```

Momento quarto

```
G2<-M4/Sqm^4 # Indice di curtosi (uno dei tanti)
```

```
cat(n,Mu,Me,Sqm,M3,G1,M4,G2,"n")
```

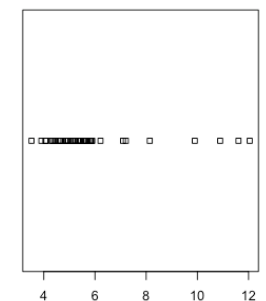
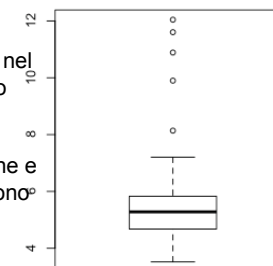
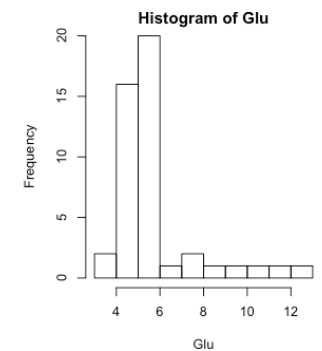
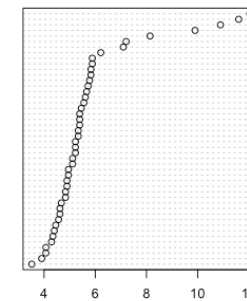
Disuguaglianza di Pearson (1916)

```
Test<-(G2+3) >= (G1^2+1);Test
```

Se i calcoli sono corretti questa variabile logica è vera.

Esempio

```
par(mfrow=c(2,2),
mar=c(4.0,4.5,1,2))
Glu<-
scan("Glucosio.txt",sep=
",")
dotchart(Glu)
hist(Glu)
boxplot(Glu)
stripchart(Glu)
```



Le opzioni **mfrow** e **mar** nel comando **par** specificano dettagli per i grafici.

c(2,2) vuol dire quattro grafici a matrice due righe e due colonne. I margini sono riferiti all'area del grafico dove R riporta le figure.

Data frame (matrici di dati)

I data frame sono tabelle di elementi non necessariamente omogenei

```
> intake.pre <- c(5260, 5470, 5640, 6180, 6390,
+ 6515, 6805, 7515, 7515, 8230, 8770)
> intake.post <- c(3910, 4220, 3885, 5160, 5645,
+ 4680, 5265, 5975, 6790, 6900, 7335)

> d <- data.frame(intake.pre, intake.post)
> d
  intake.pre intake.post
1      5260      3910
2      5470      4220
3      5640      3885
4      6180      5160
5      6390      5645
6      6515      4680
7      6805      5265
8      7515      5975
9      7515      6790
10     8230      6900
11     8770      7335
> d$intake.post
[1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335
```

Lettura di una matrice di dati

I dati possono essere importati da altri programmi. Ad esempio EXCEL

Sulla destra sono riportati I dati di 12 Paesi del Sud America (fonte UNESCO, 1990)

I nomi delle variabili sono sulla prima riga ed il nome dei Paesi è sulla prima colonna

	Birth R.	Inf.M.	GNP
Argentina	20.7	25.7	2370
Bolivia	46.6	111.0	630
Brazil	28.6	63.0	2680
Chile	23.4	17.1	1940
Colombia	27.4	40.0	1260
Ecuador	32.9	63.0	980
Guyana	28.3	56.0	330
Paraguay	34.8	42.0	1110
Peru	32.9	109.9	1160
Uruguay	18.0	21.9	2560
Venezuela	27.5	23.3	2560
Mexico	29.0	43.0	2490

```
LA<-read.table("LAmer.dat", header=TRUE, row.names=1,dec=".")
```

Scrivendo LA I dati diventano disponibili

Data frames /2

E' la tipica organizzazione dei dati che si incontra nelle analisi statistiche.

E' una tabella rettangolare con righe e colonne: le righe possono avere composizione eterogenea, ma le colonne debbono contenere dati della stessa natura (stringa, logico, numerico).

Il comando per caricare la matrice dei dati è **read.table**

Nome	Tempo_totale	Interventi	Giudizio	Perc_errore	Leadership
A.V.	98	2	scarso	0.43	F
C.D.	103	3	buono	0.28	F
N.S.	104	4	ottimo	0.44	T
S.F.	76	2	scarso	0.49	F
F.O.	92	3	medio	0.37	F
R.A.	119	4	ottimo	0.32	T
M.F.	81	0	sufficiente	0.41	F
R.T.	78	2	pessimo	0.35	T
D.S.	96	1	buono	0.37	F
G.A.	103	4	buono	0.41	F
R.S.	88	2	medio	0.40	F

Lettura di una matrice dei dati/2

```
Work<-read.table("Colloqui.csv",header=TRUE,row.names=1,sep=";",dec=".")
```

Il file deve esistere nella cartella di lavoro in formato .csv o in formato .txt

```
> Work<-read.table("Colloqui.csv",header=TRUE,row.names=1,sep=";",dec=".")
> Work
  Tempo_totale Interventi  Giudizio Perc_errore Leadership
A.V.          98         2    scarso      0.43      FALSE
C.D.         103         3     buono      0.28      FALSE
N.S.         104         4     ottimo      0.44       TRUE
S.F.          76         2    scarso      0.49      FALSE
F.O.          92         3     medio      0.37      FALSE
R.A.         119         4     ottimo      0.32       TRUE
M.F.          81         0 sufficiente      0.41      FALSE
R.T.          78         2    pessimo      0.35       TRUE
D.S.          96         1     buono      0.37      FALSE
G.A.         103         4     buono      0.41      FALSE
R.S.          88         2     medio      0.40      FALSE
>
```

Data frame e tabelle

Il data.frame può essere creato con vettori da console

```
A1<-c(1, 3, 5, 9, 11, 32, 28, 14, 17, 19, 22, 31)
A2<-c(-1, 0, 2, -2, 4, 2, 5, 2, -1, -1, 0, 2)
A3<-c(3.14, 2.71, 1.41, 3.6, 1.8, 0.9, 2.7, 1.9, 2.4, 3.3, 0.7, 0.5)
A4<-c(2,3,2,4,3,2,1,3,4,1,2,2)
A<-cbind(A1,A2,A3,A4)
rownames(A)<-month.abb
colnames(A)<-c("Intero", "Relativo", "Frazionario", "Codice")
A<-data.frame(A)
print(A)
attach(A)
table(Codice)
table(Codice)/length(Codice)
table(Codice)*100/length(Codice)
```

```
a<-read.table("Rabal.csv",header=TRUE,sep=";",dec=".",row.names=1)
```

```
> a
      localization tumorsize progress
XX348      proximal      6.3    FALSE
XX234      distal      8.0     TRUE
XX987      proximal     10.0    FALSE
XX123      distal      8.7     TRUE
XX175      distal      9.1    FALSE
```

```
> a[c(1,3),]
      localisation tumorsize progress
XX348      proximal      6.3        0
XX987      proximal     10.0        0
> a[c(T,F,T),]
      localisation tumorsize progress
XX348      proximal      6.3        0
XX987      proximal     10.0        0
> a$localisation
[1] "proximal" "distal" "proximal"
> a$localisation=="proximal"
[1] TRUE FALSE TRUE
> a[a$localisation=="proximal", ]
      localisation tumorsize progress
XX348      proximal      6.3        0
XX987      proximal     10.0        0
```

Subsetting

subset rows by a
vector of indices

subset rows by a
logical vector

subset a column

comparison resulting in
logical vector

subset the selected
rows

Data frame e tabelle (dati esterni predefiniti)

```
data("Forbes2000",package="HSAUR") # carica il data set
head(Forbes2000,3) # visualizza i primi 3 record
Posiz<-c(1,3,5);print(Forbes2000[Posiz,]) # visualizza record specifici
dim(Forbes2000)
ncol(Forbes2000)
nrow(Forbes2000)
names(Forbes2000)

Forbes2000[1:4,c("name","sales","profits","assets")] # scelta di righe e
colonne

order_sales<-order(Forbes2000$sales)
Forbes2000[order_sales[c(2000,1999,1998)],c("name","sales","profits")]

# top sellers
```

```
### Costituzione di un data frame
# Il data frame è una tabella riga/colonna in cui le unità sono sulle righe
# e le variabili sulle colonne. È una struttura molto usata in statistica
# Creazione di data frame da tastiera
State<-c("MA", "TX", "NH", "TX", "NH", "IL", "TX", "NH", "CA", "NH", "MD", "TX", "GA", "TX", "MD", "MA")
Age<-c(34, 41, 42, 37, 35, 51, 29, 33, 44, 37, 42, 38, 46, 36, 42, 46)
HART<-data.frame(State, Age); dim(HART)
# Aggiungiamo una variabile
HART$Weight<-c(195, 211, 181, 164, 223, 196, 174, 188, 153, 242, 204, 179, 184, 218, 246, 188)
Systolic<-c(135, 140, 125, 130, 135, 145, 115, 130, 140, 150, 130, 135, 125, 140, 165, 140)
Diastolic<-c(75, 80, 65, 70, 85, 85, 75, 70, 80, 90, 85, 70, 75, 80, 100, 85)
HART<-cbind(HART, Systolic, Diastolic); names(HART)
# Lettura diretta del data frame
HART1<-read.table("CARDIAC1.Csv",dec=".",sep=";",header=TRUE)
names(HART1)
# Fusione dei due data set
HART<-rbind(HART, HART1)
# Individuazione di una particolare variabile con richiamo della colonna specifica
HART$Systolic; HART[, "Systolic"]; HART[, 4]
# Righe e colonne
dd<-dim(HART)
nrow<-dd[1]; ncol<-dd[2]; cat(nrow, ncol, "\n")
# In alternativa ...
nrow<-length(HART[, 1]); ncol<-length(HART[1,]); cat(nrow, ncol, "\n")
### Distribuzioni di frequenza
table(Systolic); table(Systolic)/nrow
# Per le variabili stringa si devono ordinare i livelli
State.F<-factor(State, levels=c("CA", "GA", "IL", "MA", "MD", "NH"))
table(State.F); table(State.F)/nrow
# Dati in classi
# numero delle classi
nclassi<-round(sqrt(nrow))
Minimi<-apply(HART[, 2:5], 2, min); Massimi<-apply(HART[, 2:5], 2, max)
Classi<-seq(Minimi[1], Massimi[1], length=nclassi+1); Classi
par(mfrow=c(1,2)); par(mar=c(4,4,2,2))
hist(HART[, 2], main="Istogramma dei valori osservati", ylab="Frequenze
relative", xlab="Modalità", breaks=Classi, freq=FALSE, right=TRUE, col="lightgreen")
Classi<-seq(Minimi[2], Massimi[2], length=nclassi+1)
hist(HART[, 3], main="Istogramma dei valori osservati", ylab="Frequenze
relative", xlab="Modalità", breaks=Classi, freq=FALSE, right=TRUE, col="tomato2" )
```

Esempio

Operatività di R/3

Possibilità 3) Acquisizione di una matrice dei dati

Per analizzare alcuni indicatori sulla presenza di arsenico nell'organismo conviene caricare tutta la matrice dei dati con il comando `read.table`
Il formato dei dati può essere sia txt (testo) che csv (comma separated values)

```
> Ars<-read.table("Arsenic.csv",sep=";",header=TRUE)
> Ars
  AGE SEX DRINKUSE COOKUSE ARSWATER ARSNAILS
1  44  2      5      5  0.00087  0.119
2  45  2      4      5  0.00021  0.118
3  44  1      5      5  0.00000  0.099
4  66  2      3      5  0.00115  0.118
5  37  1      2      5  0.00000  0.277
6  45  2      5      5  0.00000  0.358
7  47  1      5      5  0.00013  0.080
8  38  2      4      5  0.00069  0.158
9  41  2      3      2  0.00039  0.310
10 49  2      4      5  0.00000  0.105
11 72  2      5      5  0.00000  0.073
12 45  2      1      5  0.04600  0.832
13 53  1      5      5  0.01940  0.517
14 86  2      5      5  0.13700  2.252
15  8  2      5      5  0.02140  0.851
16 32  2      5      5  0.01750  0.269
17 44  1      5      5  0.07640  0.433
18 63  2      5      5  0.00000  0.141
19 42  1      5      5  0.01650  0.275
20 62  1      5      5  0.00012  0.135
21 36  1      5      5  0.00410  0.175
```

Matrice dei dati in un worksheet o in un file testo

AGE	SEX	DRINKUSE	COOKUSE	ARSWATER	ARSNAILS
44	2	5	5	0.00087	0.119
45	2	4	5	0.00021	0.118
44	1	5	5	0	0.099
66	2	3	5	0.00115	0.118
37	1	2	5	0	0.277
45	2	5	5	0	0.358
47	1	5	5	0.00013	0.08
38	2	4	5	0.00069	0.158
41	2	3	2	0.00039	0.31
49	2	4	5	0	0.105
72	2	5	5	0	0.073
45	2	1	5	0.046	0.832
53	1	5	5	0.0194	0.517
86	2	5	5	0.137	2.252
8	2	5	5	0.0214	0.851
32	2	5	5	0.0175	0.269
44	1	5	5	0.0764	0.433
63	2	5	5	0	0.141
42	1	5	5	0.0165	0.275
62	1	5	5	0.00012	0.135
36	1	5	5	0.0041	0.175

Ordinamento e suddivisione (Sort & subset)

```
# Sort & Merge
# Lettura file csv Larc<-read.csv("EserSM.csv", header =
TRUE,row.names=1, sep = ";")names(Larc)
# Ordinamento dei record del file secondo i valori ascendenti della variabile
"Semilavorati"
Larc<-Larc[order(Larc[,3],decreasing=FALSE),];Larc
# Ordinamento dei record del file secondo i valori ascendenti delle etichette
in row.names
Larc<-Larc[order(row.names(Larc),decreasing=FALSE),];Larc
# Suddivisione verticale del data set (2 variabili per blocco)
LarcLeft<-Larc[,1:2];LarcLeft
LarcRight<-Larc[,3:4];LarcRight
# Suddivisione orizzontale del data set (Anni minori di 2000 e gli altri)
LarcUp<-subset(Larc, subset=row.names(Larc)<2000);LarcUp
LarcDown<-subset(Larc, subset=row.names(Larc)>=2000);LarcDown
# Riquadri (selezione di righe e colonne)
J<-seq(1,nrow(Larc),by=2)
LarcMix<-subset(Larc[J,],select=c(1,3));LarcMix
```

Fusione di due data sets

```
d1
  id sex tc
1  Nam 4.0
2  Nu 3.5
3  Nu 4.7
4  Nam 7.7
5  Nam 5.0
6  Nu 4.2
7  Nam 5.9
8  Nam 6.1
9  Nam 5.9
10 Nu 4.0

d2
  id sex tg
1  Nam 1.1
2  Nu 2.1
3  Nu 0.8
4  Nam 1.1
5  Nam 2.1
6  Nu 1.5
7  Nam 2.6
8  Nam 1.5
9  Nam 5.4
10 Nu 1.9
11 Nu 1.7

d <- merge(d1, d2, by="id",all=TRUE)
d
  id sex.x tc sex.y tg
1  1  Nam 4.0  Nam 1.1
2  2  Nu 3.5   Nu 2.1
3  3  Nu 4.7   Nu 0.8
4  4  Nam 7.7  Nam 1.1
5  5  Nam 5.0  Nam 2.1
6  6  Nu 4.2   Nu 1.5
7  7  Nam 5.9  Nam 2.6
8  8  Nam 6.1  Nam 1.5
9  9  Nam 5.9  Nam 5.4
10 10 Nu 4.0   Nu 1.9
11 11 <NA> NA   Nu 1.7
```

Merge file# Lettura di due file di testo

```
Family1<-read.table("family.txt", header = TRUE, sep = "\t")
Family2<-read.table("familyBis.txt", header = TRUE, sep = "\t")
```

Fusione dei due data set

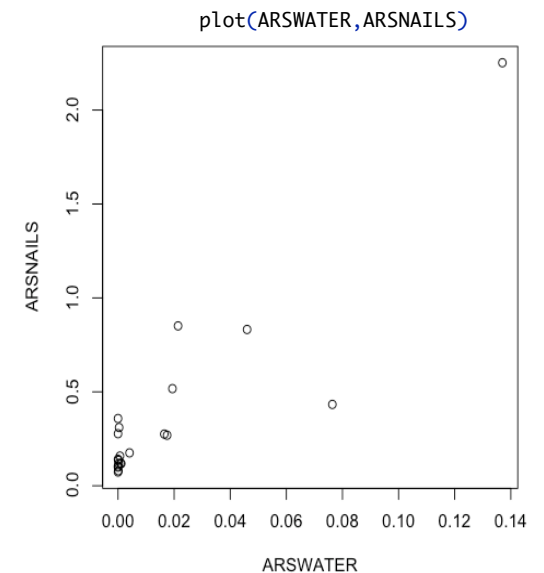
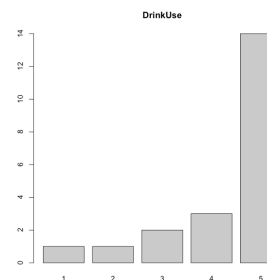
```
Family<-merge(Family1,Family2,sort = T,all=T)
```

Esempio

```
Ars<-Read.table("Arsenic.csv",sep=";",header=TRUE
attach(Ars)
```

```
# header=TRUE indica che
il file dati contiene i
nomi delle variabili, se
FALSE o assente, si
presume che i nomi non
siano indicati (R userà
V1,V2, etc.)
```

```
barplot(table(DRINKUSE
E),main="DrinkUse")
```



Esempio

- **Lettura di un file testo:**

```
> family <- read.table("family.txt", header=TRUE)
> attach(family)
```

- **Calcolo della media:**

```
> mean(age)
[1] 24.95
```

```
sex age mother father siblings
1 m 29 58 61 1
2 w 26 53 54 2
3 m 24 49 55 1
4 w 25 56 63 3
5 w 25 49 53 0
6 w 23 55 55 2
7 m 23 48 54 2
8 m 27 56 58 1
9 m 25 57 59 1
10 m 24 50 54 1
11 w 26 61 65 1
12 m 24 50 52 1
13 m 29 54 56 1
14 m 28 48 51 2
15 w 23 52 52 1
16 m 24 45 57 1
17 w 24 59 63 0
18 w 23 52 55 1
19 m 24 54 61 2
20 w 23 54 55 1
```

- **Sommario dei dati:**

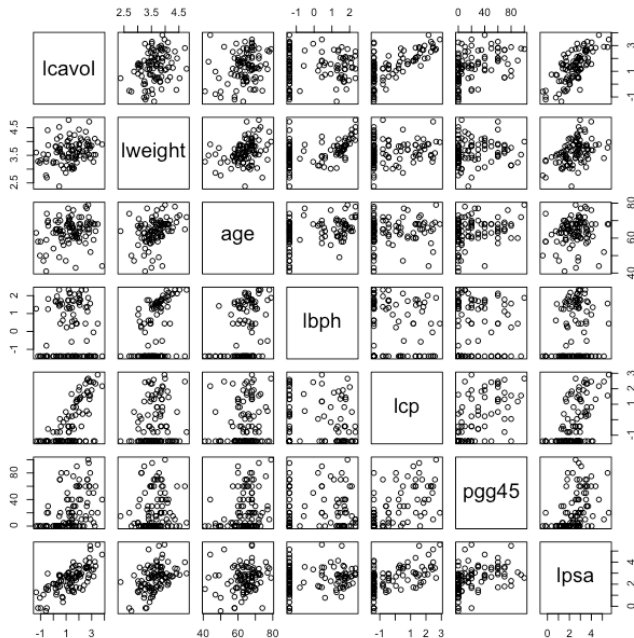
```
> summary(family)
sex      age      mother      father      siblings
m:11   Min.   :23.00   Min.   :45.00   Min.   :51.00   Min.   :0.00
w: 9   1st Qu.:23.75   1st Qu.:49.75   1st Qu.:54.00   1st Qu.:1.00
      Median :24.00   Median :53.50   Median :55.00   Median :1.00
      Mean   :24.95   Mean   :56.00   Mean   :56.65   Mean   :1.25
      3rd Qu.:26.00   3rd Qu.:56.00   3rd Qu.:59.50   3rd Qu.:2.00
      Max.   :29.00   Max.   :61.00   Max.   :65.00   Max.   :3.00
```

Esempio

```
DataMat<-
read.table("Prostate
e.csv",
Sep=";",dec=".",head
er=TRUE,
row.names=1)
names(DataMat)
attach(DataMat)
plot(DataMat[,c(1:4
,6,8:9)])
```

Il comando names
esplicita i nomi delle
variabili della matrice
dei dati.

Parentesi e numeri nel
comando plot
individuano le variabili
metriche del data set



Operatività di R/4

Possibilità 4) Acquisizione dei dati dalla rete

Analisi del dataset Prostate cancer disponibile su

<http://www-stat.stanford.edu/~tibs/ElemStatLearn/>

Selezionate il data set, copiatelo
ed incollatelo in un foglio di
lavoro di excel.

Salvate poi in formato csv

THE ELEMENTS OF STATISTICAL LEARNING										
Trevor Hastie, Robert Tibshirani, and Jerome Friedman										
About this book	train	lccavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa
How to order	1	-0.579818495	2.769459	50	-1.38629436	0	-			
Table of contents	1.38629436	6	0	-0.4307829	T	-1.38629436	0	-		
Data	2	-0.994252273	3.319626	58	-1.38629436	0	-			
Errata	1.38629436	6	0	-0.1625189	T	-1.38629436	0	-		
R Functions	3	-0.510825624	2.691243	74	-1.38629436	0	-			
Complements	1.38629436	7	20	-0.1625189	T	-1.38629436	0	-		
Short course: Statistical Learning and Data Mining	4	-1.203972804	3.282789	58	-1.38629436	0	-			
Information for instructors	1.38629436	6	0	-0.1625189	T	-1.38629436	0	-		
Book reviews	5	0.751416089	3.432373	62	-1.38629436	0	-			
	1.38629436	6	0	0.3715636	T	-1.38629436	0	-		
	6	-1.049822124	3.228826	50	-1.38629436	0	-			
	1.38629436	6	0	0.7654678	T	-1.38629436	0	-		
	7	0.737164066	3.473518	64	0.61518564	0	-			
	1.38629436	6	0	0.7654678	F	-1.38629436	0	-		
	8	0.693147181	3.539509	58	1.53686722	0	-			
	1.38629436	6	0	0.8544153	T	-1.38629436	0	-		
	9	-0.776528789	3.539509	47	-1.38629436	0	-			
	1.38629436	6	0	1.0473190	F	-1.38629436	0	-		
	10	0.223143551	3.244544	63	-1.38629436	0	-			
	1.38629436	6	0	1.0473190	F	-1.38629436	0	-		
	11	0.254642218	3.604138	65	-1.38629436	0	-			
	1.38629436	6	0	1.2669476	T	-1.38629436	0	-		

Lettura sul sito

Alcuni siti sono predisposti per il caricamento diretto in R

```
LAozone = read.table("http://www-stat.stanford.edu/~tibs/
ElemStatLearn/datasets/LAozone.data", sep=";", head=T)
```

```
> head(LAozone,5) # visualizziamo le prime 5 righe del data frame
```

```
ozone   vh wind humidity temp   ibh dpg ibt vis doy
1      3 5710   4      28  40 2693 -25  87 250   3
2      5 5700   3      37  45  590 -24 128 100   4
3      5 5760   3      51  54 1450  25 139  60   5
4      6 5720   4      69  35 1568  15 121  60   6
5      4 5790   6      19  45 2631 -33 123 100   7
```

Scrittura di un data set

```
## To write a CSV file for input to Excel one might use
x <- data.frame(a = I("a" quote"), b = pi)
write.table(x, file = "foo.csv", sep = ";", col.names = NA, qmethod =
"double")
## and to read this file back into R one needs
read.table("foo.csv", header = TRUE, sep = ";", row.names = 1)
## NB: you do need to specify a separator if qmethod = "double".
### Alternatively
write.csv(x, file = "foo.csv")
read.csv("foo.csv", row.names = 1)
## or without row names
write.csv(x, file = "foo.csv", row.names = FALSE)
read.csv("foo.csv")## End(Not run)
```

I pacchetti (packages) di R

- Il cuore di **R** è rappresentato dal modulo base (che offre gli strumenti fondamentali per l'analisi statistica) e attorno a questo modulo ruotano una serie di altre librerie aggiuntive, alcune delle quali sono già comprese nel programma **R** al momento in cui lo si installa, mentre altre librerie ancora possono essere aggiunte e installate dall'utente dopo averle scaricate dal sito succitato.
- Sul sito del **The Comprehensive R Archive Network - CRAN** è possibile scaricare un grande numero di packages che spaziano nei più disparati campi della statistica applicata.

Installazione di un pacchetto



Per analizzare il pacchetto si va sulla sua
home page

<http://cran.r-project.org/>

CRAN

[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R

[R Homepage](#)
[The R Journal](#)

Software

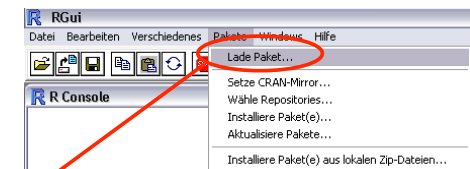
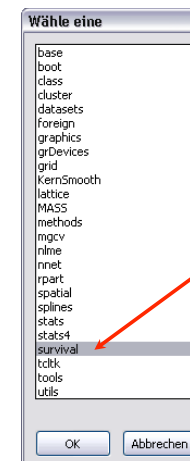
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation

[Manuals](#)
[FAQs](#)
[Contributed](#)

Packages for special functions in R

- common packages are already included in R
- they just have to be loaded



- very special packages you find on this website:

<http://cran.rakanu.com/>

Available Bundles and Packages

[aaMT](#)
[abnd](#)
[accuracy](#)

Mutual information for protein sequence alignments
Combine multi-dimensional arrays
Tools for testing and improving accuracy of statistical results.

I pacchetti disponibili

Available Packages

Currently, the CRAN package repository features 2041 available packages.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

ADGofTest	Anderson-Darling GoF test
ADaCGH	Analysis of data from aCGH experiments
AER	Applied Econometrics with R
AGSDest	Estimation in adaptive group sequential trials
AICcmoavg	Model selection and multimodel inference based on (Q)AIC(c)
AIGIS	Areal Interpolation for GIS data
AIS	Tools to look at the data ("Ad Inidicia Spectata")
ALS	multivariate curve resolution alternating least squares (MCR-ALS)
AMORE	A MORE flexible neural network package
AcceptanceSampling	Creation and evaluation of Acceptance Sampling Plans
AdMit	Adaptive Mixture of Student-t distributions
AdaptFit	Adaptive Semiparametric Regression
AlgDesign	AlgDesign
Amelia	Amelia II: A Program for Missing Data
AnalyzefMRI	Functions for analysis of fMRI datasets stored in the ANALYZE or NIFTI format
Animal	Analyze time-coded animal behavior data
AquaEnv	AquaEnv - an integrated development toolbox for aquatic chemical model generation
ArDec	Time series autoregressive-based decomposition
aCGH.Spline	Robust spline interpolation for dual color array comparative genomic hybridisation data
aaMI	Mutual information for protein sequence alignments
abind	Combine multi-dimensional arrays
accuracy	Tools for testing and improving accuracy of statistical results
actuar	Actuarial functions

Informazioni su di uno specifico pacchetto

> library(help=MASS)

Information on package 'MASS'	Functions:
Description:	Null Spaces of Matrices Try All One-Term Additions to a Model Likelihood Ratio Tests for Negative Binomial GLMs Adaptive Numerical Integration Bandwidth for density() via Normal Reference Distribution Biased Cross-Validation for Bandwidth Selection Box-Cox Transformations for Linear Models
Bundle:	VR
Priority:	recommended
Contains:	MASS class mnet spatial
Version:	7.2-44
Date:	2008-08-13
Depends:	R (>= 2.4.0), grDevices, graphics, stats, utils
Suggests:	lattice, nlme, survival
Author:	5 original by Venables & Ripley. R port by Brian Ripley <ripley@stats.ox.ac.uk>, following earlier work by Kurt Hornik and Albrecht Gebhardt.
Maintainer:	Brian Ripley <ripley@stats.ox.ac.uk>
BundleDescription:	Functions and datasets to support Venables and Ripley, 'Modern Applied Statistics with S' (4th edition).
License:	GPL-2 GPL-3
URL:	http://www.stats.ox.ac.uk/pub/MASS4/
Packaged:	Wed Aug 13 14:07:54 2008; rpley
Package:	MASS
Description:	The main library and the datasets
Title:	Main Package of Venables and Ripley's MASS
LazyLoad:	yes
LazyData:	yes
Built:	R 2.8.0; universal-apple-darwin8.11.1; 2008-10-22 17:34:40; unix
Datasets:	Aids2 Animals Belgian-phones Boston Cars93 Cushings DDT Australian AIDS Survival Data Brain and Body Weights for 28 Species Belgium Phone Calls 1950-1973 Housing Values in Suburbs of Boston Data from 93 Cars on Sale in the USA in 1993 Diagnostic Tests on Patients with Cushing's Syndrome DDT in Kale

Esempio: fbasics

fBasics: Rmetrics - Markets and Basic Statistics

Environment for teaching "Financial Engineering and Computational Finance"

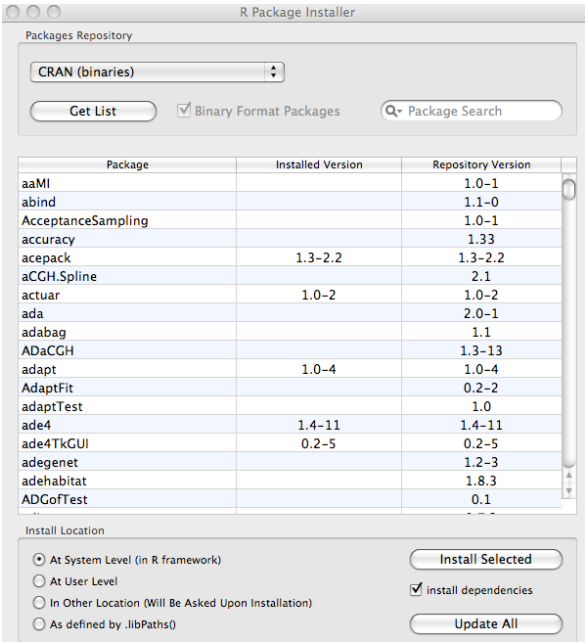
Version: 2100.78
Depends: R (≥ 2.6.0), MASS, methods, timeDate, timeSeries (≥ 2100.84)
Suggests: akima, spatial, RUnit, tcltk
Published: 2009-09-28
Author: Diethelm Wuertz and many others, see the SOURCE file
Maintainer: Rmetrics Core Team <Rmetrics-core at r-project.org>
License: GPL (≥ 2)
URL: http://www.rmetrics.org
In views: Distributions, Finance
CRAN checks: fBasics results

Downloads:

Package source: [fBasics_2100.78.tar.gz](#)
MacOS X binary: [fBasics_2100.78.tgz](#)
Windows binary: [fBasics_2100.78.zip](#)
Reference manual: [fBasics.pdf](#)
News/ChangeLog: [ChangeLog](#)
Old sources: [fBasics archive](#)

Non è da qui che si installano

Dal menù packages and data



Data sets disponibili in un pacchetto

> Data (package="cluster")

Data sets in package 'cluster':

agriculture	European Union Agricultural Workforces
animals	Attributes of Animals
chorSub	Subset of C-horizon of Kola Data
Flower	Flower Characteristics
plantTraits	Plant Species Traits Data
pluton	Isotopic Composition Plutonium Batches
ruspini	Ruspini Data
votes.repub	Votes for Republican Candidate in Presidential
Elections	
xclara	Bivariate Data Set with 3 Clusters

Una volta caricata la libreria i data sets inclusi saranno immediatamente disponibili

>library(cluster)

>agriculture

	x	y
B	16.8	2.7
DK	21.3	5.7
D	18.7	3.5
GR	5.9	22.2
E	11.4	10.9
F	17.8	6.0
IRL	10.9	14.0
I	16.6	8.5
L	21.0	3.5
NL	16.4	4.3
P	7.8	17.4
UK	14.0	2.3

Data set interni

Example, the famous data set Iris per l'analisi multivariata

data(iris)

iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
1	5.1	3.5	1.4	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa

To create objects with each column:

lengthS<-iris[,1]

widthS<-iris[,2]

nrow(iris)

ncol(iris)

Data sets disponibili in R

Data sets in package 'datasets':

AirPassengers	Monthly Airline Passenger Numbers 1949-1960
BJSales	Sales Data with Leading Indicator
BJSales.lead (BJSales)	Sales Data with Leading Indicator
BOD	Biochemical Oxygen Demand
CO2	Carbon Dioxide uptake in grass plants
ChickWeight	Weight versus age of chicks on different diets
DNase	Elisa assay of DNase
EuStockMarkets	Daily Closing Prices of Major European Stock Indices, 1991-1998
Formaldehyde	Determination of Formaldehyde
HairEyeColor	Hair and Eye Color of Statistics Students
Harman23.cor	Harman Example 2.3
Harman74.cor	Harman Example 7.4
Indometh	Pharmacokinetics of Indomethacin
InsectSprays	Effectiveness of Insect Sprays
JohnsonJohnson	Quarterly Earnings per Johnson & Johnson Share
LakeHuron	Level of Lake Huron 1875-1972
LifeCycleSavings	Intercountry Life-Cycle Savings Data
Loblolly	Growth of Loblolly pine trees
Nile	Flow of the River Nile
Orange	Growth of Orange Trees
OrchardSprays	Potency of Orchard Sprays
PlantGrowth	Results from an Experiment on Plant Growth
Puromycin	Reaction velocity of an enzymatic reaction
Seatbelts	Road Casualties in Great Britain 1969-84
Theoph	Pharmacokinetics of theophylline
Titanic	Survival of passengers on the Titanic
ToothGrowth	The Effect of Vitamin C on Tooth Growth in Guinea Pigs
UCBAdmissions	Student Admissions at UC Berkeley
UKDriverDeaths	Road Casualties in Great Britain 1969-84
UKgas	UK Quarterly Gas Consumption

Ogni pacchetto ha la propria dotazioe di data set interni

Per richiamare questi data sets bisogna inserirne il nome tra parentesi tonde precedute dal comando data:

Esempio:

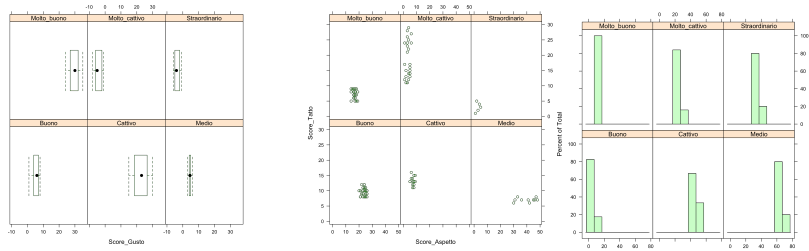
data(islands)

Per visualizzare I dati in esso contenuti basta scrivere

islands

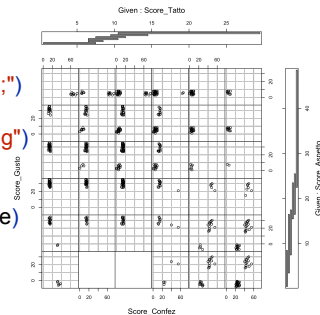
> islands	Africa	Antarctica	Asia	Australia
	11506	5500	16988	2968
Axel Heiberg		Baffin	Banks	Borneo
16		184	23	280
Britain		Celebes	Celon	Cuba
84		73	25	43
Devon		Ellesmere	Europe	Greenland
21		82	3745	840
Hainan		Hispaniola	Hokkaido	Honshu
13		30	30	89
Iceland		Ireland	Java	Kyushu
40		33	49	14
Luzon		Madagascar	Melville	Mindanao
42		227	16	36
Moluccas		New Britain	New Guinea	New Zealand (N)
29		15	306	44
New Zealand (S)		Newfoundland	North America	Novaya Zemlya
43			9390	32
Prince of Wales		Sakhalin	South America	Southampton
13		29	6795	16
Spitsbergen		Sumatra	Taiwan	Tasmania
15		183	14	26
Tierra del Fuego		Timor	Vancouver	Victoria
19		13	12	82

Grafici per dati raggruppati



Grafici per dati raggruppati

```
Base<-read.table(file="Tryprodotto.csv",header=T,sep=";")
names(Base);attach(Base)
library(lattice);lattice.options(default.theme = "col.whitebg")
bwplot(~Score_Gusto| Target, data = Base)
histogram(~Score_Confez| Target, data = Base)
xyplot(Score_Tatto ~ Score_Aspetto| Target, data = Base)
coplot(Score_Gusto ~ Score_Confez | Score_Tatto *
Score_Aspetto, data = Base)
```

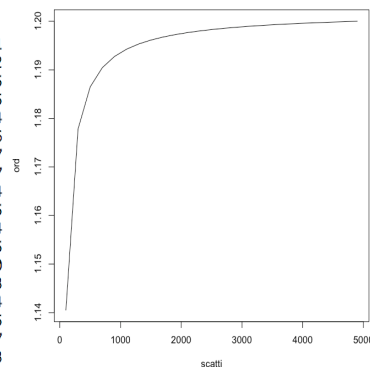


Ciclo di for e concetto di limite

```
scatti<-seq(100,5000,by=200)
conta<-0
m<-length(scatti)
ord<-matrix(0,m)
for (n in scatti)
{
  kappa <- 0;conta<-conta+1
  for (k in 2:n)
  { j <- 1:(k-1)
    kappa <- kappa + sum(1/j)/k^2
  }
  ord[conta]<-kappa
  cat(n, kappa, "\n")
}
plot(scatti,ord,type="l")
```

```
100 1.140516
300 1.177825
500 1.186488
700 1.190453
900 1.192751
1100 1.19426
1300 1.195331
1500 1.196132
1700 1.196755
1900 1.197255
2100 1.197664
2300 1.198006
2500 1.198297
2700 1.198547
2900 1.198764
3100 1.198955
3300 1.199124
3500 1.199275
3700 1.199410
3900 1.199533
4100 1.199644
4300 1.199745
4500 1.199837
4700 1.199923
4900 1.200001
```

$$\kappa = \sum_{k=2}^{\infty} \frac{1}{k^2} \sum_{j=1}^{k-1} \frac{1}{j} \quad (\approx 1.201)$$



Calcoli ripetuti: ciclo di for

Possiamo ripetere dei segmenti di calcolo senza dover ripetere la scrittura dei comandi.

Per visualizzare i numeri da uno a 10 ed i loro logaritmi naturali possiamo scrivere

```
for (i in 1:10) {cat(i,log(i),"\n")}
for (indice del ciclo)
{
  comandi
}
```

```
> for (i in 1:10) {cat(i,log(i),"\n")}
1 0
2 0.6931472
3 1.098612
4 1.386294
5 1.609438
6 1.791759
7 1.94591
8 2.079442
9 2.197225
10 2.302585
```

Ripeti i comandi tra la prima e l'ultima parentesi a graffe nel tanto che l'indice i scorre dal valore iniziale della sequenza al valore finale

Cicli annidati

I cicli possono essere annidati (uno interno all'altro, ma mai incrociati)

Un ciclo deve essere interamente interno ad un altro oppure avviarsi solo dopo che l'altro è finito

```
A=matrix(0,3,3);
for (row in 1:3)
{
  for (col in 1:3)
  {
    A[row,col]=row*col
  }
}
```

A;

Il risultato dovrebbe essere

```
[,1] [,2] [,3]
[1,] 1 2 3
[2,] 2 4 6
[3,] 3 6 9
```

Cicli annidati/2

```
p=rep(0,5)
for (init in c(1,5,9))
{
  p[1]=init;
  for (n in 2:5)
  {
    p[n]=2*p[n-1]
    cat(init,n,p[n],"\n");
  }
}
```

```
1 2 2
1 3 4
1 4 8
1 5 16
5 2 10
5 3 20
5 4 40
5 5 80
9 2 18
9 3 36
9 4 72
9 5 144
```

```
for (i in 1:10)
{
  i1<-i+1
  for (j in i1:15)
  {
    cat(letters[i],LETTERS[j],"\n")
  }
}
```

Funzioni

Alcune elaborazioni debbono essere ripetute con dati diversi o con parametri diversi.

E' possibile scrivere più volte le stesse istruzioni, ma non è necessario dato che si possono compattare le istruzioni da reiterare in un blocco che si attiva su chiamata.

Il segmento di programma che contiene le istruzioni parametrizzate è la **function**

```
W<-function(x,y,z){}
```



W è il nome con cui il blocco verrà richiamato



x, y e z sono gli argomenti attraverso i quali saranno veicolati i valori su cui il blocco dovrà operare



Tutte le istruzioni e gli oggetti comprese tra le due parentesi graffe fanno parte della funzione

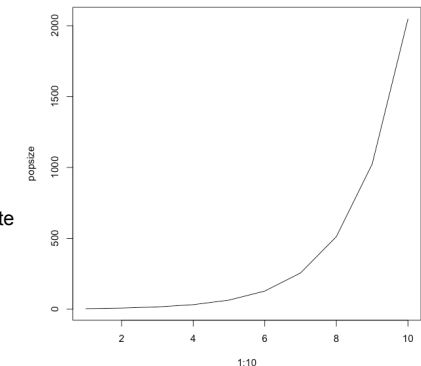
Esempio di ciclo di for

```
# Popolazione al livello iniziale
initsize=4;
# Creamo un vettore per memorizzare gli accrescimenti
popsize=rep(0,10); popsize[1]=initsize;
# Calcoliamo il livello della popolazione ai tempi da 2 e fino a 10
for (n in 2:10)
```

```
{
  popsize[n]=2*popsize[n-1];
  x=log(popsize[n]); cat(n,x,"\n");
}
plot(1:10,popsize,type="l");
```

```
2 2.079442
3 2.772589
4 3.465736
5 4.158883
6 4.85203
7 5.545177
8 6.238325
9 6.931472
10 7.624619
```

N.B. le ordinate sono in scala logaritmica

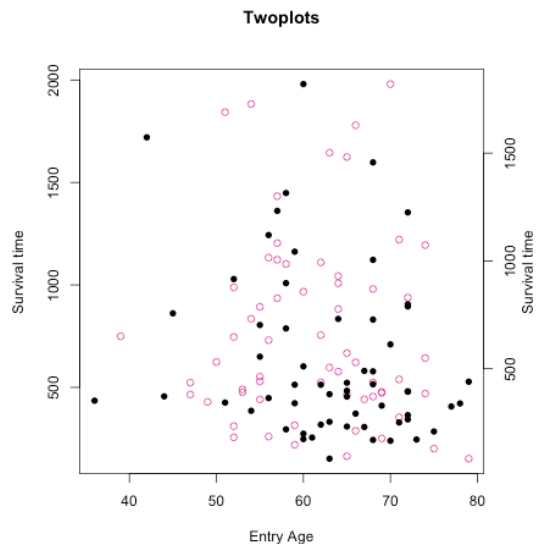


L'ipotesi è che la popolazione si raddoppi ogni anno

Esempio_1: funzioni

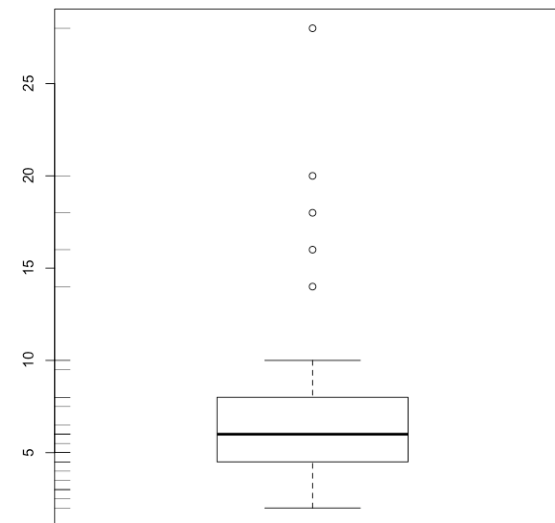
```
Twoplots<-
function(x1, y1, x2, y2, type = "l",xlim=NULL,ptitle="Twoplots",xlab
="x",ylab1="y1",ylab2 = "y2", ...) # Gli argomenti non debbono essere tutti
indicati. In mancanza, la routine usa i valori di default
{# Written by AE York Sept. 1996
# Plots y1 vs x1 and y2 vs x2 on the same plot
# Labels for y1 are on the left and labels for y2 on the right
  par(mar = rep(6, 4)) # margini del grafico
  if(missing(xlim))
    xlim <- range(pretty(range(c(x1, x2))))
  plot(x1, y1, lty = 1, pch = 1, type = type, xlim = xlim, xlab =
xlab, ylab = ylab1, ...)
  par(new = T)
  plot(x2, y2, lty = 2, pch = 16, type = type, axes = F, xlim = xlim,
xlab = "", ylab = "")
  axis(side = 4) # aggiunge l'asse a destra
  mtext(ylab2, side = 4, line = 2, outer = F)
  title(ptitle)}
Dat<-read.table("YJW.csv",sep=",".header=TRUE);attach(Dat);head(Dat)
J<-which(DummyAB==0) # Individuazione delle posizioni che hanno lo "0"
Asc1<-Entry.Age[J];Ord1<-Survtime[J]
Asc2<-Entry.Age[-J];Ord2<-Survtime[-J]
Twoplots(Asc1,Ord1,Asc2,Ord2,"p",xlim=c(min(min(Asc1,Asc2)),max(max(Asc1,Asc2)
)),xlab="Entry Age",ylab1="Survival time",ylab2="Survival time")
```

Esempio_1: funzioni/2



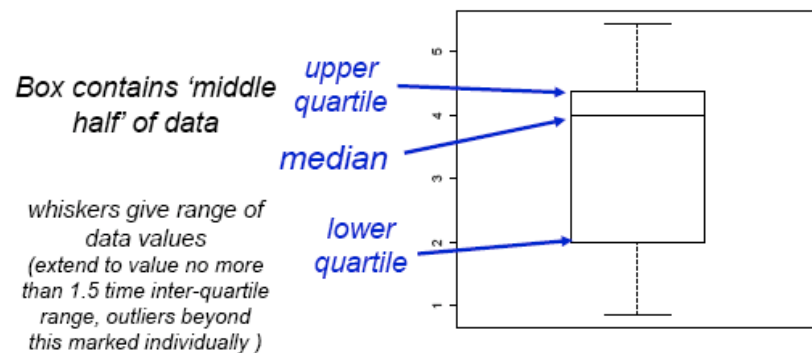
Grafica esplorativa/3

```
data(hills)
attach(hills)
summary(hills)
par(mfrow=c(1,1),
    mar=c(4,4,2,2))
boxplot(dist,sub="distan
ce",notch=T,varwidth =
TRUE)
boxplot(climb,sub="cum
ulative height")
boxplot(dist,sub="distan
ce")
rug(dist,side=2)
```



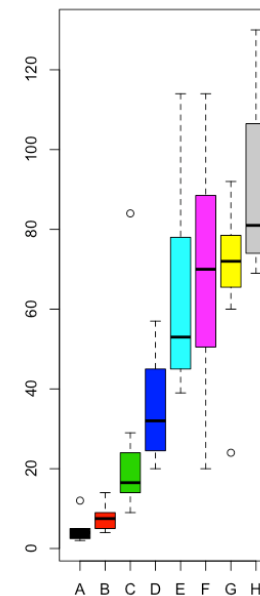
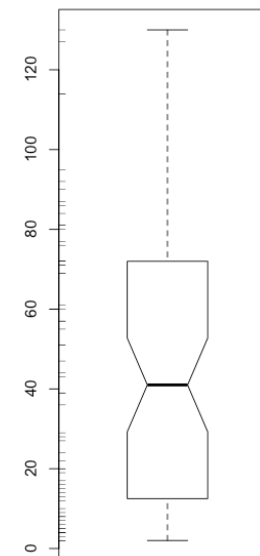
Boxplots

```
> library(MASS)
> attach(geyser)
> boxplot(duration,sub="duration")
```



Grafica esplorativa/4

```
data(OrchardSprays)
attach(OrchardSprays)
summary(OrchardSprays)
par(mfrow=c(1,2))
boxplot(decrease,
    sub="decrease in
counts",notch=T)
rug(decrease,side=2)
boxplot(decrease~treatment,
    col=1:20)
```



Esempio_1. Grafici ben fatti

Grafica di ottima qualità facilmente importabile in tesi, tesine e relazioni in qualsiasi formato e dimensione.

```
boxplot(faithful[,
1],horizontal=T,range=0,notch=TRUE,col="springgreen",
,pars =list(boxwex=0.5, staplewex=0.5,outwex = 0.5))
```

```
hist(faithful[,1],main="Istogramma dei valori
osservati",ylab="Frequenze
relative",xlab="Modalità",breaks="sturges",freq=FALS
E,right=TRUE,col="lightgreen")
```

Faithful è un dataset intrinseco ad R disponibile al semplice richiamo. E' riferito alla durata e gettito di un geyser dello Yellowstone Park (Wy-Usa)



```
set.seed(31852963) # Amelia' sequence
x <- rnorm(10)
y <- rnorm(10)
opar <- par()
par(bg="lightgray", mar=c(2.5, 1.5, 2.5, 0.25))
plot(x, y, type="n", xlab="", ylab="", xlim=c(-2, 2),ylim=c(-2, 2), xaxt="n", yaxt="n")
rect(-3, -3, 3, 3, col="cornsilk")
points(x, y, pch=10, col="red", cex=2)
axis(side=1, c(-2, 0, 2), tcl=-0.2, labels=FALSE)
axis(side=2, -1:1, tcl=-0.2, labels=FALSE)
title("Grafico personalizzato",font.main=4, adj=1, cex.main=1)
mtext("In basso a destra", side=1, line=1, at=1, cex=0.9, font=3)
mtext("In alto a sinistra", line=0.5, at=-1.8, cex=0.9, font=3)
mtext(c(-2, 0, 2), side=1, las=1, at=c(-2, 0, 2), line=0.3,col="blue", cex=0.9)
mtext(-1:1, side=2, las=1, at=-1:1, line=0.2, col="blue", cex=0.9)
par(opar)
```

Multiple Boxplot

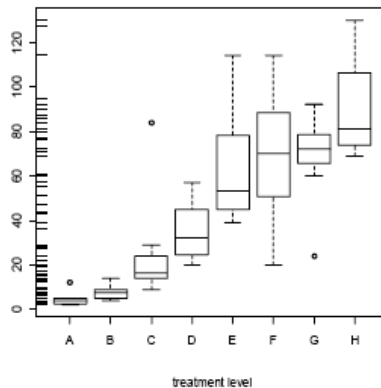
```
> boxplot(decrease~treatment)
> rug(decrease, side=2)
```

Effective for showing several data sets together

Counts increase with treatment level and variance also increases

`decrease~treatment`

"relate decrease to treatment"



Ortogrammi a colonna ed a barre

E' utile per variabili discrete ordinali e nominali

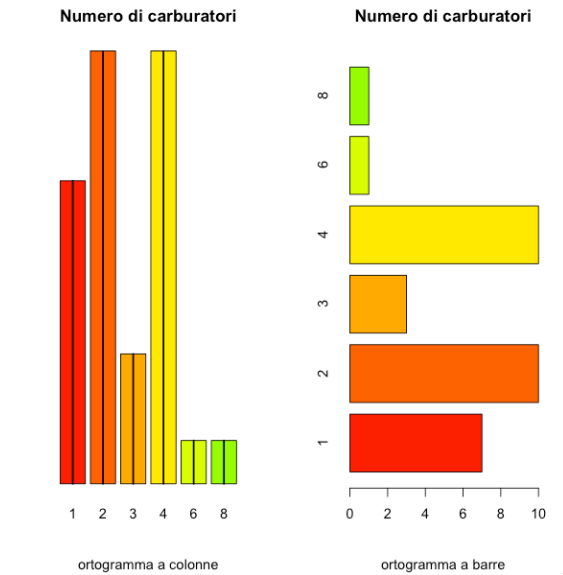
```
> data(mtcars) # carica il data set
> attach(mtcars) # integra il data set nello spazio di lavoro
> mtcars # Visualizza i dati
```

length() conta il numero di elementi del suo argomento

```
> table(cyl) # Frequenze assolute
> table(cyl)/length(cyl) # Frequenze relative
> barplot(table(cyl)/length(cyl)) # ortogramma
```

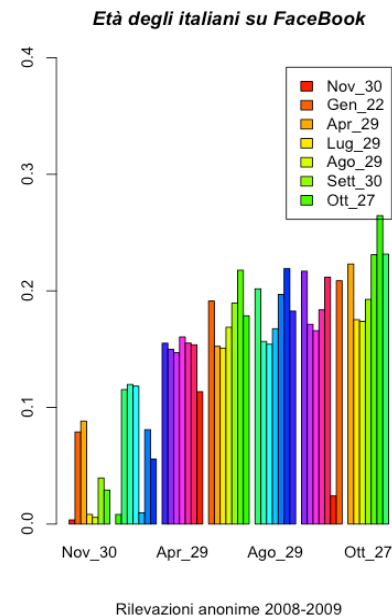
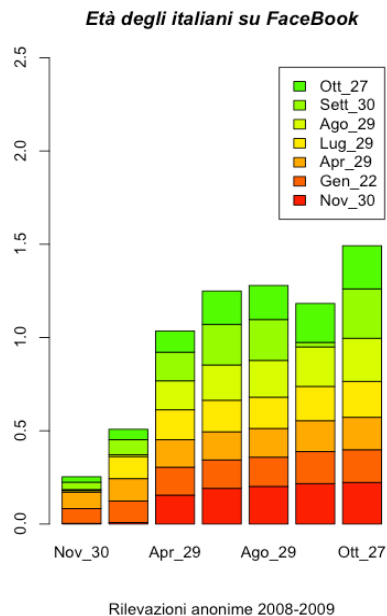
```
par(mfrow=c(1,2))
tN <- table(carb)
r <- barplot(tN, col=rainbow(20),horiz=FALSE,main = "Numero di carburatori",sub="ortogramma a colonne",axes=FALSE)
lines(r, tN, type='h', col='black', lwd=2)r <- barplot(tN, col=rainbow(20),horiz=TRUE,main = "Numero di carburatori",sub="ortogramma a barre")
```

Ortogrammi multipli



```
par(mfrow=c(1,2))
Etain<-
read.table("EtaFaceB.csv",sep=";",header=TRUE,row.names=1)
Etain<-as.matrix(Etain) # Riconoscimento come matrice
TotRig<-apply(Etain,1,sum) # Totali di riga
Etain<-diag(1/TotRig) %%% Etain # Divisione per il totale
#
mp <- barplot(Etain, beside = FALSE, col
=rainbow(20), legend = colnames(Etain), ylim=
c(0,2.5), main = "Età degli italiani su FaceBook",
font.main = 4, sub = "Rilevazioni anonime
2008-2009", cex.axis = par("cex.axis"), cex.names =
par("cex.axis"))
#
mp <- barplot(Etain, beside = TRUE, col
=rainbow(20), legend = colnames(Etain), ylim=
c(0,0.4), main = "Età degli italiani su FaceBook",
font.main = 4, sub = "Rilevazioni anonime
2008-2009", cex.axis = par("cex.axis"), cex.names =
par("cex.axis"))
```

Ortogrammi a punti



```
par(mfrow=c(1,1))
par(mar=c(4.2,4.8,0.5,1.0))
#c(5, 4, 4, 2)
x<-seq(3,12)
Ab<-
c("3", "4", "5", "6", "7", "8", "9", "10", "11", "12")
z<-rep(0,10)
f<-c(3,3,6,12,13,19,26,5,3,2)
f<-f/92
plot(x,f,frame.plot=T,
xlab="x",ylab=expression(f[n]
(x)))
points(x,f, bg = "limegreen", pch
= 21,cex=1.25)
f=f-0.005
segments(x, z, x,
f,lty="dotted",lwd=2)
```

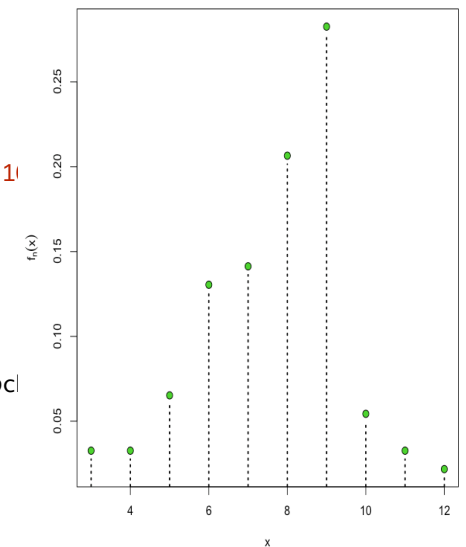
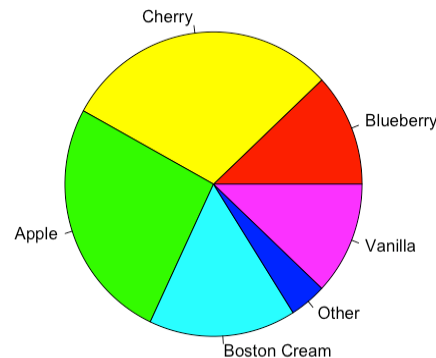


Diagramma a torta

```
par(mar=c(0, 2, 1, 2), xpd=FALSE, cex=0.5)
par(mfrow=c(1,1))
pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
names(pie.sales)<- c("Blueberry", "Cherry",
"Apple", "Boston Cream", "Other", "Vanilla")
pie(pie.sales, col = rainbow(6),cex=1.25)
```

Traferire il grafico in un editor di testo
(ad esempio Word)



Funzione di ripartizione

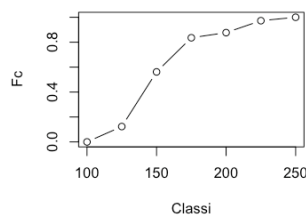
```
Fc<-cumsum(table(cut(WWWusage,breaks=c(100,125,150,175,200,225,250))))
```

Classi	(100,125]	(125,150]	(150,175]	(175,200]	(200,225]	(225,250]
Frequenze	9	41	61	64	71	73

```
> Fc<-c(0,Fc) # aggiunta del punto (0,0) per uniformare i punti
> Classi<-c(100,125,150,175,200,225,250)
> plot(Classi,Fc,type="b",main="Ogiva delle frequenze assolute")
```

```
> Fc<-Fc/73
> Fc<-c(0,Fc)
plot(Classi,Fc,type="b",main="Ogiva delle frequenze relative")
```

Ogiva delle frequenze relative



Ogiva delle frequenze assolute

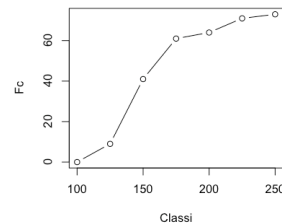
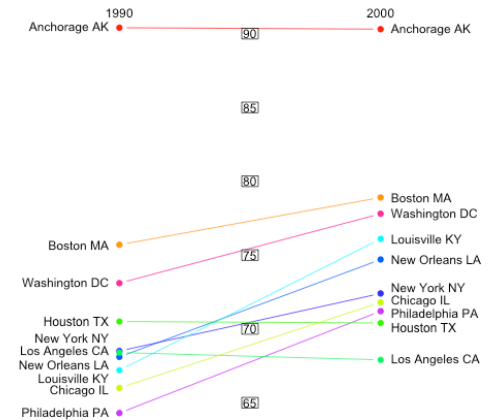


Diagramma barometrico

```
library(plotrix)
# percentage of those over 25 years having completed high school# in 10 cities in the USA in 1990 and 2000
educatn<-matrix(c(90.4,90.3,75.7,78.9,66.71,8.70,5.70,4.68,4.67,9.67,2.76,1.68,1.74,7.68,5.72,4.64,3.71,2.73,1.77,8),ncol=2,byrow=TRUE)
rownames(educatn)<-c("Anchorage AK","Boston MA","Chicago IL","Houston TX","Los Angeles CA","Louisville KY","New Orleans LA","New York NY","Philadelphia PA","Washington DC")
colnames(educatn)<-c(1990,2000)
bumpchart(educatn,main="Rank for high school completion by over 25s")
# now show the raw percentages and add central ticks
bumpchart(educatn,rank=FALSE,main="Percentage high school completion by over 25s",col=rainbow(10))
# margins have been reset, so use
par(xpd=TRUE)
boxed.labels(1.5,seq(65,90,by=5),seq(65,90,by=5))
par(xpd=FALSE)
```

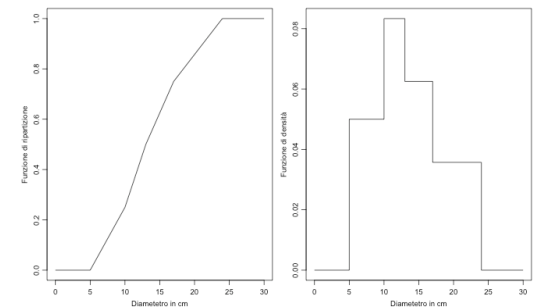
Percentage high school completion by over 25s



E' una tecnica di rappresentazione essenziale ed efficace, ma di portata limitata può essere invocata se si dispone di due blocchi di informazioni sulle medesime unità in due circostanze funzionalmente legate.

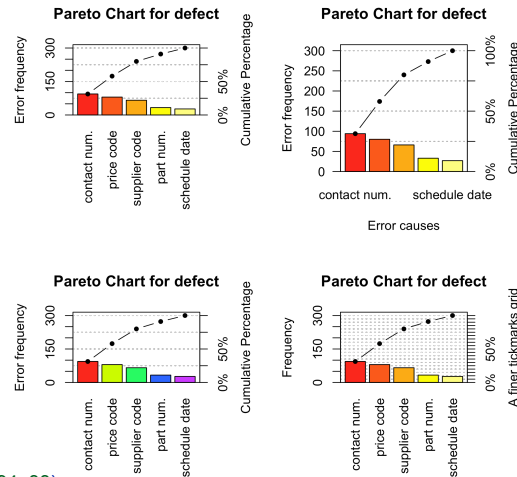
Funzione di ripartizione/2

$$F(x) = \begin{cases} 0 & x < 5 \\ -0.25 + 0.050x & 5 \leq x < 10 \\ -0.58 + 0.083x & 10 \leq x < 13 \\ -0.31 + 0.063x & 13 \leq x < 17 \\ 0.14 + 0.036x & 17 \leq x < 24 \\ 1 & x \geq 24 \end{cases}$$



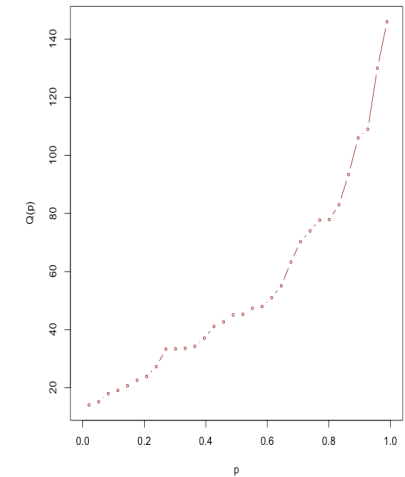
```
par(mfcol=c(1,2),mai=c(0.6,0.5,0.1,0.1),mfp=c(2,0.7,0),cex=0.8)
d<-c(5,10,13,17,24)
p<-c(0,0.25,0.5,0.75,1)
a<-(p[-1]-p[-5])/(d[-1]-d[-5])
b<-p[-5]-a*d[-5]
plot(c(0,d,30),c(0,p,1),type="l",ylab="Funzione di ripartizione",
xlab="Diametro in cm")
plot(c(0,rep(d,each=2),30),c(0,0,rep(a,each=2),0,0),type="l",
ylab="Funzione di densità", xlab="Diametro in cm")
```

Ortogramma Paretiano



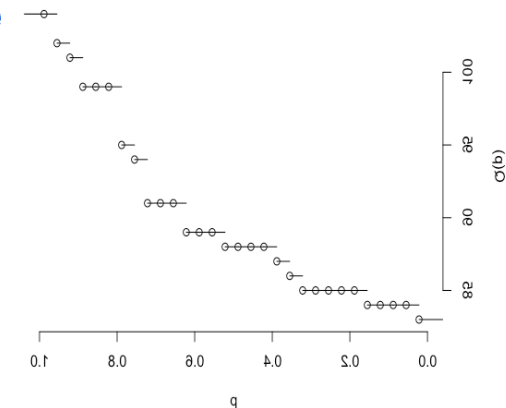
```
library(qcc)defect <- c(80, 27, 66, 94, 33)
names(defect) <- c("price code", "schedule date", "supplier code", "contact num.", "part num.")
par(mfrow = c(2,2))
pareto.chart(defect, ylab = "Error frequency")
pareto.chart(defect, ylab = "Error frequency", xlab = "Error causes", las=1)
pareto.chart(defect, ylab = "Error frequency", col=rainbow(length(defect)))
pareto.chart(defect, cumperc = seq(0, 100, by = 5), ylab2 = "A finer tickmarks grid")
```

Funzione Quantile



```
par(mar=c(4.5,4.5,1,2))
River<-read.table("Ghosh.csv",sep="," ,dec=".",header=T)
attach(River)
plot(River,main=NULL,xlab="p",ylab="Q(p)",xlim=c(0,1),type
="b",cex =0.5,col="dark red")
```

Funzione Quantile discreta



```
par(mar=c(4.0,4.5,1,2))
B<-WWWusage
B<-sort(B)
n<-30
n1<-n+1
p<-seq(1:n)
p<-(p-0.35)/n
y<-matrix(0,n1)
for(i in 1:n)
{y[i]<-B[i]}
y[n1]=B[n]
Gradf<-stepfun(p,y,f=1,right=T)
plot(Gradf,verticals=F,main=NULL,
xlab="p",ylab="Q(p)",xlim=c(0,1),
frame.plot=F)
```

Uso dei quantili

Sono delle statistiche robuste che guardano ai dati nel piano (p, X_p) e non in quello $[x, F(x)]$. Offrono interessanti spunti di analisi

```
> # Quantili
> # Decili
> QD<-quantile(Quotazione, probs = seq(0, 1, 0.1),names = TRUE, type = 5)
> QD
 0%    10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
86.00  93.40  94.95  96.50  97.95  98.30  98.65 101.70 103.80 105.40 109.00
> # Quartili
> QD<-quantile(Quotazione, probs = seq(0, 1, 0.25),names = TRUE, type = 5)
> QD
 0%    25%   50%   75%  100%
86.000 95.375 98.300 102.375 109.000
> # Quintili
> QD<-quantile(Quotazione, probs = seq(0, 1, 0.20),names = TRUE, type = 5)
> QD
 0%    20%   40%   60%   80%  100%
86.00  94.95  97.95  98.65 103.80 109.00
> # Ventili
> QD<-quantile(Quotazione, probs = seq(0, 1, 0.05),names = TRUE, type = 5)
> QD
 0%    5%   10%   15%   20%   25%   30%   35%   40%   45%   50%   55%
86.000 90.050 93.400 94.525 94.950 95.375 96.500 97.375 97.950 98.225 98.300 98.450
 60%   65%   70%   75%   80%   85%   90%   95%  100%
98.650 100.575 101.700 102.375 103.800 105.075 105.400 106.825 109.000
```

Formula per i quantili (type=)

Continuous sample quantile types 4 through 9

Type 4

$p(k) = k / n$. That is, linear interpolation of the empirical cdf.

Type 5

$p(k) = (k - 0.5) / n$. That is a piecewise linear function where the knots are the values midway through the steps of the empirical cdf. This is popular amongst hydrologists.

Type 6

$p(k) = k / (n + 1)$. Thus $p(k) = E[F(x[k])]$. This is used by Minitab and by SPSS.

Type 7

$p(k) = (k - 1) / (n - 1)$. In this case, $p(k) = \text{mode}[F(x[k])]$. This is used by S.

Type 8

$p(k) = (k - 1/3) / (n + 1/3)$. Then $p(k) \approx \text{median}[F(x[k])]$. The resulting quantile estimates are approximately median-unbiased regardless of the distribution of x .

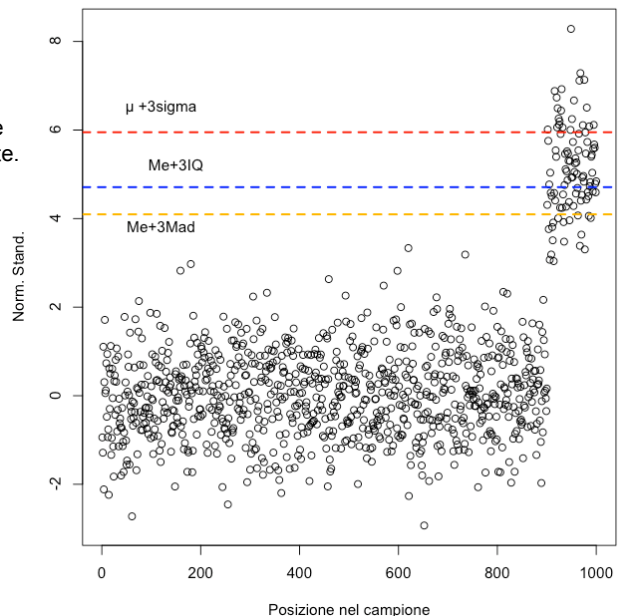
Type 9

$p(k) = (k - 3/8) / (n + 1/4)$. The resulting quantile estimates are approximately unbiased for the expected order statistics if x is normally distributed.

Hyndman and Fan (1996) recommend type 8. The default method is type 7, as used by S and by R < 2.0.0.

Esempio

Sono indicate tre diverse soglie più o meno robuste. La scelta è soggettiva



Individuazione dei valori anomali

```
set.seed(3141593)
n1=900;n2<-100;n<-n1+n2
x1<-rnorm(n1, mean=0, sd=1);
x2<-rnorm(n2, mean=5, sd=1)
x<-c(x1,x2)
plot(x,ylab="Norm. Stand.",xlab="Posizione nel campione")
std<-mean(x)+3*sd(x)
abline(h=std,lwd=2,lty=2,col="red")
text(120,6.5,"μ +3sigma")
k1=floor(n/4)+1;k2=n-k1+1
a<-rank(x); q1<-which(a==k1);q2<-which(a==k2)
cat(q1,q2,"\n")
a1<-x[q1];a2<-x[q2]
cat(k1,a1,k2,a2,"\n")
std<-median(x)+3*(a2-a1)
abline(h=std,lwd=2,lty=2,col="blue")
text(150,5.2,"Me+3IQ")
std<-median(x)+3.5*mad(x, center = median(x), constant = 1.4826,
na.rm = FALSE, low = FALSE, high = FALSE)
abline(h=std,lwd=2,lty=2,col="orange")
text(120,3.8,"Me+3Mad")
```

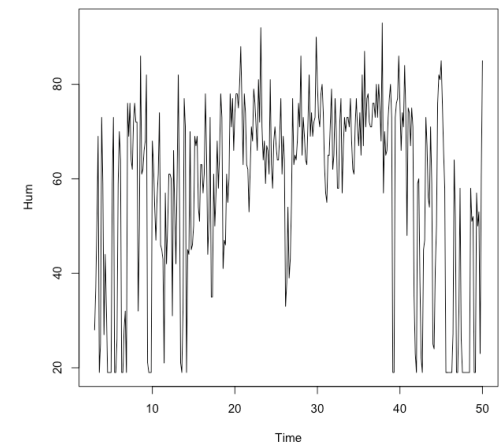
Grafico di una serie storica

Si adora il comando plot che riconosce e si adatta all'oggetto una volta che questo sia stato configurato come una serie storica.

```
Hum<-ts(LAozone$humidity,start=c(3,1),end=c(50,1),frequency=7)
plot(Hum)
```

start ed **end** indicano periodo e subperiodo di inizio e di fine del flusso di dati.

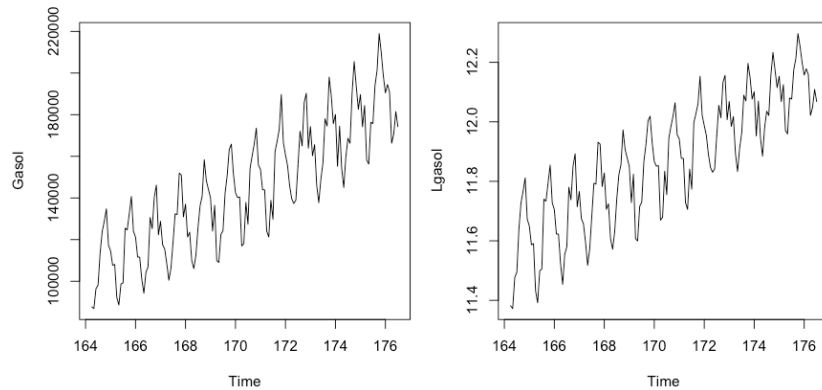
frequency indica la periodicità



Esempio

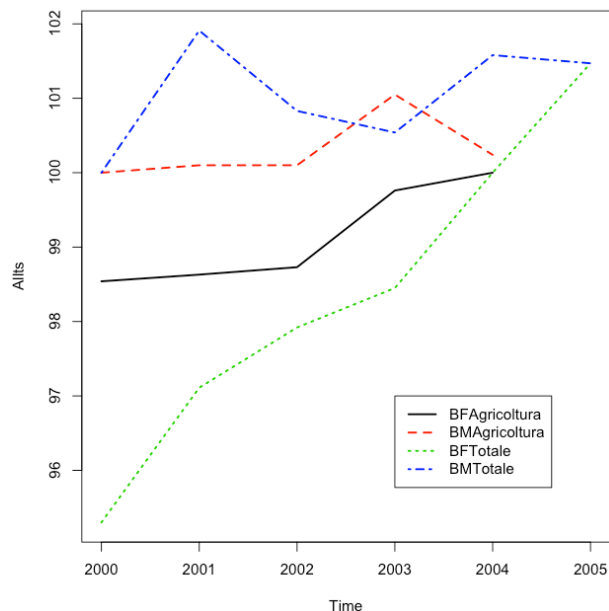
Monthly gasoline demand Ontario gallon millions 1960 - 1975.

```
Gasol <-scan("Gasol.txt",sep=" ")
Gasol<-ts(Gasol,start=c(1,1960),end=c(12,1975),frequency=12)
Lgasol<-log(Gasol)
par(mfrow=c(1,2));par(mar=c(5,4,2.0,1))
plot(Gasol);plot(Lgasol)
```



Esempio

Perché il totale ha un andamento diverso dalle sue componenti?



Grafici di serie storiche multiple

```
Acs<-c(2090,2092,2094,2116,2121)
Tot<-c(12903,13149,13258,13330,13540,13739)
Acs<-ts(Acs,start=2000,frequency=1)
Tot<-ts(Tot,start=2000,frequency=1)
plot(Acs,type="b",frame.plot=FALSE,col="gold4")
text(2001,2110,label="Imprenditrici per attività")
text(2001,2108,label="economica")
BFB04Agr<-round(Acs*100/Acs[5],2)
BFB04Tot<-round(Tot*100/Tot[5],2)
Temp<-c(Acs[1],lag(Acs))
Temp<-ts(Temp,start=2000,frequency=1)
BMAgr<-round(Acs*100/Temp,2)
BMAgr<-ts(BMAgr,start=2000,frequency=1)
Temp<-c(Tot[1],lag(Tot))
Temp<-ts(Temp,start=2000,frequency=1)
BMTot<-round(Tot*100/Temp,2);BMTot<-ts(BMTot,start=2000,frequency=1)
Allts<-cbind(BFB04Agr,BMAgr,BFB04Tot,BMTot);plot(Allts)
plot(Allts,plot.type="single",lwd=2,ty=1:4,col=1:4)
legend(2003,97,legend=c("BFAgricoltura","BMAgricoltura","BFTotale","BMTotale"),
col=1:4,lwd=2,ty=1:4)
```

Uso del comando matplot

Monthly CO2-Concentration from 1960 – 1997

Lettura in 5 distinti vettori delle serie storiche mensili di vari anni

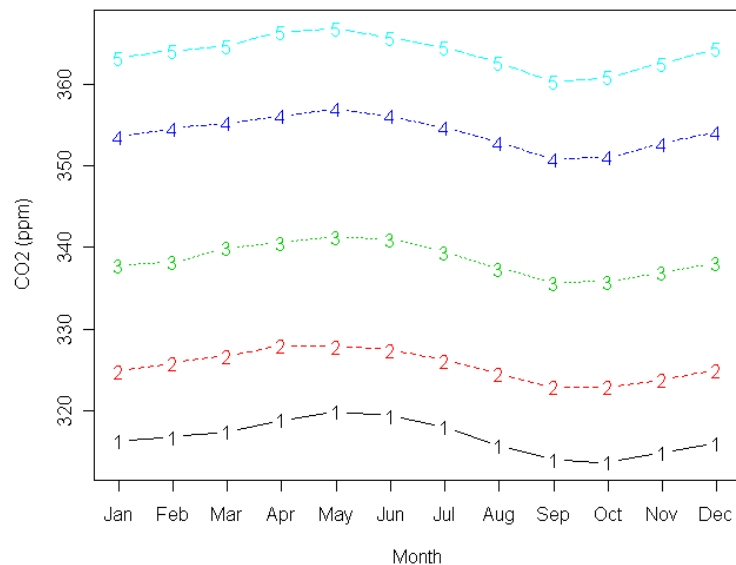
```
> y60<-c(316.27, 316.81, 317.42, 318.87, 319.87, 319.43, 318.01, 315.74, 314.00, 313.68, 314.84, 316.03)
> y70<-c(324.89, 325.82, 326.77, 327.97, 327.91, 327.50, 326.18, 324.53, 322.93, 322.90, 323.85, 324.96)
> y80<-c(337.84, 338.19, 339.91, 340.60, 341.29, 341.00, 339.39, 337.43, 335.72, 335.84, 336.93, 338.04)
> y90<-c(353.50, 354.55, 355.23, 356.04, 357.00, 356.07, 354.67, 352.76, 350.82, 351.04, 352.69, 354.07)
> y97<-c(363.23, 364.06, 364.61, 366.40, 366.84, 365.68, 364.52, 362.57, 360.24, 360.83, 362.49, 364.34)
> CO2<-data.frame(y60, y70, y80, y90, y97)
> row.names(CO2)<-c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
> CO2
```

```
CO2<-read.table("CO2m.csv",header=TRUE, sep=".",dec=".")
```

Rappresentazione grafica multipla con aggiunta del titolo

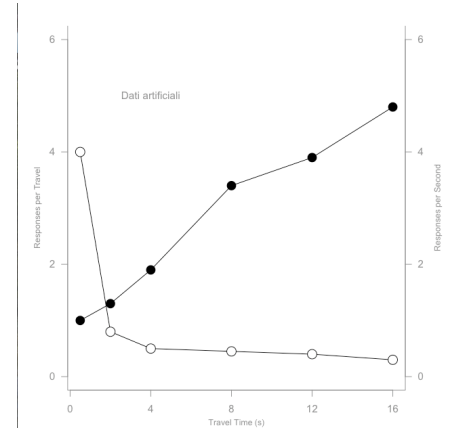
```
> matplot(CO2,axes=F,frame=T,type='b',ylab="")
> help("matplot")
> axis(2)
> axis(1, 1:12, row.names(CO2))
> title(xlab="Month")
> title(ylab="CO2 (ppm)")
> title("Monthly CO2 Concentration \n for 1960, 1970, 1980, 1990 and 1997")
```

**Monthly CO2 Concentration
for 1960, 1970, 1980, 1990 and 1997**



Serie storiche con doppia scala

```
x <- c(0.5, 2, 4, 8, 12, 16)
y1 <- c(1, 1.3, 1.9, 3.4, 3.9, 4.8)
y2 <- c(4, .8, .5, .45, .4, .3)
par(las=1, mar=c(4, 4, 2, 4))
plot.new()
plot.window(range(x), c(0, 6))
lines(x, y1)
lines(x, y2)
points(x, y1, pch=16, cex=2)
points(x, y2, pch=21, bg="white", cex=2)
par(col="grey50", fg="grey50",
col.axis="grey50")
axis(1, at=seq(0, 16, 4))
axis(2, at=seq(0, 6, 2))
axis(4, at=seq(0, 6, 2))
box(bty="u")
mtext("Travel Time (s)", side=1, line=2, cex=0.8)
mtext("Responses per Travel", side=2, line=2, las=0, cex=0.8)
mtext("Responses per Second", side=4, line=2, las=0, cex=0.8)
text(4, 5, "Dati artificiali")
par(mar=c(5.1, 4.1, 4.1, 2.1), col="black", fg="black", col.axis="black")
```



Serie storiche con doppia scala

```
x <- c(0.5, 2, 4, 8, 12, 16)
y1 <- c(1, 1.3, 1.9, 3.4, 3.9, 4.8)
y2 <- c(4, .8, .5, .45, .4, .3)
par(las=1, mar=c(4, 4, 2, 4))
plot.new()
plot.window(range(x), c(0, 6))
lines(x, y1)
lines(x, y2)
points(x, y1, pch=16, cex=2)
points(x, y2, pch=21, bg="white", cex=2)
par(col="grey50", fg="grey50",
col.axis="grey50")
axis(1, at=seq(0, 16, 4))
axis(2, at=seq(0, 6, 2))
axis(4, at=seq(0, 6, 2))
box(bty="u")
mtext("Travel Time (s)", side=1, line=2, cex=0.8)
mtext("Responses per Travel", side=2, line=2, las=0, cex=0.8)
mtext("Responses per Second", side=4, line=2, las=0, cex=0.8)
text(4, 5, "Dati artificiali")
par(mar=c(5.1, 4.1, 4.1, 2.1), col="black", fg="black", col.axis="black")
```

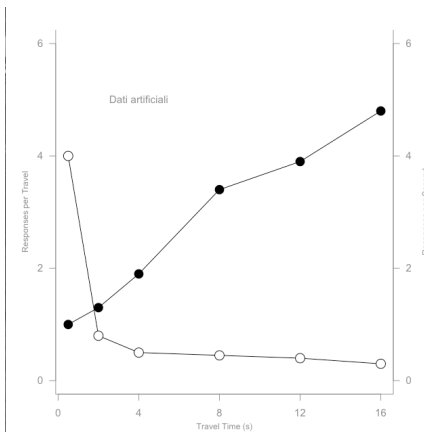
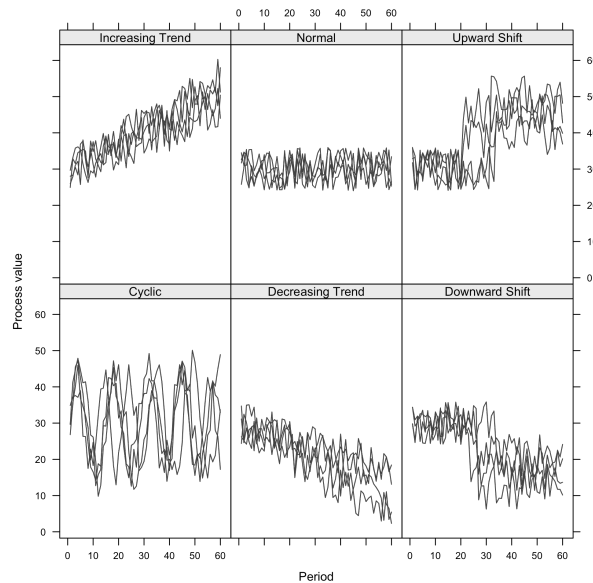


Grafico con i pacchetti lattice e latticeExtra

```
library(latticeExtra)
#
Hawk<-function(kclus,p,size,Typek,Aldat)
{Matr<-matrix(0,p,60);jpos<-matrix(0,kclus,size)
for (i in 1:kclus) {m<-Typek[i];S1<-1+(i-1)*100;S2<-i*100;Pop<-S1:S2
jpos[i,]<-sample(Pop,size,replace=FALSE)
if (i==1) kpos<-jpos[1,] else kpos<-c(kpos,jpos[i,]) }
Matr<-Aldat[kpos,];return(Matr)}
#
Titolo<-"Synthetic Control Chart Time Series"
Aldat<-read.table(file="Alcock.csv",sep=";",dec=".",header=TRUE,row.names=1)
set.seed(31852963);p<-24;kclus<-6;Typek<-1:6; Nper<-60;s<-p/kclus; Scaling<-FALSE
Az<-Hawk(kclus,p,s,Typek,Aldat)
rownames(Az)<-rownames(Az, do.NULL = FALSE, prefix = "Un.");Ay<-t(Az)
Group<-c(rep("Normal",60),rep("Cyclic",60),rep("Increasing Trend",60),rep("Decreasing Trend",
60),rep("Upward Shift",60),rep("Downward Shift",60))
Cluster<-factor(Group,levels=c("Normal","Cyclic","Increasing Trend","Decreasing Trend","Upward
Shift","Downward Shift"))
Rov<-1:60;Period<-rep(Rov,6)
costa<-rbind(Ay[,1:4],Ay[,5:8],Ay[,9:12],Ay[,13:16],Ay[,17:20],Ay[,21:24])
Datam<-cbind(costa,Period,Cluster)
colnames(Datam)<-c("PTS_01","PTS_02","PTS_03","PTS_04","Period","Cluster")
rownames(Datam)<-1:360;Datam<-data.frame(Datam)
pl <- xyplot(PTS_01+PTS_02+PTS_03+PTS_04~Period|Group,ylab="Process value",
data=Datam,type="l",lty=1,col="black",col.line="gray20", par.settings=list(fontsize=list(text=10,
points=8),strip.background=list(col="grey90")));print(pl)
```

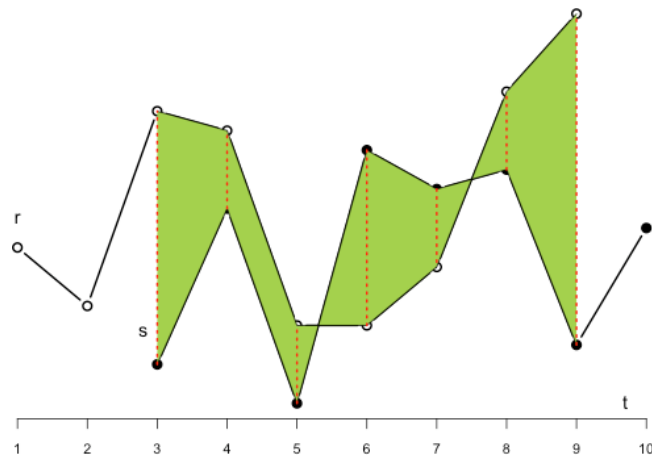
Poligonal

Provate a modificare
la font ed i colori



```
library(ggplot2)
x<-c(NA,NA,3,11,1,14,12,13,4,10);x<-ts(x);y<-c(9,6,16,15,5,5,8,17,21,NA)
y<-ts(y);ya<-y[3:9];xa<-x[3:9]
par(mfrow=c(1,1),pty="m")
two<-cbind(x,y)
matplot(two,type="b",lwd=2,lty=1,col="black",pch=c(19,21),cex=1.25,frame.plot=
FALSE,ylab="",axes=FALSE)
axis(1,at=1:10,cex.axis=0.9)
text(9.7,1.1,"t",cex=1.2);text(1,10.5,"r",cex=1.2);text(2.8,4.7,"s",cex=1.2)
cross <- data.frame(x1=3:9, y1 = xa, y2 = ya);attach(cross)
#
ymin <- min(y1,y2) #vertices for area under y1
mat1 <- cbind(c(x1[1], x1, x1[length(x1)]), c(ymin, y1, ymin))
#vertices for area under y2
mat2 <- cbind(c(x1[1], x1, x1[length(x1)]), c(ymin, y2, ymin))
#create polygons from vertices
pp1 <- as(mat1, "ggplot")
pp2 <- as(mat2, "ggplot")
plot(setdiff(pp1,pp2), poly.args=list(col="yellowgreen", border=NA), add=TRUE)
plot(setdiff(pp2,pp1), poly.args=list(col="yellowgreen", border=NA), add=TRUE)
lines(x1, y1, col="black",lwd=1); lines(x1, y2, col="black",lwd=1)
# Vertical lines
for (i in 1:10) {arrows(i,x[i],i,y[i],length=0,lty=3,lwd=2,col="red")}
```

Applicazione della poligonale



Il pacchetto ggplot2 offre molte altre possibilità per operare con le poligonali

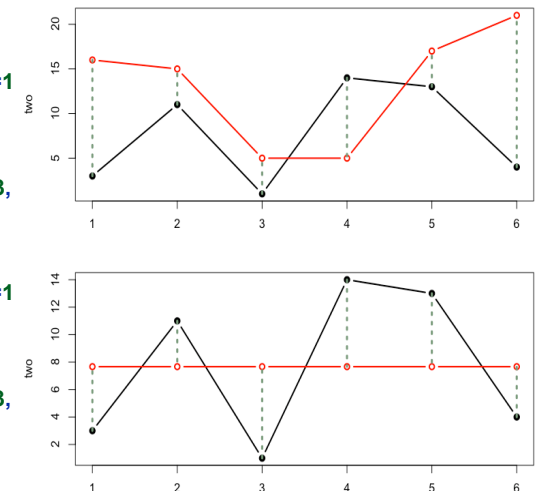
Variabilità come distanza

Misura della variabilità=Somma degli spostamenti
necessari per sovrapporre una serie all'altra

```
x<-c(3,11,1,14,13,4);x<-ts(x);
y<-c(16,15,5,5,17,21);y<-ts(y);
par(mfrow=c(2,1),pty="m")
two<-cbind(x,y)
matplot(two,type="b",lwd=2,lty=1,
,col=1:2,pch=c(19,21))
for (i in 1:6)

{arrows(i,x[i],i,y[i],length=0,lty=3,
lwd=3,col="darkseagreen4" )}
y<-rep(mean(x),6)
two<-cbind(x,y)
matplot(two,type="b",lwd=2,lty=1,
,col=1:2,pch=c(19,21))
for (i in 1:6)

{arrows(i,x[i],i,y[i],length=0,lty=3,
lwd=3,col="darkseagreen4" )}
```



Misure di variabilità

Dal confronto di due serie variabili non emerge una idea chiara della variabilità: se anche fossero sovrapposte potrebbero ancora essere tutte e due molto variabili.

Ecco perché nel secondo grafico si è eliminata una fonte della variabilità: una serie è diventata costante.

Se in una serie la variabilità è assente, la variabilità che comunque emerge è sicuramente attribuibile all'altra.

La tendenza ad assumere valori diversi dalla costante ovvero la somma delle distanze da percorrere per azzerare le differenze corrisponde all'idea di variabilità formulata da C.E. Bonferroni fin dagli anni '30 del secolo scorso.



Esempio

```
# Dati relativi ai tempi di impiego di scavatrici e camion nei cantieri
# di un'impresa di costruzioni
library(MASS)
dataf<-read.table("Aboudataset.csv",sep=";",dec=".",header=TRUE,na.strings =
"NA")
par(mfrow=c(1,2))
hist(dataf$Scavatrice,main="Istogramma dati scavatrici",ylab="Frequenze
relative",xlab="Modalità",breaks="sturges",freq=FALSE,right=TRUE,col="olivedra
b4",cex=0.60,)
hist(dataf$Camion,main="Istogramma dati camion",ylab="Frequenze
relative",xlab="Modalità",breaks="sturges",freq=FALSE,right=TRUE,col="plum4",c
ex=0.60)
dev.st<-c(sd(dataf$Scavatrice,na.rm=TRUE),sd(dataf$Camion,na.rm=TRUE))
dev.ab1<-sum(abs(dataf$Scavatrice-median(dataf
$Scavatrice,na.rm=TRUE)),na.rm=TRUE)
dev.ab2<-sum(abs(dataf$Camion-median(dataf$Camion,na.rm=TRUE)),na.rm=TRUE)
dev.ab<-c(dev.ab1,dev.ab2)
varm<-cbind(dev.st,dev.ab)
row.names(varm)<-c("Scavatrici","Camion")
print(varm)
```

Indici di variabilità

La misura della variabilità si avvale dell'idea di distanza: in particolare della metrica di Minkowski

$$Minkowski: S_{\alpha} = \left[\sum_{i=1}^n f_i |x_i - \theta_{\alpha}|^{\alpha} \right]^{\frac{1}{\alpha}}$$

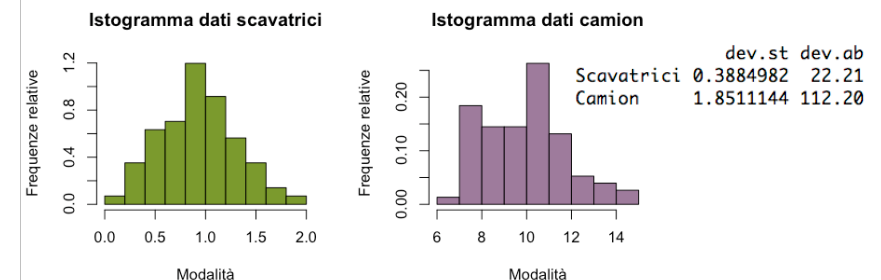
Dove f_i è la frequenza relativa della modalità x_i e θ_{α} è il valore costante di riferimento che dipende dal tipo di metrica α . In particolare, è il valore che rende minima la somma delle distanze dalle modalità.

Se $\alpha=2$ allora si ottiene la metrica euclidea e θ_{α} è la media aritmetica
Se $\alpha=1$ allora si ottiene la metrica city-block e θ_{α} è la mediana

Ad ogni scelta di $\alpha \geq 1$ corrisponde una diversa misura di variabilità

Se $\alpha < 1$ allora S_{α} non è una metrica

Indici di variabilità/2



La variabilità dei dati si avverte considerando l'ampiezza dei fianchi dell'istogramma, nonché l'allungamento verso i valori grandi.

Sebbene la forma degli istogrammi sia diversa, la variabilità appare dello stesso ordine di grandezza.

Tuttavia le misure di variabilità sembrano dare indicazioni diverse, nel senso che quella delle scavatrici è molto più bassa di quella dei camion. Perché?

Misure di variabilità relativa

Sono indici che permettono il confronto della variabilità tra variabili espresse

- Con ordini grandezza ineguali
- Per campi di variazioni diversi
- In unità di misura eterogenee

$$\text{misura di variabilità relativa} = \frac{\text{misura di variabilità assoluta}}{\text{media}}$$

La media al denominatore deve essere positiva o in valore assoluto.

Coefficiente di Variazione

$$CV = \frac{\sigma}{|\mu|} = \sqrt{\sum_{i=1}^k \left(\frac{X_i - \mu_x}{\mu_x} \right)^2}$$

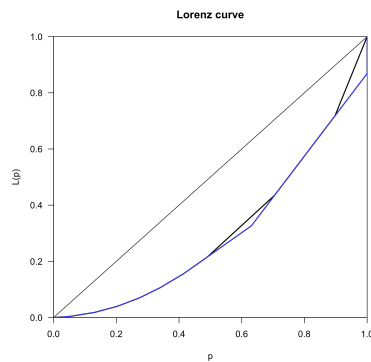
Indice di Pietra-Ricci

$$PR = \sum_{i=1}^k \left| \frac{X_i - \mu_x}{\mu_x} \right|$$

```
LA<-read.table("LAmer.dat", header=TRUE, row.names=1)
# Dimensioni della matrice dei dati
d<-dim(LA);print(d) # d E' un vettore di due elementi.
n<-d[1];m<-d[2] # numero di righe e di colonne
cat("Num.Paesi=",n,"Num.Variabili=",m,"\n") #visualizza le
assegnazioni
# la stringa: "\n" serve a cambiare linea alla prossima
scrittura.
print(Birth_R.) # Non funziona. Perché?
print(LA$Birth_R.)
Tasso.Nat<-LA[,1];print(Tasso.Nat) # Variabili con nomi
nuovi
summary(Tasso.Nat) # Riassunto numerico di una variabile
summary(LA) # Riassunto numerico di una matrice di dati
Paese.medio<- MedieLA.new<-rbind(LA,Paese.medio) # Crea una
unità
Temp<-c(rownames(LA),"Paese.medio");rownames(LA.new)<-
Temp;LA.new
# Alcune statistiche di uso frequente
Medie<-mean(LA);Dev.St<-sd(LA); Medie;Dev.St
# Variabilità relativa
CV<-Dev.St/abs(Medie);CV # Senza controllo errori
# individuazione delle variabili con eventuale media nulla.
J<-which(Medie !=0);CV<-Dev.St[J]/abs(Medie[J]);CV
PR<-matrix(0,3)for (i in 1:3){PR[i]<-sum(abs(LA[,i]-
mean(LA[,i])))/abs(mean(LA[,i]))}
cat("Indice di Pietra-Ricci:",PR,"\n")
Dev.Med<-matrix(0,3)
Dev.Med[1]<-mad(LA[,1]);Dev.Med[2]<-
mad(LA[,2]);Dev.Med[3]<-mad(LA[,3])
cat("Deviazione mediana dalla mediana:",Dev.Med,"\n")
```

Esempio

Curva di Lorenz

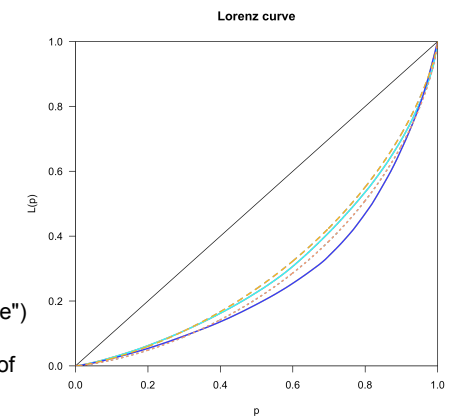


```
library(ineq)
# Distribuzione redditi USA 1968 (in 10 classi)
# x reddito medio nelle classi, f redditieri
x <- c(541, 1463, 2445, 3438, 4437, 5401,
6392, 8304, 11904, 22261)
f <- c(482, 825, 722, 690, 661, 760, 745,
2140, 1911, 1024)
# Curva di Lorenz minima (conc.nulla nelle
classi)
Lc.min <- Lc(x, n=f)
# Curva di Lorenz massima (limiti di Mehran)
Lc.max <- Lc.mehran(x,f)
# Le due curve nello stesso grafico
plot(Lc.min)
lines(Lc.max, col=4)
```

```
> # Calcola il rapporto di concentrazione
> ineq(x)
[1] 0.4620911
> # Calcola il coefficiente di Atkinson con parametro 0.5
> ineq(x, parameter=0.5, type="Atkinson")
[1] 0.1796591
```

```
## Load and attach income (and metadata)
set from Ilocos, Philippinesdata(Ilocos)
attach(Ilocos)
## extract and rescale income for the
provinces "Pangasinan" and "La Union"
income.p <-
income[province=="Pangasinan"]/10000
income.u <- income[province=="La
Union"]/10000
## compute the Lorenz curves
Lc.p <- Lc(income.p)
Lc.u <- Lc(income.u)
## plot both Lorenz curves
plot(Lc.p,col="cyan");lines(Lc.u, col="blue")
# add the theoretic Lorenz curve of a
Lognormal-distribution with an estimate of
the standard
## deviation parameter
lines(Lc.lognorm,
parameter=sd(log(income.p)),
col="orange",lty=2)
lines(Lc.lognorm,
parameter=sd(log(income.u)),
col="coral",lty=3)
```

Esempio



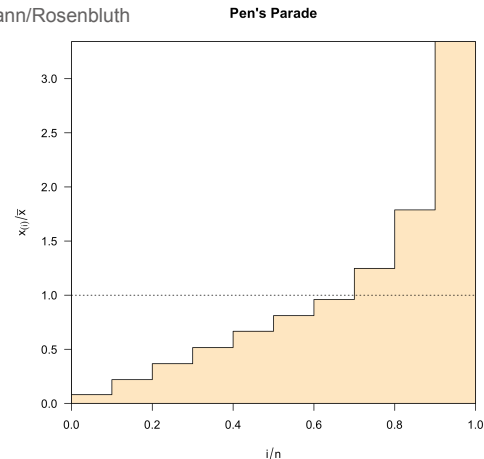
Da notare che la curva "La Union" è interna a quella di Pangasinan e quindi presenta meno concentrazione.
L'adattamento del modello Lognormale è buono per la prima, ma non per la seconda distribuzione.

Indici di concentrazione industriale

```
# X = Unità locali
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
# compute Herfindahl coefficient with parameter 1
conc(x)
# compute coefficient of Hall/Tiedemann/Rosenbluth
conc(x, type="Rosenbluth")
Pen(x)
Pen(x, fill = hsv(0.1, 0.3, 1))
```

[1] 0.1840812

[1] 0.1859051



Ricodifica

Alcune variabili sono state registrate con delle modalità nominali che potrebbero essere inadatte per le elaborazioni

Basso medio alto ----> 1 2 3

```
library(MASS)
data(Cars93)
A<-Cars93
print(A[,9])
A[,9]<-ifelse(A[,9]=="Driver & Passenger",3,ifelse(A[,9]=="Driver only",2,1))
print(A[,9])
A[,11]<-ifelse(A[,11]=="rotary",6,A[,11])
A[,18]<-ifelse(A[,18]==2,3,A[,18]);A[,18]<-A[,18]-2;
A[,16]<-ifelse(A[,16]=="No",0,1)
A[,26]<-ifelse(A[,26]=="USA",1,0)
print(cbind(Cars93[,c(11,18,26)],A[,c(11,18,26)]))
```

Data coding

```
bmd <- c(-0.92,0.21,0.17,-3.21,-1.80,-2.60,
        -2.00,1.71,2.12,-2.11)
diagnosis <- bmd
diagnosis[bmd <= -2.5] <- 1
diagnosis[bmd > -2.5 & bmd <= 1.0] <- 2
diagnosis[bmd > 1.0] <- 3
data <- data.frame(bmd, diagnosis)
data
```

	bmd	diagnosis
1	-0.92	3
2	0.21	3
3	0.17	3
4	-3.21	1
5	-1.80	2
6	-2.60	1
7	-2.00	2
8	1.71	3
9	2.12	3
10	-2.11	2

```
diagnosis <- bmd
diagnosis <- replace(diagnosis, bmd <= -2.5, 1)
diagnosis <- replace(diagnosis, bmd > -2.5 & bmd <= 1.0, 2)
diagnosis <- replace(diagnosis, bmd > 1.0, 3)
```

Modelli probabilistici univariati

Alcuni modelli per variabili continue o dense

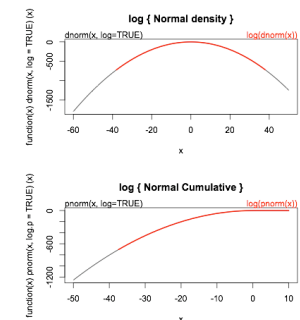


Gaussiano con denominazione base "norm"

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

```
## Using "log = TRUE" for an extended range :
par(mfrow=c(2,1))
plot(function(x) dnorm(x, log=TRUE), -60, 50,
      main = "log { Normal density }")
curve(log(dnorm(x)), add=TRUE, col="red",lwd=2)
mtext("dnorm(x, log=TRUE)", adj=0)
mtext("log(dnorm(x))", col="red", adj=1)

plot(function(x) pnorm(x, log.p=TRUE), -50, 10,
      main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add=TRUE, col="red",lwd=2)
mtext("pnorm(x, log=TRUE)", adj=0)
mtext("log(pnorm(x))", col="red", adj=1)
```



Modelli probabilistici univariati/2

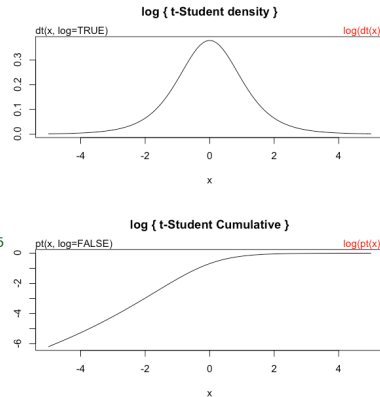


t-Student con denominazione base "t"

```
dt(x, df, ncp, log = FALSE)
pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
rt(n, df, ncp)
```

```
par(mfrow=c(2,1))
plot(function(x) dt(x, 5,0,log=FALSE),-5,5,
     main = "log { t-Student density }")
curve(log(dt(x)), add=TRUE, col="red",lwd=2)
mtext("dt(x, log=TRUE)", adj=0)
mtext("log(dt(x))", col="red", adj=1)

plot(function(x) pt(x, 5,0, log.p=TRUE), -5, 5
     main = "log { t-Student Cumulative }")
curve(log(pt(x)), add=TRUE, col="red",lwd=2)
mtext("pt(x, log=FALSE)", adj=0)
mtext("log(pt(x))", col="red", adj=1)
```



Modelli probabilistici univariati/3

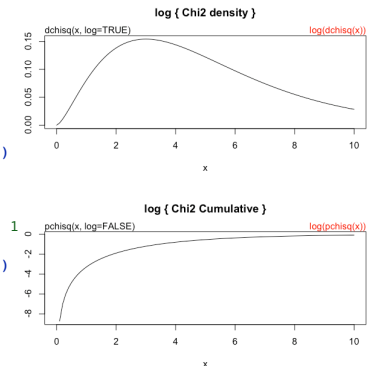


chi2 con denominazione base "chisq"

```
dchisq(x, df, ncp=0, log = FALSE)
pchisq(q, df, ncp=0, lower.tail = TRUE, log.p = FALSE)
qchisq(p, df, ncp=0, lower.tail = TRUE, log.p = FALSE)
rchisq(n, df, ncp=0)
```

```
par(mfrow=c(2,1))
plot(function(x) dchisq(x, 5,0,log=FALSE),0,10,
     main = "log { Chi2 density }")
curve(log(dchisq(x)), add=TRUE, col="red",lwd=2)
mtext("dchisq(x, log=TRUE)", adj=0)
mtext("log(dchisq(x))", col="red", adj=1)

plot(function(x) pchisq(x, 5,0, log.p=TRUE), 0, 1
     main = "log { Chi2 Cumulative }")
curve(log(pchisq(x)), add=TRUE, col="red",lwd=2)
mtext("pchisq(x, log=FALSE)", adj=0)
mtext("log(pchisq(x))", col="red", adj=1)
```



Modelli probabilistici univariati/4

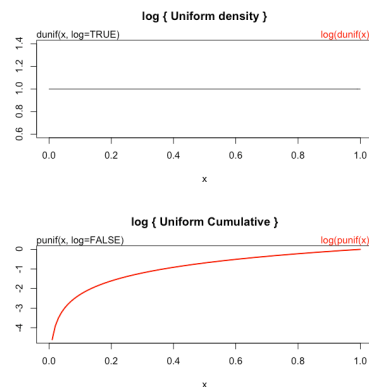


Uniforme con denominazione base "unif"

```
dunif(x, min=0, max=1, log = FALSE)
punif(q, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
runif(n, min=0, max=1)
```

```
par(mfrow=c(2,1))
plot(function(x) dunif(x, 0,1,log=FALSE),0,1,
     main = "log { Uniform density }")
curve(log(dunif(x)), add=TRUE, col="red",lwd=2)
mtext("dunif(x, log=TRUE)", adj=0)
mtext("log(dunif(x))", col="red", adj=1)

plot(function(x) punif(x, 0,1, log.p=TRUE), 0, 1,
     main = "log { Uniform Cumulative }")
curve(log(punif(x)), add=TRUE, col="red",lwd=2)
mtext("punif(x, log=FALSE)", adj=0)
mtext("log(punif(x))", col="red", adj=1)
```



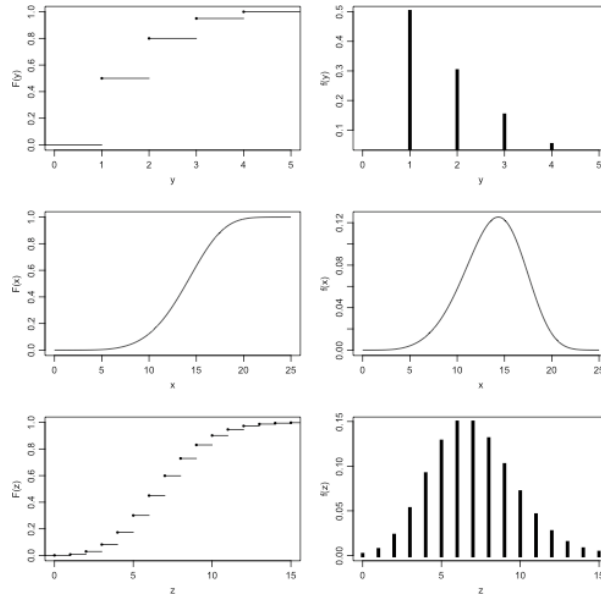
Grafica per i modelli

```
windows(width=6, height=7)
par(mfcol=c(3,2),mai=c(0.6,0.5,0.1,0.1),mfp=c(2,0.7,0))
y<-c(1,2,3,4);Fy<-c(0.5,0.8,0.95,1.0)
plot(stepfun(y,Fy),xlab="y",ylab="F(y)",verticals=FALSE,pch=16,main=NA)
# Diameter distribution, c.d.f.
x<-seq(0,25,0.1);Fx<-pweibull(x,5,15)
plot(x,Fx,type="l",xlab="x",ylab="F(x)")
# Number of trees, c.d.f.
z<-seq(0,20,1);Fz<-ppois(z,7) # Distribuzione di Poisson
plot(stepfun(z,c(0,Fz)),xlab="z",ylab="F(z)",verticals=FALSE,pch=16,
     main=NA,xlim=c(0,15))
# Tree species, density.
fy<-c(0.5,0.3,0.15,0.05)
plot(y,fy,type="n",xlim=c(0,5),xlab="y",ylab="f(y)")
sapply(1:4,function(i) lines(y[c(i,i)],c(0,fy[i]),lwd=4,lend="square"))
# Diameter distribution, density.
fx<-dweibull(x,5,15);plot(x,fx,type="l",xlab="x",ylab="f(x)")
# Number of trees, density.
fz<-dpois(z,7)
plot(z,fz,xlim=c(0,15),type="n",xlab="z",ylab="f(z)")
sapply(1:20,function(i) lines(z[c(i,i)],c(0,fz[i]),lwd=4,lend="square"))
```

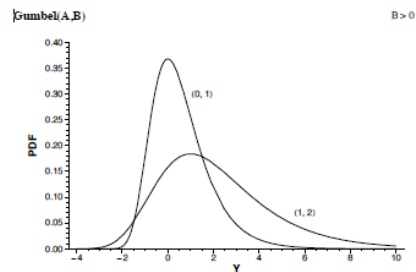
Esercizio

Effettuare lo stesso studio con i seguenti modelli

☐	weibull	☐	Lognormale (lnorm)
☐	beta	☐	gamma
☐	Esponenziale (exp)		



Modelli non presenti in R



Modella l'andamento del massimo di un campione estratto da una popolazione di tipo esponenziale

Parameters -- A (0): Location, B (0): Scale

Moments, etc.

$$\text{Mean} = A + \gamma B$$

$$\text{Variance} = \frac{1}{6} (\pi B)^2$$

$$\text{Skewness} = \frac{12 \sqrt{6} \zeta(3)}{\pi^2} \approx 1.1395$$

$$\text{Kurtosis} = \frac{12}{5}$$

$$\text{Mode} = A$$

$$A-49$$

```
par(mfrow=c(2,1))
dgumbel <- function(x,a,b) 1/b*exp((a-x)/b)*
exp(-exp((a-x)/b))
pgumbel <- function(q,a,b) exp(-exp((a-q)/b))
qgumbel <- function(p,a,b) a-b*log(-log(p))
plot(function(x) dgumbel(x, 5,10), -10, 30,
      main = "log { gumbelorm density }")
curve(log(dgumbel(x)), add=TRUE, col="red", lwd=2)
mtext("dgumbel(x, log=TRUE)", adj=0)
mtext("log(dgumbel(x))", col="red", adj=1)

plot(function(x) pgumbel(x, 5,10), -10, 30,
      main = "log { gumbelorm Cumulative }")
curve(log(pgumbel(x)), add=TRUE, col="red", lwd=2)
mtext("pgumbel(x, log=FALSE)", adj=0)
mtext("log(pgumbel(x))", col="red", adj=1)
```

Modelli probabilistici univariati/5

Alcuni modelli per variabili discrete

binom	Binomial (size, prob)
hyper	hypergeometric (m, n, k)
nbinom	negative binomial (size, prob)
pois	Poisson (lamda)
geom	geometric (prob)

four all discrete distributions you can use the 4 functions pxxxx CDFdxxxx pmf (d for discrete) qxxxx quantilerxxxx random variates

Binomiale

```
plot(0:100,pbinom(0:100,size=100,prob=0.22))
#look at the figure and guess binomial distribution
dbinom(3,size=20,prob=0.4) #pmf(3) Binomial distribution, n=20 p=0.4
dbinom(3,20,0.4) #short version of the above command
dbinom(3.5,20,0.4)

pbinom(3,20,0.4) #CDF(x) Binomial distribution, n=20 p=0.4
sum(dbinom(1:3,20,0.4))
#these commands also work for vectors! why is it not the same as pbinom above??
sum(dbinom(0:3,20,0.4))# now it is the same

qbinom(0.93,20,0.4)#quantile (inverse CDF)
pbinom(11,20,0.4)
pbinom(10,20,0.4)
rbinom(100,20,0.4)#random vector of length 100 of binomial distribution

hist(rbinom(100,20,0.4))#show the histogram with frequencies
hist(rbinom(10000,20,0.4),freq=FALSE)#show the histogram with relative frequencies
plot(0:20,dbinom(0:20,20,0.6),type="s")#plot of the pmf; "s" stands for stairs

# Calculate P(X>=10) by simulation
mean(runif(1000000)>0.8)
# calculates the empirical probability that uniform 0,1 variate larger than 0.8
```

Istogramma con curva

Prezzo medio per carato in dollari di un anello con diamanti sul mercato di Singapore.

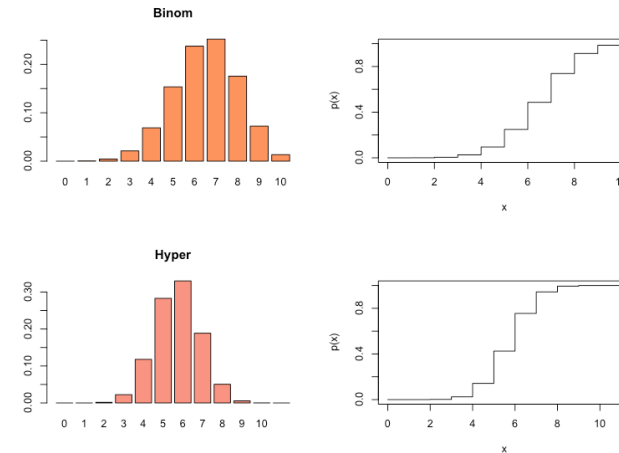
```
library(normalp)
Y<-scan("PMKarat.txt")
Y<-scale(Y) # sottrazione della media e divisione per lo scarto
Y[Y < -3.5 | Y > 3.5] <- NA # Esclusione dei valori troppo grandi
x <- seq(-3.5, 3.5, .1)
dn <- dnorm(x)
par(mar=c(4.5, 4.1, 3.1, 0))
hist(Y, breaks=seq(-3.5, 3.5), ylim=c(0, 0.5), col="lightcoral", freq=FALSE,
main="Istogramma del Prezzo")
lines(x, dnorm(x), lwd=2,col="red4")
lines(x, dnorm(x,p=1),lwd=2,col="blue")
lines(x, dnorm(x,p=1.5),lwd=2,col="green4")
par(mar=c(5.1, 4.1, 4.1, 2.1))
```

Si tenta di modellare i dati con varie curve di tipo gaussiano esponenziale

Esempio

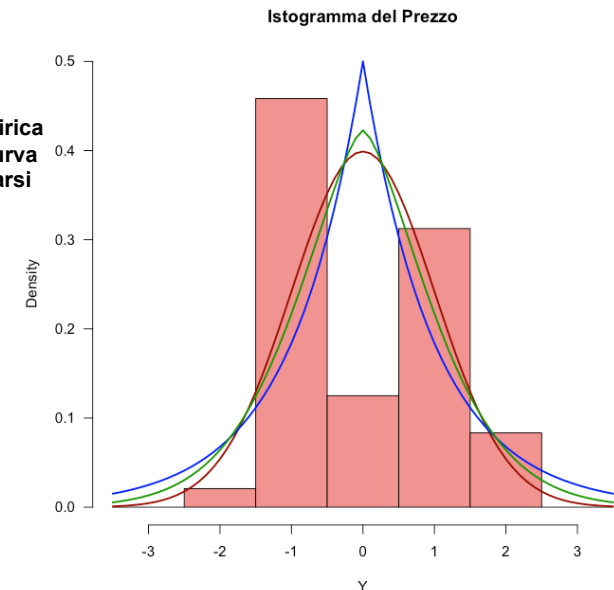
```
par(mfrow=c(2,2))
barplot(dbinom(0:10,10,0.65), col="coral", names.arg=0:10,main="Binom")
plot(0:10, pbinom(0:10, 10, 0.65), type="s", xlab="x", ylab="p(x)")

barplot(dhyper(0:11,12,9,10), col="salmon", names.arg=0:11,main="Hyper")
plot(0:11, phyper(0:11, 12, 9,10), type="s", xlab="x", ylab="p(x)")
```



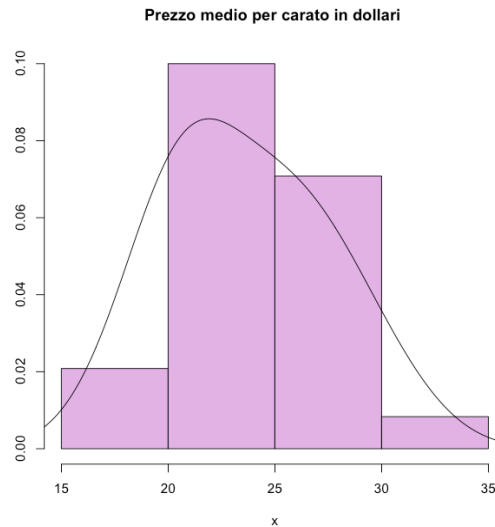
Esempio

Se la distribuzione empirica è bimodale, nessuna curva teorica potrà mai adattarsi ad essa in modo soddisfacente



Comando density

```
x<-scan("PMKarat.txt")
idq<-summary(x)[5]-
summary(x)[2]
truehist(x,probability=
TRUE,col="plum",main=
"Prezzo medio per carato in
dollari ")
lines(density(x,width=2
*idq))
```



Esempio/1

```
#Istogramma delle frequenze e spline interpolante
par(mfrow=c(1,1),mar=c(4,4,2,2))
attach(attitude)
hist(rating,breaks="sturges",freq=FALSE,main="Istogramma
delle frequenze",xlab="Modalità osservate",ylab="Densità di
frequenza",col="lightblue",border="red")
Dens<-density(rating)
xval<-range(Dens$x);yval<-range(Dens$y)
# yval[2]<-yval[2]+0.0025
hist(rating,breaks="sturges",freq=FALSE,main="Istogramma
delle frequenze",xlab="Modalità osservate",ylab="Densità di
frequenza",col="lightblue",border="red",probability=TRUE,xlim
=xval,ylim=yval)
lines(Dens,col="blue",lwd=2,lty=2)
```

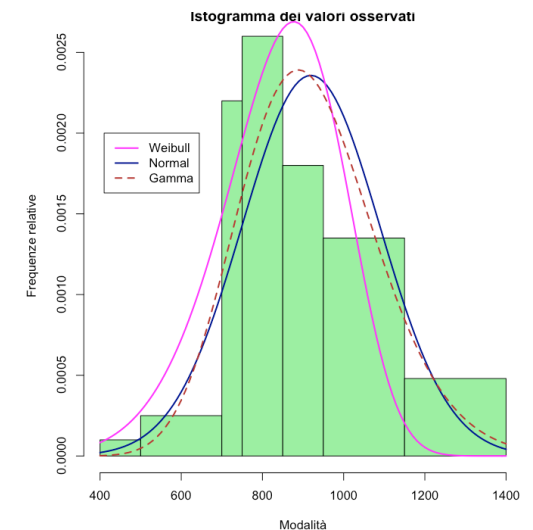
```
density(x, bw = "nrd0", adjust = 1,
kernel = c("gaussian", "epanechnikov", "rectangular",
"triangular", "biweight",
"cosine", "optcosine"),
weights = NULL, window = kernel, width,
give.Rkern = FALSE,
n = 512, from, to, cut = 3, na.rm = FALSE, ...)
```

Adattamento di una distribuzione teorica

```
hist(Nile,main="Istogramma dei valori
osservati",ylab="Frequenze
relative",xlab="Modalità",breaks=c(400,500,700,750,850,950,115
0,1400 ),freq=FALSE,right=TRUE,col="lightgreen",prob=TRUE)
sdn<-sd(Nile);xdn<-mean(Nile)
curve(dnorm(x,mean=xdn,sd=sdn),add=TRUE,lwd=2,col="navy")
cvn<-xdn/sdn^2;hdn<-xdn^2/sdn^2
curve(dweibull(x,shape=6.5,scale=900),from=400,
to=1400,add=TRUE,lwd=2,col="magenta")
curve(dgamma(x,rate=cvn,shape=hdn),add=TRUE,lwd=2,lty=2,col="
brown")
legend(410,0.0020,legend=c("Weibull", "Normal", "Gamma"),col=c("
magenta", "navy", "brown"),lwd=c(2,2,2),lty=c(1,1,2))
```

Esempio (continua)

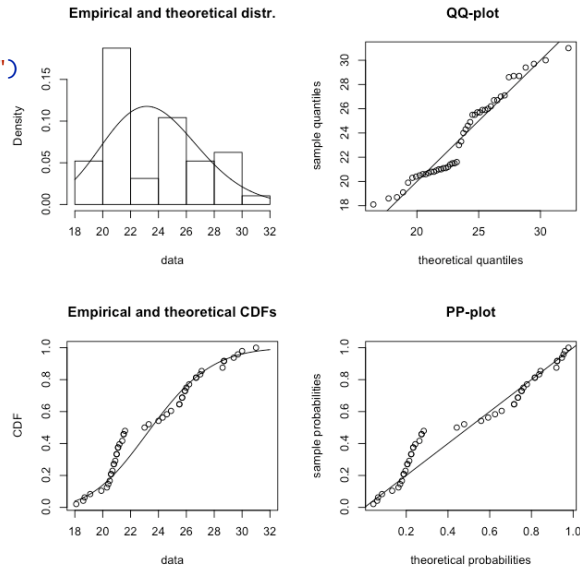
Si può notare la unimodalità dell'istogramma che viene in effetti colta dai modelli. Nessuno di essi sembra però individuare la asimmetria verso i valori grandi



Adattamento di una distribuzione/2

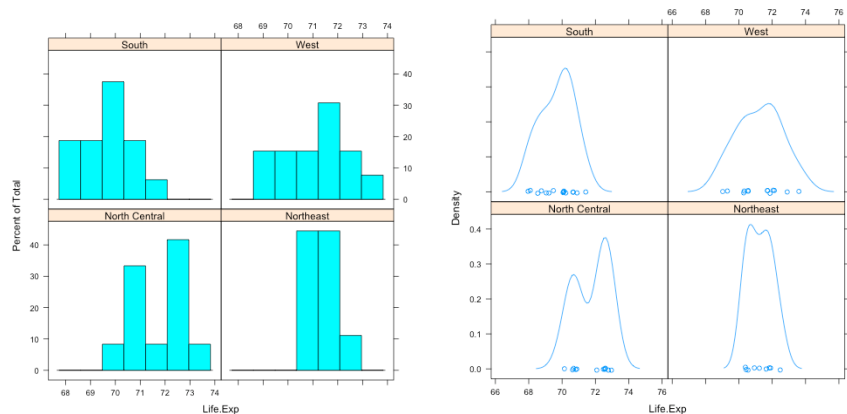
```
library(fitdistrplus)
x<-scan("PMKarat.txt")
f1c <- fitdist(x,"gamma")
print(f1c)
plot(f1c)
```

Se l'adattamento fosse buono, gli scarti nei grafici PP e QQ sarebbero piccoli



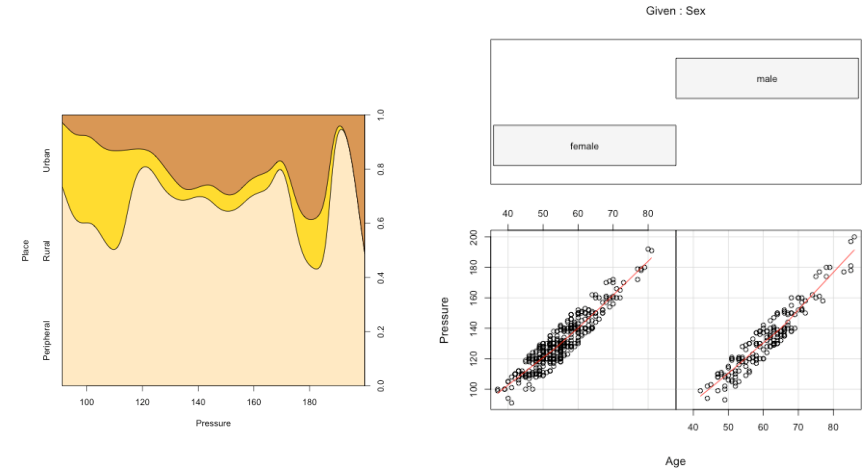
Confronto di ditribuzioni

```
USstates <- read.table("USstates.txt",header=T,sep=",")
attach(USstates)
summary(USstates) # summary stats
library(lattice) # enhanced plotting environment
histogram(~ Life.Exp | Region) # Life.exp split in 3 classes
densityplot(~ Life.Exp | Region)
```



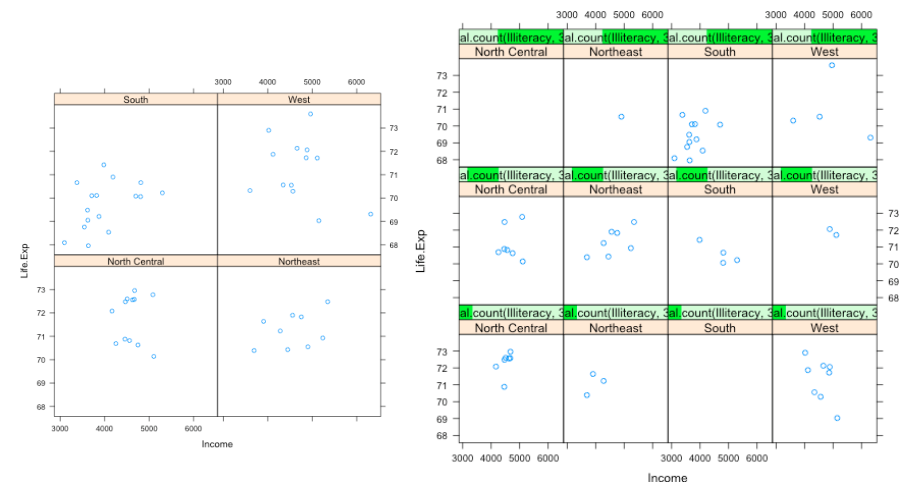
Densità condizionate

```
Older<-read.table("UrbGen.csv",header=T,sep=",")
attach(Older)
graphics.off() # reset/close all graphical devices
cdplot(Place~Pressure,bw="nrd0",col=c("moccasin","gold","peru"))
coplot(Pressure ~ Age | Sex, panel=panel.smooth)
```



Confronto di relazioni

```
xyplot(Life.Exp ~ Income | Region)
xyplot(Life.Exp ~ Income | Region*equal.count(Illiteracy,3,0))
```



Integrazione numerica

Per calcolare il valore di un integrale definito di una certa funzione in un dato intervallo si usa il comando **integrate**

```
integrate(f, lower, upper, ..., subdivisions=100,  
rel.tol = .Machine$double.eps^0.25, abs.tol = rel.tol,  
stop.on.error = TRUE, keep.xy = FALSE, aux = NULL)
```

$$\int_0^{\pi/2} 4 \sin(2x) \exp(-x^2) dx$$

```
integrand<-function(x) 4*sin(2*x)*exp(-x^2)  
print(Area<-integrate(integrand,0,pi/2))
```

2.190752 with absolute error < 2.4e-14

```
integrand<-function(x) 4*sin(2*x)*exp(-x^2)  
print(Area<-integrate(integrand,0,Inf))
```

Spiegare il risultato inferiore

2.152318 with absolute error < 2.7e-05

Generazione di numeri pseudo casuali

La simulazione di numeri casuali si basa su sequenze di numeri che nascono da ben definite relazioni matematiche e per questo si parla di numeri pseudocasuali.

Pur conservando un precipuo carattere deterministico si comportano come una successione di variabili casuali.

L'evoluzione dei generatori di numeri ha una sua pietra miliare nei generatori congruenziali lineari introdotti da Lehmer (1949).

La loro formula è

$$X_n = (aX_{n-1} + c) \text{ Mod } m; \quad n=1,2, \dots, m$$
$$p \equiv q \text{ Mod } m \text{ significa che } p = q - \left\lfloor \frac{q}{m} \right\rfloor m$$

ovvero p è il resto intero della divisione di q per m

Campionamento

```
Population<-  
c(3,4,12,5,11,35,24,21,29,38,11,17,34,6,22,13,19,24,38,2,17,  
+ 15,28,31,29,11,23,34,3,12,4,28,39,16,22,31,37,19)  
N<-length(Population);n<-floor(0.20*N)  
set.seed(262657)  
Sam1<-sample(Population,n,replace=TRUE)  
print(Sam1)  
Sam2<-sample(Population,n,replace=FALSE)  
print(Sam2)  
Sam3<-sample(Population,n,replace=TRUE,prob=1/(Population^2))  
print(Sam3)  
Sam4<-sample(Population,n,replace=FALSE,prob=1/(Population^2))  
print(Sam4)  
Sam5<-sample(Population,n,replace=TRUE,prob=1/(Population^0.1))  
print(Sam5)  
Music<-  
c("Puccini", "Rossini", "Verdi", "Mascagni", "Paisiello", "Rendano", "Doniz  
etti", "Bellini", "Leoncavallo", "Monteverdi")  
Best<-sample(Music,4,replace=FALSE)  
print(Best)
```

Generazione di numeri pseudo casuali/2

Le sequenze ottenute dalla sono deterministiche: dato un certo termine è possibile calcolarne qualsiasi altro che interverrà in successione,

La conseguenza è che ogni successione è riproducibile: per ripeterla tutta è sufficiente conservarne il valore iniziale.

Il comando in R è

```
RNGkind(kind = NULL, normal.kind = NULL)
```

Esistono diversi algoritmi per ottenere numeri pseudo casuali di qualità.

In particolare

kind="Wichmann-Hill" (da me sperimentato con successo in molte occasioni)

kind="Mersenne-Twister" (scelta di default) che ha un periodo molto elevato ($2^{19937}-1$) e si mantiene efficace anche nel caso di generazioni multivariate.

Generazione di numeri pseudo casuali/3

Dal generatore si ottengono valori pseudo casuali nell'intervallo unitario

Per ottenere valori da altre distribuzioni ci si può basare sul seguente risultato teorico

Teorema

Se la variabile casuale X ha funzione di densità $f(x)$ allora la variabile casuale

$$u = \int_{-\infty}^x f(t) dt$$

ha densità uniforme sull'intervallo $[0,1]$.

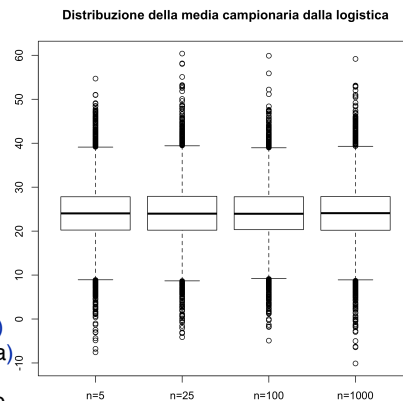
Ad esempio, per i modelli che rientrano nella famiglia Burr

$$F(x) = \frac{1}{1 + e^{-g(x)}} \quad x = g^{-1} \left[\text{Ln} \left(\frac{u}{1-u} \right) \right]$$

Per ottenere una x con distribuzione F si usa una u uniforme e la si trasforma

Simulazione della distribuzione di una media campionaria

```
Rlogis <- function(n,mu, sigma)
{u <- rep(runif(1),n)
 return(mu + sigma* log(u/(1-u)))}
set.seed(820731)
mu<-24;sigma<-3.5
Numsim<-10000
mean5<-rep(0,Numsim);mean25<-mean5
mean100<-mean5;mean1000<-mean5
for (i in 1:Numsim)
{mean5[i]<-mean(Rlogis(5,mu,sigma))
 mean25[i]<-mean(Rlogis(25,mu,sigma))
 mean100[i]<-mean(Rlogis(100,mu,sigma))
 mean1000[i]<-mean(Rlogis(1000,mu,sigma))
}
boxplot(mean5,mean25,mean100,mean1000,
names=c("n=5","n=25","n=100","n=1000"),
title("Distribuzione della media campionaria
dalla logistica"))
```



Esempio: distribuzione logistica

$$F(x;\mu,\sigma) = \frac{1}{1 + \exp\left[-\left(\frac{x-\mu}{\sigma}\right)\right]}; \quad \sigma > 0$$

$$x = \mu + \sigma \log\left(\frac{u}{1-u}\right) \quad u \sim \text{uniform } [0,1]$$

```
Rlogis <- function(n,mu, sigma)
{u <- rep(runif(1),n)
 return(mu + sigma* log(u/(1-u)))}
```

Esempio: medie progressive

```
#esponenziale: F(x)=1-e^(-lambda x)
lambda<-1;numsim<-200
mean5<-rep(0,numsim);mean25<-mean5;mean100<-mean5;mean200<-mean5
for (i in 1:numsim){
  mean5[i]<-mean(rexp(5,lambda))
  mean25[i]<-mean(rexp(25,lambda))
  mean100[i]<-mean(rexp(100,lambda))
  mean200[i]<-mean(rexp(200,lambda))
}
boxplot(mean5,mean25,mean100,mean200,names=c("n=5","n=25","n=100","n=200"))
title("distribuzione della media per campioni dalla esponenziale")
#####
Medprog<-function(x){
  n<-length(x)
  ret<-rep(0,n)
  for (k in 1:n){ret[k]<-mean(x[1:k])}
  return(ret)}
x<-rnorm(100, mean=0, sd=1)
y<-rcauchy(100, location = 0, scale = 1)
par(mfrow=c(1,2));plot(Medprog(x),type="l")
title("Medie progressive dalla normale standardizzata")
plot(Medprog(y),type="l")
title("Medie progressive dalla Cauchy")
```

Integrazione Monte Carlo

Vogliamo calcolare un integrale che ha una funzione integranda complicata

$$I = \int_a^b g(x) dx$$

Generiamo n valori indipendenti da una uniforme in $[a,b]$ e calcoliamo

$$\hat{I} = (b-a) \frac{1}{n} \sum_{i=1}^n g(X_i)$$

Per la legge dei grandi numeri si ha

$$\hat{I}_n \rightarrow (b-a) E[g(X)]$$

Il valore atteso di $g(x)$ in $[a,b]$ è dato da

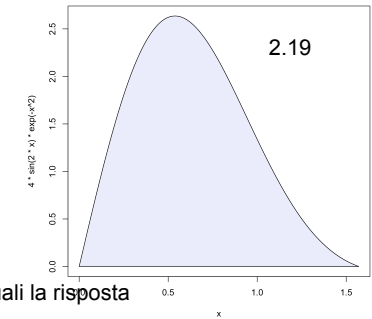
$$E[g(X)] = \int_a^b g(x) \frac{1}{b-a} dx = \left(\frac{1}{b-a} \right) I$$

Quindi I_n è una approssimazione di I che migliora all'aumentare di n

Esempio

$$\int_0^{\pi/2} 4 \sin(2x) \exp(-x^2) dx$$

```
curve(4*sin(2*x)*exp(-x^2),0,pi/2,201)
x1<-seq(0,pi/2,length=201)
y1<-4*sin(2*x1)*exp(-x1^2)
x2<-seq(0,pi/2,length=201)
y2<-4*sin(2*x2)*exp(-x2^2)
polygon(c(x1,x2),c(y1,y2),col="lavender")
u<-runif(1000000,min=0,max=pi/2)
pi/2*mean(4*sin(2*u)*exp(-u^2))
# a=0, b=pi/2, so b-a=pi/2#
Se non fissa il seed del generatore di numeri casuali la risposta
# sarà diversa per la stessa chiamata
u<-runif(1000000,min=0,max=pi/2)
pi/2*mean(4*sin(2*u)*exp(-u^2)) # a=0, b=pi/2, so b-a=pi/2
# Fissato il seed il calcolo sarà lo stesso ad ogni call. Ad esempio
set.seed(820731)
u<-runif(1000000,min=0,max=pi/2)
pi/2*mean(4*sin(2*u)*exp(-u^2)) # a=0, b=pi/2, so b-a=pi/2
set.seed(820731)
u<-runif(1000000,min=0,max=pi/2)
pi/2*mean(4*sin(2*u)*exp(-u^2)) # a=0, b=pi/2, so b-a=pi/2
```



Verifica della normalità

Una parte considerevole dello studio inferenziale si regge sull'ipotesi di normalità delle osservazioni.

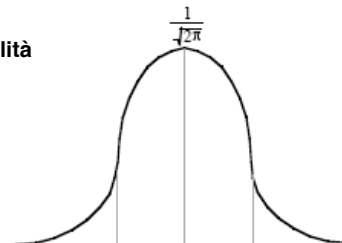
Se i valori non sono normali molte procedure conservano ancora efficienza purché

Gli errori si mostrino simmetrici

Le code si mostrino leggere

Grazie anche a queste condizioni la normalità diventa un'ipotesi plausibile per campioni moderatamente grandi.

E' quindi opportuno asseverare se i valori osservati siano riconducibili alla ipotesi di normalità



QQ_plot

Il grafico è ottenuto riportando in un sistema di assi cartesiani i valori (standardizzati ed ordinati) ed i quantili della Normale corrispondenti (da cui: grafico quantile-quantile)

In dettaglio

$$\left[Z(p_i), \frac{\hat{e}_{(i)}}{\hat{\sigma}_i} \right]; \quad i = 1, 2, \dots, n$$

Il pedice tra parentesi indica valori ordinati in senso ascendente.

$Z(p_i)$ sono i quantili della normale standardizzata corrispondenti alla frequenza cumulata p_i associata con $e_{(i)}$ osservato.

Se il grafico somiglia molto ad una retta che passa per l'origine e con inclinazione unitaria (la bisettrice del quadrante per intenderci), l'approssimazione normale è considerata buona.

Posizioni grafiche

La scelta dei quantili cui far corrispondere $e_{(i)}$ stimato è controversa.

Scartata la soluzione $p_i = i/n$ che è inadatta per un supporto infinito (raggiunge subito l'unità per $i=n$), ci si è orientati sulla seguente formula generale

$$p_i(\alpha, \beta) = \frac{i - \alpha}{n + 1 - (\alpha + \beta)}$$

Dove α e β sono tali da assicurare valori crescenti compresi tra zero ed uno.

I valori di questi parametri cambiano da distribuzione a distribuzione.

Spesso si sceglie $\alpha = \beta$

Scelta delle posizioni grafiche

Name	α	β	$p_i(\alpha, \beta)$
0) <i>Hazen</i>	0.5	0.5	$\left(\frac{i - 0.5}{n}\right)$
1) <i>Weibull</i>	0	0	$\left(\frac{i}{n+1}\right)$
2) <i>Blom</i>	0.375	0.375	$\left(\frac{i - 0.375}{n + 0.25}\right)$
3) <i>Landwehr</i>	0.35	0.65	$\left(\frac{i - 0.35}{n}\right)$
4) <i>Tukey</i>	0.3333	0.3333	$\left(\frac{i - 0.3333}{n + 0.3333}\right)$
5) <i>Cunnane</i>	0.4	0.4	$\left(\frac{i - 0.4}{n + 0.2}\right)$
6) <i>Benard</i>	0.3	0.3	$\left(\frac{i - 0.3}{n + 0.4}\right)$
7) <i>Filliben</i>	0.3175	0.3175	$\left(\frac{i - 0.3175}{n + 0.65}\right)$
8) <i>Gringorten</i>	0.44	0.44	$\left(\frac{i - 0.44}{n + 0.12}\right)$
9) <i>Larsen</i>	0.567	0.567	$\left(\frac{i - 0.3175}{n - 0.134}\right)$

La formula di Weibull gode di una certa popolarità.

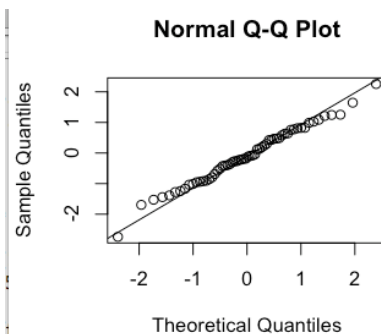
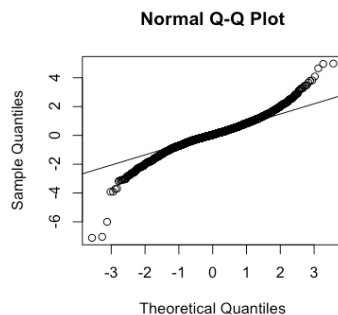
Quella che si consiglia è la Landwehr

$p_i(\alpha, \beta)$ è una approssimazione dei quantili della distribuzione uniforme nell'intervallo unitario.

Grazie al teorema dell'inversione possiamo ottenere i quantili di moltissime distribuzioni a partire proprio da $p_i(\alpha, \beta)$

Esempio

```
x<-rnorm(60)
qqnorm(x);qqline(x)
###
library(MASS)
SP500
qqnorm(SP500)
qqline(SP500)
```

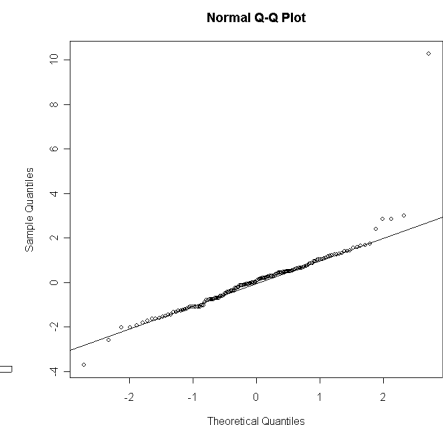
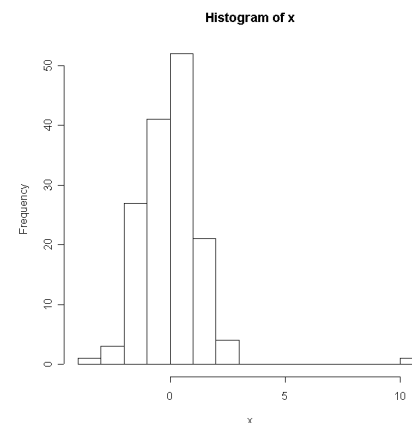


R utilizza le posizioni di Weibull

Il grafico Q-Q è un semplice ed utile ausilio per accertare l'aderenza alla Normale ovvero a qualsiasi distribuzione di cui siano noti i quantili.

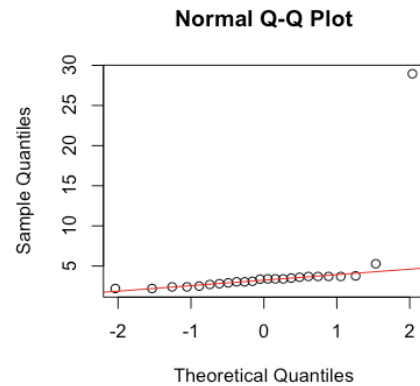
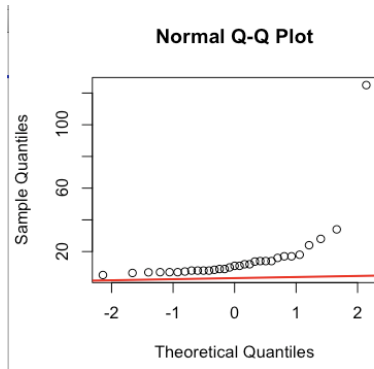
Casi di disnormalità: valori remoti

I casi di osservazioni estreme (alla luce del campione osservato) sono subito evidenti nel grafico QQ



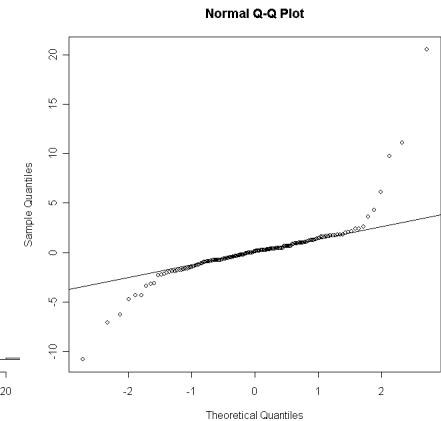
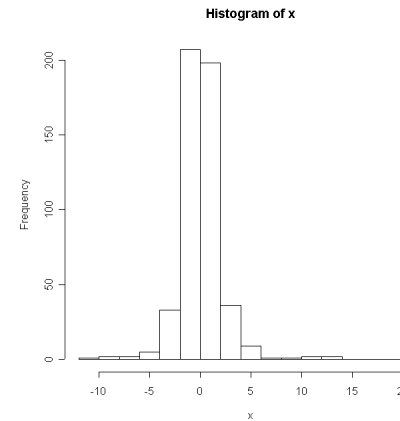
Esempio

```
library(MASS)
qqnorm(chem)
qqline(chem)
###
qqnorm(abbey)
qqline(chem,col = 2,lwd=2)
```



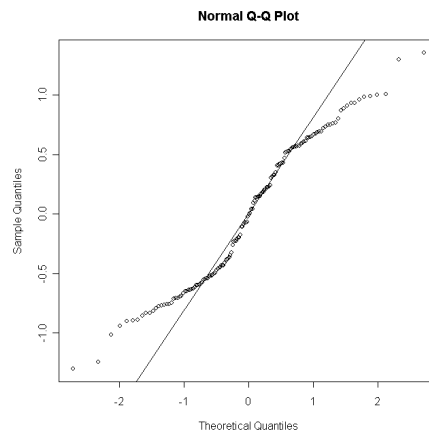
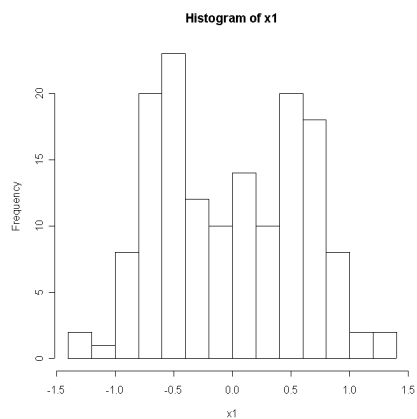
Casi di disnormalità: code pesanti

Le code pesanti si mostrano con la presenza di varie osservazioni per valori troppo bassi e/o troppo alti rispetto a ciò che ci si aspetta in un fenomeno normale



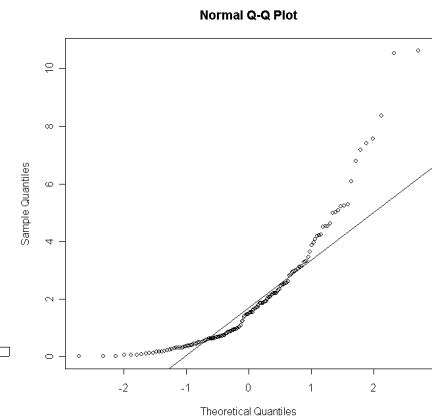
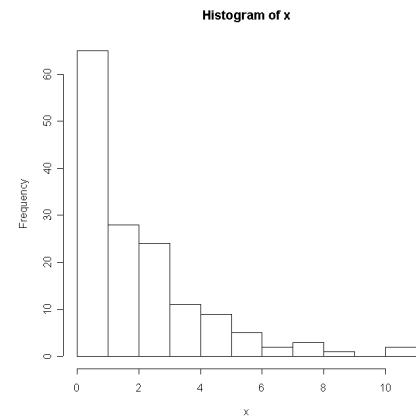
Casi di disnormalità: code smorzate

Le code smorzate si mostrano con l'assenza di osservazioni per valori bassi e/o alti laddove un fenomeno normale produrrebbe diversi valori sia pure con frequenza decrescente

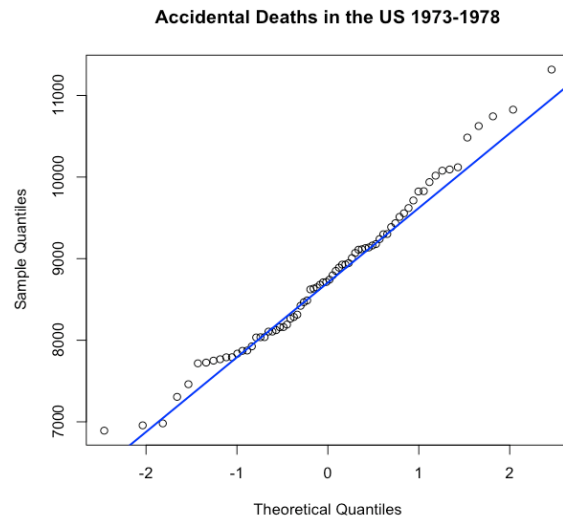


Casi di disnormalità: asimmetria

L'asimmetria della distribuzione si riscontra nella presenza di una forte curvatura del diagramma QQ.



```
library(MASS)
qqnorm(accdeaths,main="Accidental Deaths in the US 1973-1978")
qqline(accdeaths,col = 4,lwd=2)
```



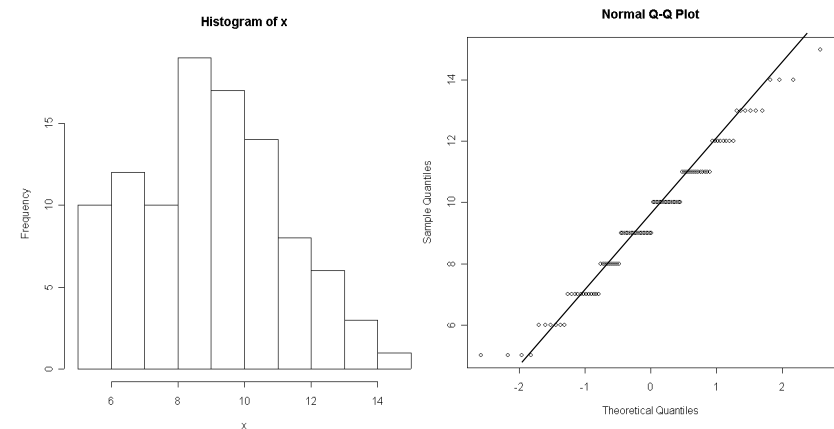
Matrici

```
> x<-matrix(1:10,ncol=5) #costruisci una matrice
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> x[,1] #seleziona la prima colonna
[1] 1 2
> x[2,] #seleziona la seconda riga
[1] 2 4 6 8 10
> x[3,2] #seleziona l'elemento [3,2]
Error: subscript out of bounds
> x[2,3] #...e quello [2,3]
[1] 6
> x[,4:5] #seleziona solo le colonne 4 e 5
      [,1] [,2]
[1,]    7    9
[2,]    8   10
> x[,-c(2,4)] #seleziona le colonne 1, 3 e 5
      [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
```

Casi di disnormalità: livellamenti

Il verificarsi di valori ripetuti (a causa di unità di misura troppo grezze) si traduce in una deviazione dalla normalità.

Anche la presenza di gap (vuoti) tra i valori è segno di disnormalità



Matrici/2

Altre funzioni sono `cbind()`, `rbind()` e `diag()` che costruisce una matrice diagonale o estrae la diagonale da una matrice:

```
> cbind(1:2,c(1,-2),c(0,9)) #dispone per colonne
      [,1] [,2] [,3]
[1,]    1    1    0
[2,]    2   -2    9
> rbind(1:4,c(0,5,-4,6)) #dispone per righe
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    0    5   -4    6
> diag(x[,4:5]) #estrae la diagonale principale
[1] 7 10
> X<-diag(1:3) #costruisce una matrice diagonale
```

Matrici/3

Il contenuto delle matrici deve essere OMOGENEO e non necessariamente NUMERICO

```
> matrix(month.name, nrow = 6)
      [,1]      [,2]
[1,] "January" | "July"
[2,] "February" "August"
[3,] "March"    "September"
[4,] "April"    "October"
[5,] "May"      "November"
[6,] "June"     "December"
```

Matrici/4

```
> mydata <- c(2.9, 3.4, 3.4, 3.7)
> colour <- c("red", "green", "blue")
> xi <- 25:30
> length(mydata)
[1] 4
> mode(mydata)
[1] "numeric"
> mode(xi)
[1] "numeric"
> colour[2:3]
[1] "green" "blue"
> xi[-1]
[1] 26 27 28 29 30
> colour != "green"
[1] TRUE FALSE TRUE
> colour[colour != "green"]
[1] "red" "blue"
> names(mydata) <- c('a', 'b', 'c', 'd')
> mydata
  a b c d
2.9 3.4 3.4 3.7
> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o"
"p" "q" "r" [19] "s" "t" "u" "v" "w" "x" "y" "z"
> mydata[letters[1:2]]
  a b
2.9 3.4
> matrix(xi, 2, 3)
      [,1] [,2] [,3]
[1,] 25 27 29
[2,] 26 28 30
> matrix(xi, 2, 3, byrow=T)
      [,1] [,2] [,3]
[1,] 25 27 29
[2,] 26 28 30
```

Matrici/5

E' possibile creare matrici con l'aggregazione di matrici più piccole e/o di vettori grazie agli operatori cbind (accostamento di colonne) e rbind (accostamento di righe)

```
> y <- cbind(A = 1:4, B = 5:8, C = 9:12)
> y
  A B C
[1,] 1 5 9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12
> rbind(y, 0)
  A B C
[1,] 1 5 9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12
[5,] 0 0 0
```

Note that the short vector (0) is replicated.

Matrici/6

```
x<-c(1:5,7:14,-7:-1)
A<-matrix(x, nrow = 5, ncol=4, byrow=TRUE, dimnames =
list(c("row_1", "row_2", "row_3", "row_4", "row_5"), c("C.1",
"C.2", "C.3", "C.4")))
n<-nrow(A);m<-ncol(A);d<-dim(A) # numero di righe e di colonne
cat(n,m,d[1],d[2],"\n")
rownames(A)<-c("Pax","Meger","Str","Vac","Lu")
colnames(A)<-c("Pz","Cs","To","Lon");A
```

```
5 4 5 4 5 4
> rownames(A)<-c("Pax","Meger","Str","Vac","Lu")
> colnames(A)<-c("Pz","Cs","To","Lon")
> A
      Pz Cs To Lon
Pax    1  2  3  4
Meger  5  7  8  9
Str   10 11 12 13
Vac   14 -7 -6 -5
Lu    -4 -3 -2 -1
```

Indici di una matrice

#Data una matrice si intende richiamarne alcune parti

```
x<-c(1:5,7:14,-7:-1)
A<-matrix(x, nrow = 5, ncol=4, byrow=TRUE, dimnames =
list(c("row_1", "row_2", "row_3", "row_4", "row_5"),
c("C.1", "C.2", "C.3", "C.4")))
A
A[,3]; ##### Terza colonna
A[4,]; ##### Quarta riga
A[,]; ##### Tutto
A[5,2]
##### Elemento in posizione
di riga 5 e colonna 2
```

```
> A
      C.1 C.2 C.3 C.4
row_1  1  2  3  4
row_2  5  7  8  9
row_3 10 11 12 13
row_4 14 -7 -6 -5
row_5 -4 -3 -2 -1
> A[,3]; ##### Terza colonna
row_1 row_2 row_3 row_4 row_5
      3  8 12 -6 -2
> A[4,]; ##### Quarta riga
C.1 C.2 C.3 C.4
14 -7 -6 -5
> A[,]; ##### Tutto
      C.1 C.2 C.3 C.4
row_1  1  2  3  4
row_2  5  7  8  9
row_3 10 11 12 13
row_4 14 -7 -6 -5
row_5 -4 -3 -2 -1
> A[5,2] ##### Elemento in posizione di riga 5 e colonna 2
[1] -3
```

Vettorizzazione

#vettorizzazione della matrice C di n righe e m colonne

```
C<-c(1,2,3,10,11,12)
C<-matrix(C,nrow=2,ncol=3,byrow=TRUE)
n<-2;m<-3
A<-matrix(0,n*m) # n righe,m colonne
A<-matrix(C,nrow=n*m,ncol=1);A
#
## Matrice di vettori
X<-matrix(c(-3,0,-1,2,0,-1,1,-1),4,2)
Y<-matrix(c(1,2,3,4,4,3,2,1),4,2)
Z<-matrix(c(-4,2,-3,1,2,0,0,1),4,2)
print(X);print(Y);print(Z)
x<-matrix(X,nrow=8,ncol=1)
y<-matrix(Y,nrow=8,ncol=1)
z<-matrix(Z,nrow=8,ncol=1)
D<-cbind(x,y,z);D
library(corpcor)
sm2vec(m, diag = FALSE)
sm.index(m, diag = FALSE)
vec2sm(vec, diag = FALSE, order = NULL)
```

sm2vec takes a symmetric matrix and puts the lower triangular entries into a vector

Trasposta di una matrice

E' la tabella che si ottiene ricopiando la matrice con le righe al posto delle colonne.

L'operatore è t()

A cosa è uguale
C<-t(t(A)) ?

```
> A<-matrix(1:12,3,4,byrow=TRUE)
> A
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> B<-t(A)
> B
      [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

Uso dell'operatore diag()

La diagonale delle matrici quadrate ha sempre un ruolo speciale nelle applicazioni pratiche dell'algebra lineare

Matrice identità

$$I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

```
> A<-matrix(0,5,5);diag(A)<-1
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

A<-diag(5)

Recupero della diagonale

```
> A<-matrix(c(1,-17,4,pi,0,8,2,-3/4,-exp(1)),3,3,byrow=TRUE)
> A
      [,1] [,2] [,3]
[1,] 1.000000 -17.00 4.000000
[2,] 3.141593  0.00 8.000000
[3,] 2.000000 -0.75 -2.718282
> D<-diag(A)
> D
[1] 1.000000 0.000000 -2.718282
```

$$A = \begin{pmatrix} 1 & -17 & 4 \\ \pi & 0 & 8 \\ 2 & -3/4 & -e \end{pmatrix}$$

Traccia e determinante di una matrice

La traccia è data dalla somma sugli elementi della diagonale

$$Tr(A) = \sum_{i=1}^n a_{i,i}$$

Il determinante di una matrice è una somma di prodotti degli elementi della matrice combinati secondo un procedimento ben preciso

A partire dalle definizioni

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \\ -1 & 0 & 7 \end{bmatrix} \quad Y = \begin{bmatrix} 6 & 0 & 0 \\ -3 & 4 & 0 \\ 0 & -5 & 2 \end{bmatrix}$$

Calcoliamo la traccia ed il determinante delle due matrici

```
> TrX<-sum(diag(X)); TrY<-sum(diag(Y));
> DtX<-det(X);DtY<-det(Y)
```

Prodotto (interno) di Matrici

Si realizza grazie all'operatore

% * %

Attenzione! Il solo asterisco definisce il prodotto di HADAMARD tra matrici.

E' diverso dal prodotto interno perché moltiplica le matrici elemento per elemento corrispondente

```
> x
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
> x * x
  [,1] [,2] [,3] [,4]
A    1    4    9   16
B   25   36   49   64
C   81   100  121  144
> x %*% t(x)
   A  B  C
A  30 70 110
B  70 174 278
C 110 278 446
```

Matrici triangolari

Le matrici triangolari sono strutture che si incontrano in varie situazioni, ad esempio nella soluzione dei sistemi di equazioni

$$T_L = \begin{bmatrix} a & e & h & j \\ 0 & b & f & i \\ 0 & 0 & c & g \\ 0 & 0 & 0 & d \end{bmatrix}; \quad T_{L,d} = \begin{bmatrix} 0 & a & e & g \\ 0 & 0 & b & f \\ 0 & 0 & 0 & c \\ 0 & 0 & 0 & 0 \end{bmatrix}; \quad T_U = \begin{bmatrix} a & 0 & 0 & 0 \\ e & b & 0 & 0 \\ h & f & c & 0 \\ j & i & g & d \end{bmatrix}; \quad T_{U,d} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 \\ e & b & 0 & 0 \\ g & f & c & 0 \end{bmatrix};$$

```
H<-matrix(c(1:5,2:6,3:7,4:8,5:9),5,5);H
```

```
#
```

```
HL<-H;HL[upper.tri(H,diag=F)]<-0HL
```

```
HLd<H;HLd[upper.tri(H,diag=T)]<-0;HLd
```

```
HU<H;HU[lower.tri(H,diag=F)]<-0;HU
```

```
HUd<-H;HUd[lower.tri(H,diag=T)]<-0;HUd
```

```
> # Hankel matrix
> H<-matrix(c(1:5,2:6,3:7,4:8,5:9),5,5);H
  [,1] [,2] [,3] [,4] [,5]
[1,]  1    2    3    4    5
[2,]  2    3    4    5    6
[3,]  3    4    5    6    7
[4,]  4    5    6    7    8
[5,]  5    6    7    8    9
```

Illustrazione

I costi unitari di trasferimento ed il numero di spedizioni da quattro magazzini per tre punti vendita sono i seguenti

	Costi unitari				Spedizioni				Costi totali			
	M ₁	M ₂	M ₃	M ₄	M ₁	M ₂	M ₃	M ₄	M ₁	M ₂	M ₃	M ₄
V ₁	10	15	9	7	4	8	7	2	40	120	63	14
V ₂	14	8	12	8	1	0	2	9	14	0	24	72
V ₃	6	14	22	17	0	5	3	6	0	70	66	102

La matrice dei costi totali è ottenuta come prodotto di Hadamard tra la matrice dei costi unitari per la matrice del numero di spedizioni

```
CU<-matrix(c(10,15,9,
7,14,8,12,8,6,14,22,17),nrow=3,ncol=4,byrow=TRUE,dimnames=list(c("V.1","V.2","V.3"),c("M1","M2","M3","M4")))
```

- 1) Costruire la Matrice
- 2) Realizzare il prodotto di Hadamard
- 3) Realizzare il prodotto t(CU) per SPE e interpretare il risultato

1) Mostrare che le due matrici

$$A = \begin{bmatrix} 1 & 5 & -5 \\ 3 & 2 & -3 \\ 6 & -2 & -5 \end{bmatrix}; \quad B = \begin{bmatrix} -3 & 2 & -6 \\ -3 & 5 & -7 \\ -2 & 3 & -4 \end{bmatrix}$$

Hanno lo stesso determinante. (-5)

2) E' plausibile che due matrici diverse abbiano lo stesso determinante?

3) Calcolare il determinante delle due trasposte e del prodotto delle due matrici

1.4 Norms

A *norm* on a vector space V over $\mathbb{C}$ is a real valued function $\| \cdot \|$ on V satisfying three axioms:

1. $\|v\| \geq 0$ for all $v \in V$ and $\|v\| = 0$ if and only if $v = 0$.
2. $\|cv\| = |c| \|v\|$ for all $c \in \mathbb{C}$ and $v \in V$.
3. $\|u + v\| \leq \|u\| + \|v\|$ for all $u, v \in V$ (*triangle inequality*).

$$\|x\|_1 = \sum_{i=1}^n |x_i|$$

$$\|x\|_2 = \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2}$$

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

1) Manhattan: $\sum |x_i|$

2) Euclidean: $\sqrt{\sum x_i^2}$

3) Minkowski

4) Thebychev

Frobenius: $\text{tr}(A'A)$

$$\|A\|_F = \left(\sum_i \sum_j |a_{ij}|^2 \right)^{1/2}$$

```
> X
      [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
```

La funzione `solve()` serve per risolvere sistemi di equazioni lineari, ma può essere utilizzata per il calcolo della matrice inversa

```
> solve(X) #1'inversa
      [,1] [,2] [,3]
[1,]    1  0.0 0.0000000
[2,]    0  0.5 0.0000000
[3,]    0  0.0 0.3333333
> X%*%solve(X) #...verifica
      [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

essendo `%*` l'operatore 'prodotto-matriciale' (riga $\times$ colonna)

In una matrice è possibile sostituire completamente una linea (riga o colonna), ammesso che le dimensioni corrispondano. Ad esempio, per la seconda riga:

```
> x[2,]<-rep(2,5)
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    2    2    2    2
```

A differenza dei vettori, le matrici sono caratterizzate da una coppia di numeri, e la funzione `dim()` è utilizzata per restituire il numero di righe e colonne

INVERSA

```
library(MASS)
ginv(A)
```

Soluzione di un sistema lineare

$$\begin{cases} 5x_1 + 3x_2 + 2x_3 = 4 \\ 3x_1 + 9x_2 + 8x_3 = 6 \\ 4x_1 + 2x_2 + 2x_3 = 3 \end{cases}$$

```
A<-matrix(c(5,3,2,3,9,8,4,2,2),nrow=3, byrow=T)
b<-c(4,6,3)
print(A);print(b)
Sol<-solve(A,b)
print(round(Sol,2))
library(MASS)
A1<-ginv(A)
Sol1<-A1 %*% b
print(round(Sol1,2))
```

```
> A<-matrix(c(5,3,2,3,9,8,4,2,2),nrow=3, byrow=T)
> b<-c(4,6,3)
> print(A);print(b)
      [,1] [,2] [,3]
[1,]    5    3    2
[2,]    3    9    8
[3,]    4    2    2
[1] 4 6 3
> Sol<-solve(A,b)
> print(round(Sol,2))
[1] 0.5 0.5 0.0

> library(MASS)
> A1<-ginv(A)
> Sol1<-A1 %*% b
> print(round(Sol1,2))
      [,1]
[1,]    0.5
[2,]    0.5
[3,]    0.0
```

2.3 Array

Così come le matrici possono intendersi come estensioni dei vettori, gli *array* costituiscono una estensione delle matrici. In un *array* (multidimensionale) ogni suo elemento è individuato da un vettore di indici (si ricordi che in vettori e matrici gli elementi sono individuati da uno e due indici rispettivamente). Ad esempio, in un *array* tridimensionale, ogni elemento è caratterizzato da una terna (i_1, i_2, i_3) . Sebbene in Statistica Applicata gli *array* possono trovare numerose applicazioni, in un approccio base tali elementi possono essere trascurati. Soltanto per completezza qualche codice per la gestione degli *array* è riportato sotto.

```
> a<-array(1:24, dim=c(3,4,2))
> a[, ,2]
      [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
> a[1, ,]
      [,1] [,2]
[1,]     1    13
[2,]     4    16
[3,]     7    19
[4,]    10    22
> a[1,2,1]
[1] 4
> dim(a)
[1] 3 4 2
```

Per cercare di fissare le idee, un *array* $3 \times 4 \times 2$ può essere pensato come '2 matrici 3×4 una dietro l'altra': ad esempio tali due matrici potrebbero rappresentare una distribuzione doppia in ciascuno dei livelli di un confondente. L'uso delle parentesi quadre, alla stregua di vettori e matrici, ha il fine di selezionare sotto-insiemi dell'*array*.

Gli strati

Matrici e Strati

Matrice: Tabella riga per colonna di dati omogenei

Example: I costi di un servizio in alcune città e alcuni periodi.

	Venezia	Palermo	Milano
Lunedì	114	102	104
Martedì	112	100	101
Mercoledì	109	104	106
Giovedì	109	99	100
Venerdì	110	95	103
Sabato	109	104	101
Domenica	111	98	109

Strato: Tabella a 3, 4, ... dimensioni di dati omogenei

Example: I costi di più servizi in alcune città e alcuni periodi

	Servizio_C	Venezia	Palermo	Milano
Lunedì		112	103	102
Martedì		105	100	104
Servizio_B				
Lunedì		114	99	108
Martedì		107	96	108
Mercoledì		114	96	107
Servizio_A				
Lunedì		109	99	104
Martedì		111	102	102
Mercoledì		109	104	109
Giovedì		108	101	103
Venerdì		111	102	104
Sabato		113	98	101
Domenica		109	97	109

Liste

- La lista è uno strumento molto potente a disposizione dell'utente di R.
- Consente di combinare oggetti di natura e tipologia diversa

```
> doe <-
  list(name="John", age=28, married=F, cars=c("bmw", "Honda", "Ford"))
```

Gli elementi della lista si individuano premettendo il nome della lista seguita dal segno `$` e poi dal nome del soggetto desiderato

```
> doe$name
[1] "John"
> doe$age
[1] 28
```

- Se un soggetto della lista è un vettore, si potrà accedere agli elementi interni con l'indice tra parentesi quadre.

```
> doe$car[2]
[1] "Honda"
```

Il comando apply()

Effettua una certa operazione su tutte le righe o su tutte le colonne di una matrice

```
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
rownames(x) <- letters[1:8]
apply(x, 2, mean, trim = .2)
col.sums <- apply(x, 2, sum)
row.sums <- apply(x, 1, sum)
```

Effetto margini

```
rbind(cbind(x, Rtot = row.sums), Ctot =
c(col.sums, sum(col.sums)))
```

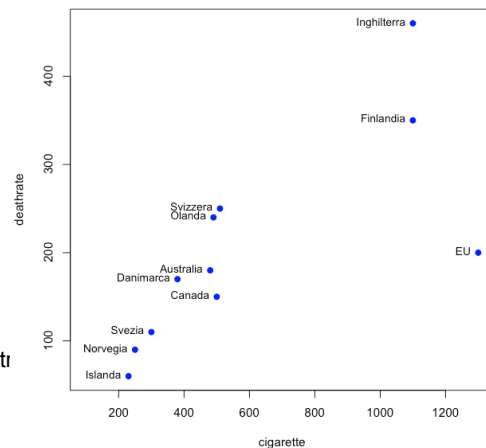
Esempio: numeri indici sintetici

```
#####
#Esempio sui numeri indici sintetici
datip<-matrix(0,3,5);datiq<-matrix(0,3,5)
datip[1,]<-c(5200,2800,5800,6100,9870)
datiq[1,]<-c(46224,14510,6265,2158,780)
datip[2,]<-c(5500,2950,6400,6400,10160)
datiq[2,]<-c(48128,15188,7128,2297,690)
datip[3,]<-c(6000,3150,6950,6900,11540)
datiq[3,]<-c(51053,16873,7059,2325,730)
datip85<-rbind(datip[1,],datip[1,],datip[1,])
datiq85<-rbind(datiq[1,],datiq[1,],datiq[1,])
M2<-datip85 * datiq #denominatore di Paasche
M3<-datip * datiq85 #numeratore di Laspeyres
M1<-datip * datiq # numeratori di Paasche e denominatore di Laspeyres
Npas<-apply(M1,1,sum)
Dpas<-apply(M2,1,sum)
Nlas<-apply(M3,1,sum)
Dlas<-rep(Npas[1,3])
PAS<-Npas*100/Dpas
LAS<-Nlas*100/Dlas
index<-cbind(LAS,PAS)
rownames(index)<-c(1985,1990,1995)
colnames(index)<-c("Laspeyres","Paasche")
print (round(index,2))
```

Scatterplot

Consumo di sigarette pro-capite nel 1930 e morti di tumore al polmone nel 1950

```
country<-c("Australia", "Canada",
"Danimarca", "Finlandia",
"Inghilterra", "Islanda", "Olanda",
"Norvegia", "Svezia", "Svizzera",
"EU")
cigarette<-c(480,500,380,1100,
1100,230,490,250,300,510,1300)
deathrate<-c(180,150,170,350,
460,60,240,90,110,250,200)
plot(cigarette,deathrate,pch=19,
col="blue",xlim=c(100,1300))
text(cigarette,deathrate,label=country,
cex=0.9,pos=2)
```

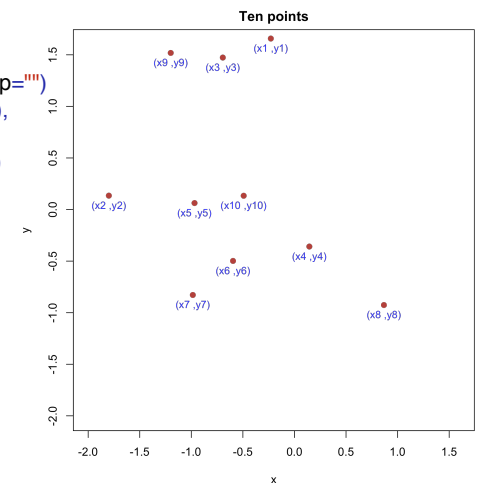


Esempio: numeri indici sintetici/2

```
#####
#Esempio sui numeri indici dell'occasione tipica
datip<-matrix(0,3,4);datiq<-matrix(0,3,4)
datip[1,]<-c(2800,3500,3900); datiq[1,]<-c(32,40,45)
datip[2,]<-c(3000,4000,4500); datiq[2,]<-c(150,195,184)
datip[3,]<-c(3200,4300,4700); datiq[3,]<-c(120,130,140)
datip[4,]<-c(3500,3800,4200); datiq[4,]<-c(25,27,31)
M1<-1/datiq
Har<-3/apply(M1,2,sum)
datis<-rbind(Har,Har,Har)
M2<-datip * datis
Numtip<-apply(M2,1,sum)
Otmh<-Numtip*100/Numtip[2]
print(Otmh)
#####
Geo<-(apply(datiq,2,prod))^(1/3)
datis<-rbind(Geo,Geo,Geo)
M2<-datip * datis
Numtip<-apply(M2,1,sum)
Otmg<-Numtip*100/Numtip[2]
print(Otmg)
index<-cbind(Otmh,Otmg)
rownames(index)<-c(1988,1989,1990)
colnames(index)<-c("Tipica:armonica","Tipica:geometrica")
print (index)
```

Esempio

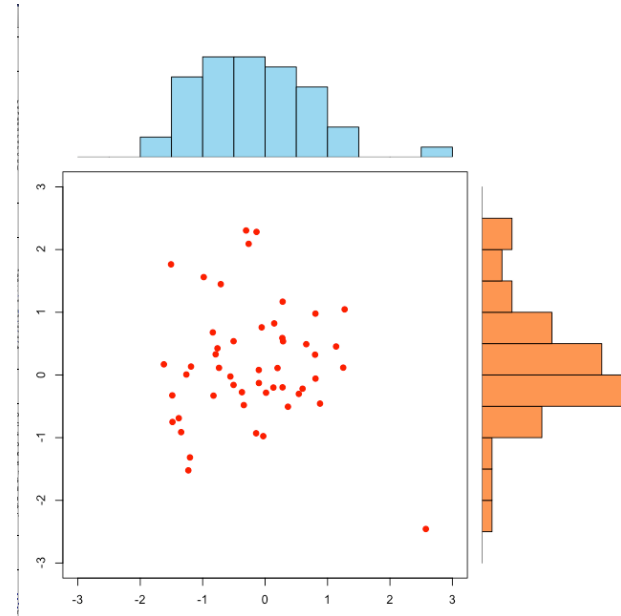
```
par(mfrow=c(1,1),mar=c(4,4,2,2))
set.seed(820731)
x <- rnorm(10)
y <- rnorm(10)
l <- paste("x",1:10,"y",1:10,"", sep="")
plot(x,y,xlim=c(-2,1.6),ylim=c(-2,1.6),
pch=19,col="brown")
text(x,y,l,pos=1,col="blue",cex=0.9)
title("Ten points")
```



Scatterplot con istogrammi a lato

```
set.seed(271828)
x <- pmin(3, pmax(-3, stats::rnorm(50)))
y <- pmin(3, pmax(-3, stats::rnorm(50)))
xhist <- hist(x, breaks=seq(-3,3,0.5), plot=FALSE)
yhist <- hist(y, breaks=seq(-3,3,0.5), plot=FALSE)
top <- max(c(xhist$counts, yhist$counts))
xrange <- c(-3,3); yrange <- c(-3,3)
nf <- layout(matrix(c(2,0,1,3),2,2,byrow=TRUE), c(3,1),
c(1,3), TRUE)
layout.show(nf)par(mar=c(3,3,1,1))
plot(x, y, xlim=xrange, ylim=yrange, xlab="",
ylab="",pch=19,col="red")
par(mar=c(0,3,1,1))barplot(xhist$counts, axes=FALSE,
ylim=c(0, top), space=0,col="skyblue")
par(mar=c(3,0,1,1))barplot(yhist$counts, axes=FALSE,
xlim=c(0, top), space=0, horiz=TRUE,col="sienna1")
par(def.par)#- reset to default
```

Notare l'impatto del
valore anomalo



Esempio sulla regressione

Input the data from the cigarette example

```
cig<-c(480,500,380,1100,1100,230,490,250,300,510,1300)
death<-c(180,150,170,350,460,60,240,90,110,250,200)
cor(cig,death)
```

will produce the correlation 0.737345

To estimate the parameters use the `lm` command ('linear model')
`lm(death~cig)`

The output is

```
Coefficients:
(Intercept) cig
 67.5609    0.2284
```

`summary(lm(death~cig))` gives a more detailed output

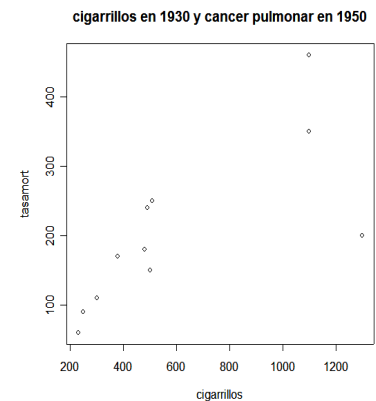
Aggiunta di informazioni

Adding titles. After obtaining the plot
with

```
plot(cigarette,deathrate)  
we write
```

```
title(main="cigarettes in 1930 and  
lung cancer in 1950")
```

b. In some plots, the title is included in
the command itself.
`hist(cigarette, main="percapita cigarette
consumption in 1930")`



Aggiunta di informazioni/2

After we have obtained the scatter plot we can use the command **abline** to add a line; we need to indicate the intercept and the slope of the line.

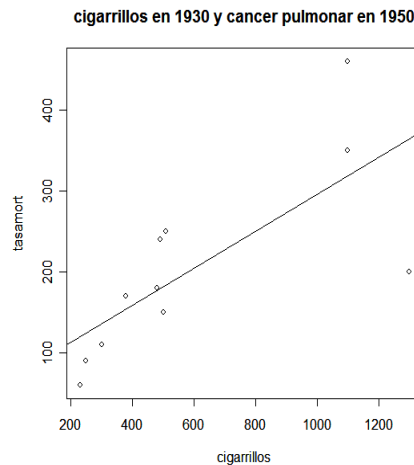
Example:

First to get the plot

```
plot(cigarette,deathrate)
```

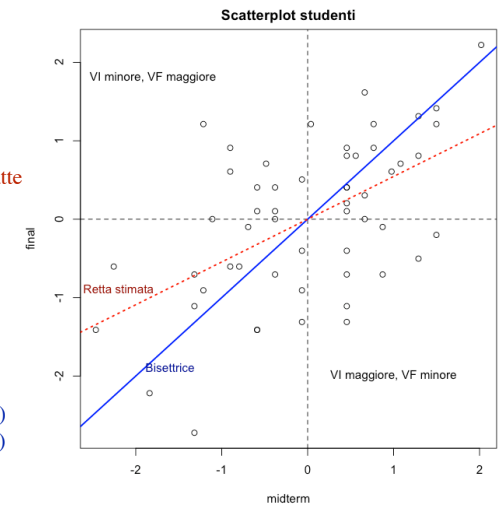
and then we write

```
abline(67.56 , 0.22844)
```



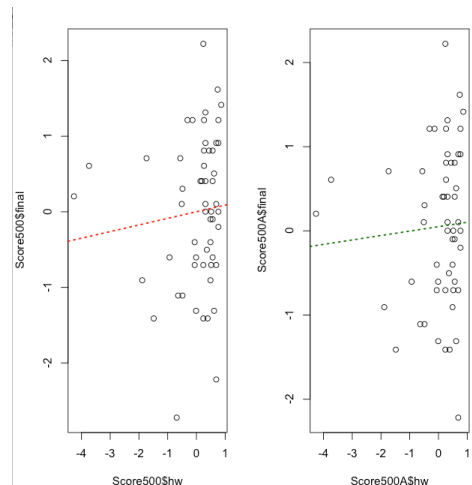
Regressione OLS

```
par(mfrow=c(1,1),mar=c(4,4,2,2))
library(faraway)
data(stat500)
Score500<-scale(stat500)
Score500<-data.frame(Score500)
names(Score500)
plot(final~midterm,Score500,main="Scatte
rplot studenti")
abline(h=0,lty=2);
abline(v=0,lty=2)
text(-1.8,1.8,"VI minore, VF maggiore")
text(1,-2,"VI maggiore, VF minore")
abline(0,1,col="blue",lwd=2)
text(-1.6,-1.9,"Bisettrice",col="navy")
Stim<-lm(final~midterm,Score500)
abline(Stim$coef,lty=3,lwd=2,col="red")
text(-2.2,-0.9,"Retta stimata",col="red4")
```



Regressione OLS/2

```
par(mfrow=c(1,2),mar=c(4,4,2,2))
cor(Score500)
plot(midterm,total)
#####
par(mfrow=c(1,2),mar=c(4,4,2,2))
plot(Score500$hw,Score500$final)
Ols<-lm(final~hw,data=Score500)
summary(Ols)
abline(Ols)
$coef,lty=3,lwd=2,col="red")Score500
A<-data.frame(Score500[-47,])
plot(Score500A$hw,Score500A$final)
OlsA<-lm(final~hw,data=Score500A)
summary(OlsA)
abline(OlsA)
$coef,lty=3,lwd=2,col="darkgreen")
```



***** Lecture Code *****

```
## Pickup data visualization
pickup <- read.csv("pickup.csv");names(pickup)

## R's summary function is pretty clever
summary(pickup)
# Histograms
par(mfrow=c(1,3)) # break the plot into 1x3 matrix (mfrow="multiframe row-wise")
hist(pickup$year, col=grey(.4), border=grey(.9))
hist(pickup$miles, col=grey(.4), border=grey(.9))
hist(pickup$price, col=grey(.4), border=grey(.9))
# Scatterplots
par(mfrow=c(1,2))
plot(price~year, col=make, data=pickup, pch=20)
plot(price~miles, col=make, data=pickup, pch=20)
legend("topright", fill=1:3, legend=levels(pickup$make), cex=.7)
# Boxplot
par(mfrow=c(1,1))
plot(log(price)~make, data=pickup, col=8)
##### Housing data: just price (in $100,000) vs size (in 1000 sq.ft.) #####
size <- c(8,.9,1,1.1,1.4,1.4,1.5,1.6,1.8,2.2,2.4,2.5,2.7,3.2,3.5)
price <- c(70.83,74.93,89.58,85,114,95,100,138,111,124,161,172)
plot(size, price, pch=20)
print( n <- length(size) )

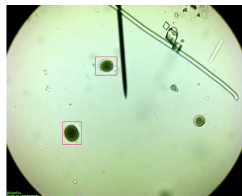
## Simple regression
reg <- lm(price ~ size)
b1 <- cor(price,size)*sd(price)/sd(size)
b0 <- mean(price) - mean(size)*b1
cbind(b0,b1)
```

```
# Labstat1. Esempi anche con regressione descrittiva
#### ***** Example 1: Rent Data ***** #####
rent <- read.csv("rent.csv")
par(mfrow=c(1,3))
boxplot(Rent ~ Bathrooms, data=rent)
boxplot(Rent ~ AC, data=rent)
boxplot(Rent ~ Parking, data=rent)
## Bathrooms looks like the most influential factor
plot(Rent ~ Rooms, data=rent, pch=20, col=as.factor(Bathrooms))
plot(Rent ~ YearBuilt, data=rent, pch=20, col=as.factor(Bathrooms))
plot(Rent ~ SqFt, data=rent, pch=20, col=as.factor(Bathrooms))
legend("topright", fill=1:3, legend=levels(as.factor(rent$Bathrooms)), title="Baths")
#
## Sq Ft looks the most influential, and it also correlates well with the # of bathrooms
par(mfrow=c(1,3))
boxplot(Rent, col=7, xlab="marginal", data=rent)
boxplot(Rent ~ Bathrooms, xlab="Bathrooms", main = "Rent Distribution", col=7, data=rent)
boxplot(Rent ~ Rooms, xlab="Rooms", col=7, data=rent)
# Looks like rent increases with the # of (bath)rooms
## There are some very high SqFt places which look like outliers
## removing these in R is very easy, but you could also just do it in excel
rent <- rent[rent$SqFt < 25]
## run the regression with 'lm'
reg <- lm(Rent ~ SqFt, data=rent)
summary(reg)
## Forecasting
yhat = reg$coef[1] + 14.8*reg$coef[2]
## clunky, but useful later with high-dimensional data
predict(reg, newdata=list(SqFt=14.8), se.fit=TRUE, interval="prediction")
```

Esempio_2: regressione lineare semplice

Dati sul saggio massimo di accrescimento (Rmax) di una alga *Chlorella vulgaris* in relazione alla intensità della luce (Light) misurata in μE per m^2 al secondo

```
Light<-c(20,20,20,20,21,24,44,60,90,94,101)
Rmax<-
c(1.73,1.65,2.02,1.89,2.61,1.36,2.37,2.08,2.69,2.32,3.67)
```

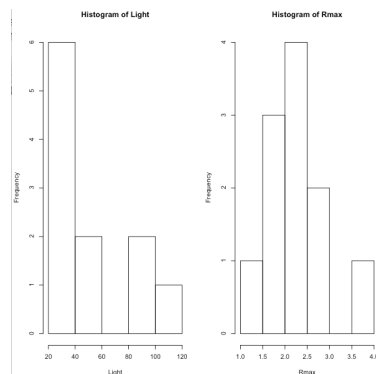


Primo passo di ogni indagine è l'analisi grafica. In Rmax c'è un interessante moda

```
par(mfrow=c(1,2),cex=0.7)
```

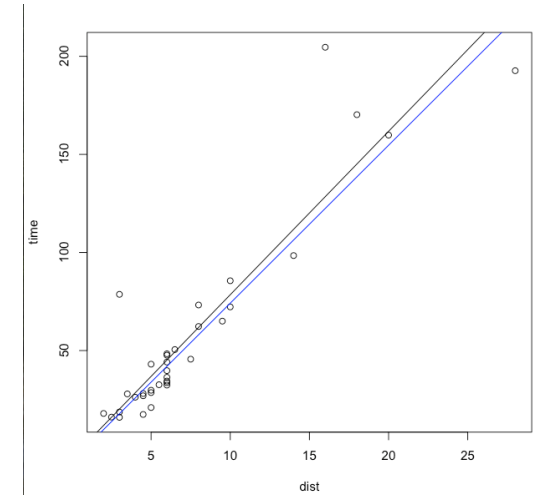
cex è un fattore di espansione/riduzione rispetto allo standard (minore di 1, rimpicciolisce)

```
hist(Light);hist(Rmax)
```



Regressione OLS e regressione robusta

```
library(MASS)
data(hills)
attach(hills)
plot(dist,time)
hillslm1<- lm(time~dist)
summary(hillslm1)
lines(abline(hillslm1))
hillslm2<- rlm(time~dist)
summary(hillslm2)
lines(abline(hillslm2,col="blue"))
```



```
fit<-lm(Rmax~Light);summary(fit)
summary(fit)
```

Esempio_2: continua

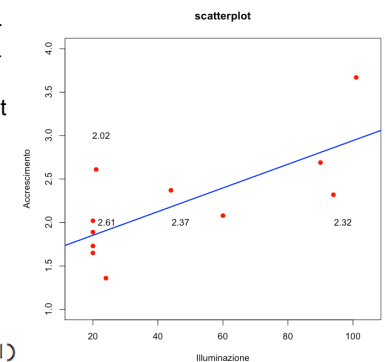
lm è l'istruzione che esegue la regressione (linear model). A sinistra la dipendente, una tilde di separazione, e poi il o i regressori. Infine, c'è il sommario. L'esito del calcolo è tutto nell'oggetto fit

```
Call:
lm(formula = Rmax ~ Light)

Residuals:
    Min       1Q   Median       3Q      Max
-0.5478 -0.2607 -0.1166  0.1783  0.7431
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.580952   0.244519   6.466 0.000116 ***
Light         0.013618   0.004317   3.154 0.011654 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.4583 on 9 degrees of freedom
Multiple R-squared: 0.5251, Adjusted R-squared: 0.4723
F-statistic: 9.951 on 1 and 9 DF, p-value: 0.01165
```



Cartografia Tematica: coroplete

```
library(maptools); library(lattice);library(sp);library(RColorBrewer)
Italy<- readShapePoly("prov2001_s") # Questo è il file di tipo .shp da reperire
Dat<-read.dbf("Proclus1.dbf") # Questo è il file dei dati aggiuntivi da graficare
j<-rank(Italy$PROVINCIA) # Abbinamento dei dati poligonal
Italy$CLUSTER<-as.factor(Dat$CLUSTER[j]) # e di osservazione
# Vari tipi di campitura e colorazione
Leg<-c(0.95,0.80,0.65,0.50,0.35,0.00);Colory<-gray(Leg)
#Colory<-brewer.pal(6,"Purples")
#Colory<-c("salmon1", "salmon3", "sienna3","red3", "brown", "black")
# Imposta la legenda
levels(Italy$CLUSTER)<-list("Fragile provinces"="1", "Provinces at risk"="2",
"Robust provinces"="3","Core provinces"="4","Smart provinces"="5","Metropolitn
areas"="6")
P<-
spplot(Italy,"CLUSTER",col.regions=Colory,main="",key.color=TRUE,col="gray40"
,par.settings = list(axis.line = list(col = "transparent")))
names(P$legend) <- "inside"
P$legend$inside$x <- 0.65
P$legend$inside$y <- 0.8
plot(P)
```

Cartografia tematica/2

```
sport<- readShapePoly("london_sport.shp")
names(sport)
colours<- brewer.pal(5, "Blues")
brks<-c(10, 15, 20, 25)
brks<-classIntervals(sport$Partic_Per,
n=5,style="quantile") plot(brks,pal=colours)
summary(brks)
brks<- brks$brks
plot(sport, col=colours[findInterval(sport$Partic_Per,
brks)], axes=F, asp=T) SpatialPolygonsRescale(layout.north.arrow(1),
offset= c(505100,160000), scale = 6000, plot.grid=F)
SpatialPolygonsRescale(layout.scale.bar(), offset= c(503800,154800),
scale= 10000, fill= c("transparent", "black"), plot.grid= F)
legend(x=548500, y=164800, legend=leglabs(brks), fill=colours, bty="n")
box()
title(paste ("London Sports Participation")) text(509000, 153500, "10KM",
cex= 1)
text(534000,152000, "Boundary Data Crown Copyright Ordnance Survey
2009.", cex= 1)
text(556500, 166000, "% Participation", cex= 1)
```

