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 sistem 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

Orientamento agli oggetti

Le tipologie elementari (atomic) in R sono

- **numeric (intere, doppia precisione, complesse)**
- **Character**
- **Logical**
- **Function**

In R, an object is anything that can be assigned to a variable. This includes constants, data structures, functions, and even graphs.

Objects have a mode (which describes how the object is stored) and a class (which tells generic functions like print how to handle it).

L'oggetto è un insieme si variabili elementari e di altri suboggetti (noti anche come slot) che condividono uno spazio di memoria e sono disponibili singolarmente o come complesso.

Gli oggetti appartengono alle class che ne caratterizzano la natura

Orientamento agli oggetti/2

Advantages:

Encapsulation Impacchettamento. Possiamo usare gli oggetti che altri hanno preparato senza preoccuparci di tutti i loro dettagli

Generic functions Sono diponibili comandi comuni per gli oggetti che semplificano la trattazione.

Inheritance Ereditarietà. Certe qualità degli oggetti si trasmettono ai loro suboggetti.

Caveat:
Esagerata sofisticazione, barocchismo nella architettura delle elaborazioni.

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 ancora disponibili sui pacchetti standard per 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) del software concorrente.

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.

"R is the most powerful & flexible statistical programming language in the world"[1]

- Capabilities
  - Sophisticated statistical analyses
  - Predictive analytics
  - Data visualization
- Applications
  - Real-time trading
  - Finance
  - Risk assessment
  - Forecasting
  - Bio-technology
  - Drug development
  - Social networks
  - .. and more





R User Community

From: The R Ecosystem
bit.ly/R-ecosystem



Local R User Groups (55)
http://blog.revolutionanalytics.com/local-r-groups.html

Video Rchive
http://www.vcasmo.com/user/drewconway

Revolutions
http://blog.revolutionanalytics.com

R-bloggers
R news and tutorials contributed by (231) R bloggers

github SOCIAL CODING
1100 repositories
https://github.com/languages/R/created

crantastic!

inside-R
http://www.inside-r.org

R users (2M+)

Quora

stackoverflow
6000 Questions
http://stackoverflow.com/questions/tagged/r

twitter
#rstats
http://bit.ly/r-twitter

REVOLUTION ANALYTICS

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

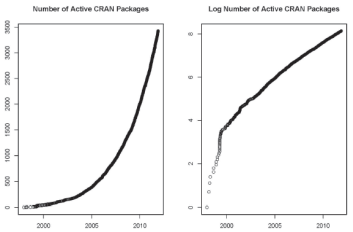
Prima di R



Students using computer terminal equipment

Breve storia

- 1991-93: Ross Dhaka and Robert Gentleman begin work on R project at U. Auckland
- 1995: R available by ftp under the GPL
- 96-97: mailing list and R core group is formed
- 2000: John Chambers, designer of S joins the Rcore (wins a prize for best software from ACM for S)
- 2001-2011: Core team continues to improve base package with a new release every 6 months.
- Many others contribute "packages" to supplement the functionality for particular problems
  - 2003-04-01: 250 packages
  - 2004-10-01: 500 packages
  - 2007-04-12: 1,000 packages
  - 2009-10-04: 2,000 packages
  - 2011-05-12 3,000 packages
  - 2012-08-23 4,000 packages



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/ModStatI/

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/iacus/corsoR/

Manualistica in inglese

Owen (2007) The R Guide,

http ://cran.r-project.org/doc/contrib/Owen-TheRGuide.pdf

Farnsworth () Econometrics in R ,

http ://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf

Short (2004) R reference card,

http ://cran.r-project.org/doc/contrib/Short-refcard.pdf

Verzani (2005) Using R for Introductory Statistics,

http ://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf

You can also find some lectures notes on R , almost everywhere, e.g.

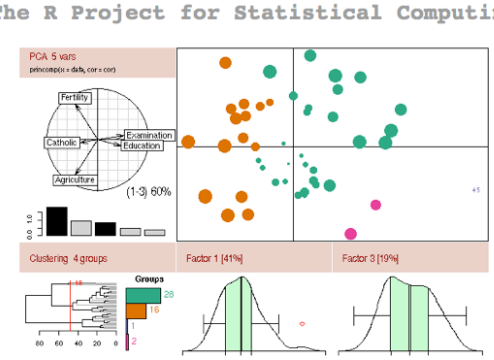
http ://gking.harvard.edu/zelig/docs/static/,

http ://www.math.ilstu.edu/dhkim/Rstuff/Rtutor.html,

http ://www.dangoldstein.com/flash/Rtutorial1/Rtutorial1.html.



Installazione



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.

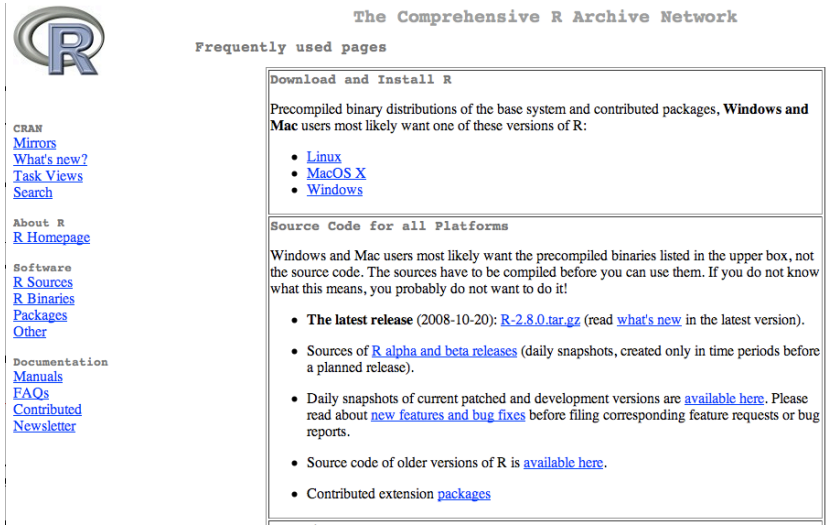
Argentina	
http://cran.patan.com.ar/	Patan.com.ar, Buenos Aires
Australia	
http://cran.ms.unimelb.edu.au/	University of Melbourne
Austria	
http://cran.at.r-project.org/	Wirtschaftsuniversitaet Wien
Belarus	
http://cran.promotionalpro.com/	www.ehost.by
Belgium	
http://www.freestatistics.org/cran/	K.U.Leuven Association
Brazil	
http://cran.br.r-project.org/	Universidade Federal do Parana
http://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
http://www.vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
http://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
Canada	
http://cran.stat.sfu.ca/	Simon Fraser University, Burnaby
http://probability.ca/cran/	University of Toronto
http://cran.skazkaforyou.com/	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/	Garr Mirror, Milano
http://cran.stat.unipd.it/	University of Padua
http://dssm.unipa.it/CRAN/	Universita degli Studi di Palermo

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

Pagina principale di R



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed
Newsletter

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

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

Link alla pagina per la versione Windows



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed
Newsletter

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

Link alla pagina per la versione Mac OS



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed
Newsletter

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. Altough 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 earlied 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.

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.

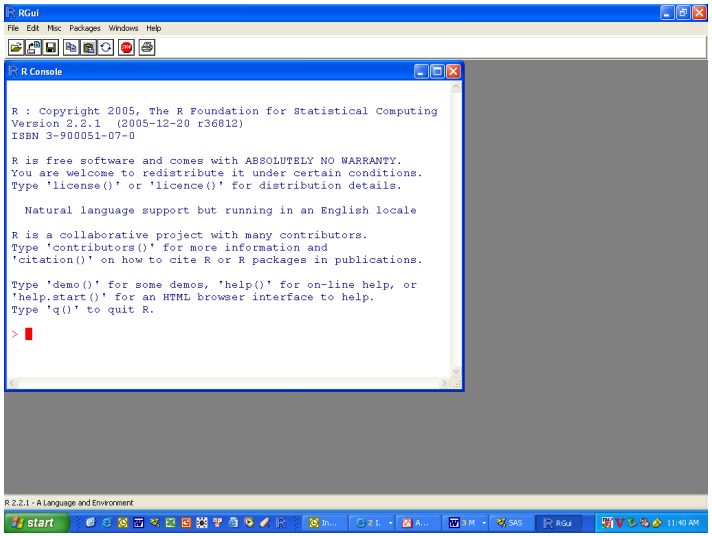
  Natural language support but running in an English locale

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)



Dice che usare R è difficile perché…

1. R doesn't have a GUI (Graphical User Interface)
   - Partly true, many use syntax
   - Partly not true, GUIs exist (e.g., R Commander, R-Studio)
   - Quasi GUIs for Mac and PCs make syntax writing easier
2. R syntax is hard to use
   - Not really, unless you think an iPhone is hard to use
   - Easier to give instructions of 1-4 lines of syntax rather than pictures of what menu to pull down.
   - Keep a copy of your syntax, modify it for the next analysis.
3. R is not user friendly: A personological description of R
   - R is introverted: it will tell you what you want to know if you ask, but not if you don't ask.
   - R is conscientious: it wants commands to be correct.
   - R is not agreeable: its error messages are at best cryptic.
   - R is stable: it does not break down under stress.
   - R is open: new ideas about statistics are easily developed.

Directory di lavoro

In R ogni operazione su file avviene a partire dalla directory corrente di lavoro, la quale `e visibile con ilcomando:

> getwd()

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

L'uso delle directory in R è di fondamentale importanza al fine di poter salvare e utilizzare successivamenteil 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

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**.

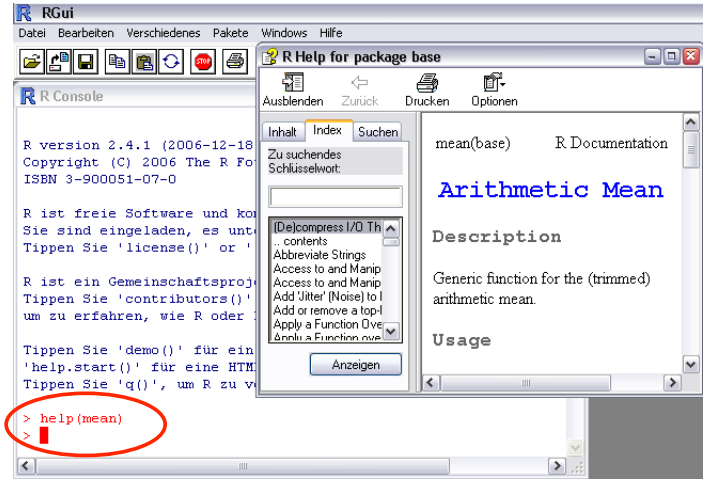
R distingue tra minuscole e maiuscole

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

Aiuti in R

- Help function

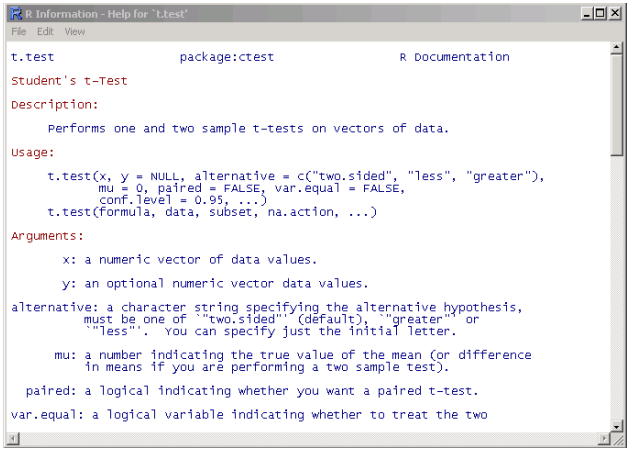


Aiuti in R/ continua

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



Riga di comando

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

Espressioni

Assegnazioni.

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

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

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

> summary(x)
Min. 1st Qu. Median Mean 3rd Qu. Max.
 2.000 3.000   4.000  4.889  7.000  9.000

> 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
```

```
Permulation: 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
```

Altri esempi

```
a <- 2     # create variable a, assign the number 2 to it.
class(a)                    # what is it?
is.numeric(a)               # is it a number?
b <- 4                      # create variable b, assign the number 4 to it.
a + b                       # addition
a - b                       # subtraction
a * b                       # multiplication
a / b                       # division
a ^ b                       # exponentiation
(a + b)^a                   # parentheses
a == b                      # logical test of equality
a < b                       # comparison
max(a,b)                    # largest
min(a,b)                    # smalles
order(c(a,b))               # return the indices of a and b in increasing order
c(a,b)[order(c(a,b))]       # return a and b in increasing order
a<- "string"                # create variable a, assign the value "string" to it.
is.numeric(a)               # is it a number?
is.character(a)             # is it a string?
b <- "ketchup"              # create variable b
paste(a, b)                 # join the strings
paste(a, b, sep="")         # join the strings with no gap
d <- paste(a, b, sep="_")
```

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)
```

- Nei nomi si possono 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

Ortografia di R/2

Il punto o il trattino basso possono comparire nei nomi senza difficoltà, ma il dollaro ($) separa un oggetto da un suboggetto e quindi non può essere usato come elemento del nome.

R non cosente commenti che dispongano su più linee. Per I commenti che occupino più linee è necessario cominciare la riga di commento con il cancelletto.

Le variabili non debbono essere predichiarate, ma nascono nel puno in cui ricevono la prima assegnazione

www.johndcook.com/R_language_for_programmers.html

http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html

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)

Carattero o stringa (tra apici)

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

Sub è un comando sulle stringhe

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

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

Logiche: TRUE/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)
[1] "double"
is.double(8.9)
[1] TRUE
test <- 1223.456
is.double(test)
[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)
y <- 2.0
z <- x/y
```

```
as.integer(2^31 - 1)
[1] 2147483647
as.integer(2^31)
[1] NA
Warning message:
NAs introduced by coercion

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
```

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

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

```
> 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 indicarla

```
> 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
```

N.B. "per default" significa "in automatico" oppure "pre-definito" cioè quello che si applica se non di indica esplicitamente di operare in modo diverso

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

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:

```
> 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
```

Max richiama il massimo in un oggetto conentente dei valori

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
```

Funzioni intrinseche

Sono richiamate a comando, senza ulteriori specificazioni

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

Standardizzazione

La standardizzazione costringe le nuove variabili ad avere varianza uno.

La unitarizzazione costringe le nuove variabili a variare nell'intervallo [0,1]

La divisione per la media porta al coefficiente di variazione.

$$\textit{Standardizzazione}: \frac{x}{\sigma(x)}$$

$$\textit{Unitarizzazione}: \frac{x}{\max(x)-\min(x)}$$

$$\textit{Rapporto alla media}: \frac{x}{|\mu(x)|}$$

$$\textit{Scarto medio assoluto}: \frac{x-Me(x)}{n^{-1}\sum_{r=1}^{n}|x_r-Me(x)|}$$

```
scale(x, center = TRUE, scale = TRUE)
Arguments
x        a numeric matrix(like object).
Center   either a logical value or a numeric vector of length equal to the number of columns of x.
Scale    either a logical value or a numeric vector of length equal to the number of columns
```

```
scale(x,center=T,scale=T)
scale(x,center=min(x),scale=range(x)[2]-range(x)[1])
scale(x,center=F,scale=abs(mean(x)))
scale(x,center=median(x),scale=mean(abs(x-median(x))))
```

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'
```

```
> log(100)
[1] 4.60517
> log2(16)     # same as log(16,base=2) or just log(16,2)
[1] 4
> log(1000,base=10)     # same as log10(1000)
[1] 3
>
```

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
```

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) Exponenziazione
(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

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
Esclude la parte decimale di un numero

Signif
Arrotonda al numero di ciffre significative indicate

Per fissare il numero di cifre da visualizzare si usa il comando

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
```

Operatori di R

Operatori relazioni	Significato
`x < y`	minore di
`x > y`	maggiore di
`x <= y`	minore o uguale di
`x >= y`	maggiore o uguale di
`x == y`	uguale a
`x != y`	non uguale a

Operatori logici	Significato
`!x`	negazione logica (NOT)
`x & y`	intersezione logica (AND)
`x | y`	unione logica (OR)
`xor(x, y)`	OR eslcusivo (XOR)

Esempi sugli operatori logici

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

Operatore & (and)

```
# OPERATORE and
#confroni 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)
#confroni 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)
```

```
> NA | TRUE
[1] TRUE
> NA & TRUE
[1] NA
```

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"
```

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.

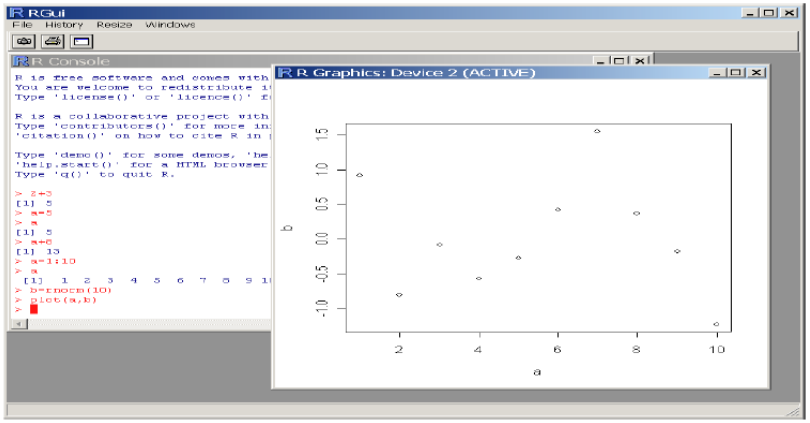
Ogni sistema ha il suo modo di rendere disponibili gli script (win ha il notepad). Anche Word va bene.

```
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)
```

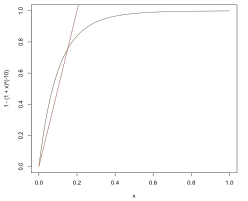
Finestra dei grafici



Applicazione elementare

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

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



```
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)
```

Inserimento dati da istruzione

Scegliere il nome del'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    Netherland
      480       500      380      1100      1100      230       490
   Norway    Sweden Switzerland      USA
      250       300       510      1300
```

Esempio

Data set relativo al livello delle piene del fime 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 potarlo nello spazio di lavoro di R

DIGITAZIONE DEI DATI (naif)

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, 45.1, 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)

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

L'oggetto River è ora 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
```

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 di grandezza generale del fenomeno

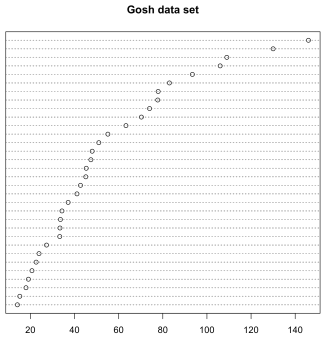
Preliminari

# Diagramma ramo foglia
stem(River)

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

```
 0 | 4589
 2 | 134733447
 4 | 13557815
 6 | 30488
 8 | 33
10 | 69
12 | 0
14 | 6
```

# Cleveland dot chart
dotchart(River, main = "Gosh data set")



# Riassunto dei dati
summary(River)

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
14.10   31.80   45.20   54.34   74.92  146.00
```

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.

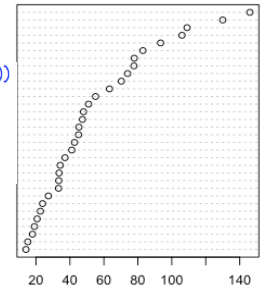
Funzioni statistiche

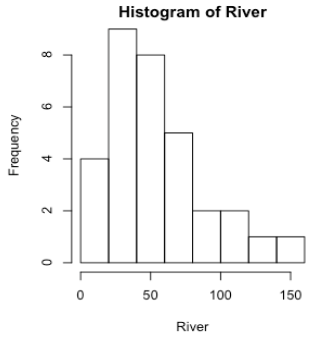
Per calcolarne la media, la varianza, la deviazione standard e la mediana si usano le seguenti funzioni:
> mean(River)
> var(River)
> sd(River)
> median(River)
Il numero dei casi di River, il valore massimo e il minimo si ottengono con le chiamate:
> length(River)
> max(River)
➢min(River)

```
mad(x, center = median(x), constant = 1.4826, na.rm = FALSE,
    low = FALSE, high = FALSE)
```

Esempio

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

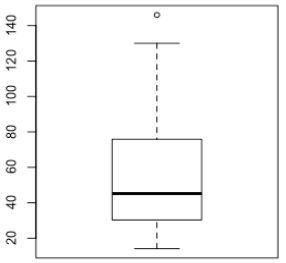


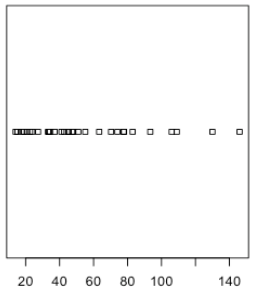


I quattro comandi richiamano altrettanti grafici standard per la descrizione dei dati.

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





Esempio

Data set predefinito

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

```
> stem(duration)

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

  8 | 3
 10 |
 12 |
 14 |
 16 | 223370023357778
 18 | 000222233333355577788800223333335555577778
 20 | 00000000000000000000000000223578023578
 22 | 0278
 24 | 7807
 26 | 05
 28 | 373
 30 | 00
 32 | 583
 34 | 523
 36 | 00235
 38 | 0277802377
 40 | 00000000000000000000000000
```

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
```

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

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/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 cartteri
> gg <- c("1",3,4)
> is.vector(gg)
[1] TRUE
> gg
[1] "1" "3" "4"
> is.character(gg)
[1] TRUE
```

Costruzione di vettori con cut

La funzione cut serve per assegnare i valori di un vettore a classi di ampiezza (dicretizzazione 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] (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 Secon Terzo Primo Primo Terzo Secon Primo Primo
[14] Quart Quart
Levels: Primo Secon Terzo Quart
```

Accesso agli elementi del vettore

Si deve indicare la posizione cercata tra parentesi quadre

Gli indici in indirizzo posono 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
```

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 <- 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

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

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
```

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")
> x1 <- 25:30
> length(mydata)
[1] 4
> mode(mydata)
[1] "numeric"
> mode(x1)
[1] "numeric"
> colour[2:3]
[1] "green" "blue"
> x1[-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(x1, 2, 3)
     [,1] [,2] [,3]
[1,]   25   27   29
[2,]   26   28   30
> matrix(x1, 2, 3, byrow=T)
     [,1] [,2] [,3]
```

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,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 triagonal entries into a vector

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 tralasciati. 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_A	Venezia	Palermo	Milano
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

Servizio_B	Venezia	Palermo	Milano
Lunedì	114	99	108
Martedì	107	96	108
Mercoledì	114	96	107
Giovedì			100
Venerdì			109
Sabato			107
Domenica			101

Servizio_C	Venezia	Palermo	Milano
Lunedì	112	103	102
Martedì	105	100	104
Mercoledì			104
Giovedì			108
Venerdì			105
Sabato			103
Domenica			102

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"
```

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

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

```
> 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

> 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 speficarlo 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/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
```

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 rst. 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
```

La matrice dei dati

Il risultato di molti percorsi di ricerca è l'individuazione della matrice dei dati, X, che costituirà poi l'oggetto dell'analisi.

Si tratterà di una matrice rettangolare di dimensioni (n x m) , in cui le n righe saranno costituite dalle unità oggetto di indagine, e le m colonne rappresenteranno le variabili che sono stati rilevate per ciascuna unità.

Vettori colonna
Per le variabili

Vettori riga
per le unità

	Indicatori					
Unità	X_1	X_2	.....	X_j	.....	X_m
I_1	x_{11}	x_{12}	.....	x_{1j}	.....	x_{1m}
I_2	x_{21}	x_{22}	.....	x_{2j}	.....	x_{2m}
.....	.....	.....	.....	.....	.....	.....
I_i	x_{i1}	x_{i2}	.....	x_{ij}	.....	x_{im}
.....	.....	.....	.....	.....	.....	.....
I_n	x_{n1}	x_{n2}	.....	x_{nj}	.....	x_{nm}

Con X_{ij} possiamo indicare quindi il valore che il j-esimo indicatore di base assume nell'unità i-esima

DATA APPENDIX

Country	y_{1980}	y_{2000}	n	$\bar{y}_R$	s_R
Argentina	11747	17494	1.4	17.6	5.7
Australia	17245	38079	1.8	24.7	11.1
Austria	11263	34566	0.4	26.1	8.3
Burundi	1073	1158	1.9	5.0	0.5
Belgium	12065	35625	0.3	24.6	10.2
Benin	2148	2248	2.5	6.5	1.9
Burkin Faso	1281	1898	1.9	8.4	0.7
Bangladesh	1964	3855	2.5	10.0	3.9
Bolivia	3975	4854	2.4	10.1	5.0
Brazil	4644	10901	2.7	20.6	4.8
Canada	17247	39512	1.7	21.9	10.5
Switzerland	21154	39476	0.8	27.8	6.0
Chile	7103	15097	2.1	16.0	7.3
Cote d'Ivoire	3686	3464	3.7	8.2	2.6
Cameroon	3158	4051	2.4	6.9	3.5
Congo	1191	4394	2.6	23.0	5.1
Colombia	4840	8634	2.8	11.5	8.0
Costa Rica	7158	8753	3.5	14.2	6.3
Denmark	16786	40343	0.5	23.4	10.8
Dominican Rep.	3354	8472	2.9	12.4	5.6
Algeria	4729	9247	3.0	17.9	5.8
Ecuador	4113	5650	3.0	20.1	7.0
Egypt	3006	6813	2.5	7.0	8.1
Spain	7222	26556	0.8	24.5	9.6
Ethiopia	1026	1294	2.5	4.4	1.3
Finland	11850	34318	0.6	26.4	11.4
France	12903	34931	0.8	24.7	9.6
United Kingdom	14819	35009	0.3	18.3	9.8
Ghana	3273	2297	2.8	10.1	4.8
Greece	6178	21747	0.7	25.8	8.6
Guatemala	5090	7656	2.7	8.0	2.9
Hong Kong	4946	36058	2.6	25.8	6.6
Honduras	3214	3678	3.2	12.3	4.0
Indonesia	1354	5811	2.3	12.2	4.7
India	1521	4028	2.3	11.6	5.2
Ireland	9029	38195	1.1	17.9	12.8
Israel	10283	28838	2.8	28.2	9.2
Italy	10328	31764	0.4	24.9	7.3
Jamaica	4343	5249	1.5	19.1	10.2
Jordan	3911	6940	4.7	13.1	9.7
Japan	7386	35982	0.9	31.0	10.6
Kenya	1764	2277	3.4	11.1	2.9
Korea	2685	19552	2.3	27.3	9.7

Esempio

Analisi di un modello di crescita

Nel corso di base di statistica si incontrano uno o due variabili

Variabili

Unità

Detta anche

"osservazione"

Di solito le righe cominciano con un identificatore di unità detto row.names

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
```

A data frame is a structure in R that holds data and is similar to the datasets found in standard statistical packages (for example, SAS, SPSS, and Stata).

The columns are variables and the rows are observations. You can have variables of different types (for example, numeric, character) in the same data frame. Data frames are the main structures you'll use to store datasets.

Caricamento dati da istruzione

Manager	Date	Country	Gender	Age	q1	q2	q3	q4	q5
1	10/24/08	US	M	32	5	4	5	5	5
2	10/28/08	US	F	45	3	5	2	5	5
3	10/01/08	UK	F	25	3	5	5	5	2
4	10/12/08	UK	M	39	3	3	4		
5	05/01/09	UK	F	99	2	2	1	2	1

```
manager <- c(1, 2, 3, 4, 5)
date <- c("10/24/08", "10/28/08", "10/1/08",
"10/12/08", "5/1/09")
country <- c("US", "US", "UK", "UK", "UK")
gender <- c("M", "F", "F", "M", "F")
age <- c(32, 45, 25, 39, 99)
q1 <- c(5, 3, 3, 3, 2)
q2 <- c(4, 5, 5, 3, 2)
q3 <- c(5, 2, 5, 4, 1)
q4 <- c(5, 5, 5, NA, 2)
q5 <- c(5, 5, 2, NA, 1)
leadership <- data.frame(manager, date,
country, gender, age,
q1, q2, q3, q4, q5, stringsAsFactors=FALSE)
```

```
  manager     date country gender age q1 q2 q3
q4 q5
1       1 10/24/08      US      M  32  5  4  5  5  5
2       2 10/28/08      US      F  45  3  5  2  5  5
3       3  10/1/08      UK      F  25  3  5  5  5  2
4       4 10/12/08      UK      M  39  3  3  4 NA NA
5       5   5/1/09      UK      F  99  2  2  1  2  1
```

Esempio

```
### Costituzione di un data frame
# Il data frame è una tabella riga/colonna in cui le unità solo sulle righe
# e le variabili sulle colonne. E' 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" )
```

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 tempertura 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 16.0 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

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

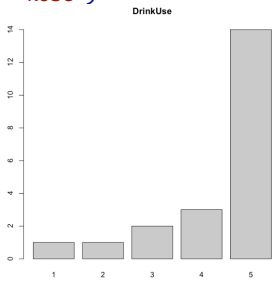
```
# Questa lettura non va bene perché i nomi sulle righe sono ripetuti.
 Demog<-read.table("demoitalia.txt",sep="",dec=".",header=TRUE,row.names=1)
#
Demog<-read.table("demoitalia.txt",sep="",dec=".",header=TRUE)
# Questa lettura va bene, ma ora la chiave del record diventa una variabile nominale
#
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

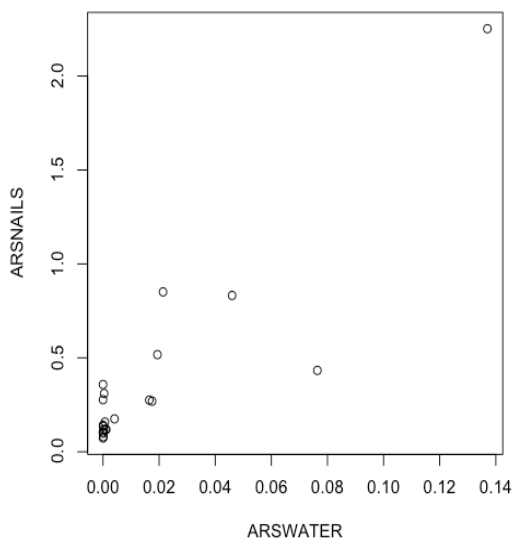
```
Ars<-Read.table("Arsenic.csv",sep=",",header=TRUE
attach(Ars)

# 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),main="Drin
kUse")
```




```
plot(ARSWATER,ARSNAILS)
```



Importare i dati come 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 separatatore può essere anche ";" o " "
```

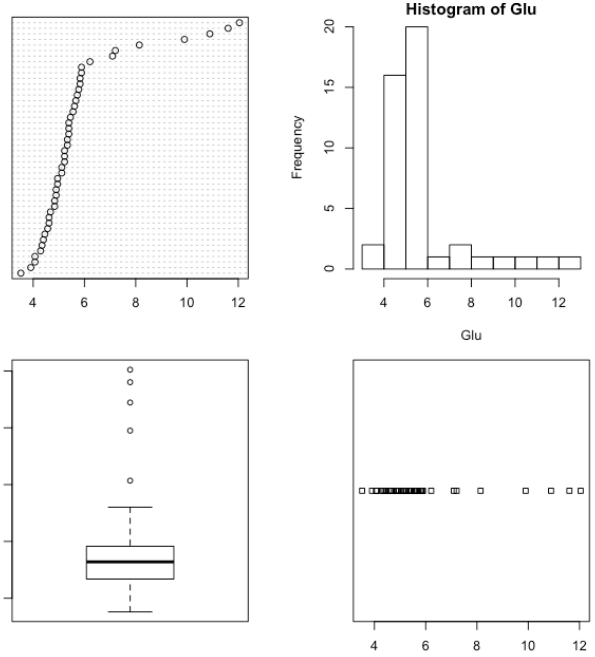
I valori confluiscono nell'oggetto Glu che si autoconfigura come vettore di valori.

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
```

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.

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

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
>
```

The best way to read an Excel file is to export it to a comma-delimited file from within Excel and import it to R using scan, read.table, read.csv

Lettura di una matrice di dati su file

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

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   :53.00   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
```

names, rownames, colnames

Per elencare la denominazione delle righe e/o delle colonne R dispone di comandi specifici.

A<-read.table("EserSM.csv",row.names=1,header=T,sep=",",dec=".")

> names(A)
[1] "Investimenti" "Consumi" "Semilavorati" "Scorte"

> colnames(A)
[1] "Investimenti" "Consumi" "Semilavorati" "Scorte

> rownames(A)
[1] "1992" "1993" "1994" "1995" "1996" "1997" "1998" "1999" "2000"
[10] "2001" "2002" "2003" "2004" "2005" "2006"

Anno	Investimenti	Consumi	Semilavorati	Scorte
1992	832.38	11.11	3.12	7499.72
1993	333.44	13.32	2.85	4448.63
1994	360.04	7.61	2.83	2755.55
1995	761.88	9.29	2.98	7092.21
1996	537.14	13.93	3.03	13279.41
1997	231	11.8	2.38	2728.44
1998	206.03	8.1	2.59	1673.06
1999	573.99	14.06	2.92	8078.62
2000	51.15	2.13	1.77	114.29
2001	778.8	10.95	3.13	8532.73
2002	284.97	4.87	2.57	1387.55
2003	209.16	2.29	2.56	490.86
2004	455.36	9.6	2.85	4386.65
2005	50.81	5.38	1.84	281.99
2006	271.81	15.94	2.53	3033.25

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
```

Ordinamento e suddivisione (Sort & subset)

```
# Sort & Merge
# Lettura file csvLarc<-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(rownames(Larc),decreasing=FALSE),];Lar
# 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=rownames(Larc)<2000);LarcUp
LarcDown<-subset(Larc, subset=rownames(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)
```

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

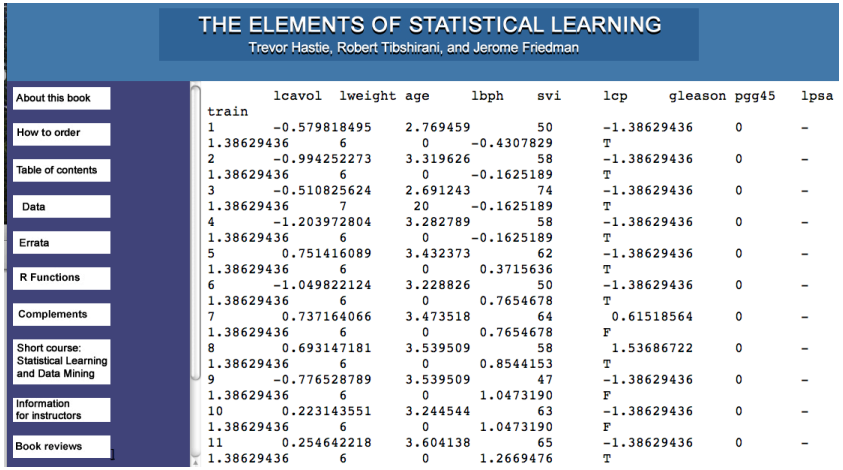
Operatività di R/4

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



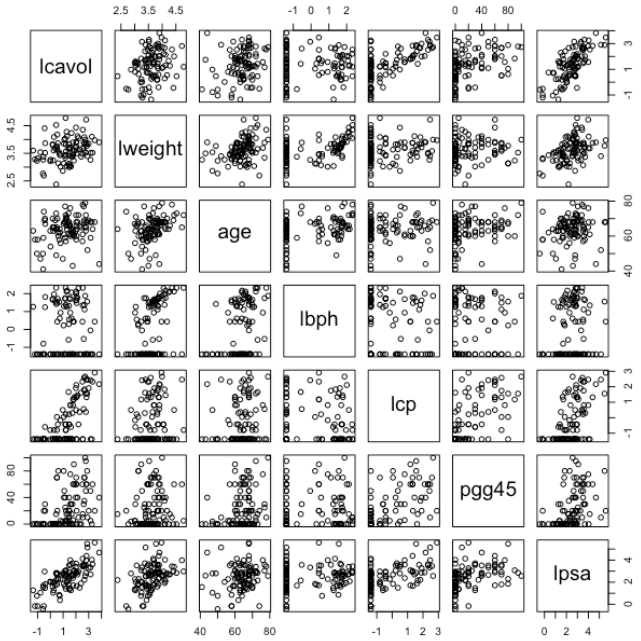

```
        lcavol  lweight age      lbph     svi      lcp      gleason pgg45   lpsa
train
1       -0.579818495    2.769459        50      -1.38629436     0       -
1.38629436      6         0     -0.4307829      T
2       -0.994252273    3.319626        58      -1.38629436     0       -
1.38629436      6         0     -0.1625189      T
3       -0.510825624    2.691243        74      -1.38629436     0       -
1.38629436      7        20     -0.1625189      T
4       -1.203972804    3.282789        58      -1.38629436     0       -
1.38629436      6         0     -0.1625189      T
5        0.751416089    3.432373        62      -1.38629436     0       -
1.38629436      6         0      0.3715636      T
6       -1.049822124    3.228826        50      -1.38629436     0       -
1.38629436      6         0      0.7654678      T
7        0.737164066    3.473518        64       0.61518564     0       -
1.38629436      6         0      0.7654678      F
8        0.693147181    3.539509        58       1.53686722     0       -
1.38629436      6         0      0.8544153      T
9       -0.776528789    3.539509        47      -1.38629436     0       -
1.38629436      6         0      1.0473190      F
10       0.223143551    3.244544        63      -1.38629436     0       -
1.38629436      6         0      1.0473190      F
11       0.254642218    3.604138        65      -1.38629436     0       -
1.38629436      6         0      1.2669476      T
```

Esempio

```
DataMat<-read.table
("Prostate.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



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)
```

Codifica

```
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)
```

Ricodifica

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

Basso medio alto ----> 1 2 3

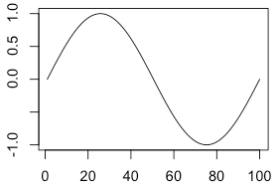
```
library(MASS)
data(Cars93)
A<-Cars93
print(A[,9])
A[,9]<-ifelse(A[,9]=="Driver & Passenger",3,ifelse(A[,9]==
"Driver only",2,1))
print(A[,9])
A[,11]<-ifelse(A[,11]=="rotary",6,A[,11])
A[,18]<-ifelse(A[,18]==2,3,A[,18]);A[,18]<-A[,18]-2;
A[,16]<-ifelse(A[,16]=="No",0,1)
A[,26]<-ifelse(A[,26]=="USA",1,0)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

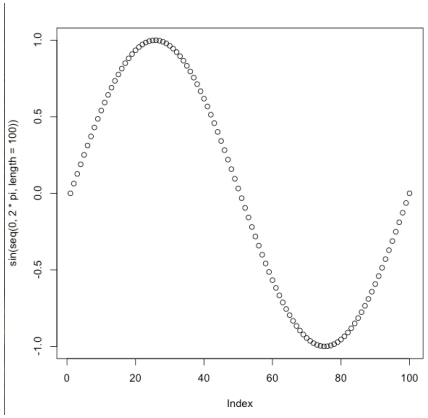
Plot

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

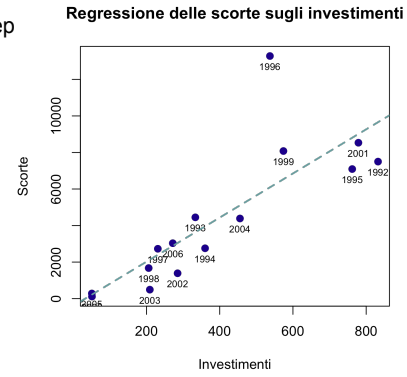




Esempio

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

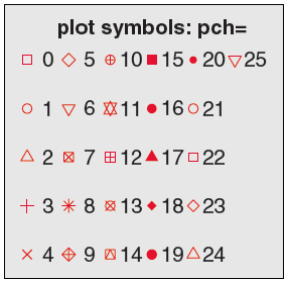
```
A<-read.table
("EserSM.csv",row.names=1,header=T,sep
=";",dec=".")
attach(A)
plot(Investimenti,
Scorte,pch=19,cex=1.1,col="darkblue")
text(Investimenti, Scorte,label=rownames
(A),cex=0.7,pos=1)
abline(lm(Scorte~Investimenti),
col="cadetblue",lwd=2,lty=2)
title("Regressione delle scorte sugli
investimenti")
detach(A)
```

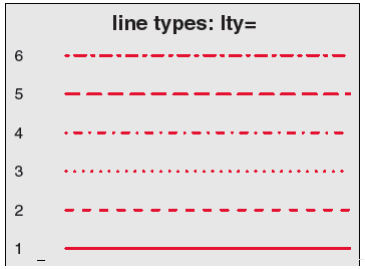


Il grafico deve essere copiato dalla pagina grafica di R e poi incollato nel testo dove è necessario.

Parametri grafici

Parameter	Description
pch	Specifies the symbol to use when plotting points (see figure 3.4).
cex	Specifies the symbol size. cex is a number indicating the amount by which plotting symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, and so forth.
lty	Specifies the line type (see figure 3.5).
lwd	Specifies the line width. lwd is expressed relative to the default (default=1). For example, lwd=2 generates a line twice as wide as the default.





Parametri grafici/2

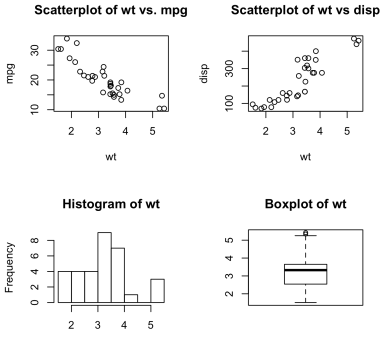
Parameter	Description
col	Default plotting color. Some functions (such as lines and pie) accept a vector of values that are recycled. For example, if col=c("red", "blue") and three lines are plotted, the first line will be red, the second blue, and the third red.
col.axis	Color for axis text.
col.lab	Color for axis labels.
col.main	Color for titles.
col.sub	Color for subtitles.
fg	The plot's foreground color.
bg	The plot's background color.

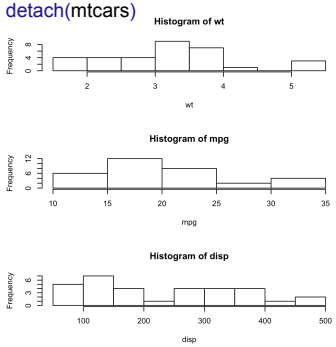
Parameter	Description
cex	Number indicating the amount by which plotted text should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
cex.axis	Magnification of axis text relative to cex.
cex.lab	Magnification of axis labels relative to cex.
cex.main	Magnification of titles relative to cex.
cex.sub	Magnification of subtitles relative to cex.

Parametri grafici/3

Per disporre grafici multipli su una tabella si può usare il comando par con l'opzione mfrow (ovvero con l'opzione mfcol):

```
attach(mtcars)
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(wt,mpg, main="Scatterplot of wt vs. mpg")
plot(wt,disp, main="Scatterplot of wt vs disp")
hist(wt, main="Histogram of wt")
boxplot(wt, main="Boxplot of wt")
par(opar)
detach(mtcars)
```

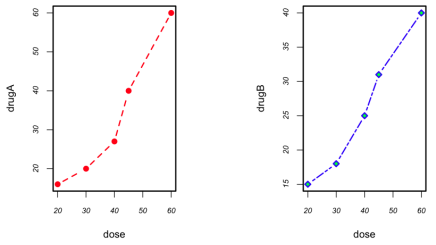


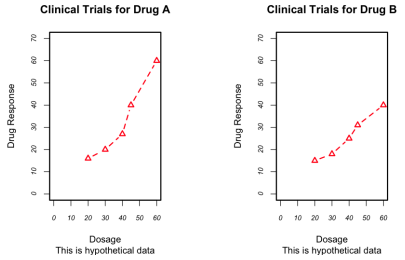



```
attach(mtcars)
opar <- par(no.readonly=TRUE)
par(mfrow=c(3,1))
hist(wt)
hist(mpg)
hist(disp)
par(opar)
detach(mtcars)
```

Esempio

```
dose <- c(20, 30, 40, 45, 60)
drugA <- c(16, 20, 27, 40, 60)
drugB <- c(15, 18, 25, 31, 40)
#
opar <- par(no.readonly=TRUE)
par(pin=c(2, 3))
par(lwd=2, cex=1.5)
par(cex.axis=.75, font.axis=3)
par(mfrow=c(1,2))
plot(dose, drugA, type="b", pch=19, lty=2,
col="red")
#plot(dose, drugA, type="b", col="red", lty=2,
pch=2, lwd=2,main="Clinical Trials for Drug
A",sub="This is hypothetical
data",xlab="Dosage", ylab="Drug
Response",xlim=c(0, 60), ylim=c(0, 70))
plot(dose, drugB, type="b", pch=23, lty=6,
col="blue", bg="green")
#plot(dose, drugB, type="b", col="red", lty=2,
pch=2, lwd=2,main="Clinical Trials for Drug
B",sub="This is hypothetical
data",xlab="Dosage", ylab="Drug
Response",xlim=c(0, 60), ylim=c(0, 70))
par(opar)
```

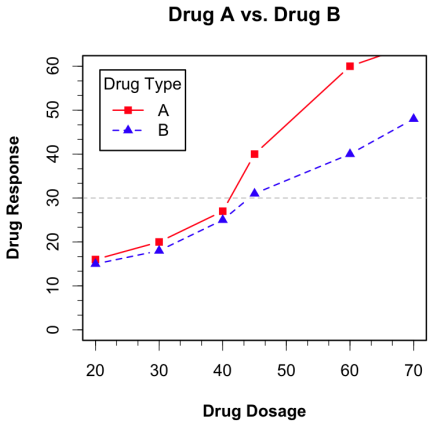




Altro esempio

```
library(Hmisc)
dose <- c(20, 30, 40, 45, 60,70)
drugA <- c(16, 20, 27, 40, 60,65)
drugB <- c(15, 18, 25, 31, 40,48)
#
opar <- par(no.readonly=TRUE)
# Aumenta lo spessore della riga, la
grandezza del simbolo e delle etichette
par(lwd=2, cex=1.5, font.lab=2)

# Costruisci il grafico
plot(dose, drugA, type="b",
pch=15, lty=1, col="red", ylim=c(0, 60),
main="Drug A vs. Drug B",
xlab="Drug Dosage", ylab="Drug Response")
lines(dose, drugB, type="b",
pch=17, lty=2, col="blue")
abline(h=c(30), lwd=1.5, lty=2, col="gray")
# Aggiungi i trattini minori sugli assi
minor.tick(nx=3, ny=3, tick.ratio=0.5)
# Inserisci legenda
legend("topleft", inset=.05, title="Drug Type",
c("A","B"),
lty=c(1, 2), pch=c(15, 17), col=c("red",
"blue"))
par(opar)
```

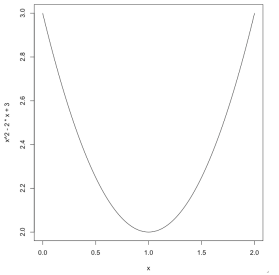


Rappresentazioni di curve analitiche

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

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

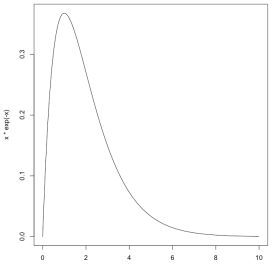
```
par(mfrow=c(1,1))
par(mar=c(4.2,4.8,0.5,1.0))
curve(x^2-2*x+3,0,2,101)
```



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)
```



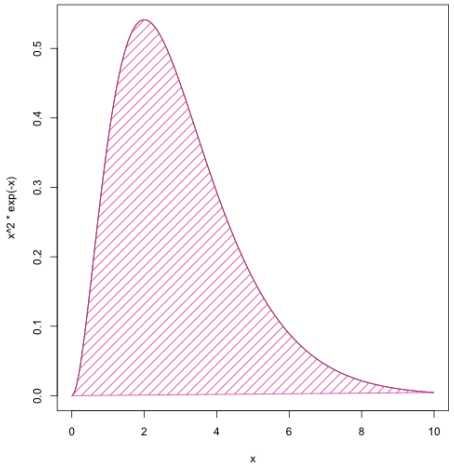
Coloratura dell'area sottesa

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

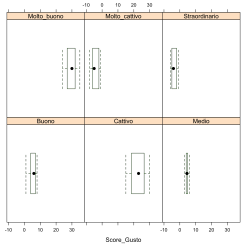
$$f(x) = x^2 e^{-x}$$

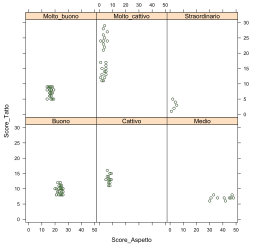
```
par(mfrow=c(1,1))
par(mar=c(4.2,4.8,0.5,1.0))
curve(x^2*exp(-x),0,10,201)
asc<-seq(0,10,length=201)
ord<-asc^2*exp(-asc)
polygon(asc,ord,
10,45,col="violetred")
```

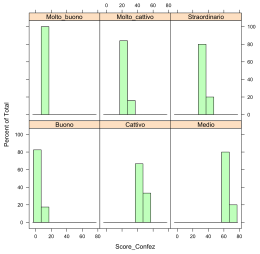
Il primo numero indica la densità della campitura ed il secondo indica l'inclinazione delle linee



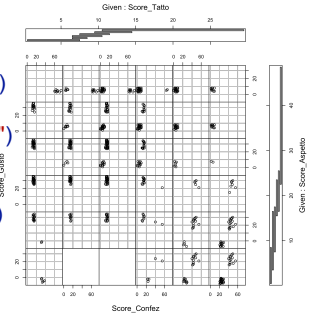
Grafici per dati raggruppati








```
# 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)
```



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
```

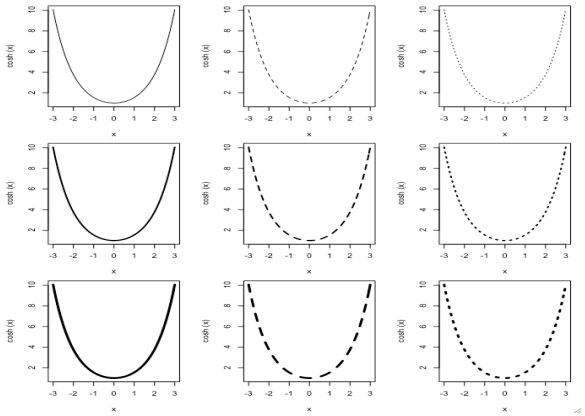
Ripeti i comandi tra la prima e l'ultima parentesi a graffe nel tanto che l'indice i scorre dal valore iniziale della sequenza al valore finale

Esempi

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

```
par(mfrow=c(3,3))
par(mar=c(4.2,4.8,0.5,1.0))
for(i in 1:3)
{for(j in 1:3)
  {plot
(cosh,-3,3,lwd=i,lty=j)
    }
  }
```

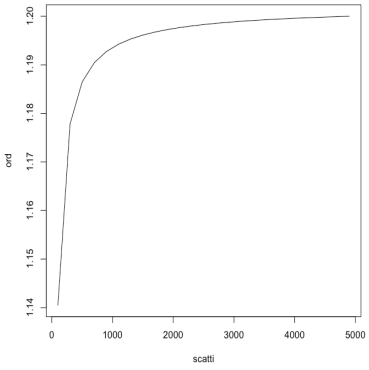


Ciclo di for e concetto di limite

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

```
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
```



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")
              }
}
```

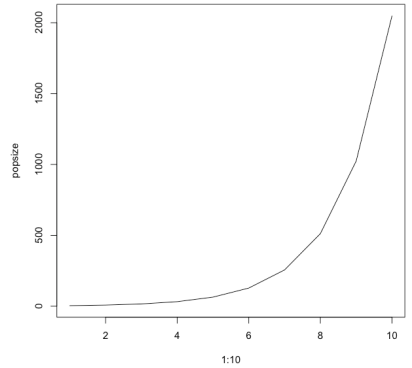
Esempio di ciclo di for

L'ipotesi è che la popolazione si raddoppi ogni anno

```
# Popolazione al livello iniziale
initsize=4;
# Creamo un vettore per memorizzare gli acrescimenti
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



Istogrammi

> Sintetizzano graficamente un collettivo statistico (data set)

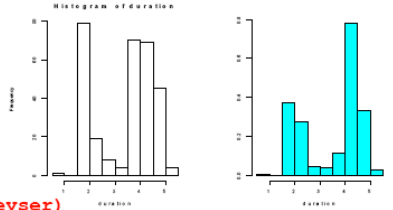
> Con essi si propone una stima della densità dei valori nella popolazione

>Per realizzarlo occorre sceglie 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

Esempio

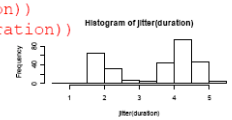


Qual è la scala degli assi?

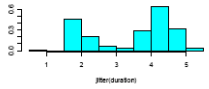
```
> attach(geyser)
> hist(duration)
> truehist(duration)
```

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

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

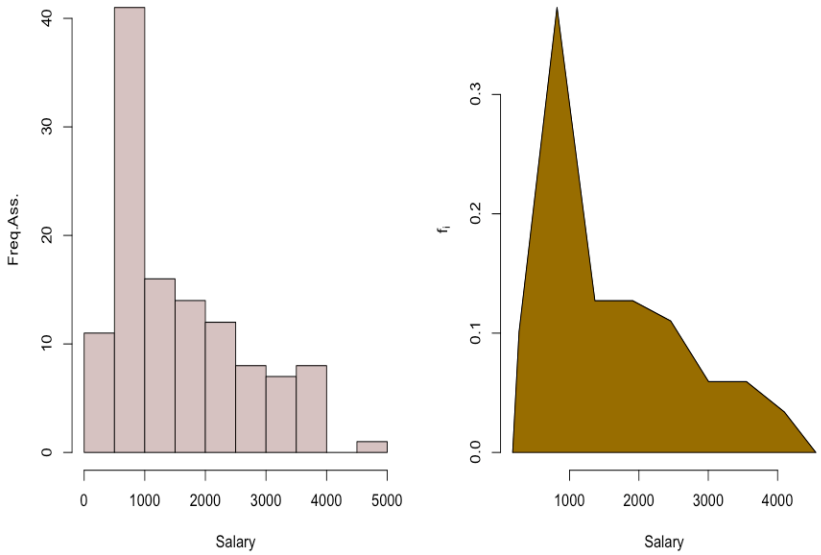


Add random amount to avoid values on class boundary



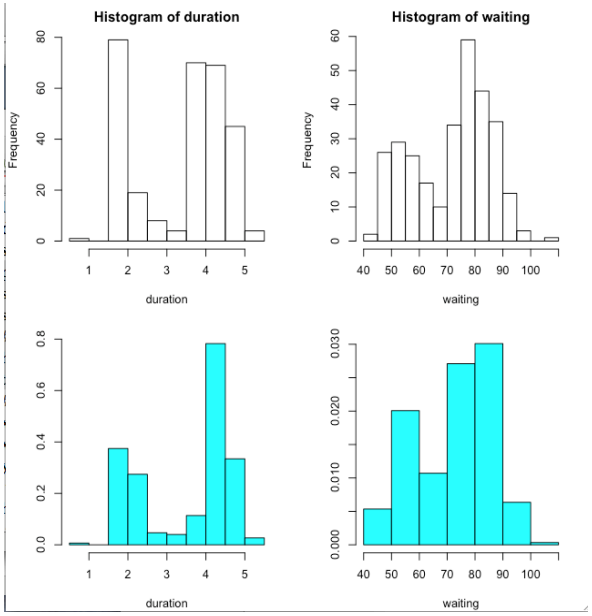
Esempio

```
# 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.Ass.",m
ain="")
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=expression
(f[i]))
polygon(A, col="orange4", border = "black")
```



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

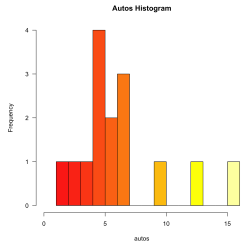
cars<-c(1,3,6,4,9)
trucks<-c(2,5,4,5,12)
suvs<-c(4,4,6,6,16)
autos.data<-data.frame(cbind
(cars,trucks,suvs))

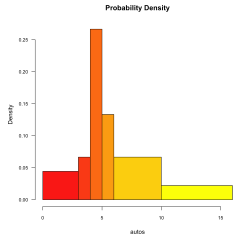
autos.dat

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)
```