

Università degli Studi di Tor Vergata
Facoltà di Lettere
Corso di Statistica per la valutazione dei servizi turistici

Introduzione ad R

dr. Simone Celant

Principali software **A PAGAMENTO** per l'elaborazione di dati statistici:

MS Excel

SPSS

Eviews

SAS

S-Plus

Gauss

MatLab

Tutti questi software, a parte Excel, sono più o meno duttili, nel senso che è possibile scrivere degli scripts, ossia delle righe di codice, per impostare delle funzioni che non sono predefinite: purché uno sappia programmare, è possibile far fare a questi software quello che si vuole.

In questa sede ci occuperemo di un software per l'elaborazione dei dati
completamente gratuito:

R

E' scaricabile gratuitamente, non è richiesta nemmeno la registrazione, su
sito <http://www.r-project.org>

Per scaricarlo basta cliccare sul link “Download R”. Si verrà reindirizzato
alla lista dei CRAN (gli archivi di R).



The R Project for Statistical Computing

About R

[What is R?](#)

[Contributors](#)

[Screenshots](#)

[What's new?](#)

Download

[CRAN](#)

R Project

[Foundation](#)

[Members & Donors](#)

[Mailing Lists](#)

[Bug Tracking](#)

[Developer Page](#)

[Conferences](#)

[Search](#)

Documentation

[Manuals](#)

[FAQs](#)

[Newsletter](#)

[Wiki](#)

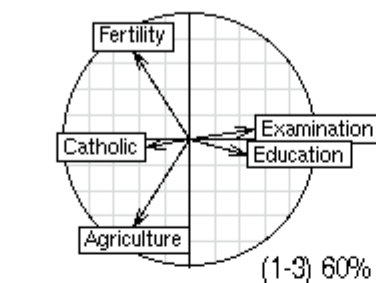
[Books](#)

[Certification](#)

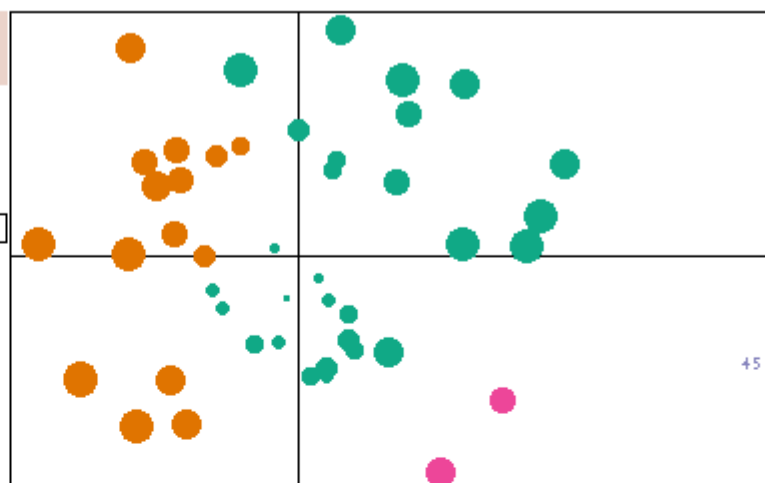
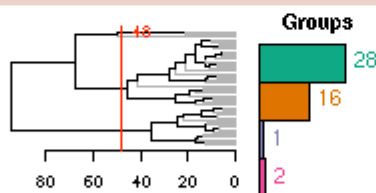
[Other](#)

PCA 5 vars

`princomp(x = data, cor = cor)`



Clustering 4 groups



Factor 1 [41%]

Factor 3 [19%]



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 or [answers to frequently asked questions](#) before you send an email.

Se ne seleziona uno (ce ne sono 3 con l'interfaccia base in italiano), dopodiché si sceglie il sistema operativo sul quale si intende installare R (esistono versioni per Windows, Linux e Mac).

A questo punto si clicca sul link “base”, che manda ad una pagina in cui è possibile scaricare l'eseguibile per l'installazione – si tratta di un file di circa 31 Mb.

L'installazione non richiede di specificare niente di particolare: dopo aver lanciato il file di installazione è sufficiente dare l'ok a tutte le specifiche richieste.

La versione base di R è molto scarna: dopo averlo scaricato ed installato, quando lo si lancia ci si trova davanti una finestra chiamata RGui contenente 7 menu con pochissimi comandi, qualche pulsante standard (Salva, Copia, Incolla...) ed una finestra di dialogo (*R Console*): è qui che bisogna essere possibile scrivere i comandi da tastiera.

L'ultima riga della finestra di dialogo è un segno di maggiore seguito dal cursore: significa che il software è in attesa. E' lì che bisogna iniziare a digitare i comandi.

Bisogna tenere presente che la compilazione dei comandi da tastiera è il metodo principale per dare istruzioni ad R. Per chi conosce il sistema operativo Linux, R si esegue addirittura direttamente da Terminale.

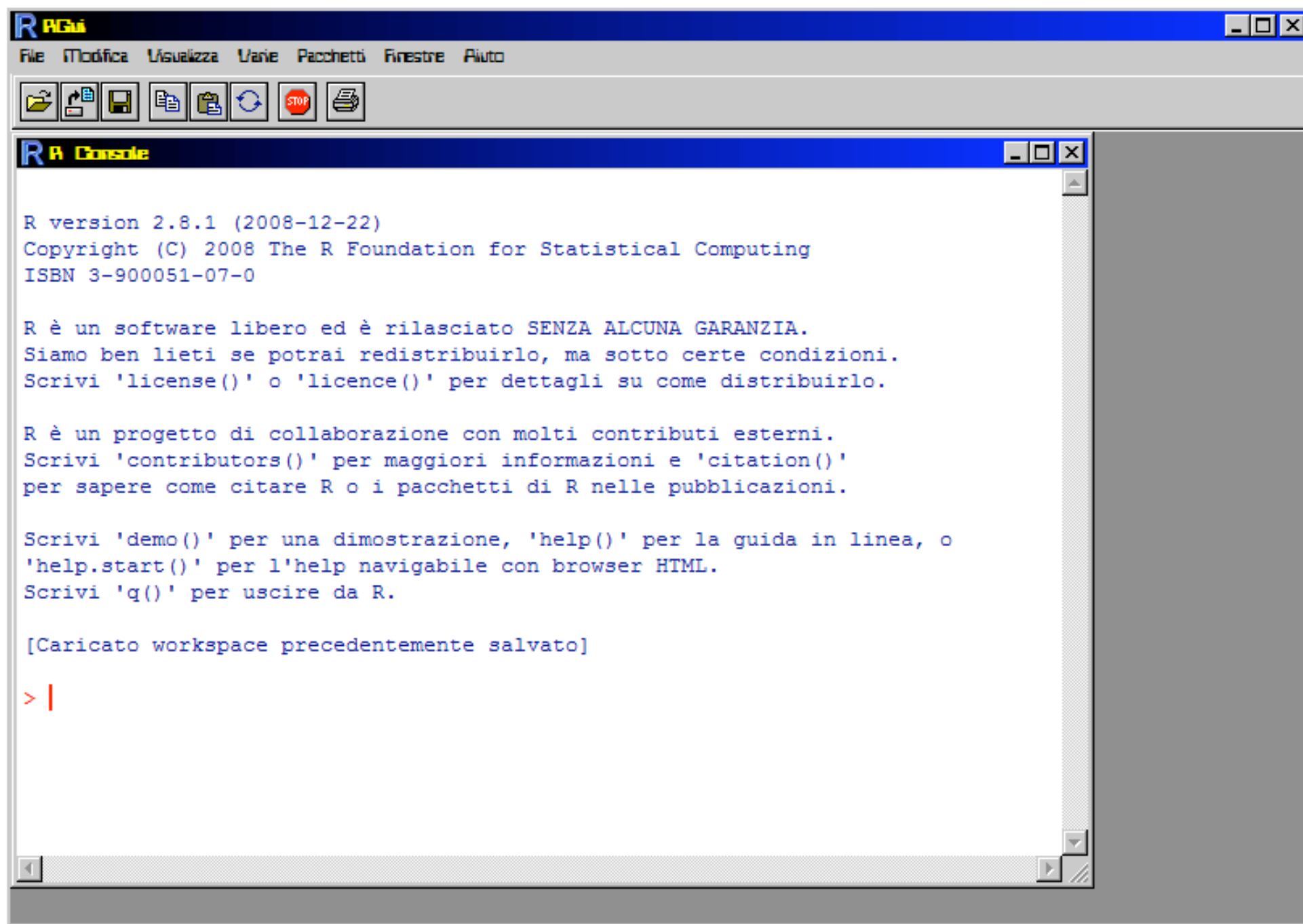
Questo non impedisce che esistano delle estensioni che permettono di usare interfacce grafiche più complete, tramite le quali dare comandi od istruzioni tramite menu a tendina.

Tuttavia l'utilizzo di un software come R rende estremamente consigliabile saper utilizzare la sua parte “programmabile”.

R è in buona sostanza la versione gratuita di S-Plus.

S-Plus è molto più completo dal punto di vista dell'interfaccia utente, ed è più maneggevole da usare perché i menu a tendina sono molto più articolati in pratica permettono di eseguire molte più elaborazioni rispetto a quello che consente R. Tuttavia la sintassi per l'utilizzo e la compilazione delle funzioni e la struttura degli output sono sostanzialmente analoghe.

Il linguaggio di riferimento per la compilazione degli scripts è il C. Questo non significa che in R si programmi in C: è solo un riferimento. SAS per esempio si basa su un linguaggio diverso, il Pascal.



Piccole istruzioni di digitazione

Un'istruzione (ad esempio un'operazione aritmetica) viene passata ad R digitandola e premendo invio: se l'istruzione è corretta il software scriverà il risultato a partire dalla riga successiva.

Il cursore può essere spostato solo con le freccette (sinistra e destra) della tastiera, il mouse può servire a selezionare, ad esempio per fare un copia e incolla, ma non sposta la posizione del cursore.

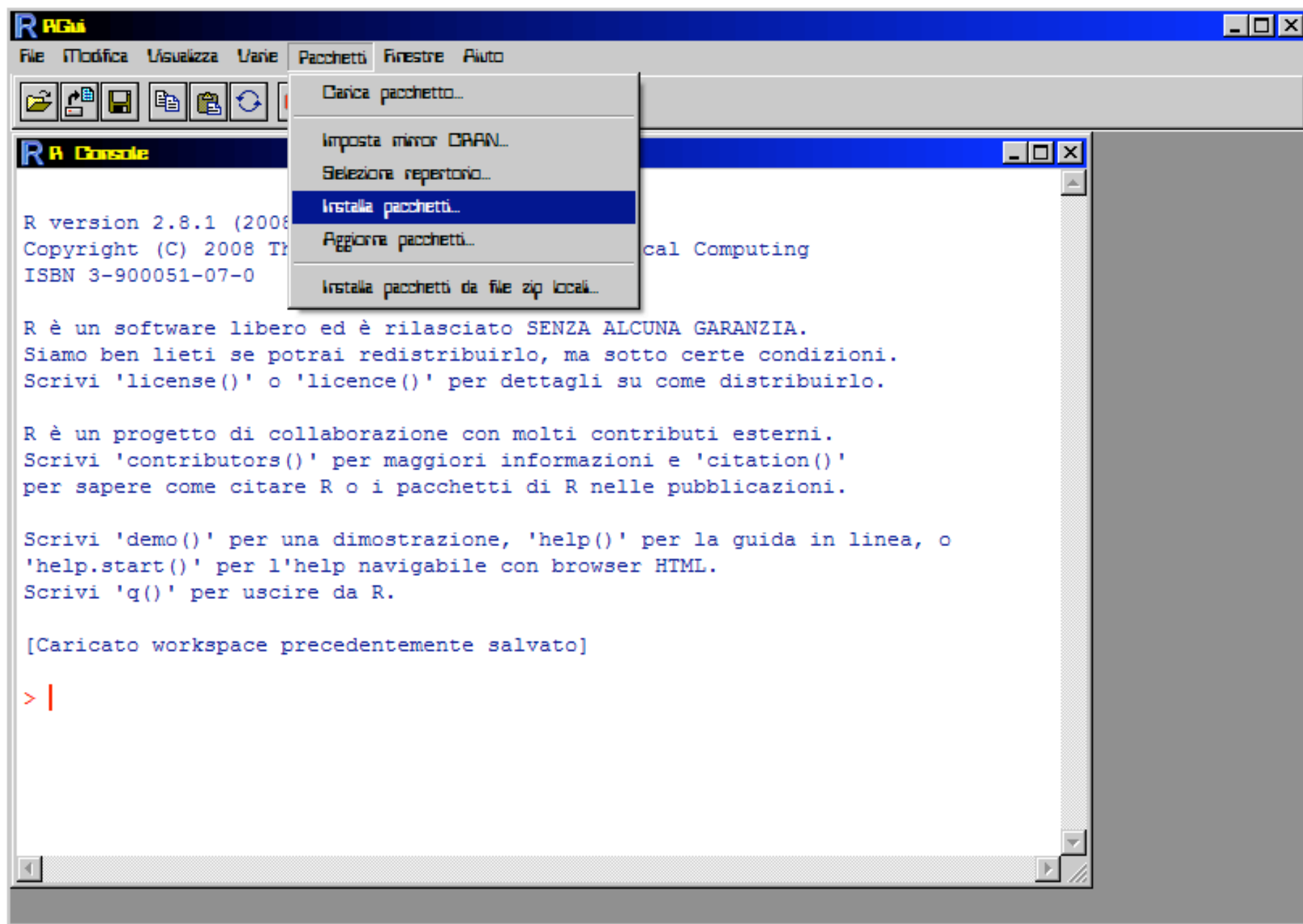
E' possibile modificare solamente la riga di comando corrente, le righe precedenti possono essere selezionate, ma non modificate.

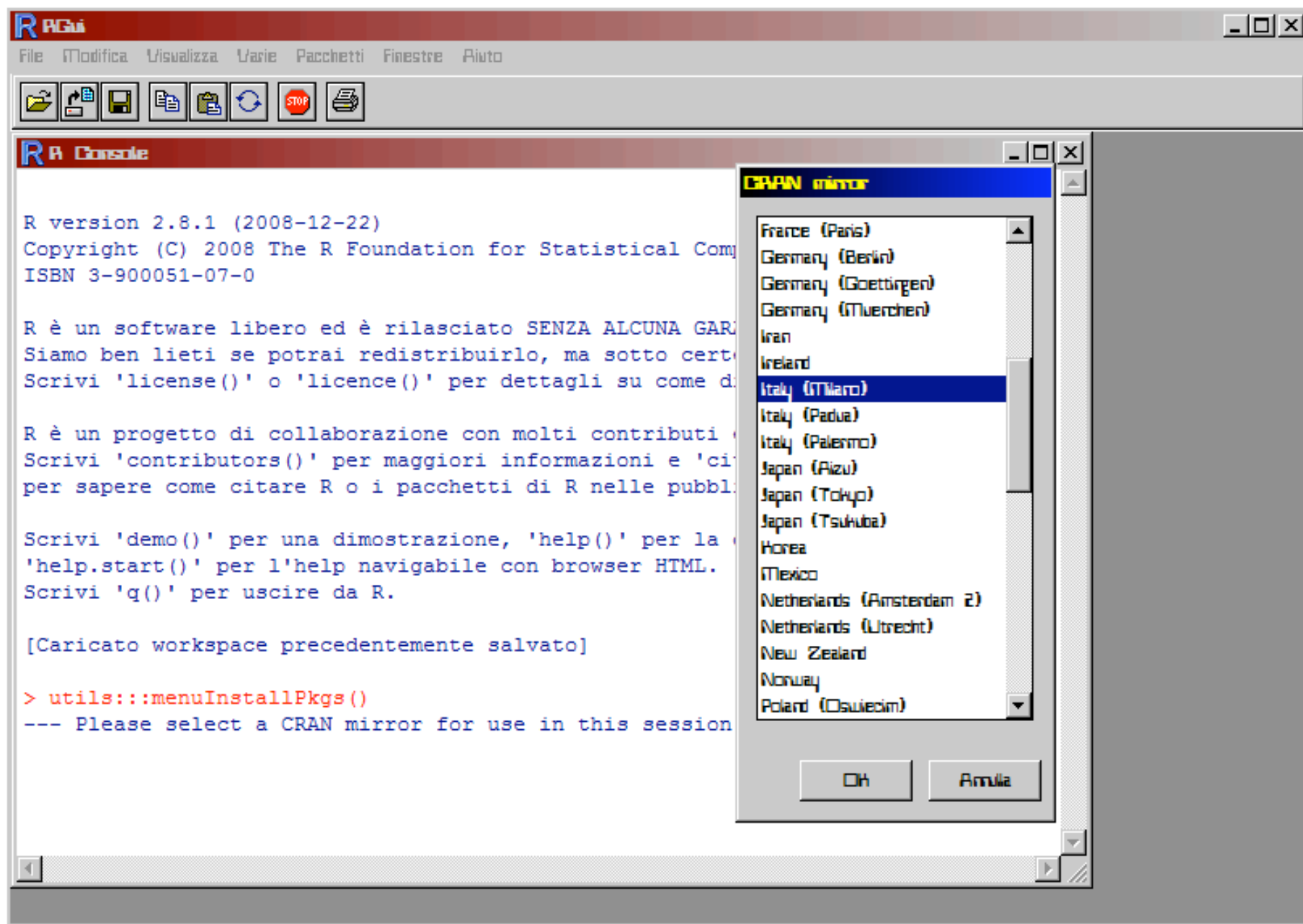
Dallo stesso sito da cui si scarica la versione base di R sono scaricabili anche molti pacchetti (*libraries*) che permettono di aumentare la fruibilità.

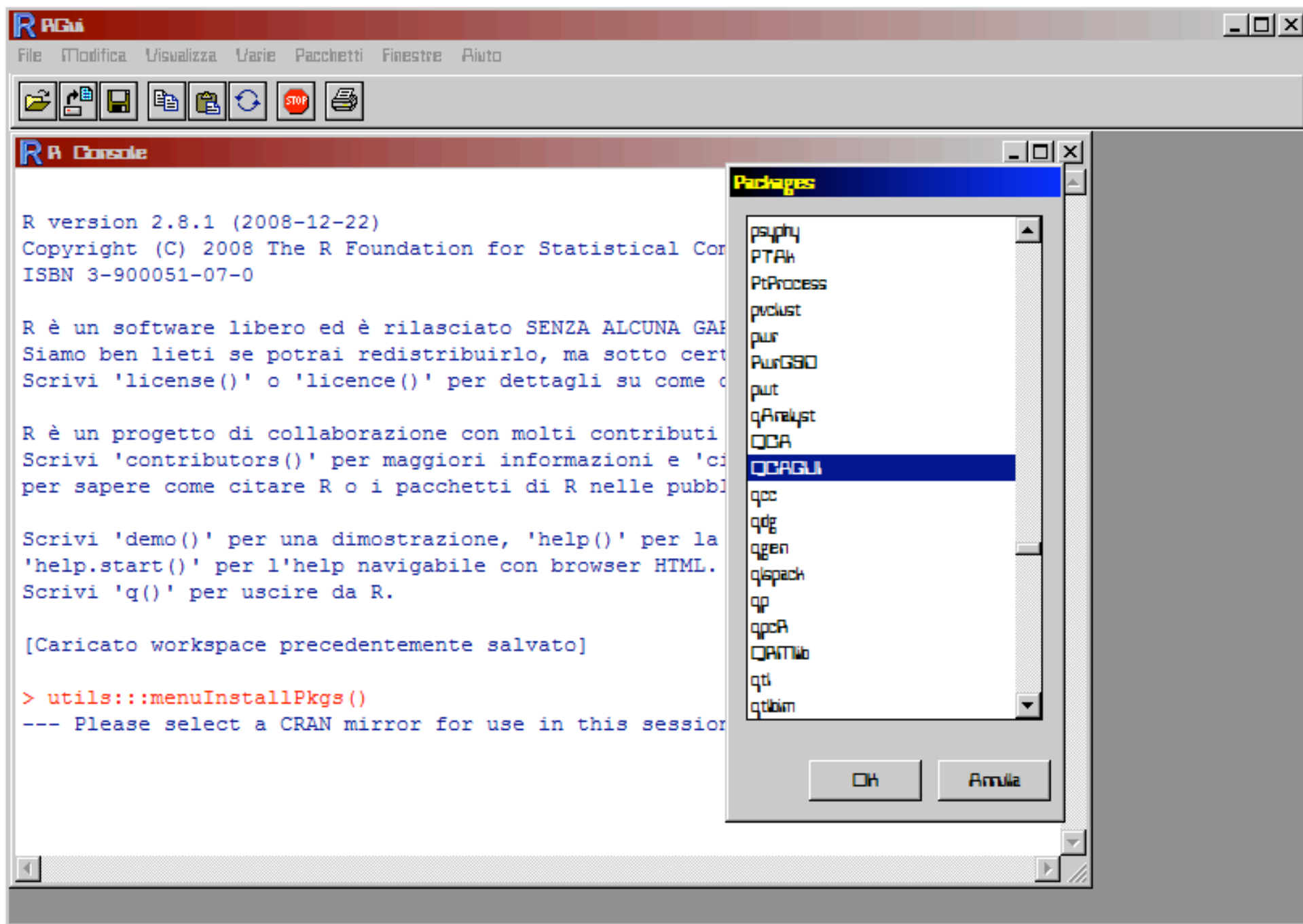
Alcuni di essi sono altre cosiddette GUI (*graphical user interface*), che permettono di utilizzare R con menu a tendina molto più completi.

E' possibile scaricare ed installare pacchetti direttamente da R: dal menu *Pacchetti* selezionare *Installa pacchetti*; il software chiederà di selezionare da quale CRAN scaricarli, poi quale dei tanti pacchetti scaricare ed installare. Lo si seleziona e si aspetta: salvo problemi, fa tutto da solo.

Due buone interfacce grafiche per lavorare con i menu a tendina sono Rcmdr e QCAGUI.







Quando si installa una library, questa diviene disponibile sul proprio sistema. Per poterla utilizzare bisogna lanciarla. questo viene fatto attraverso il comando:

```
library(nome library)
```

Se ad esempio si vuole lanciare l'interfaccia grafica Rcmdr, basta scrivere “library (Rcmdr)”.

Alla prima esecuzione di una library è possibile che il software si accorga che è necessario installare altri pacchetti per poterla utilizzare: a questo punto il sistema chiederà di poterli installare, perché senza di essi potrebbe non funzionare correttamente. Dando l'ok, parte il processo di download e installazione da CRAN. E' sufficiente dare l'ok e selezionare il CRAN da cui si vuole che il software scarichi questi ulteriori pacchetti: salvo problemi, l'R fa tutto da solo.

Le library che non fanno parte del pacchetto base devono essere lanciate
AD OGNI ESECUZIONE DI R.

In altre parole, se si chiude R e lo si riavvia, bisogna lanciare di nuovo i pacchetti di cui si ha bisogno. Se si vuole utilizzare di nuovo l'interfaccia grafica Rcmdr, bisogna digitare nuovamente “library(Rcmdr)”, l'interfaccia grafica non parte automaticamente all'avvio. A quel punto il pacchetto parte immediatamente, non deve di nuovo installare i pacchetti necessari al suo funzionamento.

Quando si chiude R, il software chiede se si vuole salvare l'area di lavoro: ovviamente, se si intende continuare a lavorare con quello che si è fatto durante la sessione corrente, bisogna rispondere di sì.

ATTENZIONE!

Un'interfaccia grafica come Rcmdr traduce in un menu a tendina una funzione già esistente che può essere chiamata tramite tastiera: è SEMPRE possibile fare da tastiera quello che si può fare da menu a tendina, mentre non è vero il contrario. Se un'elaborazione non può essere fatta da menu, non è detto che non esista una funzione che la esegue e che può essere solo richiamata da tastiera.

Infatti è ad esempio possibile installare un pacchetto digitando sulla console “`install.packages()`” e premendo invio: parte esattamente la stessa procedura.

Vediamo alcune nozioni fondamentali per l'uso dell'interfaccia base, quella che richiede di scrivere i comandi da tastiera.

Se si scrive sulla finestra di dialogo un'operazione aritmetica, R restituirà il risultato: Scrivendo $3+2$ e premendo invio, R scriverà

```
> 3+2  
[1] 5  
> |
```

[1] è un contatore dei risultati (è possibile che ce ne sia più di uno), 5 è il risultato dell'operazione richiesta.

E' possibile eseguire qualunque genere di operazione aritmetica, +, -, *, /. Quindi, come prima, $5/4$ darà come risultato

```
> 5/4  
[1] 1.25  
> |
```

L'elevamento a potenza si richiede utilizzando ^: 3^2 darà come risultato

```
> 3^2  
[1] 9  
> |
```

E' possibile eseguire operazioni aritmetiche più complesse, come

```
> 3*4+7-16/4  
[1] 15  
> |
```

R segue le norme generali delle operazioni aritmetiche per quello che riguarda la priorità delle operazioni (quindi esegue prima l'elevamento a potenza, poi moltiplicazione e divisione, infine addizione e sottrazione); a parità di priorità, esegue le operazioni da sinistra a destra.

Per modificare le priorità si possono utilizzare le parentesi.

Per questo scopo si utilizzano SOLO LE PARENTESI TONDE!!!

Le parentesi quadre e le parentesi graffe hanno un utilizzo diverso.

Quindi il comando

$$3+4*2$$

equivale al comando

$$3+(4*2)$$

Se si vuole eseguire prima la somma basta scrivere

$$(3+4)*2$$

In presenza di un dubbio se mettere o meno una parentesi, è meglio metterla.
Se ce n'è una di troppo, non succede niente, se ne manca una il risultato è diverso:

$$(3+4)*2+7$$

e

$$((3+4)*2)+7$$

sono la stessa espressione, ma se c'è una coppia di parentesi di troppo l'elaboratore non segnala errori.

Se si apre una parentesi senza chiuderla e si preme invio il software considera ancora aperta l'espressione:

```
> ( (3+4) *2+7  
+ |
```

Questo “+” indica che l'espressione non è finita. E' possibile proseguire nella linea successiva, ma occorre fare attenzione che quello che risulta scritto si quello che si voleva.

```
> ( (3+4) *2+7  
+ )  
[1] 21  
> |
```

Se il calcolo che serviva era

$$((3+4)*2+7)*3$$

e si è dimenticata l'ultima parentesi chiusa, premendo invio si ottiene in seguente output:

```
> ((3+4)*2+7*3  
+ |
```

il software dà l'opportunità di chiudere l'ultima parentesi, ma il risultato che si ottiene sarà ovviamente diverso da quello corretto.

Per uscire senza far calcolare niente al software basta premere ESC.

Le parentesi sono essenziali anche nell'elevamento a potenza. Come è noto la radice quadrata di un numero equivale ad elevare quel numero alla potenza di $1/2$, e la radice ennesima all'elevamento a $1/n$. Se si vuole ottenere la radice di un numero, ad esempio 9, e si scrive $9^{1/2}$ si ottiene 4.

```
> 9^1/2  
[1] 4.5  
> |
```

Ciò che è scritto non è 9 elevato a $1/2$: è 9 elevato alla 1 il tutto diviso per 2.

L'espressione corretta è $9^{(1/2)}$, cioè 9 elevato alla potenza di 1 diviso 2.

```
> 9^(1/2)  
[1] 3  
> |
```

Ulteriori istruzioni di digitazione

La freccia verso l'alto serve a richiamare le istruzioni precedenti: se si preme una volta appare sulla riga di comando presente l'ultima riga di comando digitata. Attenzione, non l'ultimo comando, che come abbiamo visto può essere digitato su più righe, semplicemente l'ultima riga scritta ed inviata.

Si può procedere a ritroso fino all'inizio della sessione (ossia fino all'avvicinamento del programma), continuando a premere la freccia verso l'alto, mentre con la freccia verso il basso si compie il percorso inverso.

Le righe così richiamate vengono scritte come erano state scritte prima che si premesse invio: in questo modo se si vuole ripetere l'operazione basta premere nuovamente invio, se la si vuole correggere, magari inserendo una parentesi che si era dimenticata, la si può inserire spostandosi con il cursore nella posizione desiderata (sempre tramite le frecce, non con il mouse).

Per eseguire di nuovo operazioni già eseguite in precedenza, o comunque per farle comparire scritte per poterle magari correggere e poi eseguire nuovamente, è possibile anche copiare ed incollare una parte del testo digitato in precedenza sulla console.

In questo caso, è necessario utilizzare il mouse: l'unica cosa che è possibile fare con il mouse è selezionare del testo: si possono selezionare comandi digitati in precedenza, o anche risultati. Una volta selezionato il testo che si intende copiare, per incollarlo sulla console come nuova istruzione è sufficiente selezionare “Copia” e poi “Incolla” dal menu “Modifica”, oppure CTRL+C e poi CTRL+V, esattamente come in qualunque altro software.

ATTENZIONE!!!

R, come tutti i software ed in generale i computer, è velocissimo, fa tutto quello che gli viene detto, ma è stupido e fa SOLO quello che gli viene detto: non capisce le cose ad intuito, bisogna sempre spiegargliele. Per lui $1/2$ non è un numero frazionario pari a 0.5, è una divisione, uno diviso due

L'assegnazione: <-

Con questa specie di freccetta si ASSEGNA un valore ad un oggetto.

Se si scrive:

```
a<-3
```

si crea un oggetto di nome “a” e lo si pone uguale a 3.

A questo punto se si digita semplicemente “a” e si preme invio, R restituirà il valore di a.

```
> a<-3  
> a  
[1] 3  
> |
```

ATTENZIONE!!!

L'espressione `a<-3` non significa “a minore di -3”, e tutto questo non c'entra niente con la programmazione ad oggetti in Java.

La sequenza di caratteri “<-” equivale ad un'uguaglianza.

Scrivere “`a<-3`” equivale a dire “crea un oggetto di nome a e assegnagli un valore pari a 3”.

Se volessi porre a uguale a -3 basterebbe scrivere

`a<--3`

Nel momento in cui un oggetto viene creato, e gli viene assegnato un valore, questo rimane in memoria. E' dunque possibile richiamarlo più tardi ed utilizzarlo per eventuali operazioni.

Gli oggetti creati sono delle variabili a tutti gli effetti. Oltre ad essere richiamati possono essere utilizzati nelle operazioni:

```
> a<-3  
> 4*a  
[1] 12  
> 12/a  
[1] 4  
> 2^a  
[1] 8  
> |
```

ATTENZIONE!!!

Un oggetto una volta creato non è intoccabile:

```
> a<-4  
> a  
[1] 4  
> a<-"Forza Inter"  
> a  
[1] "Forza Inter"  
> |
```

Se si assegna un valore diverso ad un oggetto già creato, R esegue l'operazione senza neanche chiedere conferme. Se l'oggetto “a” è già stato creato, e tramite una nuova assegnazione gli viene assegnato un valore diverso, R elimina il valore precedente e crea di nuovo l'oggetto da zero.

Gli oggetti possono essere il risultato di operazioni, sui quali è possibile eseguire ulteriori operazioni, che possono a loro volta essere salvate in memoria come ulteriori oggetti:

```
> a<-3*2
> b<-16-4*3
> a
[1] 6
> b
[1] 4
> a-b
[1] 2
> c<-a-b
> c
[1] 2
> |
```

Passiamo ora ad un concetto fondamentale su R: le **FUNZIONI**.

Non ci si può aspettare che l'utente programmi ogni singolo step di tutto quello che deve fare. Per questo, già nella versione base, è possibile richiamare alcune funzioni che eseguono determinate operazioni.

Quasi tutte le libraries che possono essere installate all'interno di R contengono delle funzioni che ne arricchiscono le potenzialità.

Le funzioni prendono in input uno o più valori e restituiscono uno o più output. Gli input vengono messi tra parentesi tonde, ed eventualmente separati da virgole.

Ad esempio, la funzione `sum()` calcola la somma degli elementi tra parentesi:

```
> sum(2,3)
[1] 5
> sum(a,b)
[1] 10
> |
```

Le funzione `min()` e `max()` ne calcolano invece rispettivamente il valore minimo e massimo:

```
> min(2,3)
[1] 2
> max(2,3)
[1] 3
> min(a,b)
[1] 4
> max(a,b)
[1] 6
> |
```

E' possibile assegnare ad un valore l'output di una funzione:

```
> c<-sum(a,b)
> c
[1] 10
> d<-min(4,6,2,43,-3,7,0,4)
> d
[1] -3
> |
```

Gli input delle funzioni possono essere espressioni o a loro volta funzioni:

```
> sum(4+3,2*5,3)
[1] 20
> min(sum(-2,4),12/3,3, 0.5*sum(1,1))
[1] 1
> |
```

Un paio di funzioni particolari: ls e rm.

Scrivendo ls() si ottiene la lista di oggetti in memoria:

```
> ls()  
[1] "a"      "b"      "c"      "d"  
> |
```

rm() permettere di rimuovere un oggetto dalla memoria.

```
> rm(d)  
> ls()  
[1] "a"      "b"      "c"  
> |
```

Due funzioni utilissime: la funzione “**c**” e la funzione “**list**”.
Entrambe richiedono che il risultato sia assegnato ad un oggetto.

“c” permette di creare di un VETTORE:

```
> vec<-c(3, 5, 0, 1, -5, 2, 11)
> vec
[1] 3 5 0 1 -5 2 11
> |
```

con `vec<-c(3, 5, 0, 1, -5, 2, 11)` si crea un oggetto di nome “vec”, e ad esso assegna il vettore creato dalla funzione “c”, contenente i valori inseriti tra parentesi. I differenti elementi del vettore sono separati da virgola (per i decimali in R si usa il punto).

La dimensione di R risulta dall'assegnazione: se vengono inseriti 7 numeri la dimensione del vettore sarà automaticamente 7

La funzione “c” permette di creare anche vettori non numerici.

Se si vuole che R consideri il vettore `vec` sopra creato come un vettore contenente caratteri, o si vuole creare un vettore composto di stringhe di caratteri, ad esempio un vettore di nomi, bisogna mettere i valori tra virgolette.

```
vec<-c(“Giorgio”, “Francesco”, “Michele”).  
vec<-c(“1”, “2”, “3”).
```

In quest'ultimo caso R considererà 1, 2 e 3 come CARATTERI e non come numeri, quindi non sarà possibile eseguire su di essi nessuna operazione matematica.

E' possibile inserire nel vettore che si intende creare delle variabili create in precedenza.

```
> a<-3  
> vec<-c(a, 6, 7, 1, -3)  
> vec  
[1] 3 6 7 1 -3  
> |
```

Una di queste variabili può a sua volta essere un vettore precedentemente creato con la funzione “c”

```
> vec2<-c(-vec, a, 1)  
> vec2  
[1] -3 -6 -7 -1 3 3 1  
> |
```

I singoli elementi di un vettore si identificano tramite le PARENTESI QUADRE (ecco a cosa servono!!!): ad esempio `vec[4]` è il quarto elemento del vettore `vec`. Sono anch'essi variabili scalari a tutti gli effetti, possono essere assegnati ad altre variabili e su di loro è possibile effettuare delle operazioni.

```
> vec
[1] 3 6 7 1 -3
> vec[4]
[1] 1
> a<-vec[3]-2
> a
[1] 5
> |
```

“list” permette di creare di un LISTA DI OGGETTI

```
> lista<-list(vec, 2, 3)
> lista
[[1]]
[1] 3 6 7 1 -3

[[2]]
[1] 2

[[3]]
[1] 3

> |
```

Il funzionamento di “list” è del tutto analogo a quello di “c”, solo che in questo modo l'output non è un vettore, ma una lista.
Gli elementi di una lista possono essere numeri, vettori, stringhe di caratter e liste a loro volta.

Un vettore creato con “c” richiede che tutti gli elementi siano dello stesso tipo: se uno di essi è una stringa di caratteri, gli altri, pur essendo numeri, vengono automaticamente convertiti dal computer in stringhe di caratteri. La conversione si fa al livello “più basso”: essendo un numero un carattere, ma non essendo vero il viceversa, l'unico modo per leggere un numero ed un carattere allo stesso modo è convertire entrambi in un carattere

```
> vec<-c("stringa", a, 4)
> vec
[1] "stringa" "3"      "4"
```

Questo non vale per le liste. Una lista può contenere degli elementi numerici e delle stringhe di caratteri:

```
> a<-3
> lista<-list(vec, "stringa", a)
> lista
[[1]]
[1] 3 6 7 1 -3

[[2]]
[1] "stringa"

[[3]]
[1] 3
```

Gli elementi di una lista si identificano attraverso le DOPPIE PARENTESI QUADRE. Quindi l'elemento `lista[[1]]` è il primo elemento della lista denominata “lista”.

Qualora il primo elemento di una lista fosse un vettore, esso si comporta a tutti gli effetti come un vettore creato con “c”: quindi per identificare un elemento di quel vettore si ricorre di nuovo alle parentesi quadre:

```
> lista[[1]]  
[1] 3 6 7 1 -3  
> lista[[1]][2]  
[1] 6  
> |
```

Questi elementi sono di nuovo variabili a tutti gli effetti

Una funzione importantissima: la funzione “**matrix**”

“matrix” permette di creare una MATRICE di elementi e dimensioni date:

```
> mat<-matrix(0, 5, 3)
> mat
```

	[,1]	[,2]	[,3]
[1,]	0	0	0
[2,]	0	0	0
[3,]	0	0	0
[4,]	0	0	0
[5,]	0	0	0

```
> |
```

Con `mat<-matrix(0, 5, 3)` si crea un oggetto di nome “mat” e ad esso si assegna una matrice di zeri di dimensione 5x3 (ossia di 5 righe e 3 colonne)

La formulazione esatta per chiamare la funzione `matrix` sarebbe:

```
mat<-matrix(data, nrow=x, ncol=y, byrow=FALSE, dimnames=NULL)
```

data sono i dati che si intende inserire nella matrice. Può trattarsi di un oggetto già esistente (ad esempio un'altra matrice); nell'esempio `data` è stato posto uguale a 0. Se si vuole una matrice vuota basta scrivere `data=NA` (NA in R indica l'assenza di un dato).

nrow=x, **ncol=y** sono il numero di righe e di colonne della matrice `mat`.

Tornando ai dati:

E' possibile utilizzare la funzione “c” per inserire i valori in una matrice:

```
> mat<-matrix(c(1, 4, 3, 7, 1, 4, 1, 2, 1), 3, 3)
> mat
      [,1] [,2] [,3]
[1,]    1    7    1
[2,]    4    1    2
[3,]    3    4    1
> mat<-matrix(c(1, 4, 3, 7, 1), 3, 3)
Warning message:
In matrix(c(1, 4, 3, 7, 1), 3, 3) :
  data length [5] is not a sub-multiple or multiple of the number of rows [3]
> mat
      [,1] [,2] [,3]
[1,]    1    7    4
[2,]    4    1    3
[3,]    3    1    7
> |
```

Come si può notare, se la matrice viene riempita con una funzione “c”, R sistema i valori che gli vengono passati PER COLONNA.

Questo avviene di default, se non si specifica niente questa è l'impostazione base: si tratta infatti di un parametro OPZIONALE: se il parametro “byrow” non viene specificato, per default, R sistema gli eventuali valori per colonne.

Per modificare questa procedura e riempire la matrice per riga, basta scrivere “byrow=TRUE”.

Se la dimensione degli elementi inseriti tramite “c” è minore del numero degli elementi della matrice, R ricomincia a compilare dal primo elemento.

Se il numero di elementi inseriti non è sottomultiplo del numero degli elementi della matrice, o delle sue righe o colonne, R dà un avviso (ma **non** un messaggio di errore), e compila ugualmente.

La matrice così creata contiene solo i dati inseriti, le righe e le colonne non hanno intestazione. Se si vuole inserire un'intestazione, si può utilizzare l'input OPZIONALE “dimnames”.

dimnames deve essere posto uguale ad una lista contenente i due vettori che hanno al loro interno le intestazioni di riga e di colonna.

```
> mat<-matrix(c(182, 77, 168, 54), 2, 2, byrow=T,  
+ dimnames=list(c("Simone", "Alessia"), c("Altezza", "Peso")))  
> mat
```

	Altezza	Peso
Simone	182	77
Alessia	168	54

```
> |
```

Il nome della variabile di input in R non è necessario: nella funzione `matrix` ad esempio, il primo valore è SEMPRE `data`, il secondo valore è SEMPRE numero di righe della matrice che si intende creare, il terzo valore è SEMPRE il numero di colonne. Quindi, per sintesi, i nomi dei parametri d'input si possono omettere.

```
> mat<-matrix(c(182, 77, 168, 54), 2, 2, T,  
+ list(c("Simone", "Alessia"), c("Altezza", "Peso")))  
> mat
```

	Altezza	Peso
Simone	182	77
Alessia	168	54

```
> |
```

Per identificare gli elementi di una matrice si utilizzano le parentesi quadre: gli indici sono separati da virgola. Così “vec[4]” è il quarto elemento del vettore vec; “mat[3,2]” è l'elemento che si trova nella posizione 3,2 (riga 3 colonna 2) di mat:

```
> mat<-matrix(0, 5, 3)
> mat[3,2]
[1] 0
> |
```

E' possibile effettuare assegnazioni all'interno di un vettore e di una matrice con le stesse modalità fin qui osservate:

```
> a<-6
> b<-4
> mat[1,2]<-3+2
> mat[3,1]<-a
> mat[4,3]<-sum(a,b)*2
> mat
```

	[,1]	[,2]	[,3]
[1,]	0	5	0
[2,]	0	0	0
[3,]	6	0	0
[4,]	0	0	20
[5,]	0	0	0

Come si vede, le colonne della matrice “mat” sono indicate da [,1], [,2] e [,3], mentre le righe sono indicate da [1,], [2,], [3,], [4,] e [5,]. Possono dunque essere richiamate ed essere eventualmente assegnate con questa sintassi:

```
> mat[1,]  
[1] 0 5 0  
> mat[,1]  
[1] 0 0 6 0 0  
> d<-mat[4,]  
> d  
[1] 0 0 20  
> |
```

L'oggetto d così definito è un vettore.

Per sapere quali sono le dimensioni di una matrice di R si può utilizzare la funzione “dim()”:

```
> dim(a)
NULL
> dim(a1)
[1] 40 40
> |
```

L'output della funzione dim() è un vettore contenente due elementi, il primo (che, nell'esempio, indica con dim(a1)[1]) è il numero di righe, il secondo (dim(a1)[2]) il numero di colonne della matrice in input. Se l'input non è una matrice la funzione restituisce il valore NULL.

Se si vuole sapere solamente quante sono le righe o le colonne di una matrice invece di scrivere dim(a1)[1] o dim(a1)[2] si può scrivere nrow(a1) e ncol(a1)

```
> nrow(a1)
[1] 40
> ncol(a1)
[1] 40
> |
```

Ritornando alle funzioni introdotte sopra, `sum`, `max` e `min` sono definite su vettori e matrici:

```
> sum(vec)
[1] 80
> max(mat)
[1] 20
> max(mat[3,])
[1] 6
> |
```

Inoltre è possibile calcolare la **media aritmetica** di un vettore e di una matrice: la funzione è semplicemente “`mean`”:

```
> mean(mat)
[1] 2.066667
> mean(vec)
[1] 11.42857
> |
```

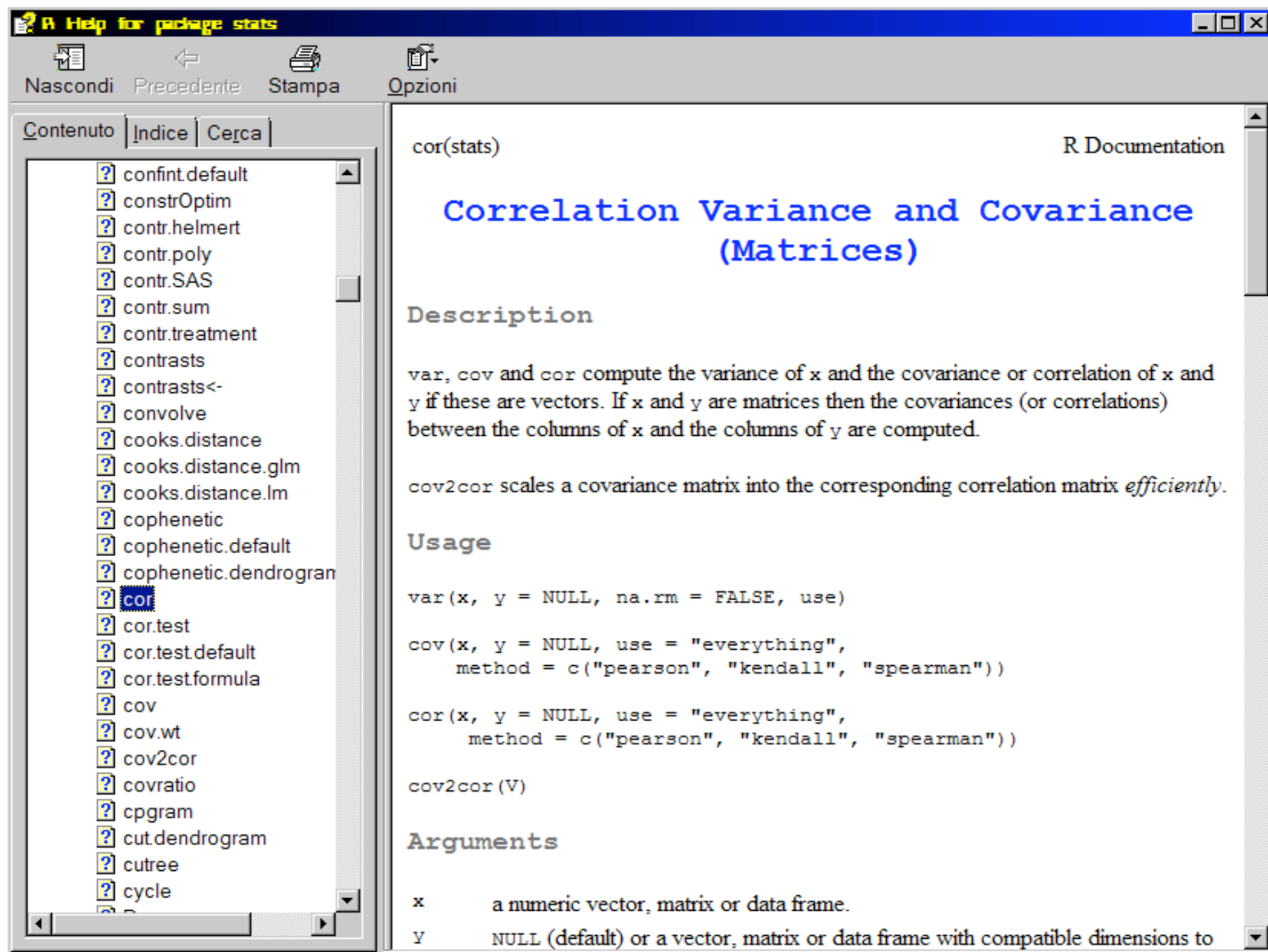
La **varianza** si calcola attraverso la funzione “var”: questa funzione è definita solo su un vettore. Se la si applica ad una matrice il risultato è diverso (si tratta della matrice di varianze e covarianze tra le colonne della matrice originaria):

```
> var(vec)
[1] 581.9524
> var(mat[1,])
[1] 8.333333
> e<-var(mat[,2])
> e
[1] 5
> |
```

In qualunque momento sia necessario un aiuto per l'utilizzo di una funzione è sufficiente digitare il nome della funzione preceduto dal **PUNTO INTERROGATIVO**. Se ad esempio si vuole visualizzare l'aiuto della funzione var, si scrive “?var” e si preme invio.

In questo modo si aprirà una finestra contenente le informazioni della funzione: cosa calcola, che valori prende in input, gli eventuali input opzionali, che tipo di output restituisce, spesso qualche esempio.

Se invece si scrive il nome della funzione e si preme invio, viene visualizzato il cosiddetto corpo della funzione, ossia il codice che la compone.



```
> var
function (x, y = NULL, na.rm = FALSE, use)
{
  if (missing(use))
    use <- if (na.rm)
      "na.or.complete"
    else "everything"
  na.method <- pmatch(use, c("all.obs", "complete.obs", "pairwise.complete.obs",
    "everything", "na.or.complete"))
  if (is.data.frame(x))
    x <- as.matrix(x)
  else stopifnot(is.atomic(x))
  if (is.data.frame(y))
    y <- as.matrix(y)
  else stopifnot(is.atomic(y))
  .Internal(cov(x, y, na.method, FALSE))
}
<environment: namespace:stats>
> |
```

Alcune funzioni sono presenti solamente su determinati pacchetti. Come gi detto, se si ha bisogno di un pacchetto questo deve essere lanciato ad ogni esecuzione di R.

Ad esempio, la funzione “gden()”, che prende in input un grafo e ne calcol la densità, è presente ad esempio sul pacchetto “sna”. Se si scrive “gden(x)” senza aver prima lanciato la library “sna” si ottiene un messaggio di errore

```
> gden(a)  
Errore: non trovo la funzione "gden"  
> |
```

Alcune funzioni sono presenti su più di un pacchetto: basta lanciarne uno qualunque che le contenga per averle a disposizione.

La funzione **read.table()**

Permette di importare in R una tabella costruita con altri software (ad esempio Excel o SPSS) e salvata all'interno del computer.

```
Dataset<-read.table("C:/Documents and Settings/Worksheet1.xls")
```

In questo modo creo un oggetto di nome Dataset e gli assegno i valori contenuti nel documento Worksheet1.xls (che è un foglio di lavoro Excel) salvato nella cartella Document and Settigs nel disco C. Il percorso tra file deve essere messo tra vorgolette, ed i separatori sono le **barre** ("/"), NON backslash ("\"), come è usuale in ambiente Windows.

Su Dataset è pssibile eseguire le operazioni fin qui introdotte.

La funzione `read.table()` ha una serie di input opzionali. Sostanzialmente servono a spiegare al software come deve leggere la tabella che gli si sta dicendo di andare a prendere.

```
read.table(file, header = F, sep = " ", dec = ".", row.names, col.names...)
```

file è l'input indispensabile, che viene passato tramite indirizzo di allocazione come detto sopra.

header è un valore binario tramite il quale si impone al software di considerare la prima riga del file di input come una riga di intestazione delle variabili. Se si pone “header=F” gli si dice di NON farlo, se si vuole che lo faccia basta scrivere “header=T”. Per default, R considera la prima riga come un vettore di intestazioni se c'è qualche valore mancante. Se la prima riga ha lo stesso numero di valori significativi delle altre, è considerata una riga di dati.

```
read.table(file, header = F, sep = " ", dec = ".", row.names, col.names...)
```

sep = " " serve a comunicare al software qual è il separatore di colonna. Per default, questo valore è lo spazio: in altre parole, ogni volta che trova uno spazio nella scansione del file di input, R mette il valore precedente allo spazio in una colonna, quello successivo in un'altra. Se si vuole che il separatore di colonna sia diverso, ad esempio la virgola, è sufficiente scrivere `sep = ","`

dec = "." serve a stabilire qual è il separatore tra interi e decimali. Per default è il punto. R in generale separa interi e decimali con un punto. Excel, ad esempio, usa la virgola. Come sopra, se si vuole che R legga il numero 1,5 come un decimale, basta scrivere `dec = ","`

ATTENZIONE AI CONFLITTI!!!

Se gli viene detto che separatore di colonna e separatore di decimali sono lo stesso carattere, R non capisce ad intuito quando la virgola vuol dire una cosa e quando ne vuol dire un'altra.

```
read.table(file, header = F, sep = " ", dec = ".", row.names, col.names...)]
```

row.names e **col.names** sono due variabili tramite le quali è possibile passare al software i nomi delle righe e delle colonne del data set che si sta importando. Si tratta di due vettori, ad esempio creati tramite la funzione “c” vista sopra.

La funzione `read.table` prende in input diversi altri parametri opzionali, questi sostanzialmente sono i più importanti. Possono ovviamente essere visualizzati digitando “`?read.table`”. L'unico necessario è l'indirizzo del file che si desidera importare.

```

> Dataset<-read.table("F:/Documenti/MS Excel/EIES1.csv")
> Dataset
      V1 V2 V3 V4 V5 V6 V7 V8 V9 V10
1      0  4  2  2  2  2  2  2  2  2
2      4  0  2  0  1  0  3  3  4  1
3      3  1  0  4  1  0  0  2  0  2
4      2  0  2  0  2  0  0  2  2  2
5      3  0  0  2  0  0  0  2  3  2
6      3  0  0  0  0  0  0  2  0  0
7      3  2  1  0  0  0  0  2  2  0
8      2  2  2  2  2  0  0  0  1  0
9      3  4  0  0  2  0  0  2  0  0
10     2  1  3  3  2  0  1  2  2  0
> |

```

In questo caso la tabella di origine è una matrice quadrata priva di intestazioni, quindi per importarla non è stato necessario specificare niente parte il percorso in cui andarla a prendere. Il formato “csv” è un formato testuale di Excel, ed è molto utile per l'importazione delle tabelle in R, perché consente di specificare ad esempio i separatori di colonna.

La matrice creata dalla funzione `read.table` non è dello stesso tipo delle matrici create tramite la funzione `matrix`. La funzione `read.table` crea una matrice di tipo “**data frame**”. Buona parte delle funzioni più complesse che è possibile utilizzare su R prendono in input matrici di tipo data frame, non matrici semplici.

Se si vuole trasformare una matrice semplice, come quelle create con la funzione `matrix`, in un data frame è sufficiente scrivere:

```
mat1<-data.frame(mat)
```

In questo modo l'oggetto di nome `mat1` viene posto uguale alla matrice `mat`, ma in formato data frame.

Il percorso inverso, ossia per trasformare un data frame in una matrice semplice, consiste nello scrivere:

```
mat2<-as.matrix(mat1)
```

In questo caso, i due oggetti `mat` e `mat2` saranno identici, mentre `mat1` avrà lo

Un qualunque oggetto, uno scalare, una matrice, un vettore, una lista o un data frame possono essere modificati tramite i comandi da tastiera visti fino ad ora. Inoltre è possibile modificarli su una finestra separata tramite un editor che si apre eseguendo andando al menu “Modifica” e selezionando “Editor dei dati”.

Parte a quel punto una finestra che chiede di digitare il nome dell'oggetto che si intende modificare.

A seconda del tipo di oggetto che si è indicato parte un editor graficamente molto scarso, che permette di intervenire sui dati, modificandoli semplicemente da tastiera, come su un qualunque editor come Word o Excel.

Quando si chiude l'editor, i dati rimangono in memoria modificati. R non chiede conferma prima di salvare le modifiche.

Il corrispettivo comando da tastiera per modificare un oggetto tramite l'editor è “fix()”.

```
> fix(a1)
```

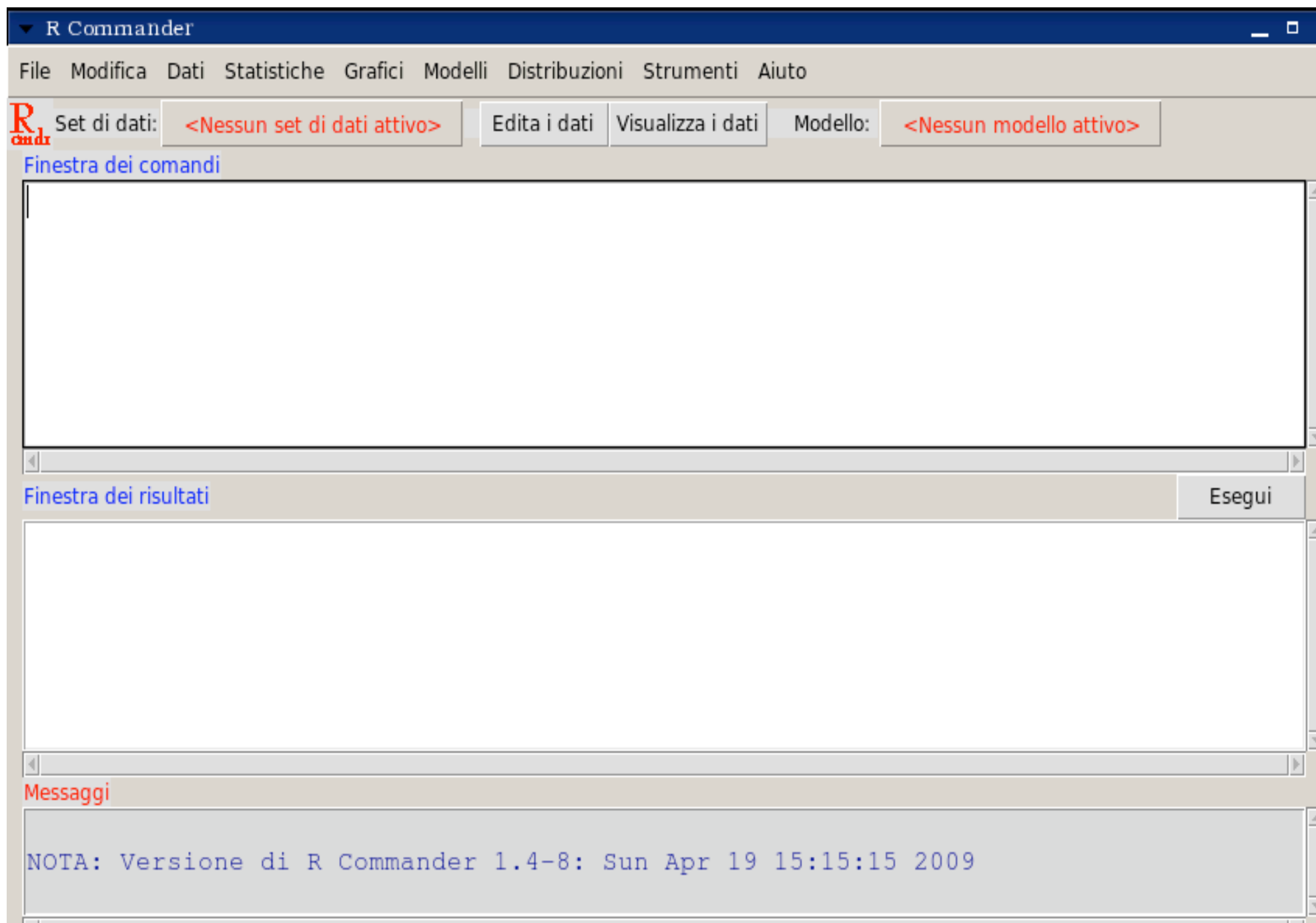
	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10
1	0	1	1	1	1	1	1	1	1	1
2	1	0	1	1	1	1	0	1	1	1
3	1	1	0	1	1	1	1	1	1	1
4	1	1	1	0	1	0	1	1	0	0
5	1	1	1	0	0	1	1	1	1	1
6	1	1	1	0	1	0	1	1	1	0
7	1	1	1	1	1	1	0	0	0	0
8	1	1	1	1	1	1	0	0	1	1
9	1	1	1	1	1	1	0	1	0	1
10	1	1	1	0	1	0	0	1	1	0
11	1	1	1	1	1	1	0	1	1	1
12	0	1	1	1	1	0	0	1	1	1
13	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0
15	1	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	0	0	1	0	0	0	0	1	0	1
19	0	0	0	1	0	0	0	1	0	0

L'interfaccia grafica Rcmdr (sta per “R Commander”) permette di richiamare alcune delle funzioni della versione base di R tramite menu a tendina, e contemporaneamente di visualizzare l'istruzione che deve essere digitata da tastiera qualora si volesse utilizzare la console.

A parte la barra dei menu, è suddivisa in quattro parti:

Una prima parte riga che contiene 4 pulsanti: il set di dati attivo (che indica il nome del data set sul quale si sta lavorando), due pulsanti che permettono di visualizzarlo e di modificarlo in una finestra vagamente simile ad Excel, anche se molto più scarna, ed il modello che si sta utilizzando. Il set di dati attivo dev'essere in formato data frame.

Ci sono poi la finestra dei comandi, la finestra dei risultati e la finestra dei messaggi.



La finestra dei comandi può essere utilizzata per scrivere da tastiera dei comandi come sulla console: la differenza è che premendo invio non succede niente, R non esegue l'istruzione digitata. Per eseguire un'istruzione, bisogna digitarla, selezionarla e schiacciare il pulsante “Esegui” in basso a destra rispetto alla finestra dei comandi.

In questo modo è possibile far eseguire all'elaboratore più di un comando alla volta: se se ne selezionano due, R li eseguirà in sequenza, dall'alto in basso.

Se si richiama una funzione da menu a tendina, si vedrà che il comando necessario ad avviarla viene scritto dall'elaboratore nella finestra dei comandi ma non è necessario selezionarlo e premere “Esegui” per eseguirlo.

In questo modo è possibile “scoprire” ed “imparare” i comandi per richiamare alcune funzioni di R tramite tastiera.

La finestra dei comandi, eccezion fatta per la diversa modalità di esecuzione delle istruzioni (selezione e clic su “Esegui” invece di invio) è esattamente come la console sulla versione base di R. Qualunque funzione possa essere eseguita sulla console può essere eseguita anche sulla finestra di comando.

Ad esempio, è possibile lanciare un pacchetto digitando “library(nome pacchetto)”. Le funzioni eventualmente presenti su questo pacchetto non diventano eseguibili tramite menu a tendina, possono essere chiamate solo tramite tastiera, ma diventano disponibili.

E' possibile lanciare i pacchetti anche da menu: Da “Strumenti” si esegue la voce “Carica il/i pacchetto(i)”. Parte una finestra di dialogo con la lista dei pacchetti disponibili. Si seleziona quello desiderato e si dà l'ok.

I comandi relativi alle funzioni che sono state eseguite rimangono scritti nella finestra: in questo modo se si vuole eseguire di nuovo una funzione è sufficiente selezionarla di nuovo e premere Esegui. E' inoltre possibile modificare le istruzioni prima di eseguirle di nuovo.

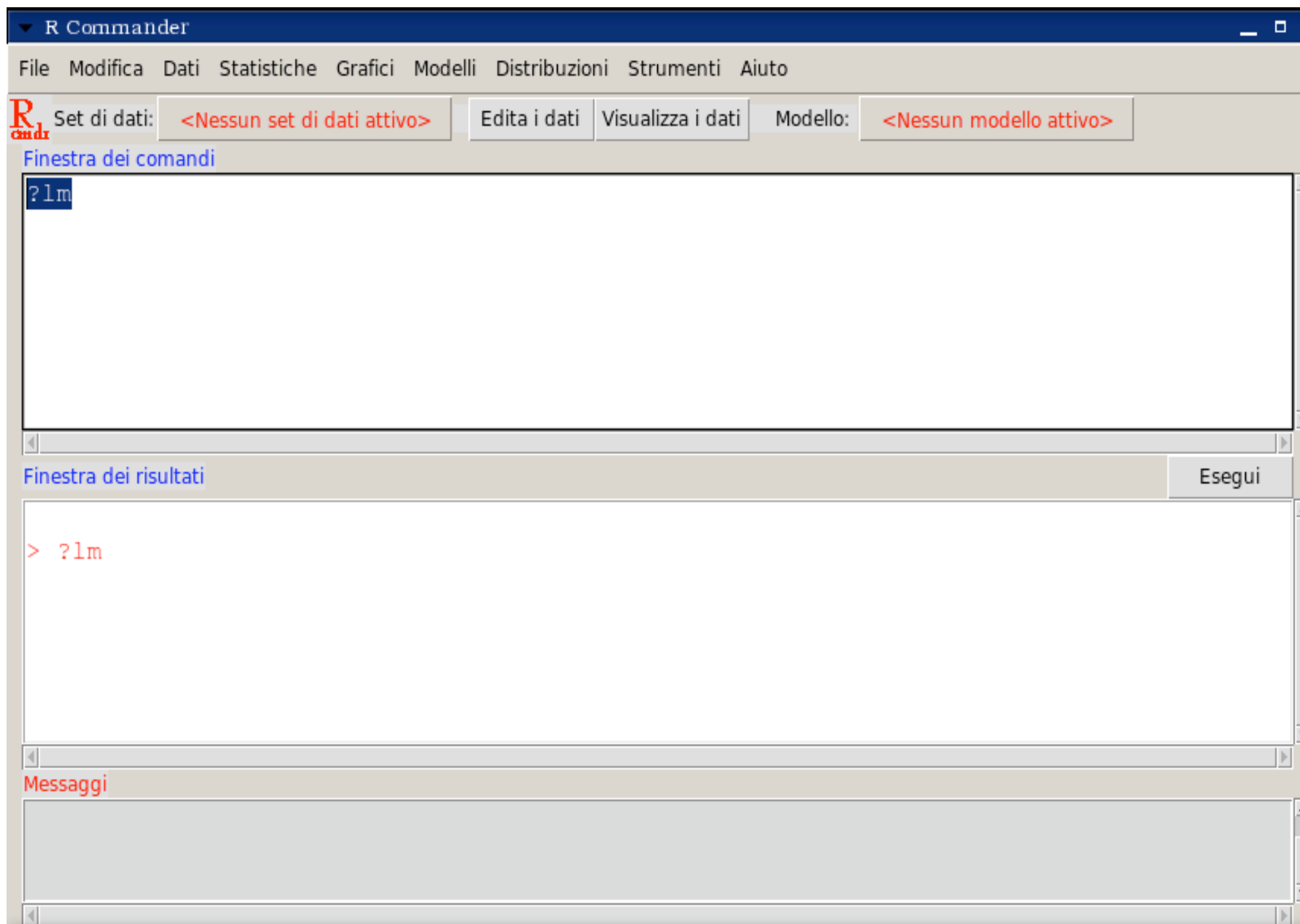
La finestra dei risultati è dove R scrive i risultati dei comandi che sono stati mandati in esecuzione. Sulla console, i risultati appaiono nello stesso posto in cui vengono digitate le istruzioni. L'interfaccia grafica Rcmdr (QCAGUI è assolutamente analoga) invece separa i comandi dai risultati.

Eventuali commenti, messaggi di errore o semplicemente di attenzione vengono scritti nella finestra più in basso, la finestra “Messaggi”.

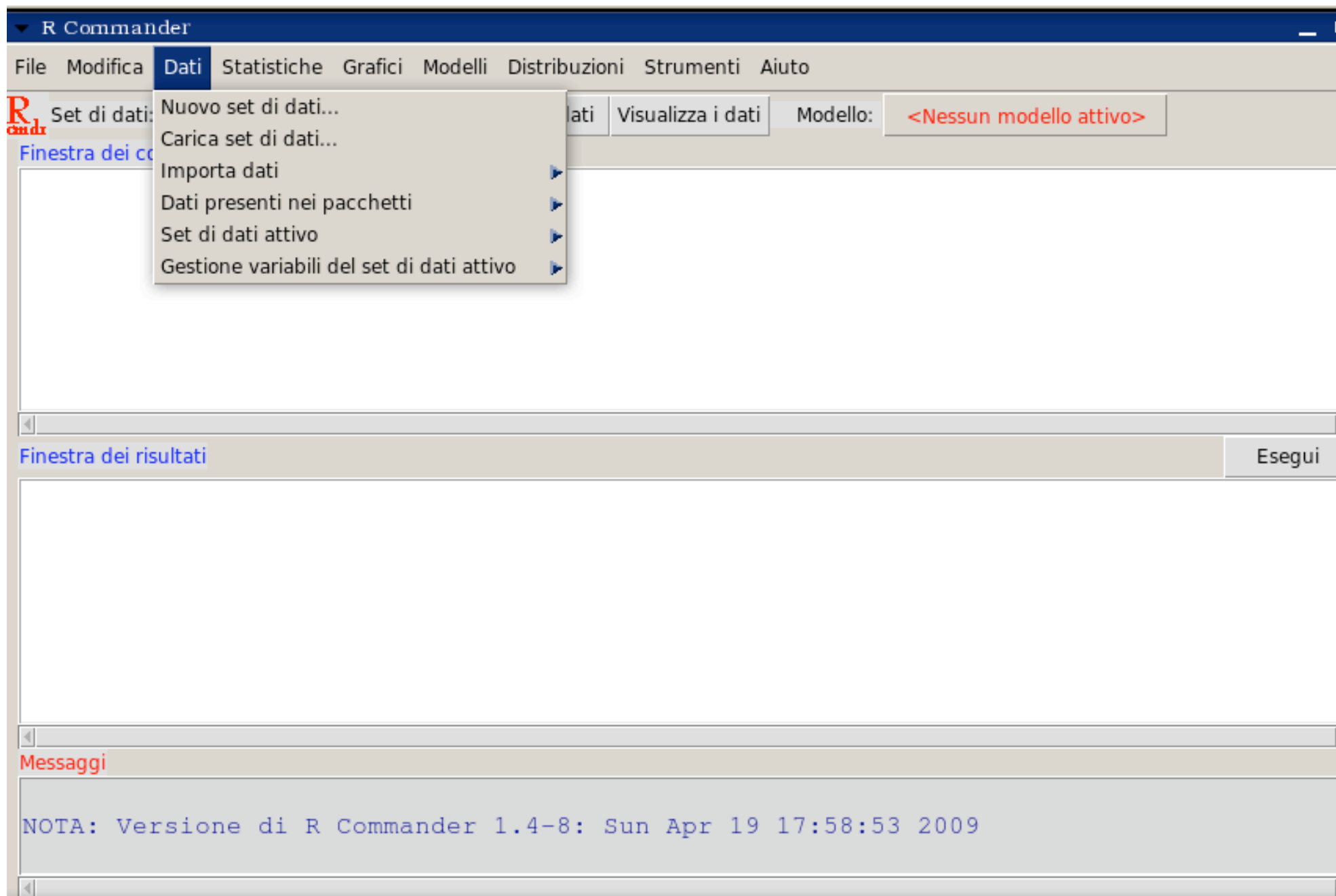
Se si ha bisogno di consultare la finestra di aiuto relativa ad una funzione che si intende eseguire da menu è possibile, invece di digitarne il nome preceduto dal punto interrogativo sulla console, premere il pulsante “Aiuto” che si trova in basso a destra sulla finestra di dialogo.

Si noti comunque che la console della versione base resta in esecuzione anche se si lancia una finestra come Rcmdr, quindi se si vuole eseguire qualche operazione digitando i comandi da tastiera è possibile utilizzarla. Allo stesso modo, le finestre di aiuto possono essere sempre lanciate dalla console.

Tutti i comandi che possono essere eseguiti da tastiera dalla console della versione base di R possono essere eseguiti da tastiera anche su Rcmdr, è sufficiente scriverli, selezionarli e premere il pulsante “Esegui”.



</



Nel menu “Dati” compaiono 5 voci: la prima, “Crea set di dati”, avvia una finestra di dialogo sulla quale è possibile digitare il nome del data frame che intende creare, poi, dando l'ok, parte una visualizzazione in forma di tabella tramite la quale è possibile inserire i dati come su Excel.

Il data frame risultante avrà un numero di righe e colonne pari a quelle rilevanti digitate nella tabella.

La seconda, “Carica set di dati”, permette di caricare un data frame precedentemente salvato, in formato Rda. Non è possibile caricare un set di dati salvato tramite un altro programma: per fare questo è necessario **IMPORTARE** il data set desiderato.

R Commander (RGB, 1 layer) 1278x750

File Modifica Dati Statistiche Grafici Modelli Distribuzioni Strumenti Aiuto

Set di dati: <Nessun set di dati attivo> Edita i dati Visualizza i dati Modello: <Nessun modello attivo>

Finestra dei comandi

Nuovo set di dati

Inserire il nome del set di dati: Dataset

OK Annulla Aiuto

Finestra dei risultati

R Data Editor

Copy Paste Quit

	var1	var2
1	2	1
2	3	5
3	2	4
4	4	2
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

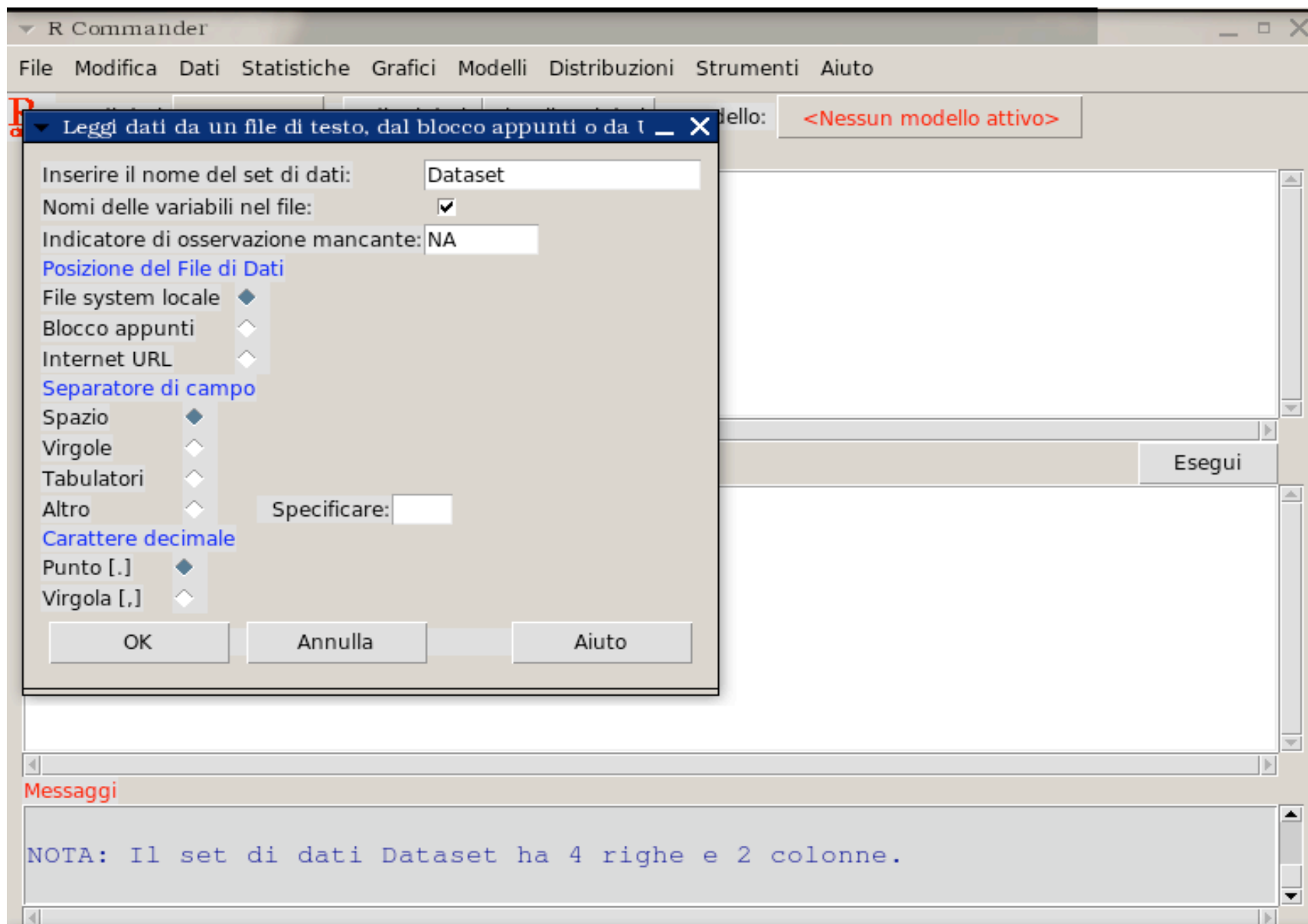
Esegui

Maccanri

L'importazione di dati si esegue dalla voce “Importa dati” del menu “Dati”.
formati possibili sono file di testo (è possibile con questa voce importare dati presenti su una pagina web, ad esempio delle tabelle che si trovano on line su sito dell'ISTAT), o in formato SPSS, Minitab o STATA. Parecchi data set sono resi disponibili in formato SPSS. Se bisogna importare un data set da Excel, esiste il già citato formato testuale “csv”, che tra l'altro permette di salvare solamente una tabella alla volta.

Una volta selezionato il tipo di dati che si desidera importare parte una finestra di dialogo, che, ad esempio, nel caso di importazione da file di testo, permette di specificare alcune chiavi di lettura del file, esattamente come la funzione `read.table` (in effetti è esattamente la stessa funzione che si esegue tramite menu invece che specificando tutto da tastiera), quale ad esempio il carattere che R dovrà considerare un separatore di campo, o se la prima riga deve essere considerata una riga di intestazioni.

Quando viene dato l'ok compare una seconda finestra di dialogo per specificare il percorso ed il nome del file da importare.



Come si vede, in basso a destra sulla finestra di dialogo c'è il pulsante “Aiuto”: premerlo equivale a scrivere “?read.table” sulla console o sulla finestra dei comandi.

Quando viene dato l'ok al file da importare, sia nella finestra dei comandi che in quella dei risultati comparirà la scritta relativa alla funzione che è stata eseguita, qualcosa come:

```
Dataset <- read.table("F:/Documenti/MS Excel/temp1.csv", header=TRUE,  
  sep=" ", na.strings="NA", dec=".", strip.white=TRUE)
```

In altre parole, si è eseguita una funzione `read.table` per creare un oggetto di nome “Dataset”, trasferendo il contenuto del documento testuale Excel “temp1.csv” contenuto nella cartella specificata, considerando la prima riga come una riga di intestazioni, utilizzando lo spazio come separatore di campi, codificando con NA i dati mancanti e utilizzando il punto come separatore di decimali.

Contrariamente a quanto succede importando i dati con comando diretto da tastiera, utilizzando le finestre di dialogo se si cerca di creare un data frame con un nome già utilizzato per un altro oggetto, R chiede la conferma.

L'altro effetto che si ottiene dall'importazione o creazione di un data frame è che il data frame appena importato o creato diventa immediatamente il data set attivo: questo è possibile vederlo verificando che il suo nome si trova sul campo “Set di dati” sotto la barra dei menu.

I due pulsanti “Edita i dati” e “Visualizza i dati” ora si riferiscono a lui, e qualunque funzione che prende in input un data set che venga lanciata da un menu lo considererà il data set di partenza.

Le funzioni che prendono in input un tipo diverso di dati o che non possono essere calcolate su quel data set specifico non sono attive e non possono essere lanciate da menu.

Sul data set attivo è possibile intervenire anche tramite i comandi presenti nelle ultime due voci del menu “Dati”, “Set di dati attivo” e “Gestione variabili del set di dati attivo”.

La prima contiene comandi che permettono di eseguire delle operazioni sul data frame, come ad esempio rinominare le variabili, eliminare delle righe o delle colonne (ad esempio quelle contenenti valori mancanti) o esportarlo in formato testuale.

La seconda contiene comandi che consentono di intervenire all'interno del data frame, ad esempio è possibile ricodificare alcune variabili, standardizzarle, od inserirne di nuove che risultano da operazioni su una o più variabili già presenti.

I comandi contenuti dentro “Set di dati attivo” e “Gestione variabili del set di dati attivo” permettono di eseguire operazioni più complesse delle semplici modifiche da tastiera che possono essere effettuate tramite l'editor che si può lanciare dalla versione base con il comando “fix”.

Ad esempio, è possibile codificare i valori mancanti: quando all'interno di un matrice (o di un vettore, o di un data frame) mancano uno o più valori, R inserisce la stringa “NA”. R non vede la rispettiva casella come una casella testuale, si tratta di una codifica per indicare il fatto che il dato non c'è.

Tuttavia alcune elaborazioni richiedono che tutti i dati siano presenti, quindi molto spesso è necessario attribuire un valore al dato mancante. Viceversa, a volte in alcuni data set i valori mancanti sono codificati con dei numeri particolari (ad esempio nelle indagini Eurostat i dati mancanti su variabili come “Numero di figli” o “Titolo di studio” sono codificati con numeri negativi), e può essere necessario a volte specificare che il valore non c'è.

Dall'editor della versione base questo si può fare solo sostituendo i valori uno alla volta, un'operazione noiosa ed a volte (se ad esempio si hanno migliaia di osservazioni) virtualmente impossibile.

Ad esempio, se si è interessati al numero medio di figli e si è in presenza di un data set con risposte mancanti, non ha senso fare la media comprendendo le codifiche dei valori mancanti. Ad esempio, nei data set Eurostat come l'ECH il valore che indica il dato mancante è -8: se, su 10 intervistati, 9 hanno risposto che hanno 2 figli e 1 non ha risposto, con questa codifica il vettore contenente le risposte è il seguente:

```
> a  
[1] 2 2 2 2 2 2 2 2 2 -8  
> mean(a)  
[1] 1  
> |
```

Calcolando la media sul vettore così com'è, risulta una media di 1 figlio a testa: ovviamente non ha senso.

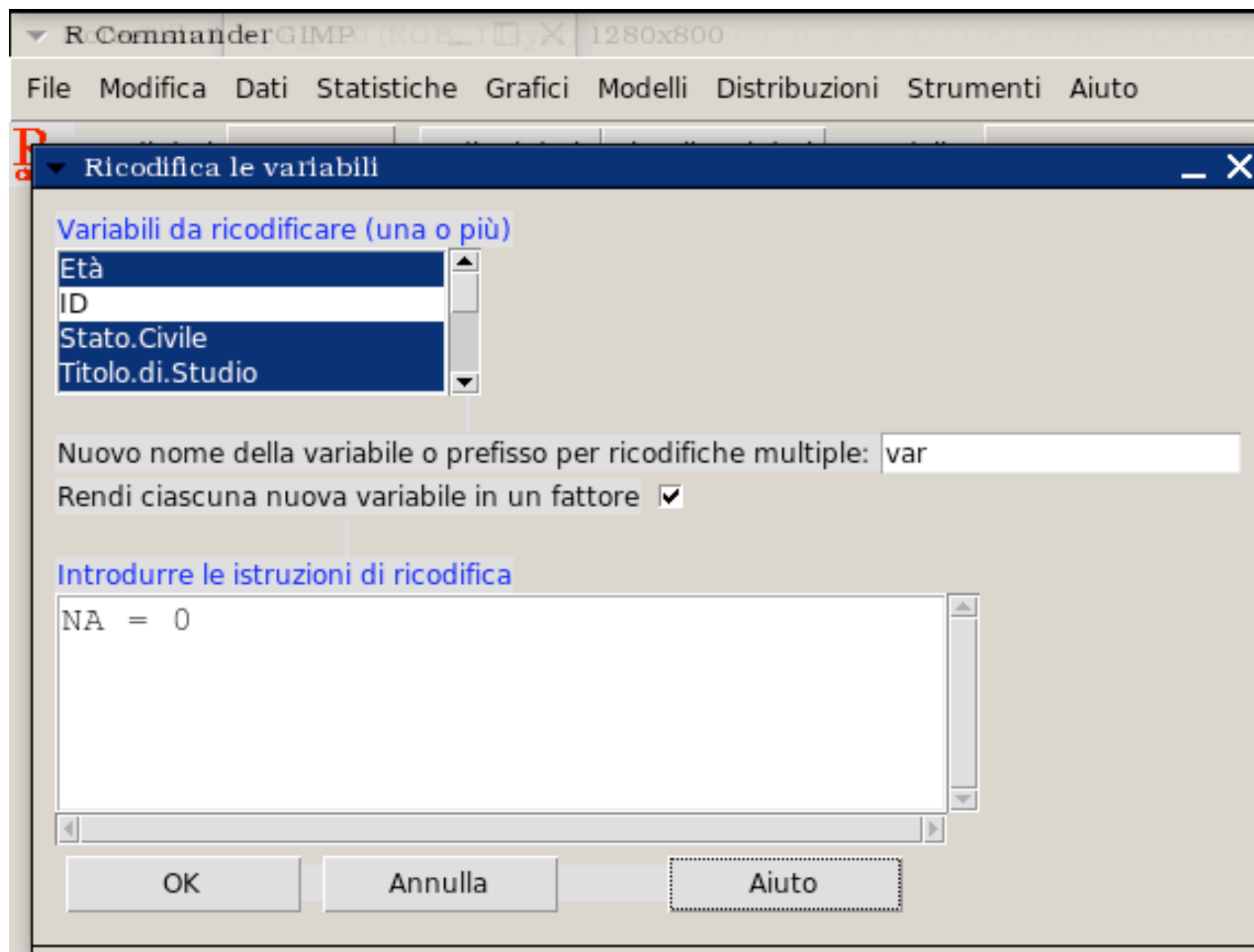
Si immagini di avere lo stesso problema su centinaia di osservazioni: la possibilità di intervenire modificando, o addirittura eliminando, tutti i valori mancanti è uno strumento molto potente ed altrettanto utile.

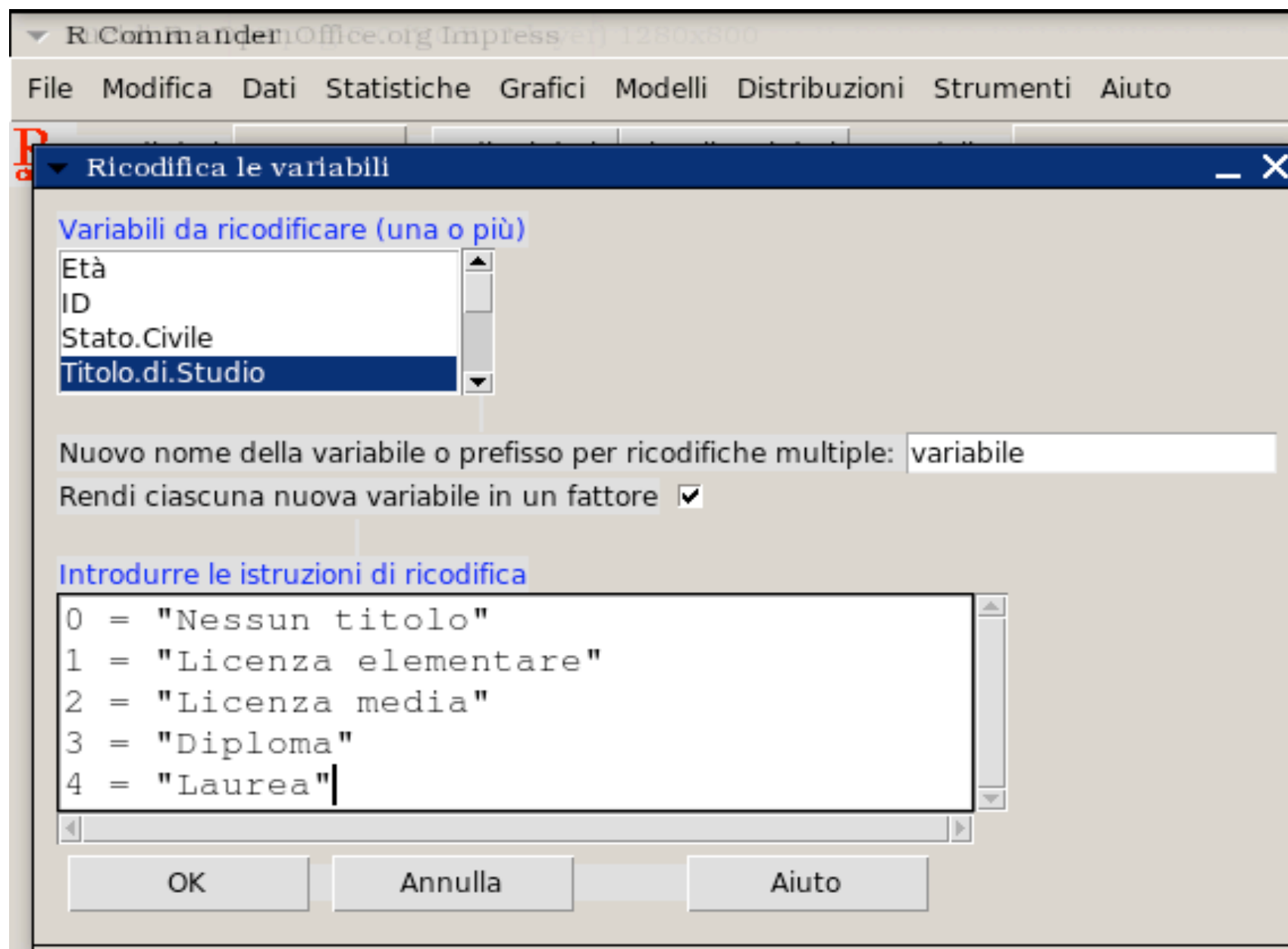
Ad esempio, per trasformare i dati mancanti codificati con NA in valori nulli bisogna selezionare il comando “Ricodifica le variabili...” dalla voce “Gestione variabili del set di dati attivo” del menu “Dati”: si aprirà una finestra di dialogo nella quale si devono selezionare le variabili interessate dalla ricodifica, e poi scrivere le istruzioni di ricodifica nel campo apposito. Nel caso specifico, $NA = 0$.

E' anche possibile trasformare variabili numeriche in variabili testuali: ad esempio il titolo di studio è indicato con 1 in caso di licenza elementare, 2 di licenza media, e via dicendo, le istruzioni di recodifica saranno:

```
1 = "Licenza elementare"  
2 = "Licenza media"  
3 = "Diploma"  
4 = "Laurea"
```

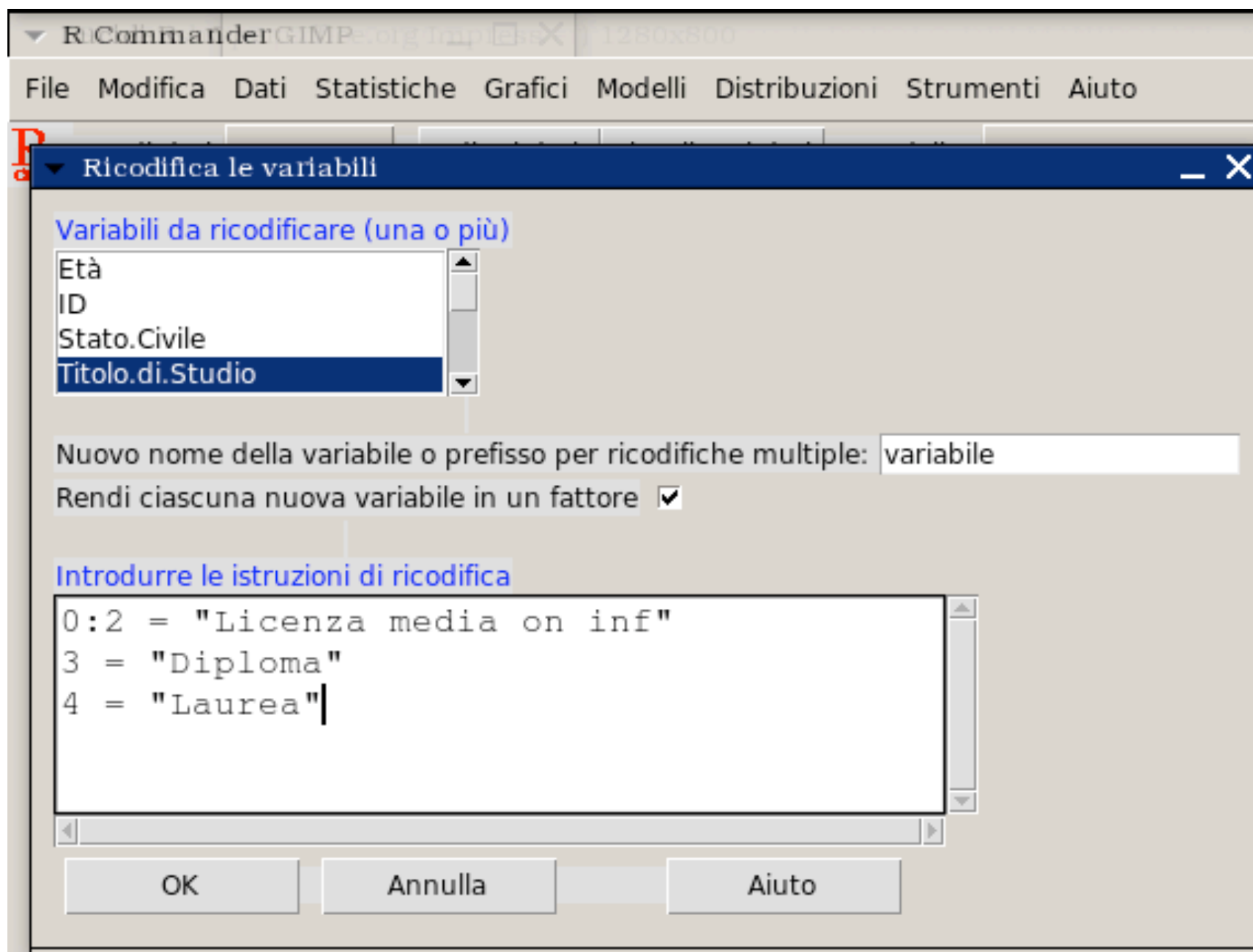
Le stringhe di caratteri vanno messe tra virgolette, e le diverse istruzioni vanno su righe diverse





E' possibile anche raggruppare alcune modalità sotto la stessa dicitura: ad esempio, nel caso del titolo di studio, se si vogliono raggruppare gli individui che non hanno raggiunto il diploma sotto la dicitura “Licenza media o inf”, si può procedere in due modi:

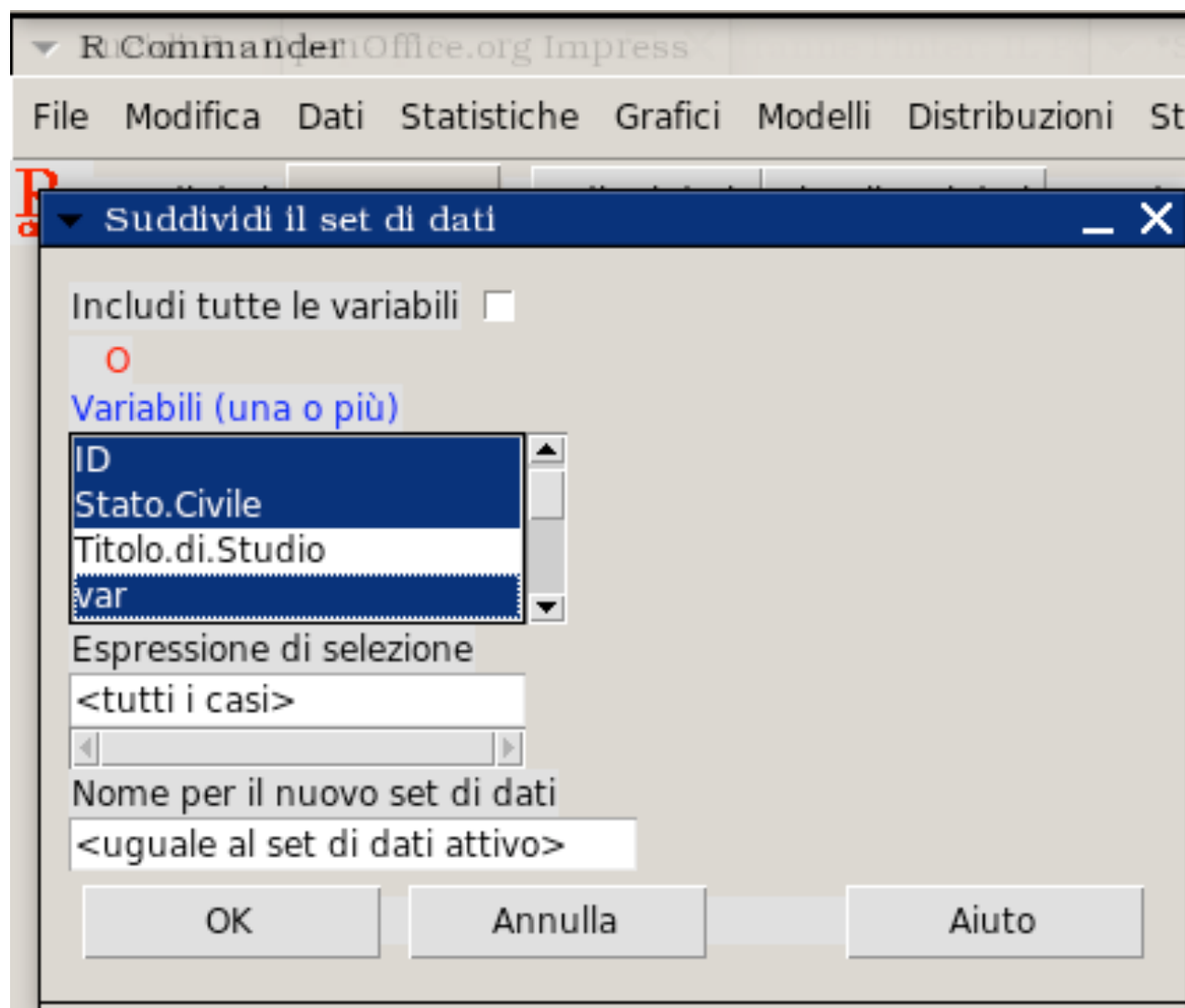
1. Scrivendo nelle istruzioni di ricodifica il valore minimo e massimo dell'intervallo di valori che si vuole raggruppare separati dai due punti: '0:2 = “Licenza media o inf”;
2. Elencando nelle istruzioni di codifica tutti i valori che si vogliono raggruppare separati da virgola: ' 0,1,2 = “Licenza media o inf”.



Come si vede, al centro della finestra di dialogo c'è l'opportunità di specificare il nome della variabile che si intende creare: infatti, nel ricodificare una variabile, R non sovrascrive la variabile esistente, ne crea un'altra e la mette in coda al data frame. Se il data frame originario aveva 15 colonne e se ne ricodificano 3, il nuovo data frame avrà lo stesso nome, ma 18 colonne, le ultime 3 sono le variabili ricodificate con il nome originario cui è stato aggiunto il prefisso specificato nel campo apposito.

Per ricondursi ad un data set con 15 colonne e la nuova variabile al posto di quella prima della ricodifica bisogna selezionare il comando “Estrai un sottoinsieme del set di dati attivo...” dalla voce “Set di dati attivo” del menu “Dati” e, dopo aver tolto la spunta dall'opzione “Includi tutte le variabili”, selezionare le variabili che si intende utilizzare.

In questo caso, è possibile specificare un nuovo nome per il data frame, o conservare quello vecchio. In questo secondo caso, R chiede conferma prima di sovrascrivere.



Per rinominare le variabili create con la ricodifica, in modo da utilizzare i nomi delle variabili originarie, si ricorre al comando “Rinomina le variabili...” dalla voce “Gestione variabili del set di dati attivo” del menu “Dati”. Si selezionano dalla finestra di dialogo le variabili che si intende rinominare, e parte una nuova finestra di dialogo che consente di effettuare il cambio di intestazione.

Nel rinominare le variabili bisogna tenere conto che gli spazi, così come diversi caratteri particolari (ad esempio “&”) non sono ammessi: per separare le parole di un'intestazione, R usa solitamente il punto. Quindi la variabile “Titolo di studio” solitamente comparirà come “Titolo.di.studio”.

ATTENZIONE!!!

Nella voce “Set di dati attivo” del menu “Dati” compare il comando “Elimin i casi con dati mancanti...”: lanciandolo è possibile selezionare su quali variabili un dato non presente determina l'eliminazione. In R con il termine “casi” si intendono le righe. Un tipico data set in cui le righe rappresentano i profili delle persone intervistate e le colonne rappresentano le variabili è chiamato “casi per variabili”.

Ne segue che eliminare i casi con dati mancanti equivale ad eliminare le righe che contengono un dato mancante nelle variabili specificate.