

Corso Introduttivo all'Utilizzo di R

"Sapienza Università di Roma"

Marco Alfó, Serena Arima, Pierpaolo Brutti, Alessio Farcomeni, Luca Tardella

Febbraio 2007

Hanno collaborato alla stesura delle presenti note: Marco Alfò, Serena Arima, Pierpaolo Brutti, Alessio Farcomeni, Luca Tardella.

©2007 Marco Alfò, Serena Arima, Pierpaolo Brutti, Alessio Farcomeni, Luca Tardella.
Il presente materiale non può essere copiato né utilizzato in tutto o in parte senza l'esplicito consenso scritto degli autori.

Premessa

Queste note costituiscono una prima raccolta di materiale di supporto per un corso introduttivo a R, un software distribuito gratuitamente, che costituisce sia un ambiente per le elaborazioni statistiche e per le rappresentazioni grafiche, sia un vero e proprio linguaggio di programmazione. Non hanno nessuna pretesa di rappresentare materiale definitivo ed esaustivo, ma dovrebbero semplicemente fornire un ausilio ai partecipanti del corso per ritrovare in forma organica già scritta alcuni aspetti essenziali trattati durante le lezioni frontali con l'ausilio dei PC in aula didattica.

Prima di cominciare cerchiamo di fare chiarezza su cosa ci si può aspettare da questo corso e cosa non ci si deve aspettare. Il corso è rivolto a persone che non hanno alcuna familiarità con l'ambiente R ma che comunque hanno necessità, per la propria attività lavorativa o di ricerca, di poter disporre di un software gratuito in grado di produrre analisi statistiche dal livello più elementare di organizzazione dei dati e descrittivo, ai più complessi modelli statistici e tecniche inferenziali avanzate basate su simulazioni.

Una raccomandazione da rivolgere ai partecipanti del corso è la seguente: per rendere il corso più efficace possibile è importante poter integrare il materiale che è stato predisposto in modo ovviamente generale e, necessariamente, un po' generico, con le esigenze pratiche dei partecipanti. Insomma l'invito è quello di sottoporre durante il corso quelli che sono i vostri problemi di analisi dei dati.

Cercheremo di rendere più interessante il corso proponendo dei percorsi per raggiungere obiettivi predeterminati più o meno specifici. Solo come spunti iniziali:

- *come possiamo importare i dati sperimentali raccolti in un foglio Excel?*
- *e quelli disponibili sul sito del mio Ente?*
- *come posso estrarre dai miei dati l'informazione di cui ho bisogno?*
- *ci sono indizi di brogli nei dati elettorali del 2006?*
- *e il surriscaldamento della Terra? (o quantomeno di Roma)*
- *Greggio, Renzi o?*

Capitolo 1

Elementi di Programmazione in R

1.1 Origini e un po' di filosofia su R e i software open source

Il software R è distribuito gratuitamente come software Open Source con licenza GPL (General Public License) ed è disponibile per diverse architetture e i più comuni sistemi operativi: Linux, MacOS, Unix, Windows.

Ricordiamo alcune tappe fondamentali per la nascita di R e per il suo sviluppo e diffusione:

- Nel 1996 Ross Ihaka e Robert Gentleman pubblicano la prima versione del software R per il sistema operativo MacOS
- Nel 1997 nasce il gruppo formato da statistici e informatici di tutto il mondo che si occupano dello sviluppo e della distribuzione del programma denominato R Development Core Team
- Nel 2003 nasce un'organizzazione non profit *R Foundation for Statistical Computing* che si occupa di promuovere lo sviluppo e la diffusione di fornire supporto legale per le questioni di copyright e di rappresentare un punto di riferimento per il mondo istituzionale, imprenditoriale e accademico.
- Nel 2004 l'Austrian Association for Statistical Computing (AASC) organizza la prima di una serie di conferenze internazionali dedicate alla diffusione di nuove funzionalità e pacchetti di R e allo scambio di nuove idee e promozione di nuovi progetti
- Nel 2006 la casa editrice Springer inaugura una serie dedicata agli utenti di R denominata proprio *Use-R*

1.2 Ottenere ed installare il programma R

La facilità di reperimento e l'assenza di costi sono tra i fattori che hanno favorito il successo di R. E' sufficiente aprire un programma per la navigazione di internet comunemente denominato *browser* e visitare il sito <http://www.r-project.org>. Il modo più semplice per installare il programma per un sistema operativo come Windows consiste nel prelevare il file binario autoinstallante che si trova seguendo i link appropriati

Download>CRAN>Mirror + vicino> Windows (95 and later) >Base >
una volta clickato sul nome del file corrispondente alla versione più aggiornata R-2.4.1-win32.exe (alla data del primo febbraio 2007) verrà scaricato il file su una cartella del vostro PC e quindi, mandando in esecuzione tale file con un doppio click sull'icona corrispondente, verrà attivata la procedura di installazione automatica che, in un paio di minuti, vi consentirà di avere R già funzionante sul vostro PC.

Ancora più brevemente il file dell'ultima versione distribuita del programma si trova direttamente al seguente link:

<http://rm.mirror.garr.it/mirrors/CRAN/bin/windows/base/R-2.4.1-win32.exe>.

Se invece usate un sistema operativo diverso da Windows ci limitiamo a segnalare che potete navigare dal sito <http://www.r-project.org> nelle cartelle appropriate ed avete la possibilità di reperire sia i file binari per il vostro sistema operativo sia i file sorgenti per una eventuale compilazione del programma dal codice sorgente. Per gli amanti di Linux può essere utile sapere che esiste una distribuzione *live* basata su Knoppix denominata Quantian¹ che include R tra i software di base ovvero come programma già installato e pronto per l'utilizzazione.

1.3 Entrare ed uscire da casa R (console)

La procedura di caricamento ed attivazione del programma R consiste nel procedere ad un doppio click sull'icona del programma che normalmente viene visualizzato sul Desktop del sistema operativo. Apparirà la cosiddetta *console* di R con il seguente simbolo

>

detto *prompt* dei comandi che segnala che R è disponibile ad accettare comandi da digitare sulla corrispondente linea. I comandi verranno eseguiti solo se scritti in modo sintatticamente corretto e completati con il tasto di Invio [Return].

Tanto per cominciare proviamo a mandare in esecuzione, nell'ordine, i seguenti 4 comandi:

¹Un sito con ampia descrizione della distribuzione è <http://dirk.eddelbuettel.com/quantian.html>, un mirror europeo per il download è <http://sunsite.informatik.rwth-aachen.de/ftp/pub/Linux/quantian/>.


```
a
2+2
a=2+2
a
```

Sulla console vedrete apparire qualcosa del genere

```
R version 2.4.0 (2006-10-03)
Copyright (C) 2006 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
```

```
R  un software libero ed  rilasciato SENZA ALCUNA GARANZIA.
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.
Scrivi 'license()' o 'licence()' per dettagli su come distribuirlo.
```

```
R  un progetto di collaborazione con molti contributi esterni.
Scrivi 'contributors()' per maggiori informazioni e 'citation()'
per sapere come citare R o i pacchetti di R nelle pubblicazioni.
```

```
Scrivi 'demo()' per una dimostrazione, 'help()' per la guida in linea, o
'help.start()' per l'help navigabile con browser HTML.
Scrivi 'q()' per uscire da R.
```

```
> 2+2
[1] 4
> a
Errore: oggetto "a" non trovato
> a=2+2
> a
[1] 4
>
```

A leggere attentamente il messaggio iniziale si ricavano già numerose informazioni utili, ma noi adesso non ci soffermeremo e procederemo rapidamente a spiegare come R interpreta i 4 comandi, come viene modificata la memoria di lavoro e come si può uscire dal programma o perdendo completamente la traccia di tutto quello che è stato eseguito oppure conservando in memoria sia i comandi eseguiti sia l'eventuale risultato.

Mandando in esecuzione il comando

q()

il sistema fa apparire la seguente finestra



Rispondendo “No” otterremo come risultato la chiusura del programma senza che rimanga traccia di tutto ciò che abbiamo fatto nella breve sessione descritta nei comandi appena eseguiti. Rispondendo “Sì” invece il programma termina lasciandoci però la possibilità di riaprire la console successivamente in modo da poter disporre, sia dei comandi digitati nella precedente sessione, sia degli eventuali risultati ottenuti, ad esempio il contenuto dell’oggetto denominato `a` ovvero il numero 4 corrispondente al risultato dell’addizione $2+2$. Per rendersene conto ci sono 2 modi:

- Aprire nuovamente R e mandare in esecuzione il comando

`a`

oppure

Aprire nuovamente R e digitare la freccia in su e in giù in modo da visualizzare i 4 comandi già eseguiti nella precedente sessione.

- Andando nell’apposita directory da dove il programma viene normalmente eseguito e che costituisce la directory di lavoro di default per R. Nella directory compaiono due file che prima non esistevano denominati

`.RData`
`.Rhistory`

Aprendoli con un editore di testo (e.g. Blocco Note) ci si rende conto subito che il primo file è un file di tipo binario dal contenuto illeggibile, mentre il secondo contiene un file di tipo ASCII leggibile e che in ciascuna linea contiene i comandi mandati in esecuzione nella precedente sessione di lavoro. Torneremo sulla directory di lavoro e la memoria di lavoro nelle sezioni successive.

1.4 Come orientarsi e a chi chiedere aiuto: documentazione ed help

Per schematizzare in modo scherzoso i diversi approcci al problema di imparare ad usare un software statistico elencherei i seguenti:

HW *Hard Way* Autoapprendimento

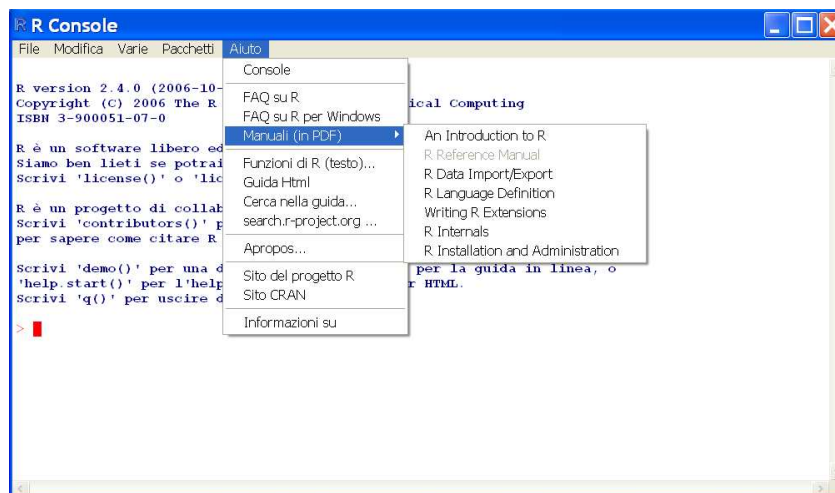
EW *Easy Way* Un corso introduttivo

IW *Italian Way* Un amico

alle brutte se gli amici competenti scarseggiano ritornate al punto EW e chiedete agli amici della Facoltà di Scienze Statistiche!

Per organizzare seriamente le idee faremo riferimento ai diversi modi di ottenere informazione tramite documentazione

1. Un primo modo di documentarsi su R può, e deve, basarsi sulla documentazione ufficiale ovvero i file in formato PDF consultabili direttamente dalla console (vedi figura) attraverso il Menu di aiuto a tendina



e comunque presenti in una delle sottocartelle della cartella dove R è stato installato. Nel nostro caso probabilmente sarà la cartella

C:\Programmi\R\R-2.4.0\doc>manual

2. In effetti esiste un altro modo ibrido un po' più flessibile di ottenere aiuto on-line con il comando
`help.start()`
3. Per quanto riguarda la documentazione sulla sintassi di specifici comandi ed i relativi argomenti opzionali sono utili i comandi in linea
 - `help(...)`
 - `help.search(...)`

Con il primo ad esempio con `help(quit)` o, equivalentemente ma più brevemente `?quit` si ottiene la sintassi del comando esteso che consente di uscire dalla sessione di R, cioè di chiudere il programma. Invece, inserendo una stringa all'interno delle parentesi del comando `help.search(...)` si ottiene in una nuova finestra una lista (eventualmente vuota) dei comandi che nella documentazione di help contengono quella stringa. Ad esempio con

`help.search('working directory')`

si ottiene una finestra che indica che nel programma standard di R esiste un comando/funzione denominata `getwd(...)` che contiene informazioni utili a riguardo. Dunque procedendo con `help(getwd)` si ottiene una descrizione completa di quel comando/funzione nonché il riferimento, nella sezione *see also* ad un'altra funzione collegata denominata `setwd(...)`

4. Si possono consultare libri pubblicati da editori commerciali. Ci limitiamo a segnalarne solo un numero molto ristretto dei più recenti e in particolare quelli in lingua italiana

- Franco Crivellari. *Analisi statistica dei dati con R*. Apogeo, 2006.
- John Verzani. *Using R for Introductory Statistics*. Chapman & Hall/CRC, Boca Raton, FL, 2005. ISBN 1-584-88450-9.
- Stefano Iacus and Guido Masarotto. *Laboratorio di statistica con R*. McGraw-Hill, Milano, 2003. ISBN 88-386-6084-0.
- John Maindonald and John Braun. *Data Analysis and Graphics Using R*. Cambridge University Press, Cambridge, 2003. ISBN 0-521-81336-0.
- Peter Dalgaard. *Introductory statistics with R*. Springer, New York, 2002. ISBN 0-387-95475-9.

5. Maggiori riferimenti a documentazione di varia natura possono essere consultati sul sito ufficiale in particolare

<http://www.r-project.org/other-docs.html>

6. Un altro modo per ricevere supporto è quello di iscriversi alle diverse mailing list

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

con tematiche più o meno specifiche. Per gli aggiornamenti in particolare è utile un periodico chiamato R News e scaricabile liberamente da

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

7. Un'altra opportunità per risolvere problemi iniziali è quella di consultare le Frequently Asked Questions sul sito

<http://cran.r-project.org/doc/FAQ/R-FAQ.html>.

1.5 Organizzazione e filosofia del programma: gli oggetti e le funzioni

Innanzitutto, bisogna mettere in evidenza il fatto che R viene concepito sin dalle origini come un interprete di comandi ovvero un programma che, piuttosto che organizzare il lavoro con una combinazione di attività controllate attraverso menu, finestre e mouse predilige l'uso di comandi da inserire sequenzialmente nella console

o, in alternativa, da eseguire in modalità non interattiva ovvero *batch* attraverso dei file ascii che possono essere letti ed interpretati/eseguiti in un'unica soluzione²

La grande flessibilità e potenza di R è dovuta tra l'altro anche al modo astratto con cui è organizzato e alla sua estendibilità e tale modo si ispira alla programmazione orientata agli oggetti. In effetti l'oggetto è la terminologia appropriata per descrivere le unità elementari contenute della memoria di lavoro di R detta anche *workspace*. Nelle seguenti sezioni ci accingiamo dunque ad entrare in questa logica prevalentemente attraverso esempi. Vedremo

- come assegnare un nome ad un oggetto
- quanti tipi di oggetti R può gestire
- modi, classi, attributi
- cambiare modi, classi, attributi
- errori, avvertimenti e coercizioni.

Un'altra nozione importantissima per comprendere e sfruttare a pieno R è quella di funzione che corrisponde in buona sostanza al comando che noi chiediamo di interpretare nella console. Sarà molto importante capire la sintassi appropriata di ciascuna funzione e come è strutturato ed organizzato l'output del comando o della funzione stessa. Per il momento ci limitiamo ad usare una suggestiva similitudine ... *le funzioni sono come cuccioli: non si fanno vedere finché non le chiami (con il loro nome possibilmente pronunciato bene!). Hanno una bocca (le parentesi) attraverso cui sfamarle (nel nostro caso con gli argomenti della funzione), e si lamenteranno nel caso in cui non le nutriamo in modo appropriato*³

Ma innanzitutto descriveremo brevemente le operazioni numeriche elementari con le quali possiamo usare R come una calcolatrice⁴.

1.6 Le operazioni numeriche elementari

Nella pagina che segue viene visualizzata una schermata della console che dovrebbe essere sufficiente, senza ulteriori commenti, a comprendere il significato delle diverse operazioni, degli annidamenti e delle usuali priorità.

²Il comando per eseguire l'elenco dei comandi presenti in un file ascii è `source(...)`. L'argomento ... deve essere una stringa corrispondente al nome, comprensivo di eventuale percorso o path, ad esempio `source('c:/temp/prova.txt')`.

³Liberamente tradotta da Verzani (2005) [p.11].

⁴Per ulteriori dettagli si possono invocare i seguenti aiuti:

```
help(Arithmetic)
help(Sqrt)
help(Logaritmico)
help(Trig)
help(Special)
```



```

R Console
File Modifica Varie Pacchetti Aiuto

R version 2.4.0 (2006-10-03)
Copyright (C) 2006 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R è un software libero ed è rilasciato SENZA ALCUNA GARANZIA.
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.
Scrivi 'license()' o 'licence()' per dettagli su come distribuirlo.

R è un progetto di collaborazione con molti contributi esterni.
Scrivi 'contributors()' per maggiori informazioni e 'citation()'
per sapere come citare R o i pacchetti di R nelle pubblicazioni.

Scrivi 'demo()' per una dimostrazione, 'help()' per la guida in linea, o
'help.start()' per l'help navigabile con browser HTML.
Scrivi 'q()' per uscire da R.

> 1+2*3
[1] 7
> (1+2)*3
[1] 9
> 9-3^2
[1] 0
> (9-3)^2
[1] 36
> log(1)
[1] 0
> log(10)
[1] 2.302585
> log(10,base=10)
[1] 1
> exp(1)
[1] 2.718282
> 100*10%
Errore: errore di sintassi in "100*10%"
> 100*(0.10)
[1] 10
> 100*(1+0.10)^10
[1] 259.3742
> abs(15-40)
[1] 25
> sqrt(25)
[1] 5
> 

```

1.7 I diversi tipi di oggetto: modi e classi

Ecco un elenco di esempi di assegnazione di oggetti di natura diversa

```

> # Le entità più semplici che possono costituire un oggetto di R
> # (a cui assegnare un nome) sono
>
> # [1] elementi scalari (numerici {[interi, reali] o complessi})
> a=10
> b=15.53
> c=3.51+1.2i
>
> # [2] elementi stringhe di caratteri
> d="Ciao"
> e="Luca"
>
> # [3] elementi logici
> f=TRUE

```

```
> g=FALSE
> h=F
> i=T
```

Possiamo interrogare R con il comando `mode(...)` ed ottenere stampato sulla console il *modo* dell'oggetto. Per gli oggetti `a`, `b` e `c` avremo come risultato del comando la stringa `'numeric'`. Per gli oggetti `d` ed `e` avremo come risultato del comando la stringa `'character'`. Per gli oggetti `f`, `g`, `h` e `i` avremo come risultato del comando la stringa `'logical'`.

Di tutti questi tipi di oggetto possiamo in realtà costruire dei *vettori* usando il comando `c(...)` la cui sintassi prevede che si possa inserire all'interno delle parentesi la lista degli elementi del vettore separando opportunamente con una virgola. Ad esempio i seguenti comandi

```
> c(1,2,3)
[1] 1 2 3
> primovettore=c("a","b","c")
> secondovettore=c(f,g,h,i)
```

determinano la stampa di un vettore di modo `"numeric"` con i primi tre numeri interi naturali, l'assegnazione di un vettore (`primovettore`) di modo `"numeric"` con le prime tre lettere dell'alfabeto e un vettore di modo `"logical"` contenente la sequenza dei seguenti quattro elementi logici.

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

I vettori possono anche essere costituiti da 0 elementi ovvero possono essere vuoti. Il modo più semplice di fare ciò è il seguente

```
> vuoto1=c()
> vuoto1
NULL
```

Alternativamente anche le seguenti istruzioni producono vettori vuoti ma con qualche differenza nei modi

```
> vuoto2=vector(length=0,mode="numeric")
> vuoto3=vector(length=0,mode="character")
> vuoto2
numeric(0)
> vuoto3
character(0)
```

Forniremo maggiori dettagli sui vettori nella sezione dedicata alle strutture dei dati. Al momento ci limitiamo a segnalare che i vettori devono contenere tutti elementi dello stesso modo. Infine sottolineiamo che gli oggetti che abbiamo appena chiamato scalari sono in realtà concepiti in R come vettori di lunghezza uno. Anticipiamo che per conoscere la lunghezza di un vettore la funzione che rivela tale informazione è `length(...)`.

Un'altra cosa importante da sapere sui modi degli oggetti è che con il comando `mode(...)`, non solo siamo in grado di conoscere il modo, ma, possiamo anche

modificarlo anche se non sempre il risultato della modifica consente di assegnare valori coerenti agli elementi del vettore modificato. Ad esempio nel caso del vettore `secondovettore` di modo `'logical'` è possibile convertirlo in un oggetto di modo `'numeric'` nei seguenti modi

```
> as.numeric(secondovettore)
[1] 1 0 0 1
> secondovettore
[1] TRUE FALSE FALSE TRUE
> mode(secondovettore)
[1] "logical"
> mode(secondovettore)="numeric"
> secondovettore
[1] 1 0 0 1
```

dove la trasformazione degli elementi avviene secondo la conversione di un `TRUE` in un `1` e un `FALSE` in un `0`. E' ancor più naturale la conversione di oggetti di modo `'numeric'` in `'character'`

```
> terzovettore=as.character(c(1,2,3))
> terzovettore
[1] "1" "2" "3"
```

Talvolta le conversioni di modo avvengono automaticamente ad esempio quando si mettono (volontariamente o involontariamente) elementi di modo diverso nello stesso vettore. In tal caso esistono delle priorità tra i modi che si possono convertire. A titolo di esempio

```
> c(3,"4")
[1] "3" "4"
> c(3,T)
[1] 3 1
> c("T",T)
[1] "T" "TRUE"
```

L'ultimo modo di cui tratteremo è il modo `'list'`. Un oggetto di modo `'list'` può contenere elementi di natura differente ma il suo modo di essere visualizzato e trattato dipenderà da un'altra caratteristica dell'oggetto denominata classe. Solo come esempio introduttivo consideriamo il seguente

```
> lista=list("Luca",15,TRUE)
> lista
[[1]]
[1] "Luca"

[[2]]
[1] 15

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

In R anche previsto il modo `function` che peraltro corrisponde al modo di tutte le funzioni/comandi che hanno richiesto l'uso di argomenti (al limite anche vuoti) che abbiamo impiegato finora.


```
> mode(quit)
[1] "function"
```

Iniziamo ad anticipare che è possibile definire delle funzioni personalizzate nuove, in aggiunta a quelle presenti nei pacchetti di base, attraverso la funzione – scusate il gioco di parole – `function(..)`. Ad esempio per calcolare la superficie di un cerchio usando il raggio come argomento della funzione

```
> area.cerchio=function(raggio){
+ a=raggio*raggio*pi
+ return(a)
+ }
> area.cerchio(1)
[1] 3.141593
> area.cerchio(2)
[1] 12.56637
```

1.8 Le principali operazioni elementari sugli oggetti

Cercheremo di fornire attraverso esempi alcune semplici operazioni tra oggetti di R non necessariamente di modo numerico. Ci limitiamo a segnalare solo le più importanti e frequenti nell'uso comune.

In particolare, per quanto riguarda gli oggetti (vettori o matrici) di modo numerico consideriamo le seguenti funzioni più utilizzate:

```
> vettore=c(1,2,3,4,5)
> length(vettore)
[1] 5
> sum(vettore)
[1] 15
> prod(vettore)
[1] 120
> min(vettore)
[1] 1
> max(vettore)
[1] 5
> cumsum(vettore)
[1] 1 3 6 10 15
> cumprod(vettore)
[1] 1 2 6 24 120
> diff(cumprod(vettore))
[1] 1 4 18 96
```

Un altro principio di funzionamento importantissimo quando si effettuano operazioni su vettori è capirne l'effetto quando si trattano vettori di dimensione uguale o diversa

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

```
> vettore*vettore
[1] 1 4 9 16 25
> vettore*5
[1] 5 10 15 20 25
> vettore*c(5)
[1] 5 10 15 20 25
> vettore*c(1,2)
[1] 1 4 3 8 5
Warning message:
lunghezza dell'oggetto pi  lungo
      non  un multiplo di quella dell'oggetto pi  piccolo in: vettore * c(1, 2)
```

Notiamo il fatto che in presenza di due oggetti vettoriali di dimensione una multipla dell'altra R si comporta come se l'oggetto la cui lunghezza è sottomultiplo sia esteso replicando ciclicamente i suoi elementi il numero giusto di volte per far corrispondere le lunghezze e quindi effettua l'operazione sui termini omologhi. Tuttavia anche se due oggetti vettoriali di lunghezza diversa non hanno lunghezza multipla l'uno dell'altro R usa l'estensione dell'oggetto più piccolo replicando il suo contenuto sequenzialmente a partire dal primo elemento fino ad arrivare al momento in cui l'oggetto vettoriale replicato con questa semplice regola non raggiunge la stessa dimensione dell'oggetto più grande. In tal caso però avvisa l'utente con un messaggio di avvertimento. In sostanza le ultime due operazioni appena viste sono equivalenti alle seguenti

```
> vettore*c(5,5,5,5,5)
[1] 5 10 15 20 25
> vettore*c(1,2,1,2,1)
[1] 1 4 3 8 5
```

Per quanto riguarda gli oggetti (vettori o matrici) di modo `character` consideriamo le seguenti funzioni.

Per gli oggetti di modo `character` la funzione `paste(...)`

```
> ?paste
> paste("Ciao","Luca")
[1] "Ciao Luca"
> paste("Ro","ma",sep="")
[1] "Roma"
> paste("Roma","Milano","Torino",sep="-")
[1] "Roma-Milano-Torino"
> paste(c("Roma","Milano","Torino"),collapse="-")
[1] "Roma-Milano-Torino"
> paste(c("Roma","Milano","Torino"),"Palermo",sep="-")
[1] "Roma-Palermo" "Milano-Palermo" "Torino-Palermo"
```

Per gli oggetti di modo `logical` le operazioni importanti da conoscere sono desumibili dai seguenti esempi.

```
> # NOT ovvero Negazione
> # [carattere ! o punto esclamativo]
> !TRUE
[1] FALSE
```

```

> # AND ovvero Congiunzione o Intersezione
> # [carattere & o lettera e-commercial e o ampersand]
> TRUE & FALSE
[1] FALSE
> TRUE & TRUE
[1] TRUE
> c(T,T,F,F)&c(T,F,T,F)
[1] TRUE FALSE FALSE FALSE
> # OR ovvero Alternativa o Unione
> # [carattere | o barra verticale]
> TRUE | FALSE
[1] TRUE
> c(T,T,F,F)|c(T,F,T,F)
[1] TRUE TRUE TRUE FALSE
> vettore.logico=c(T,T,F,F,T)
> vettore.logico
[1] TRUE TRUE FALSE FALSE TRUE
> any(vettore.logico)
[1] TRUE
> all(vettore.logico)
[1] FALSE
> which(vettore.logico)
[1] 1 2 5

```

Ovviamente non tutte le operazioni hanno senso! Ad esempio è difficile pretendere da R il calcolo della somma di stringhe. In effetti

```

> sum(c("Luca","Alessandra"))
Errore in sum(..., na.rm = na.rm) : invalid 'type' (character) of argument

```

`sum` si aspetta di operare su dati di modo numerico e di fatto non è neanche attribuibile per coercizione un valore numerico a Luca.

```

> cosa.esce=as.numeric("Luca")
Warning message:
si  prodotto un NA per coercizione
> cosa.esce
[1] NA

```

1.9 NA, NaN e dintorni

Vediamo se siete bravi e riuscite ad indovinare cosa viene visualizzato come risultato dei seguenti comandi digitati sulla console:

```

> 0/4
[1] 0
> 4/0
[1] Inf
> 0/0
[1] NaN

```

```
> zero
Errore: oggetto "zero" non trovato
> #(oppure luca al posto di zero)
> as.numeric("zero")
[1] NA
Warning message:
si  prodotto un NA per coercizione
```

Spesso nei dati statistici si avrà a che fare con dati non disponibili e bisognerà saperli trattare opportunamente talvolta identificando quali dati sono NA e quali non lo sono. A tale scopo è molto utile la funzione `is.na(...)` eventualmente in combinazione con l'operatore logico che ne forma la sua negazione

```
> is.na(c(1,2,3,NA,4))
[1] FALSE FALSE FALSE  TRUE FALSE
> !is.na(c(1,2,3,NA,4))
[1]  TRUE  TRUE  TRUE FALSE  TRUE
```

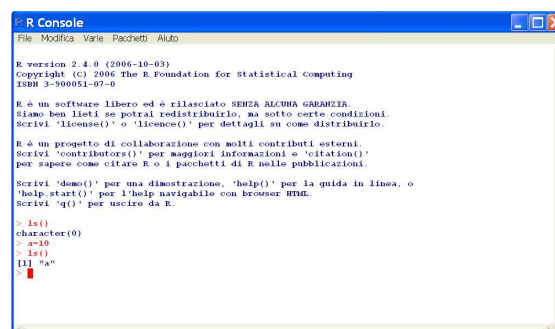
1.10 Gestione della memoria interna ed esterna

Il workspace e la directory di lavoro

Come è organizzata la memoria della sessione di lavoro e che fa in modo che R riesca a ricordare il contenuto dell'oggetto denominato `a`? In effetti il contenuto della memoria (temporanea) della sessione corrente di R, cominciata nel momento in cui è apparsa la console, può essere visualizzato attraverso il comando

```
ls()
```

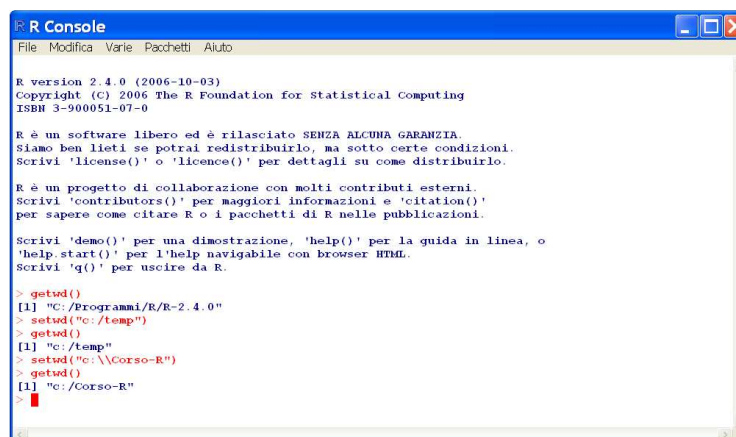
Se mandiamo in esecuzione il comando `ls()` proprio all'apertura della sessione vedremo apparire la scritta `character(0)`



che corrisponde ad un vettore di modo `'character'` composto da 0 elementi ovvero vuoto. Una nuova assegnazione come `a=10` fa in modo che una nuova invocazione del comando `ls()` produce come risultato la visualizzazione di un vettore di modo stringa composto da un solo elemento e precisamente `'a'`.

```
> a=10  
> ls()  
[1] "a"
```

È possibile anche cancellare rimuovere dalla memoria oggetti indesiderati (magari perché occupano troppo spazio in memoria) con il comando `rm(...)`. La sintassi della funzione `rm(...)` prevede come argomento o il singolo nome dell'oggetto oppure una sequenza di nomi separati da una virgola oppure ancora un vettore (di modo 'character') contenente la lista degli oggetti da rimuovere. Tali oggetti saranno memorizzati nella directory di lavoro corrente laddove si decidesse di uscire dalla sessione di lavoro con il comando `q()` rispondendo affermativamente alla richiesta di salvataggio dell'area (memoria) di lavoro. Già, ma come facciamo a sapere qual è la directory di lavoro corrente e come possiamo eventualmente modificarla? I due comandi che svolgono tali compiti sono `getwd()` e `setwd(...)`. Quest'ultimo richiede come argomento una stringa contenente tutto il percorso desiderato con il particolare accorgimento che il percorso anziché i simboli *backslash* \ usualmente impiegati nei sistemi Windows richiede l'uso dei simboli *slash* / tipicamente adottato nei sistemi Unix-Linux-like. Alternativamente si può sostituire il *backslash* \ con un doppio *backslash* \\. Come esempio visualizziamo i comandi della seguente sessione.



```
R Console  
File Modifica Varie Pacchetti Aiuto  
  
R version 2.4.0 (2006-10-03)  
Copyright (C) 2006 The R Foundation for Statistical Computing  
ISBN 3-900051-07-0  
  
R è un software libero ed è rilasciato SENZA ALCUNA GARANZIA.  
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.  
Scrivi 'license()' o 'licence()' per dettagli su come distribuirlo.  
  
R è un progetto di collaborazione con molti contributi esterni.  
Scrivi 'contributors()' per maggiori informazioni e 'citation()' per  
sapere come citare R o i pacchetti di R nelle pubblicazioni.  
  
Scrivi 'demo()' per una dimostrazione, 'help()' per la guida in linea, o  
'help.start()' per l'help navigabile con browser HTML.  
Scrivi 'q()' per uscire da R.  
  
> getwd()  
[1] "C:/Programmi/R/R-2.4.0"  
> setwd("c:/temp")  
> getwd()  
[1] "c:/temp"  
> setwd("c:\\Corso-R")  
> getwd()  
[1] "c:/Corso-R"  
>
```

1.11 Uscire senza dimenticare nulla, rientrare e ritrovare tutto come lo si era lasciato

Supponiamo di aver mandato in esecuzione nella sessione corrente i comandi da 001 a 020. Nella memoria di lavoro *workspace* sono presenti gli oggetti risultanti o da assegnazioni dirette o da elaborazioni. Abbiamo già visto che tali oggetti potranno essere memorizzati nella memoria fisica del computer e non solo nella memoria temporanea della sessione di lavoro a seguito dell'uscita tramite `q()` seguita dalla richiesta di salvataggio dell'area (memoria) di lavoro. Un modo alternativo per raggiungere lo stesso scopo senza uscire dalla sessione di lavoro è quello di usare il

comando `save('nomefile')` che produce un file nella directory di lavoro corrente denominato *nomefile* e che corrisponde ad un file binario analogo a quello denominato *.RData* prodotto dall'uscita della sessione di lavoro.

Un modo alternativo per poter ritornare ad avere nella memoria di lavoro gli stessi oggetti con lo stesso contenuto consiste nell'effettuare esattamente gli stessi comandi eseguiti fino al momento in questione. I comandi

```
history
savehistory
source
```

aiutano a tener traccia e precisamente, nell'ordine, a visualizzare, memorizzare ed eventualmente eseguire nuovamente i comandi eseguiti fino al momento corrente della sessione di lavoro. Ad esempio provate a replicare i comandi della seguente sessione

```
R Console
File Modifica Varie Pacchetti Aiuto
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.
Scrivi 'license()' o 'licence()' per dettagli su come distribuirlo.

R è un progetto di collaborazione con molti contributi esterni.
Scrivi 'contributors()' per maggiori informazioni e 'citation()'
per sapere come citare R o i pacchetti di R nelle pubblicazioni.

Scrivi 'demo()' per una dimostrazione, 'help()' per la guida in linea, o
'help.start()' per l'help navigabile con browser HTML.
Scrivi 'q()' per uscire da R.

> getwd()
[1] "C:/Programmi/R/R-2.4.0"
> ls()
character(0)
> a=10
> b=3+3
> ls()
[1] "a" "b"
> history()
> setwd("C:/Corso-R")
> dir(all.files=TRUE)
[1] ""
> savehistory()
> dir(all.files=TRUE)
[1] ""
> q()

R History
File Modifica
getwd()
ls()
a=10
b=3+3
ls()
history()
```

Dopodiché uscite dalla sessione di lavoro senza salvare l'area di lavoro. Riaprite una nuova sessione (la cui memoria, verificatelo, dovrebbe essere vuota) ed eseguite

Il comando `loadhistory('C:/Corso-R/.Rhistory')` può essere utile per rieseguire con il solo uso delle frecce comandi mandati in esecuzione in una precedente sessione di lavoro.

I comandi visti in questa sezione sono tutti disponibili anche sotto Windows nel Menu File alle voci Salva area di lavoro Carica area di lavoro Sorgente codice R Salva cronologia Carica cronologia.

1.12 Le più importanti strutture per i dati statistici

Qual è la cosa più importante per la Statistica? I dati, naturalmente!

Un paio di premesse sono doverose.

In primo luogo il titolo della sezione è volutamente forzato. In realtà il titolo più appropriato sarebbe stato: *vettori* e *liste*. In effetti R concepisce e si organizza come se

tutte le informazioni numeriche fossero organizzate in vettori e, laddove ne interessa uno solo come nel caso degli scalari esso viene pensato come un vettore costituito di un solo elemento.

In secondo luogo si deve chiarire la distinzione fondamentale tra vettori e liste. I primi saranno oggetti costituiti da tanti elementi, tutti della stessa natura, ragion per cui, il modo del vettore corrisponde a quello dei suoi elementi e la loro numerosità determina la lunghezza del vettore. In una lista invece possiamo mettere insieme in modo ordinato ed indicizzato elementi di modo e natura completamente diversa ad esempio possiamo mettere come primo elemento un vettore di elementi logici, come secondo un vettore di stringhe di lunghezza diversa dal precedente e come terzo un vettore di un solo elemento di modo logico.

Ad esempio i seguenti sono 5 vettori di lunghezza diversa e di cui l'ultimo addirittura di lunghezza 0.

```
> vettore1=3
> vettore2=c(4,5,6)
> vettore3=c(T,F,T,F)
> vettore4=c("Luca","Alessandra")
> vettore5=vector(mode="character",length=0)
>
>
> lista=list(vettore3,vettore4,vettore1)
```

Dal precedente esempio apprendiamo che per combinare più elementi della stessa natura (modo) in un vettore dobbiamo usare la funzione `c(...)` separando con virgole gli elementi costitutivi mentre per costruire una lista abbiamo bisogno della funzione/comando `list(...)`. In effetti nella funzione `c(...)` è lecito mettere tra gli elementi costitutivi anche vettori non necessariamente di uguale dimensione, come ad esempio nel seguente

```
c(1,2,vettore1,vettore2)
c(vettore3,vettore4,vettore1)
```

Sarà anche diversa, in parte, la sintassi con cui potremo accedere ad una parte di un vettore e quella con cui potremo accedere a parti di una lista.

Introduciamo ora gli oggetti di tipo matriciale, o meglio, di classe matriciale. Il modo più semplice per generare una matrice numerica in R è il seguente

```
> matrice1=matrix(0,nrow=6,ncol=4)
> matrice1
  [,1] [,2] [,3] [,4]
[1,]  0   0   0   0
[2,]  0   0   0   0
[3,]  0   0   0   0
[4,]  0   0   0   0
[5,]  0   0   0   0
[6,]  0   0   0   0
```

```
> mode(matrice1)
[1] "numeric"
> class(matrice1)
[1] "matrix"
```

In pratica il primo argomento della matrice (un vettore formato dal solo numero 0) è stato ripetuto un numero sufficiente di volte per formare una griglia di $6 \times 4 = 24$ numeri disposti su sei righe e 4 colonne. Con l'aiuto di `help(matrix)` riusciamo anche a capire che laddove abbiamo bisogno di riorganizzare in forma matriciale un vettore di lunghezza $6 \times 4 = 24$ è sufficiente usare tale vettore come primo argomento (prima della prima virgola) della funzione `matrix(...)` e specificare nei successivi argomenti `nrow=...` e `ncol=...` rispettivamente, il numero di righe e di colonne opportuno.

```
> matrice2=matrix(1:24,nrow=6,ncol=4)
> matrice2
      [,1] [,2] [,3] [,4]
[1,]    1    7   13   19
[2,]    2    8   14   20
[3,]    3    9   15   21
[4,]    4   10   16   22
[5,]    5   11   17   23
[6,]    6   12   18   24
```

Di default i primi elementi del vettore saranno utilizzati per riempire la prima colonna, i successivi per la seconda colonna e così via. Per fare in modo che venga riempita la prima riga anziché la prima colonna possiamo usare un terzo argomento della funzione `matrix(...)` e precisamente `byrow=...` dove al posto di `...` R si attende un elemento logico di tipo TRUE o FALSE. In definitiva nel seguente modo otteniamo il risultato desiderato:

```
> matrix(1:24,nrow=6,byrow=T)
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
[5,]   17   18   19   20
[6,]   21   22   23   24
```

Dall'esempio si evince anche che non c'è bisogno di specificare il numero di colonne qualora il numero di elementi del vettore e il numero di righe dichiarato nelle parentesi tonde portano ad un numero intero di colonne necessarie per esaurire nelle righe richieste il numero di elementi nel vettore.

Alcune funzioni che possono tornare utili per la manipolazioni delle matrici sono le seguenti:

```
> dim(matrice2)
[1] 6 4
> nrow(matrice2)
[1] 6
```



```
> ncol(matrice2)
[1] 4
> t(matrice2)
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]     1     2     3     4     5     6
[2,]     7     8     9    10    11    12
[3,]    13    14    15    16    17    18
[4,]    19    20    21    22    23    24
```

Analogamente alle matrici numeriche possiamo anche definire oggetti-matrici di elementi character oppure oggetti-matrici di elementi logical

```
> matrix4=matrix(c(TRUE,TRUE,FALSE),nrow=3,ncol=5,byrow=T)
> matrix4
      [,1] [,2] [,3] [,4] [,5]
[1,] TRUE TRUE FALSE TRUE TRUE
[2,] FALSE TRUE TRUE FALSE TRUE
[3,] TRUE FALSE TRUE TRUE FALSE
> mode(matrix4)
[1] "logical"
> class(matrix4)
[1] "matrix"
```

Molto brevemente un array è una struttura regolare del tipo matriciale ma utilizza un numero di dimensioni anche superiori a 2.

```
> array(1:24,dim=c(4,3,2))
, , 1

      [,1] [,2] [,3]
[1,]     1     5     9
[2,]     2     6    10
[3,]     3     7    11
[4,]     4     8    12

, , 2

      [,1] [,2] [,3]
[1,]    13    17    21
[2,]    14    18    22
[3,]    15    19    23
[4,]    16    20    24
```

In effetti la matrice è un caso particolare di array quando le sue dimensioni sono solo 2.

```
> array(1:24,dim=c(6,4))
      [,1] [,2] [,3] [,4]
[1,]     1     7    13    19
[2,]     2     8    14    20
[3,]     3     9    15    21
[4,]     4    10    16    22
[5,]     5    11    17    23
[6,]     6    12    18    24
```

Sorprenderà ora il fatto che per trattare le matrici di dati tipicamente impiegate come punto di partenza delle analisi statistiche si ricorre ad un diverso tipo di oggetto o meglio classe di oggetto in R: il `data.frame`. La ragione è presto spiegata. Abbiamo già detto che gli oggetti di tipo vettoriale, matriciale o array devono avere tutti gli elementi dello stesso modo (ad esempio tutti numerici) mentre tipicamente con i dati statistici abbiamo rilevato diversi tipi di informazione sulle stesse unità statistiche. Ad esempio di 4 persone possiamo aver registrato le seguenti informazioni:

```
> nome=c("Alessandra","Luca","Anna","Paolo")
> eta=c(25,40,5,2)
> sesso=c("F","M","F","M")
> vaccino=c(T,T,F,F)
> dati=data.frame(nome,eta,sesso,vaccino)
> dati
```

	nome	eta	sesso	vaccino
1	Alessandra	25	F	TRUE
2	Luca	40	M	TRUE
3	Anna	5	F	FALSE
4	Paolo	2	M	FALSE

```
> dim(dati)
[1] 4 4
```

In realtà esiste un comando che consente la possibilità di aggregare le 4 informazioni vettoriali in un unico oggetto di classe matrice, ma il risultato di tale operazione può non corrispondere alle aspettative per la ragione che, laddove necessario, come nel precedente esempio, viene automaticamente effettuata una coercizione che trasforma il modo degli elementi dei vari vettori ad un unico modo.

```
> mat.dati=cbind(nome,eta,sesso,vaccino)
> mat.dati
```

	nome	eta	sesso	vaccino
[1,]	"Alessandra"	"25"	"F"	"TRUE"
[2,]	"Luca"	"40"	"M"	"TRUE"
[3,]	"Anna"	"5"	"F"	"FALSE"
[4,]	"Paolo"	"2"	"M"	"FALSE"

```
> mode(mat.dati)
[1] "character"
> dim(mat.dati)
[1] 4 4
```

Qualche informazione in più su come un oggetto di R è strutturato internamente ed alcune caratteristiche del suo contenuto possono essere ricavate dai seguenti comandi

```
> str(dati)
'data.frame': 4 obs. of 4 variables:
 $ nome : Factor w/ 4 levels "Alessandra","Anna",...: 1 3 2 4
 $ eta : num 25 40 5 2
 $ sesso : Factor w/ 2 levels "F","M": 1 2 1 2
 $ vaccino: logi TRUE TRUE FALSE FALSE
```

```
> attributes(dati)
$names
[1] "nome"      "eta"       "sesso"     "vaccino"

$row.names
[1] 1 2 3 4

$class
[1] "data.frame"

> str(mat.dati)
chr [1:4, 1:4] "Alessandra" "Luca" "Anna" "Paolo" "25" ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:4] "nome" "eta" "sesso" "vaccino"
> attributes(mat.dati)
$dim
[1] 4 4

$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "nome"      "eta"       "sesso"     "vaccino"
```

1.13 Come accedere all'interno delle strutture di dati (e come modificarne il contenuto)

Affrontiamo ora il problema di estrarre da un oggetto di R la sola informazione che ci interessa ovvero come selezionare dati all'interno di un oggetto avendo in mente la situazione tipica in cui i dati a nostra disposizione sono molteplici e organizzati in forma vettoriale o matriciale.

La regola aurea di sintassi di R è che per estrarre informazioni da oggetti di tipo vettoriale o matriciale vengono usate le parentesi quadre (semplici, ovvero ripetute una sola volta) come segue

```
vettore1[...]
matrice1[...,...]
array1[...,...,...]
```

mentre per gli oggetti di modo `list` si possono usare anche (ma hanno un significato diverso) le parentesi quadre doppie (ovvero ripetute due volte) come segue

```
lista[...]
lista[[...]]
```

Vedremo nelle prossime sezioni come la sintassi dei vettori si può estendere anche ad oggetti di classe `table` oppure ad oggetti di classe `data.frame` anche se per questi ultimi oggetti il discorso è un po' più complicato.

Partiamo dal caso più semplice di un vettore. I puntini ... all'interno delle parentesi quadre dovranno contenere un vettore che identifichi quali elementi dell'oggetto vettoriale denominato `vettore1` vogliamo estrarre.

Supponiamo di avere a disposizione i dati delle temperature massime assolute registrate nei 50 anni (1906-1955) a Roma nel mese di Febbraio.

```
#vettore1=dati.apat[seq(2,by=12,length=50),"temp"]
> vettore1
 [1] 15.3 14.6 15.9 13.2 16.4 16.6 17.4 16.1 16.2 15.2
[11] 19.2 15.1 15.2 16.6 16.4 15.0 18.8 15.2 14.2 17.9
[21] 18.5 17.7 17.2 15.6 15.0 NA 13.8 17.9 17.6 15.5
[31] 17.8 18.8 16.3 17.9 18.4 16.0 15.4 19.8 14.8 18.2
[41] 18.4 17.6 16.8 17.0 18.4 17.2 15.6 17.1 16.3 17.6
> vettore1[]
 [1] 15.3 14.6 15.9 13.2 16.4 16.6 17.4 16.1 16.2 15.2
[11] 19.2 15.1 15.2 16.6 16.4 15.0 18.8 15.2 14.2 17.9
[21] 18.5 17.7 17.2 15.6 15.0 NA 13.8 17.9 17.6 15.5
[31] 17.8 18.8 16.3 17.9 18.4 16.0 15.4 19.8 14.8 18.2
[41] 18.4 17.6 16.8 17.0 18.4 17.2 15.6 17.1 16.3 17.6
> vettore1[1]
[1] 15.3
> vettore1[c(1,2,3,4,5,6,7,8,9,10)]
 [1] 15.3 14.6 15.9 13.2 16.4 16.6 17.4 16.1 16.2 15.2
```

Se non inseriamo nulla nelle parentesi quadre ovvero le lasciamo vuote il verranno considerati tutti gli elementi. Se indichiamo la sequenza da 1 a 10 verranno considerati solo i primi 10 elementi in questo caso corrispondenti agli anni 1906-1915. Ovviamente, nulla vieta, se utile, di prelevarli in ordine sparso e replicato come nel seguente esempio:

```
> vettore1[c(1,50,30,1,1,50,50)]
 [1] 15.3 17.6 15.5 15.3 15.3 17.6 17.6
```

In effetti dobbiamo aprire una breve digressione sul fatto che per estrarre parti di un oggetto spesso usiamo sequenze regolari di indici. A tal proposito segnaliamo le seguenti funzioni che definiscono per via incrementale o per replicazione delle sequenze regolari.

```
> ?seq
> seq(1,10)
 [1] 1 2 3 4 5 6 7 8 9 10
> 1:10
 [1] 1 2 3 4 5 6 7 8 9 10
> seq(2,20,by=2)
 [1] 2 4 6 8 10 12 14 16 18 20
> seq(1,20,by=2)
 [1] 1 3 5 7 9 11 13 15 17 19
```

```
> seq(0,1,length=10)
[1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667 0.7777778 0.8888889
[10] 1.0000000
> rep("A",10)
[1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
> rep(c(T,F),10)
[1] TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE
[16] FALSE TRUE FALSE TRUE FALSE
> rep(c(0,1),c(2,8))
[1] 0 0 1 1 1 1 1 1 1 1
```

Un altro modo di estrarre elementi da un oggetto vettoriale fa riferimento all'uso all'interno della parentesi quadra di vettori di modo logico con lunghezza uguale a quella del vettore da cui estrarre gli elementi. Il principio è il seguente: se il primo elemento del vettore logico è TRUE l'elemento viene considerato se invece è FALSE l'elemento non viene considerato. Analogamente funzionerà per il secondo, terzo elemento è così via. Riprendendo le temperature degli anni 1906-1915 possiamo estrarre le sole temperature degli anni dispari nel seguente modo.

```
temp.1906.1915=vettore1[1:10]
temp.1906.1915[c(F,T,F,T,F,T,F,T)]
temp.1906.1915[c(F,T)]
```

In realtà vale anche la scorciatoia di mettere un vettore di 2 soli logici che verrà considerato da R come replicato per il numero di volte sufficiente ad raggiungere la lunghezza del vettore delle temperature di partenza⁵.

L'importanza dell'uso dei logici per estrarre informazioni dai vettori risiede nell'uso degli operatori di confronto all'interno delle parentesi quadre. Ad esempio per estrarre tutte le temperature massime superiori a 18 gradi in quei 50 anni

```
vettore1=dati.apat[seq(2,by=12,length=50),"temp"]
> vettore1
[1] 15.3 14.6 15.9 13.2 16.4 16.6 17.4 16.1 16.2 15.2
[11] 19.2 15.1 15.2 16.6 16.4 15.0 18.8 15.2 14.2 17.9
[21] 18.5 17.7 17.2 15.6 15.0 NA 13.8 17.9 17.6 15.5
[31] 17.8 18.8 16.3 17.9 18.4 16.0 15.4 19.8 14.8 18.2
[41] 18.4 17.6 16.8 17.0 18.4 17.2 15.6 17.1 16.3 17.6
> vettore1>18
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[9] FALSE FALSE TRUE FALSE FALSE FALSE FALSE
[17] TRUE FALSE FALSE FALSE TRUE FALSE FALSE
[25] FALSE NA FALSE FALSE FALSE FALSE FALSE TRUE
[33] FALSE FALSE TRUE FALSE FALSE TRUE FALSE TRUE
[41] TRUE FALSE FALSE FALSE TRUE FALSE FALSE
```

⁵Bisogna fare attenzione perché eventuali errori nel dimensionamento del vettore di logici potrebbe determinare un risultato indesiderato senza che R produca messaggi di errore o di avvertimento. In effetti è anche vero che se il vettore dei logici ha lunghezza superiore del vettore da cui estrarre gli elementi saranno proprio gli elementi di quest'ultimo ad essere ripetuti a partire dal primo in poi per adeguarsi alla lunghezza del vettore dei logici.

```
[49] FALSE FALSE
> vettore1[vettore1>18]
[1] 19.2 18.8 18.5 NA 18.8 18.4 19.8 18.2 18.4 18.4
```

Notate il fatto che il dato non disponibile NA produce un logico NA.

Infine è data anche la possibilità di estrarre tutti gli elementi di un vettore a meno degli elementi i cui indici sono specificati da un opportuno vettore di indici preceduto dal segno meno - all'interno delle quadre

```
> vettore1[-1]
[1] 14.6 15.9 13.2 16.4 16.6 17.4 16.1 16.2 15.2 19.2
[11] 15.1 15.2 16.6 16.4 15.0 18.8 15.2 14.2 17.9 18.5
[21] 17.7 17.2 15.6 15.0 NA 13.8 17.9 17.6 15.5 17.8
[31] 18.8 16.3 17.9 18.4 16.0 15.4 19.8 14.8 18.2 18.4
[41] 17.6 16.8 17.0 18.4 17.2 15.6 17.1 16.3 17.6
> vettore1[-(1:25)]
[1] NA 13.8 17.9 17.6 15.5 17.8 18.8 16.3 17.9 18.4
[11] 16.0 15.4 19.8 14.8 18.2 18.4 17.6 16.8 17.0 18.4
[21] 17.2 15.6 17.1 16.3 17.6
> vettore1[-length(vettore1)]
[1] 15.3 14.6 15.9 13.2 16.4 16.6 17.4 16.1 16.2 15.2
[11] 19.2 15.1 15.2 16.6 16.4 15.0 18.8 15.2 14.2 17.9
[21] 18.5 17.7 17.2 15.6 15.0 NA 13.8 17.9 17.6 15.5
[31] 17.8 18.8 16.3 17.9 18.4 16.0 15.4 19.8 14.8 18.2
[41] 18.4 17.6 16.8 17.0 18.4 17.2 15.6 17.1 16.3
```

Ricordiamo che per rimuovere il dato mancante è più semplice usare i logici oppure la funzione `complete(...)`

```
> vettore1[!is.na(vettore1)]
[1] 14.6 15.9 13.2 16.4 16.6 17.4 16.1 16.2 15.2 19.2
[11] 15.1 15.2 16.6 16.4 15.0 18.8 15.2 14.2 17.9 18.5
[21] 17.7 17.2 15.6 15.0 NA 13.8 17.9 17.6 15.5 17.8
[31] 18.8 16.3 17.9 18.4 16.0 15.4 19.8 14.8 18.2 18.4
[41] 17.6 16.8 17.0 18.4 17.2 15.6 17.1 16.3 17.6
> complete(vettore1)
```

A questo punto dovrebbero essere chiare tutte le premesse per capire come estrarre parti di un oggetto matriciale. Usando i dati delle temperature di Roma prelevati dal sito <http://www.scia.sinanet.apat.it/avvertenze.asp#contenuti> nell'oggetto di classe `data.frame` denominato `dati.apat`

```
> dati.apat
  mese.anno temp      Data n.dat
1  Gen-1906 13.6 12/01/1906   30
2  Feb-1906 15.3 27/02/1906   28
3  Mar-1906 17.8 10/03/1906   26
4  Apr-1906 24.6 21/04/1906   30
5  Mag-1906 29.1 29/05/1906   29
6  Giu-1906 32.2 28/06/1906   29
7  Lug-1906 32.9 19/07/1906   31
```

```

8  Ago-1906 34.4      04/08/1906      30
9  Set-1906 31.7      05/09/1906      26
10 Ott-1906 25.1      05/10/1906      31
    .....          .....          ..
    .....          .....          ..

```

cerchiamo di capire come e perché estraiamo le seguenti informazioni

```

> str(dati.apat)
'data.frame': 1212 obs. of 4 variables:
 $ mese.anno: chr "Gen-1906" "Feb-1906" "Mar-1906" "Apr-1906" ...
 $ temp : num 13.6 15.3 17.8 24.6 29.1 32.2 32.9 34.4 31.7 25.1 ...
 $ Data : chr " 12/01/1906" " 27/02/1906" " 10/03/1906" " 21/04/1906" ...
 $ n.dati : num 30 28 26 30 29 29 31 30 26 31 ...
> dim(dati.apat)
[1] 1212 4
> dati.apat[5,2]
[1] 29.1
> dati.apat[5,1:2]
 mese.anno temp
5 Mag-1906 29.1
> dati.apat[5,]
 mese.anno temp Data n.dati
5 Mag-1906 29.1 29/05/1906 29
> dati.apat[dati.apat[,1]=="Gen-2000",]
 mese.anno temp Data n.dati
1129 Gen-2000 14 30/01/2000 31
> which(dati.apat[,1]=="Gen-2000")
[1] 1129
> temperature=dati.apat[seq(1,1212,by=12),]
> temperature
 mese.anno temp Data n.dati
1 Gen-1906 13.6 12/01/1906 30
13 Gen-1907 13.7 03/01/1907 31
25 Gen-1908 13.7 01/01/1908 29
37 Gen-1909 13.8 17/01/1909 30
49 Gen-1910 15.2 04/01/1910 31
    .....          .....          ..
    .....          .....          ..

```

Ma ancora non basta. Per essere più flessibile ed intuitivo R consente di estrarre parti di un oggetto attraverso i nomi assegnati ad essi! Un oggetto vettoriale oppure di classe `matrix`, `array` o `data.frame` è prevista la possibilità di assegnare delle etichette o nomi alle posizioni delle diverse dimensioni. Supponiamo di voler registrare in un oggetto vettoriale i primi numeri estratti nell'estrazione del 7 gennaio del 1939

```

> est1=c(58,27,40,85,73,73,19,9)
> est1
[1] 58 27 40 85 73 73 19 9
> est1[3]
[1] 40
> names(est1)

```

NULL

Può essere utile assegnare a ciascun elemento del vettore un'etichetta.

```
> names(est1)=c("BA","FI","MI","NA","PA","RM","TO","VE")
> est1
BA FI MI NA PA RM TO VE
32  1 68 90 22 32 87 21
> est1[6]
RM
32
> est1["RM"]
RM
32
```

Attenzione, il fatto che venga visualizzato anche l'etichetta riferita a quell'elemento del vettore non deve trarre in inganno, il contenuto di `est1['RM']` è il solo numero 32. Analogo discorso vale per le etichette di ciascuna delle due dimensioni di oggetti di tipo `matrix` o `data.frame`. In particolare avendo a disposizione anche i vettori dei secondi estratti, terzi estratti etc. possiamo costruire una matrice affiancando i vettori (di uguale lunghezza) dei vari estratti usando la funzione `cbind(...)` come segue

```
> est2
[1] 22 57 38 44 80 24 43 43
> est3
[1] 47 81 57 48 39  4 10 61
> est4
[1] 49 43 67 88 38 39 31 14
> est5
[1] 69 61  7 55 57 22 27 75
> lotto.1939=cbind(est1,est2,est3,est4,est5)
> lotto.1939
      est1 est2 est3 est4 est5
BA    58   22   47   49   69
FI    27   57   81   43   61
MI    40   38   57   67    7
NA    85   44   48   88   55
PA    73   80   39   38   57
RM    73   24    4   39   22
TO    19   43   10   31   27
VE     9   43   61   14   75
> mode(lotto.1939)
[1] "numeric"
> class(lotto.1939)
[1] "matrix"
> dim(lotto.1939)
[1] 8 5
> dimnames(lotto.1939)
[[1]]
[1] "BA" "FI" "MI" "NA" "PA" "RM" "TO" "VE"

[[2]]
```



```
[1] "est1" "est2" "est3" "est4" "est5"
```

```
> rownames(lotto.1939)
[1] "BA" "FI" "MI" "NA" "PA" "RM" "TO" "VE"
> colnames(lotto.1939)
[1] "est1" "est2" "est3" "est4" "est5"
```

In realtà la funzione `cbind(...)` può affiancare più oggetti, separandoli opportunamente con una virgola all'interno delle tonde, anche di tipo matriciale purché abbiano numero di colonne uguali. Usando i nomi-etichetta delle due dimensioni della matrice possiamo ricavare il terzo estratto sulla ruota di Napoli nel seguente modo

```
> lotto.1939["NA", "est3"]
[1] 48
```

come pure vari estratti su varie ruote usando all'interno delle quadre vettori di modo `character` come segue

```
> lotto.1939[c("RM", "NA"), c("est1", "est5")]
  est1 est5
RM   73   22
NA   85   55
```

Facciamo ora un breve cenno alla possibilità di accedere a parti di una lista. Nel caso intendiamo usare i soli indici di posizione

```
> lista
[[1]]
[1] TRUE FALSE TRUE FALSE

[[2]]
[1] "Luca"      "Alessandra"

[[3]]
[1] 3

> lista[1]
[[1]]
[1] TRUE FALSE TRUE FALSE

> lista[[1]]
[1] TRUE FALSE TRUE FALSE
> lista[[2]][2]
[1] "Alessandra"
> lista[[c(2,2)]]
[1] "Alessandra"
> lista[[c(1,3)]]
[1] TRUE
> lista[c(1,3)]
[[1]]
[1] TRUE FALSE TRUE FALSE
```

```
[[2]]
[1] 3

> mode(lista[c(1,3)])
[1] "list"
> mode(lista[[c(1,3)]])
[1] "logical"
```

Si possono anche assegnare nomi ed usarli per estrarre le 3 componenti della lista nel seguente modo

```
> lista["nomi"]
$nomi
[1] "Luca"      "Alessandra"
> lista[["nomi"]]
[1] "Luca"      "Alessandra"
> lista$nomi
[1] "Luca"      "Alessandra"
```

Merita un discorso più approfondito l'accesso a parti di un `data.frame`. In effetti questa è la tipica struttura della matrice di dati che rappresenta il punto di partenza delle analisi statistiche di un fenomeno di interesse. La matrice dei dati prende spesso il nome di matrice *unità×variabili* o *data-set*. Le diverse unità statistiche sulle quali sono stati rilevati i valori (modalità) delle variabili (caratteri) sono identificate dalle etichette registrate nella parte dell'oggetto `data.frame` accessibile e ridefinibile attraverso la funzione `rownames(...)`. analogamente per le colonne le cui etichette identificative accessibili e ridefinibili attraverso la funzione `colnames(...)`. Per aiutare le elaborazioni e analisi statistiche che riguardano un `data-set` R prevede la possibilità di accedere alle colonne come se fossero oggetti presenti workspace evitando così di usare le parentesi quadre ed espressioni troppo lunghe. Per attivare questa possibilità è necessario mandare in esecuzione la seguente istruzione `attach(dati.apat)`

```
> ls()
[1] "dati.apat"
> names(dati.apat)
[1] "mese.anno" "temp"      "Data"      "n.dati"
> Data[1:4]
Errore in Data[1:4] : questo oggetto non indicizzabile
> attach(dati.apat)
> Data[1:4]
[1] "12/01/1906" "27/02/1906" "10/03/1906" "21/04/1906"
> temp[1:10]
[1] 13.6 15.3 17.8 24.6 29.1 32.2 32.9 34.4 31.7 25.1
```

Il funzionamento della funzione `attach(...)` è un po' complicato e per il momento ci limitiamo solo a dire che per tornare ad avere la situazione preesistente dell'accesso ai soli oggetti che compaiono nell'output della lista `ls()` si deve mandare in esecuzione l'istruzione inversa.

```
> detach(dati.apat)
> Data[1:4]
Errore: oggetto "Data" non trovato
```

Questa istruzione può rendersi indispensabile ad esempio qualora dovessimo avere a che fare successivamente definiti con nomi uguali alle colonne del `data.frame` o con altri `data.frame` usati come argomento nel comando `attach(...)` contenenti colonne con nomi uguali.

1.14 Lettura di dati esterni

Generalmente memorizzati nei formati piu' disparati in veri e propri file. Daremo qualche cenno anche su come poter usare dati memorizzati in una memoria temporanea come i dati tagliati o copiati da altri documenti.

In questa sezione daremo solo brevi cenni alle funzioni più importanti inclusi nei comandi del pacchetto base di R. Rimandiamo alle sezioni successive l'uso di pacchetti per dati in formato proprietario come quello di Excel (`.xls`) SAS (`.sav`) SPSS (`.sav`)

Per i files nei formati nativi di R vedremo come usare

- `load(...)`
- `dget(...)`
- `source(...)`

Per i files in formato ASCII vedremo

- `read.table(...)`
- `read.csv(...)`
- `read.fwf(...)`
- `scan(...)`

1.15 Esportazione di dati, risultati ed oggetti

Faremo alcuni esempi di utilizzo dei seguenti comandi per salvare oggetti nel formato nativo di R

- `save(...)`
- `\dump(...)`

oppure in formato ASCII

- `\write(...)`
- `\write.table(...)`

il primo per salvare oggetti di tipo vettoriale o matriciale il secondo per oggetti di classe `data.frame`

1.16 Come riorganizzare i dati: ordinamenti

Vediamo ora come usare gli strumenti di ordinamento di R per riorganizzare in modo conveniente i dati a nostra disposizione. Ovviamente dobbiamo premettere che il comportamento di R nell'ordinamento degli elementi di un oggetto dipenderà anche dalla natura degli elementi stessi.

Per illustrare gli aspetti basilari degli ordinamenti prenderemo ad esempio gli oggetti con i quali avevamo costruito il `data.frame` denominato `dati`.

```
> nome
[1] "Alessandra" "Luca"      "Anna"      "Paolo"
> mode(nome)
[1] "character"
> sort(nome)
[1] "Alessandra" "Anna"      "Luca"      "Paolo"
> sort(nome,decreasing=T)
[1] "Paolo"      "Luca"      "Anna"      "Alessandra"
> eta
[1] 25 40 5 2
> mode(eta)
[1] "numeric"
> sort(eta)
[1] 2 5 25 40
> sort(eta,decreasing=T)
[1] 40 25 5 2
> vaccino
[1] TRUE TRUE FALSE FALSE
> mode(vaccino)
[1] "logical"
> sort(vaccino)
```

La funzione `sort(...)` applicata ad un vettore restituisce gli elementi dello stesso ordinati in ordine (numerico, alfabetico, ...) crescente a meno che non venga usato il secondo argomento `decreasing` posto uguale a `TRUE` nel qual caso gli elementi vengono ordinati in ordine decrescente. Attenzione a non confondere le funzioni `sort(...)` e `order(...)`. Quest'ultima restituisce non già gli elementi riordinati, ma l'elenco delle posizioni corrispondenti a ciascun elemento

```
> nome
[1] "Alessandra" "Luca"      "Anna"      "Paolo"
> order(nome)
[1] 1 3 2 4
```

ovvero nell'ordinamento in senso crescente Alessandra occupa il primo posto Luca il terzo, Anna il secondo e Paolo il quarto. La funzione `order(...)` può contenere un secondo (un terzo, ...) argomento che viene utilizzato per risolvere eventuali parità di posizioni nell'ordinamento che, laddove mancassero gli ulteriori argomenti verrebbero risolte solo in base alla posizione originaria nel vettore. Chiariamo con il seguente esempio

```
> sesso
```

```
[1] "F" "M" "F" "M"
> order(sesto)
[1] 1 3 2 4
> order(sesto,eta)
[1] 3 1 4 2
> dati[order(sesto,eta),]
      nome eta sesso vaccino
3      Anna  5     F   FALSE
1 Alessandra 25     F    TRUE
4      Paolo  2     M   FALSE
2       Luca 40     M    TRUE
```

da cui si evince anche l'utilità della funzione `order(...)` per riorganizzare `data.frame` in modo opportuno secondo chiavi di ordinamento primarie, secondarie etc.. Una funzione collegata agli ordinamenti e che si comporta in modo simile ad `order` è `rank(...)` ma non intendiamo approfondire il suo utilizzo in questa sede.

Un ultimo cenno alla funzione `rev(...)` che restituisce il vettore originario con l'inversione dell'ordine degli indici per cui il primo elemento del vettore restituito corrisponde all'ultimo elemento del vettore usato nell'argomento della funzione, il secondo elemento sarà il penultimo di quello usato nell'argomento e così via.

```
> nome
[1] "Alessandra" "Luca"      "Anna"      "Paolo"
> rev(nome)
[1] "Paolo"      "Anna"      "Luca"      "Alessandra"
```

1.17 Il factor: che c'entra in tutto questo?

Definizione, utilizzo e *ricodifica* di oggetti vettoriali di classe `factor` non ordinata e ordinata.

Spesso le variabili rilevate sulle unità di un collettivo sono dati categoriali ovvero misurati su scala nominale. Ne sono un esempio il sesso, il colore di capelli, il titolo di studio di un individuo. Tipicamente le modalità con le quali si manifesta il dato rilevato su un'unità è una stringa e potrebbe essere dunque memorizzato e manipolato come un elemento di tipo `character`. Tuttavia, anche per ragioni di economia di spazio di memoria, è utile usare una ri-codifica delle varie modalità come numeri interi da 1 al numero di modalità distinte presenti nelle unità rilevate. Per tale scopo è prevista una classe di oggetti di R denominata `factor` che può essere definita a partire dalla variabile o carattere ovvero dal vettore delle modalità rilevate nelle unità statistiche attraverso la funzione `factor(...)`.

```
> str(telefonate)
'data.frame': 142 obs. of 7 variables:
 $ Tipologia      : chr  "Fonia Nazionale " "Fonia Nazionale " "Fonia Na
 $ Descrizione    : chr  "Chiamata " "Chiamata " "Chiamata " "Chiamata "
 $ Numero.chiamato: chr  "3388160****" "3388160****" "3333119****" "3333119
 $ Data.e.ora     : chr  "19/12/2006 - 10:46:10" "19/12/2006 - 14:33:02"
 $ Durata.Volume  : chr  "0:0:19" "0:0:31" "0:0:13" "0:0:9" ...
 $ Costo..Euro.   : num  0.05 0.08 0.03 0.02 0.34 0.24 0.27 0.06 0.01 0.
```


il giudizio di gradimento di un prodotto, in tal caso è prevista la possibilità di specificare un opportuno ordinamento dei livelli del fattore come secondo argomento del fattore, eventualmente usando anche un terzo argomento per dichiarare la rilevanza dell'ordinamento nella specificazione della classe

```
> giudizio=c("buono","sufficiente","buono","mediocre","ottimo","sufficiente")
> giudizio.fact=factor(giudizio,levels=c("mediocre","sufficiente","buono","ottimo"))
> class(giudizio.fact)
[1] "factor"
> giudizio.fact
[1] buono      sufficiente buono      mediocre    ottimo      sufficiente
Levels: mediocre sufficiente buono ottimo
> levels(giudizio.fact)
[1] "mediocre" "sufficiente" "buono"      "ottimo"
> c(giudizio.fact)
[1] 3 2 3 1 4 2
```

1.18 Approfondimenti su funzioni e programmazione

Una volta capito come inviare in esecuzione un programma dalla console attraverso il comando `source('nomefile.txt')` rimane da capire come inviare in esecuzione lo stesso script di comandi dalla shell (Command Prompt) di Windows

```
C:\Programmi\R\R-2.4.0\bin\R CMD BATCH nomefile.txt
```

Inoltre è possibile costruire funzioni aggiuntive dalle più semplici come quella del calcolo dell'area del cerchio a funzioni complicate che operano su dati esterni.

Cercheremo di costruirne insieme qualcuna anche con l'ausilio dei suggerimenti provenienti dai vostri problemi sul campo, magari per produrre tutta una serie di analisi e grafici a partire da dati di nostro interesse.

Ci limitiamo ad accennare con alcuni esempi alla sintassi solo di alcune delle strutture di programmazione più comuni come

- costrutti iterativi: i cicli `for(...)`
- costrutti condizionati: `if(...)`

1.18.1 R, di tutto e di più!

R base e i pacchetti aggiuntivi

Spiegheremo con pochissime schermate come reperire funzionalità aggiuntive di R attraverso i pacchetti messi a disposizione della comunità Open Source. In teoria nei manuali on-line di R è presente anche un manuale che dà tutte le indicazioni dettagliate su come costruire pacchetti aggiuntivi. Ne vedremo in funzione solo alcuni che serviranno per le elaborazioni nelle successive parti del corso. In particolare segnaliamo per il prosieguo del corso i seguenti pacchetti

- datasets
- gdata
- Rcmdr

Una lista completa e aggiornata dei pacchetti disponibili è presente sul sito

<http://rm.mirror.garr.it/mirrors/CRAN/src/contrib/PACKAGES.html>

Capitolo 2

Grafica in R

Oltre ad essere un linguaggio di programmazione di alto livello estremamente flessibile, R fornisce anche un potente motore grafico che permette di realizzare figure ed immagini di elevata qualità. In questo capitolo prenderemo in considerazione le routine grafiche di base rimandando ai capitoli successivi la trattazione di quelle a carattere più prettamente statistico.

2.1 Alcuni utili comandi: le funzioni **d**, **p**, **q** ed **r**.

Prima di entrare nel vivo della discussione, ci soffermeremo su alcuni comandi R, largamente utilizzati nel proseguo, che ci permettono di “maneggiare” famiglie note di distribuzioni di probabilità sia continue, come Normale, *t* di Student e Uniforme; che discrete, come Binomiale e Geometrica. La funzione **d**, ad esempio, restituisce i valori della densità (famiglie continue) o della distribuzione di probabilità (famiglie discrete), mentre il comando **p** quelli della funzione di ripartizione. Il comando **q**, invece, fornisce i quantili della distribuzione, ed infine il comando **r** genera un campione (pseudo) casuale.

L'utilizzo di questo set di comandi è alquanto intuitivo: in R ciascuna famiglia di distribuzioni ha associato un nome/acronimo ed un certo numero di parametri. Il nome completo delle funzioni che andremo ad utilizzare lo si ottiene combinando **d**, **p**, **q** ed **r** con l'opportuno acronimo (ovviamente nome e numero dei parametri cambiano con la famiglia di distribuzioni che stiamo considerando). Ad esempio, la distribuzione uniforme sull'intervallo $[a, b]$ in R ha come acronimo la stringa **unif** ed ha associati due parametri chiamati **min** e **max**. Di conseguenza:

```
# La densit di una Uniforme in [0,3] valutata in x = 1 pari a 1/3
> dunif(x=1, min = 0, max = 3)
[1] 0.3333
```

```
# L'area alla sinistra di q = 2 (funzione di ripartizione) sotto la
# densit di una Uniforme in [0,3] pari a 0.6667
> punif(q=2, min = 0, max = 3)
[1] 0.6667
```

```
# La mediana (o quantile di livello 1/2) di una Uniforme in [0,3] 1.5
> qunif(p=1/2, min = 0, max = 3)
[1] 1.5

# Campione estratto da una Uniforme in [0,3]
# N.B.: Il valore ottenuto cambierà ogni volta che manderemo in
#       esecuzione il comando...perché?
> runif(n=1, min = 0, max = 3)
[1] 2.563
```

Notare che queste funzioni accettano anche vettori in input.

2.1.1 Le principali funzioni

sample: Estrae con o senza ripetizione, un campione da un insieme prefissato.

d: Funzione di densità/distribuzione di probabilità.

p: Funzione di ripartizione.

q: Quantili.

r: Generatore di numeri (pseudo) casuali.

2.1.2 Esempio

```
#-----
# Simulazione di variabili casuali discrete
#-----

help(sample)    # default: SENZA ripetizione

# Ad esempio, il *NUMERO* di teste per due lanci
# di una moneta può essere simulato così:
# k = {Numero di teste in 2 lanci} può assumere
# solo tre valori: 0, 1 o 2. Assumiamo la moneta
# bilanciata => Pr(k=0)=Pr(k=2)=1/4 e Pr(k=1)=1/2.

k = 0:2
p = c(1,2,1)/4
sample(k, size = 1, prob = p)

sample(k, size = 1, prob = p)

# il valore di default per "prob=" prevede tutti i
# risultati ugualmente probabili => nell'esempio
# precedente: Pr(k=0)=Pr(k=1)=Pr(k=2)=1/3.

# Possiamo creare un'urna contenente, ad esempio,
```

```

# quattro palline bianche numerate da 1 a 4 e tre nere
# numerate da 1 a 3:

urna = c("b1","b2","b3","b4","n1","n2","n3")
urna

# ed estrarre due palline senza e poi con reinserimento

sample(urna, size = 2)

sample(urna, 2, replace = T)

# ...e se "size" > del numero degli elementi nell'urna?

length(urna)

sample(urna, size = 10)

sample(urna, 10, replace = T)

#...ovviamente...

#-----
# Distribuzioni (discrete e continue) celebri!
#-----

#-----
# Per le principali variabili aleatorie,
# abbiamo a disposizione:
# 1) funzione di densit 
#    [iniziale - "d"]
# 2) funzione di ripartizione
#    [iniziale - "p"]
# 3) quantili
#    [iniziale - "q"]
# 4) generatore di numeri pseudo-casuali
#    [iniziale - "r"]
#-----
# Ad esempio, per la BINOMIALE abbiamo:
#    "dbinom", "pbinom", "qbinom", "rbinom"
#-----

# Ma andiamo per gradi

#-----
# Distribuzione uniforme in [a,b]
# Ha due parametri denominati (in R): "min="
# (ossia "a") e "max=" (ossia "b")
#-----

```

```

# Consideriamo un'uniforme in [0,3]:

dunif(x = 1, min = 0, max = 3)    # densita' in x=1 e' 1/3

punif(q = 2, min = 0, max = 3)    # f. di ripartizione in q=2
                                   # L'area alla sx di 2 e' 2/3

qunif(p = 1/2, min = 0, max = 3)  # quantile di livello p=1/2
                                   # La mediana = 1.5

runif(n = 1, min = 0, max = 3)    # numero aleatorio da U[0,3]

# A queste funzioni possiamo passare anche vettori
# Ad esempio, possiamo calcolare un certo numero di
# quantili in un sol colpo come segue:

ps          = seq(0,1,by = .2)      # vettore
names(ps) = as.character(seq(0,100, by = 20)) # qualche nome

qunif(ps, min = 0, max =1)

# Viceversa il seguente comando fornisce 5 campioni da 5
# diverse distribuzioni uniformi:

runif(5, min = 0, max = 1:5)      # riutilizziamo il "min="

#-----
# Distribuzione di Bernoulli
# di parametro "p" -> Bernoulli(p)
#-----
# * Solo due valori 0 e 1 (ad esempio)
# * Media = p
# * Var   = p(1-p)
# Possiamo usare "sample" per generare
# campioni (indep) da questa distribuzione

n = 10
p = 1/4
sample(0:1,size = n, replace = TRUE, prob = c(1-p,p))

#-----
# Variabile Aleatoria Binomiale di parametri
# "p" ed "n" -> Binomiale(p,n)
#-----
# * Assume valori tra 0 ed "n"
# * Conta i numeri di successi (=di 1) in "n"
#   prove Bernoulliane *indipendenti*
# * Media = n*p
# * Var   = n*p(1-p)

```

```

# Il coefficiente binomiale che compare nella
# definizione di questa distribuzione conta
# il numero di modi in cui "k" oggetti possono
# essere scelti da "n" *distinti* oggetti.
# Per calcolarlo in R possiamo usare:

help(choose)

# Esempio: supponiamo di aver lanciato una moneta
# bilanciata 10 volte. Sia X = {numero di testa}
# Allora  $X \sim \text{Binomiale}(10, 1/2)$ .
# La probabilita' che X = 5 possiamo calcolarla a
# mano:

choose(10,5)*(1/2)^5*(1-1/2)^(10-5)

# Piu' semplicemente

dbinom(5, size = 10, prob = 1/2)

# La probabilita' che ci siano NON PIU' di 6 testa:
# (probabilita' totali):  $\Pr(X \leq 6)$  possiamo
# calcolarla in due modi:

# utilizzando "dbinom"
sum(dbinom(0:6, size = 10, prob = 1/2))

# oppure "pbinom"

pbinom(6, size = 10, p = 1/2)

# Se stiamo cercando la probabilita' di ALMENO 7
# testa abbiamo la bellezza di 3 modi:

sum(dbinom(7:10, size = 10, prob = 1/2))

1 - pbinom(6, size = 10, p = 1/2)

pbinom(6, size = 10, p = 1/2, lower.tail = FALSE)    # k = 6, non 7!

# Finora abbiamo visto una distribuzione continua (l'uniforme su)
# un intervallo [a,b]), e due discrete. Ora vediamo la piu' famosa
# tra quelle continue!

#-----
# Variabile Aleatoria Normale di parametri
# mu e sigma
#-----

dnorm(0)          # default = normale standard

```

```

dnorm(0,mean = 0, sd = 1)

dnorm(0,mean = 1, sd = 1)
dnorm(0,1,1)

#-----
# Funzione di ripartizione
#-----

pnorm(0)
pnorm(0,2,2)

#-----
# Quantili della normale
#-----

qnorm(0.975)

qnorm(0.975,2,2)

# Quanta "area" (sotto la campana) sta a non piu' di
# una deviazione standard dalla media? Vediamo:

pnorm(1) - pnorm(-1)

# Circa il 68%! Per due e tre (deviazioni standard) i valori
# sono 95% e 99.7% ! Infatti:

# circa due deviazioni standard...
1 - 2*pnorm(-2)

# ...piu' precisamente...
1 - 2*pnorm(-1.96)

#...ancora piu' precisamente:
qnorm(.025)
1 - 2*pnorm(qnorm(.025))

# Per tre deviazioni standard, invece, abbiamo (circa):

diff(pnorm(c(-3,3))) # usiamo "diff" per sottrarre

#-----
# Generatore di campioni casuali
#-----

rnorm(10)
rnorm(10,7,5)

# Poc'anzi abbiamo visto che il 95% delle volte una v.a. Normale(0,1)

```

```
# cade entro due deviazioni standard dalla media. Testiamo il tutto
# mediante una piccola simulazione che utilizzi il comando "rnorm"

mu    = 100
sigma = 10
res   = rnorm(1000, mean = mu, sd = sigma) # 1000 campioni da N(100,10)

k = 1 # 1 dev. stand.
sum((res > mu - k*sigma) & (res < mu + k*sigma))

k = 2 # 2 dev. stand.
sum((res > mu - k*sigma) & (res < mu + k*sigma))

k = 3 # 3 dev. stand.
sum((res > mu - k*sigma) & (res < mu + k*sigma))

#-----
# Acronimi di altre famiglie di distribuzioni:
# beta, cauchy, chisq, gamma, f, pois, exp, t
#-----
```

2.2 Comandi grafici di base

In questa sezione prenderemo in considerazione i comandi grafici di base disponibili in R. In particolare ci soffermeremo sul comando `plot(...)` il quale ci permette di ottenere grafici in maniera estremamente intuitiva per una grande varietà di *oggetti* (nel senso del Capitolo precedente) definibili in R. Ciò fatto, impareremo ad utilizzare il comando `par(...)` per recuperare prima e modificare poi le impostazioni più importanti della nostra figura (vedi Tabella 2.1 alla fine di questo capitolo); ed infine utilizzeremo `lines(...)`, `points(...)` ed `arrows(...)` per aggiungere delle *annotazioni* rilevanti che migliorino la leggibilità della figura.

Per comprendere al meglio le possibilità offerte dal motore grafico di R, è opportuno soffermarsi brevemente sulla struttura della finestra grafica in cui ci troveremo ad operare. Come si vede in Figura 2.1 e 2.2, essa è suddivisa in tre aree principali, e una tipica funzione grafica di alto-livello andrà a disegnare dati, simboli e linee all'interno dell'area del grafico, relegando assi ed etichette nell'area della figura o nei margini esterni. Lo strumento di base per controllare dimensione e posizione delle differenti aree è la funzione `par(...)`.

Ciascun grafico ha associati uno o più sistemi di coordinate rispetto a cui vengono tracciate linee e punti al suo interno. Il sistema di coordinate più semplice da utilizzare è quello detto “user coordinates” dato che corrisponde all'intervallo di valori rappresentati sugli assi del grafico stesso (vedi Figura 2.3)

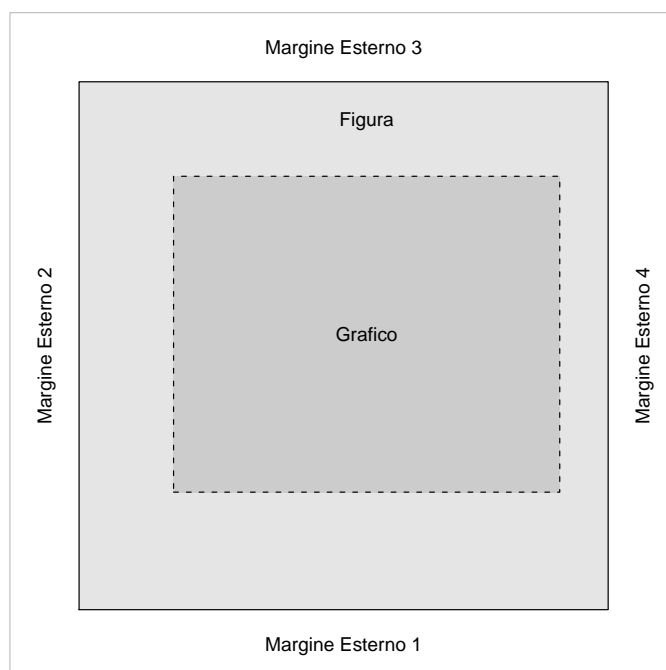


Figura 2.1: Margini esterni, area figura e area grafico, quando abbiamo a che fare con un singolo grafico (adattato da (1)).

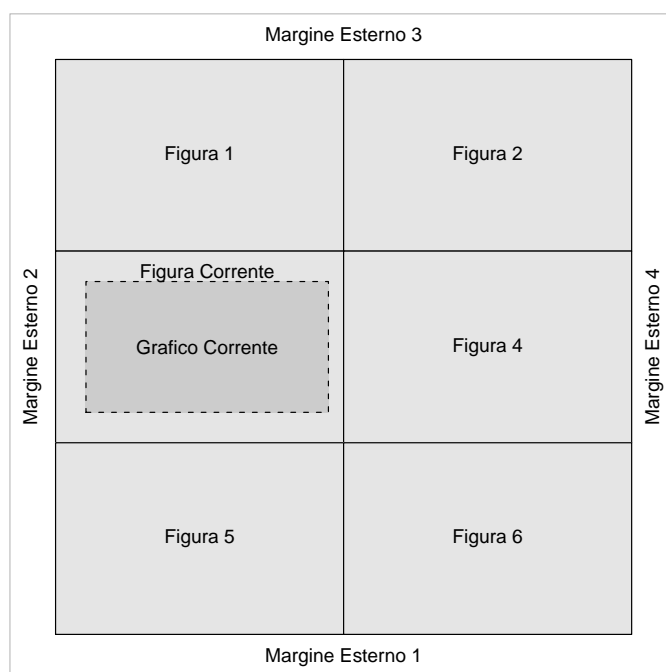


Figura 2.2: Margini esterni, area figura *corrente* e area grafico *corrente*, quando ci sono piú grafici nella stessa pagina (adattato da (1)).

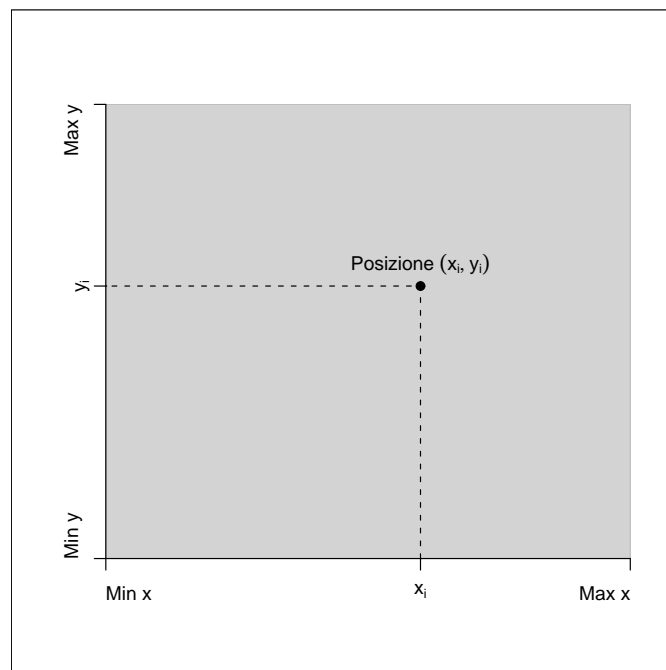


Figura 2.3: Sistema di coordinate all'interno dell'area grafica (adattato da (1)).

2.2.1 Le principali funzioni

`plot`: Funzione generica per disegnare grafici in R (vedi Figura 2.6).

`windows`: Apre una nuova finestra grafica.

`curve`: Disegna una curva corrispondente ad una data funzione su di un intervallo `[from, to]`.

`lines`: Disegna/aggiunge linee ad un grafico (vedi Figura 2.7).

`points`: Disegna/aggiunge dei punti ad un grafico (vedi Figura 2.8).

`abline`: Aggiunge una o più rette al grafico corrente (vedi Capitolo 5).

`arrows`: Disegna una freccia tra coppie di punti.

`rug`: Aggiunge un *rug-plot* al grafico corrente.

`par`: Imposta/recupera i parametri grafici della figura corrente (vedi Table 2.1).

`colors`: Restituisce i nomi dei colori built-in.

`legend`: Aggiunge una legenda alla figura corrente.

`box`: Aggiunge un riquadro alla figura corrente.

`axis`: Aggiunge/modifica gli assi alla figura corrente.

text: Inserisce una stringa di testo alle coordinate x-y fornite.

mtext: Inserisce una stringa di testo in uno dei quattro margini della figura corrente o uno dei margini esterni della finestra grafica attiva.

2.2.2 Esempio

```
#-----
# Specificare una distribuzione (discreta)
# e disegnarla (spike plot)
#-----

k = 0:4
p = c(1,2,3,2,1)/9

# L'argomento type="h" disegna le linee verticali
# Vedi Figura 2.4 per le altre opzioni disponibili
plot(k, p,
      type = "h",
      xlab = "k",
      ylab = "Probabilita'",
      ylim = c(0,max(p))
)

# Aggiungiamo i pallini!
# Vedi Figura 2.6 per le altre opzioni disponibili
points(k,p,
       pch = 16,
       cex = 2
)

# Costruiamone un'altra uniforme (equidistribuita)
# su 1, 2 e 3: Pr(k=1)=Pr(k=2)=Pr(k=3)=1/3
# dopo aver aperto una seconda finestra grafica

windows()

k = 1:3
p = c(1,1,1)/3

plot(k,p,
     type = "h",
     xlab = "k",
     ylab = "Probabilita'",
     ylim = c(0,max(p))
)

# aggiungiamo i pallini!
points(k,p,
      pch = 16,
```

```

    cex = 2
  )

#-----
# Distribuzioni (discrete e continue) celebri!
#-----

#-----
# Distribuzione uniforme in [a,b]
# Ha due parametri denominati (in R): "min="
# (ossia "a") e "max=" (ossia "b")
#-----

#
# Qualche plot comparativo
#

# INIZIO #

# Simuliamo 100 valori da una uniforme in (0,10)
res = runif(100,min = 0,max = 10)

# Costruiamo il plot principale (Istogramma, vedi Capitolo 3)

hist(res,
      prob = TRUE,
      main = "",
      col = gray(.9)
    )

# Aggiungiamo un riquadro
box()

# Ora il grafico della vera distribuzione
# "add=" sovrapponi o no al plot corrente

help(curve)

curve(dunif(x, min = 0, max = 10),
      lwd = 2,
      add = TRUE
    )

# mostriamo il campione con un "rug" plot
rug(res)

# FINE #

#-----
# Variabile Aleatoria Binomiale di parametri

```

```

# "p" ed "n" -> Binomiale(p,n)
#-----

# Spike plot della distribuzione prodotta da "dbinom"

altezze = dbinom(0:10, size = 10, prob = 1/2)

plot(0:10,altezze,
     type = "h",
     main = "Spike plot di X",
     xlab = "k",
     ylab = "d.p."
    )

points(0:10,altezze,
      pch = 16,
      cex = 2
    )

#-----
# Variabile Aleatoria Normale di parametri
# mu e sigma
#-----

# Disegniamo la densita' con delle aree evidenziate, ossia:
# come aggiungere retinature alle figure. Utilizzeremo
# il comando "curve".

# INIZIO #

# La nostra campanella
curve(dnorm(x),-3,3,
     axes = FALSE,
     ylab = "",
     xlab = "",
     ylim = c(0,.5),
     main = "I retini"
    )

# Ridenominiamo le ascisse
axis(1,c(-3,-1,0,1,3),c("", "-z",0, "z",""))

# Evidenziamo la coda di SINISTRA
vals = seq(-3,-1, length = 100)
x     = c(-3, vals, -1, -3)
y     = c(0, dnorm(vals),0, 0)
polygon(x,y, density = 20, angle = 45)

# Evidenziamo la coda di DESTRA
vals = seq(1,3,length = 100)

```

```
x      = c(1, vals, 3, 1)
y      = c(min(y), dnorm(vals), min(y), min(y))
polygon(x,y,density = 20,angle = 45)

# Qualche annotazione qua e la'
abline(h = min(y))
text(0,0.45, expression(Phi(-z)== 1-Phi(z)))
lines(c(0,0),c(0,dnorm(0)))

# FINE #

#-----
# ESPONENZIALE
#-----

# Vediamo ad esempio la distr. esponenziale
# di parametro 1/5 (molto utilizzata in
# analisi di sopravv.)

# INIZIO #

# Generiamo 50 campioni da una exp(1/5)
res = rexp(50, rate = 1/5)

# Ora...SE CI RIUSCIAMO...mettiamo un boxplot nel 35%
# "basso" della finestra grafica...prima di farlo allargato al
#
par(fig = c(0,1,0,.35))

boxplot(res,
        horizontal = TRUE,
        bty = "n",
        xlab = "campione da una esponenziale"
        )

# mettiamo il plot principale (istogramma)
# nel 75% "alto" della finestra grafica
par(fig = c(0,1,.25,1), new = TRUE)

# calcoliamo l'estremo superiore dell'asse delle y
# per poi utilizzarlo nel definire il parametro
# "ylim="
tmp.hist = hist(res, plot = FALSE)
tmp.edens = density(res)
tmp.dens = dexp(0, rate = 1/5)
y.max     = max(tmp.hist$density, tmp.edens$y, tmp.dens)

# Ok, ora siamo pronti a plottare:

# istogramma del campione
```

```

hist(res,
      ylim = c(0,y.max),
      prob = TRUE,
      main = "",
      col = gray(.9)
    )

# kernel del campione
lines(density(res),
      lty = 2
    )

# grafico della vera distribuzione
# "add=" sovrapponi al plot
curve(dexp(x,rate = 1/5),
      lwd = 2,
      add = TRUE
    )

# mostriamo il campione con un "rug" plot
rug(res)

# FINE #

#-----
# CHI QUADRATO
#-----

# Diamo un'occhiata la funzione di densita' di un Chi quadrato

help(dchisq)

# The chi-squared distribution with 'df'= n degrees
# of freedom has density
#
#  $f_n(x) = 1/(2^{n/2} \Gamma(n/2)) x^{(n/2-1)} e^{(-x/2)}$ 
#
# for  $x > 0$ . The mean and variance are  $n$  and  $2n$ .

# INIZIO #

x = seq(0,50,.1)
y = dchisq(x, df = 1)
plot(x,y,
     type = "l",
     ylim = c(0,0.6),
     lwd = 2
    )

# Aggiungiamo allo stesso grafico le densita' che si

```

```
# ottengono all'aumentare dei gradi di liberta'
# Dato che  $E[X]=2*n$ , all'aumentare dei gradi di
# liberta', ci spostiamo sempre piu' verso destra

# Vogliamo colorare le varie curve in modo automatico
# Per avere informazioni su come impostare i colori
# guardare verso la fine dell'help del comando "par"

help(par)

help(colors)

# Per avere una lista di tutti i colori "con nome"
cl = colors()
cl

# Utilizziamo sfumature di rosso
my.df = c(2,3,10,15,20,30)
my.col = c("red","red2","red3","red4","orange","yellow")

for(i in 1:length(my.df)){
  lines(x,dchisq(x,df = my.df[i]),
        lty = 2,
        lwd = 2,
        col = my.col[i]
      )
}

# Aggiungiamo una legenda
legend(30,.5,
      c("df=1","df=2","df=3","df=10","df=15","df=20","df=30"),
      col = c("black", my.col),
      lty = c(1,rep(2,length(my.col))),
      lwd = 2,
      bty = "n",
      cex = .8
    )

# FINE #

# Somiglia alla densita' di una v.a. normale!!
# Vediamo un po'...

# INIZIO #

par(mfrow = c(1,3)) # dividiamo la finestra grafica in due

# Iniziamo con 10 gradi di liberta'...
ni = 10
```

```
x = seq(0,50,.1)

plot(x, dchisq(x,df = ni),
     type = "l",
     lwd = 2,
     ylab = "Densita'"
     )

lines(x,dnorm(x, mean = ni, sd = sqrt(2*ni)),
      lty = 2,
      lwd = 2
      )

lab1 = paste("Chi2( df = ",ni,")" )
lab2 = paste("Normale( mean = ",ni," , sd = sqrt( 2 * ", ni, ") )" )

legend(15,.08,
      c(lab1,lab2),
      lty = c(1,2),
      lwd = 2,
      bty = "n",
      cex = .8
      )

# ...vediamo cos'accade a 30 gradi di liberta'...
ni = 30

x = seq(0,70,.1)

plot(x,dchisq(x,df = ni),
     type = "l",
     lwd = 2,
     ylab = "Densita'"
     )

lines(x,dnorm(x, mean = ni, sd = sqrt(2*ni)),
      lty = 2,
      lwd = 2
      )

# ...chiudiamo con 100 gradi di liberta'...
ni = 100

x = seq(20,200,.1)

plot(x,dchisq(x,df = ni),
     type = "l",
     lwd = 2,
     ylab = "Densita'"
     )
```



```

lines(x,dnorm(x, mean = ni, sd = sqrt(2*ni)),
      lty = 2,
      lwd = 2
    )

# FINE #

#-----
# GAMMA
#-----

help(dgamma)

# The Gamma distribution with parameters 'shape' = a
# and 'scale' = s has density
#
#       $f(x) = 1/(s^a \Gamma(a)) x^{a-1} e^{-(x/s)}$ 
#
# for  $x > 0$ ,  $a > 0$  and  $s > 0$ . The mean and variance
# are  $E(X) = a*s$  and  $Var(X) = a*s^2$ .

# La densita' della Gamma(alpha,beta) in x e' dunque:
#
#      dgamma(x, shape=alpha, scale=beta)

x      = seq(0,10,.1)
alpha = 1
beta  = 1
plot(x,dgamma(x, shape = alpha, scale = beta),
     type = "l",
     ylim = c(0,0.6),
     ylab = "Densita'"
    )

# cambiando i parametri...

lines(x,dgamma(x,shape = 2,scale = 1),lty = 2)
lines(x,dgamma(x,shape = 2,scale = 3),lty = 3)

#-----
# T di Student
#-----

help(dt)

# Per avere la densita' della t in x:
#
#      dt(x, df)

# INIZIO #

```

```

x = seq(-5,5,.1)

plot(x, dt(x,df = 10),
     type = "l",
     ylim = c(0,0.5),
     ylab = "Densita'"
     )

# aggiungiamo allo stesso grafico le densita'
# che si ottengono all'aumentare dei gradi di
# liberta'
for (i in 2:4){
  lines(x,dt(x,df = i),
        lty = 2)
}

# FINE #

#
# Confronto con N(0,1)
#

# INIZIO #

# Iniziamo disegnando la Normale
# senza assi
curve(dnorm(x),-4,4,
      lwd = 2,
      ylab = " ",
      main = "Confronto tra Normale e t di Student",
      axes = FALSE
      )

# scegliamo i gradi di liberta' delle t di Student
miei.df = c(3,5,25)

# scegliamo il colore con cui disegnare ciascuna t
miei.col = c("yellow","orange","red")

# Ora aggiungiamo le tre t di Student
for(i in 1:length(miei.df)){
  curve(dt(x,df = miei.df[i]),-4,4,
        add = TRUE,
        col = miei.col[i],
        lwd = 2
        )
}

# Aggiungiamo una retta di riferimento...
lines(c(0,0),c(0,dnorm(0)),type="l",lty=2)

```

```
#... una legenda...
legend(-4,.35,
      c("Normale(0,1)","df=3","df=5","df=25"),
      col = c("black",miei.col),
      lwd = 2,
      bty = "n",
      cex = .9
    )

#...un asse orizzontale.
axis(1,c(-4,-3,-2,-1,0,1,3,2,4))
abline(h = 0)
```

2.3 Grafici 3d

Dopo aver visto le routine grafiche di base, passiamo brevemente in rassegna alcuni comandi utili nella visualizzazione di dati e funzioni tridimensionali.

2.3.1 Le principali funzioni

persp Disegna il *perspective-plot* di una superficie.

image Rappresenta graficamente una matrice di dati.

contour Crea un *contour-plot*, o aggiunge delle linee di livello ad un grafico esistente.

outer Calcola il prodotto *esterno* tra due array X e Y.

2.3.2 Esempio

```
# Partiamo con l'esempio fornito da R per il comando di base
```

```
help(persp)
```

```
example(persp)
```

```
# Guardiamo il primo esempio in dettaglio...
```

```
# Costruiamo le nostre x...
```

```
x = seq(-10, 10, length = 30)
```

```
# Costruiamo le nostre y...
```

```
y = x
```

```
# Costruiamo una piccola funzioncina 2D...
```

```
f <- function(x, y) {
```

```

    r <- sqrt(x^2 + y^2)
    10 * sin(r)/r
  }

# Costruiamo il griglione...

# help(outer)

z <- outer(x, y, f)

# ...dettaglio tecnico per rimuovere eventuali NA
z[is.na(z)] <- 1

# Disegniamo! (...facciamo le cose semplici...)
persp(x, y, z)

# Un po' piu' bellino...
persp(x, y, z,
      theta = 30,
      phi = 30,
      expand = 0.5,
      col = "lightblue"
    )

# Ancora piu' bellino...
persp(x, y, z,
      theta = 30, phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120,
      shade = 0.75,
      ticktype = "detailed",
      xlab = "X", ylab = "Y", zlab = "Sinc( r )"
    )

# Cambiamo esempio...prendiamo una Normale...
# e facciamo un grafico rispetto a mean e sd

# Valori di mu e sigma che vogliamo disegnare
mu.seq = seq(-2, 2, length = 50)
sd.seq = seq(0.5, 2, length = 50)

# valutiamo "dnorm(x=0, mu.seq, sd.seq)" su questa griglia
# di valori. Il comando "outer" c'è (di nuovo) amico...
# ...MA PRIMA...bisogna ridefinire l'ordine degli argomenti
# di "dnorm"...

mio.dnorm = function(mu = 0, sigma = 1, x = 0){
  out = dnorm(x, mean = mu, sd = sigma)
}

val.ver = outer(mu.seq, sd.seq, FUN = "mio.dnorm")

```

```
# E si grafica
persp(mu.seq, sd.seq, val.ver,
      theta = 30, phi = 30,      # angolo di visuale
      expand = 0.5,
      col = "orange",
      ltheta = 120,
      shade = 0.75,
      ticktype = "detailed",
      main = "Verosimiglianza Normale(mu,sigma) in x = 0",
      xlab = "mu",
      ylab = "sigma",
      zlab = "F. di Ver."
    )
```

2.4 Layouts

Finora abbiamo visto un solo modo per farlo, e cioè utilizzando l'opzione `mfrow` del comando `par(...)`. In questo modo, però, non abbiamo alcun controllo sulla dimensione delle varie parti in cui la finestra grafica risulta scomposta: sono tutte uguali e non possiamo farci niente. Un comando molto flessibile che ci permette il massimo controllo sulla "lottizzazione" della nostra finestra grafica è `layout(...)`. `layout(...)` come (unico) input obbligatorio, richiede una matrice. Il numero di righe e colonne di questa matrice andranno a determinare il numero di righe e colonne in cui sarà divisa la finestra grafica, mentre il contenuto di ciascuna entrata della matrice dovrà essere un numero intero che determina quali righe e colonne ciascuna figura che produrremo andrà *ordinatamente* ad occupare. Le Figure 2.4 e 2.5 mostrano vari esempi di utilizzo del comando `layout(...)`

2.4.1 Le principali funzioni

`layout`: Divide la finestra grafica in un certo numero di righe e colonne.

`layout.show`: Rappresenta la scomposizione della finestra grafica ottenuta utilizzando `layout(...)`.

2.4.2 Esempio

```
# Finora abbiamo visto un solo modo per farlo, e cioè'
# utilizzando
# par(mfrow=c(2,2))      # 2 righe x 2 colonne
# In questo modo, pero' non abbiamo controllo sulla
# dimensione delle varie parti...sono tutte uguali.
# Un comando molto flessibile, invece, e' "layout"

help(layout)
```

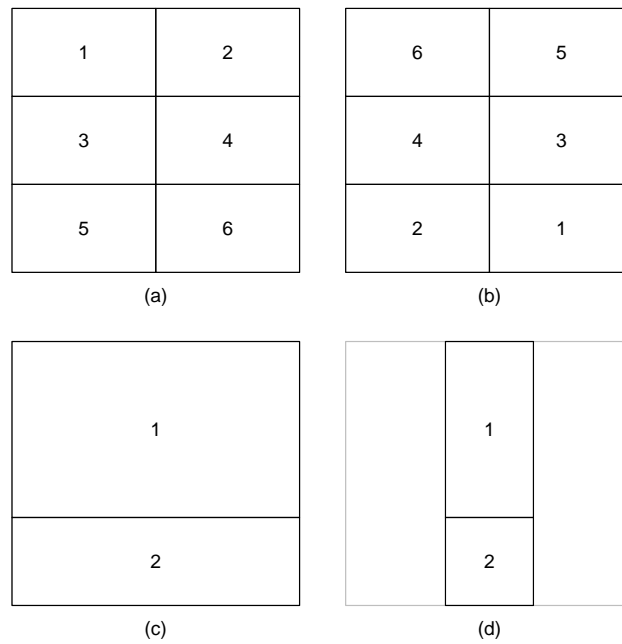


Figura 2.4: Alcuni *layout* di base. (a) Layout equivalente al comando `par(mfrow=c(3, 2))`. (b) Come in (a) ma le figure vengono introdotte in ordine inverso. (c) Layout con altezze di riga differenti. (d) Come in (c), ma con `respect = TRUE` (adattato da (1)).

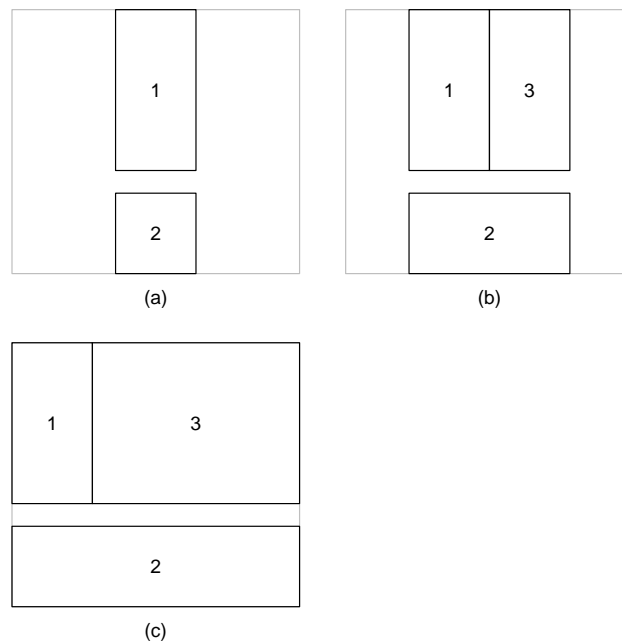


Figura 2.5: Alcuni *layout* piú complessi. (a) Layout con altezza di riga specificato in centimetri. (b) Layout con una figura che occupa piú di una colonna (c) Come in (b), ma con solo la colonna 1 e la riga 3 legate dall'argomento `respect` (adattato da (1)).

```
# Layout come (unico) input obbligatorio, vuole una matrice
# Il numero di righe e colonne di questa matrice determina
# il numero di righe e colonne in cui sara' divisa la fi-
# nestra grafica.

# Il contenuto di ciascuna entrata della matrice dovra'
# essere un numero INTERO che determina QUALI righe&colonne
# ciascuna figura occupera' (cioe' una figura puo' occupare
# piu' di una riga o colonna)

# Costruiamo la matrice
matrice.lay = rbind(c(1,2), c(3,4), c(5,6))
matrice.lay
# Impostiamo il layout
layout(matrice.lay)
# Una volta impostato il layout possiamo visualizzarlo con:
layout.show(6)
# ...quindi il primo grafico che faremo andra' ad occupare
# il box indicato con (1), il secondo quello indicato con (2)
#...etc etc...

# Come si vede, di default, tutte le righe e colonne hanno la
# stessa dimensione. Cosa dobbiamo fare per cambiarle e fare
# box piu' o meno grandi? Ma utilizzare opportunamente le op-
# zioni "heights" e "widths", naturalmente!

# Esempio: supponiamo di voler disegnare 3 grafici cosi' disposti
# 1) Il primo ed il terzo affiancati sulla prima riga con il pri-
#     di questi piu' stretto dell'altro
# 2) Il secondo, invece, occupa tutta la seconda riga
# 3) vogliamo anche un po' di spazio tra le due righe

# Costruiamo la matrice
matrice.lay = rbind(c(1,3),      # prima riga ci mettiamo il primo ed il terzo
                   c(0,0),      # seconda riga vuota: lo spazio che volevamo!
                   c(2,2)       # ultima riga tutta dedicata al secondo plot
                   )

# Ok, ora impostiamo le altezze di ciascuna delle 3 righe
# La prima e' alta 2 volte la terza e la seconda e' meta'
# della terza (...altezze relative, non assolute!)

altezze.righe = c(2,0.5,1)

# Quindi le ampiezze delle 2 colonne: la prima e' la meta'
# della seconda.

ampiezze.col = c(1,2)
```

```

# Impostiamo il layout
layout(matrice.lay, heights = altezze.righe, widths = ampiezze.col)

# Una volta impostato il layout possiamo visualizzarlo con:
layout.show(3)

# A questo punto non ci resta che fare 3 grafici!

# Ad esempio:

# Grafico (1) - In alto a destra - Veros relativa
persp(mu.seq, sd.seq, fun.ver.rel,
      theta = 30, phi = 30,
      expand = 0.5, col = "orange", ltheta = 120, shade = 0.75,
      main = "Verosimiglianza Relativa",
      xlab = "mu", ylab = "sigma", zlab = ""
    )
# Grafico (2) - In basso - boxplot dei dati
boxplot(dati.norm, border = "grey", col = "light grey", horizontal = TRUE)
# Grafico (3) - In alto a sinistra - Density plot
hist(dati.norm, prob = TRUE, border = "grey", col = "light grey", xlab = "", ylab = "", main = "")
lines(density(dati.norm), col = "royalblue", lwd = 2)
box()

##### FINE - Comando layout() #####

# Fatta conoscenza con il comando layout(), vediamo altri modi
# di disegnare superfici 3D
layout(rbind(c(1,0,2),c(0,5,0),c(3,0,4)), heights = c(1,1.3,1), widths = c(1,1.3,1))
layout.show(5)

# Grafico (1) -- help(image)
image(mu.seq, sd.seq, fun.ver.rel,
      xlab = "mu", ylab = "sigma",
      main = "Verosimiglianza Relativa \n Distribuzione Normale(mu,sigma)",
      col = topo.colors(200)
    )
# Grafico (1) -- help(image)
image(mu.seq, sd.seq, fun.ver.rel,
      xlab = "mu", ylab = "sigma",
      main = "Verosimiglianza Relativa \n Distribuzione Normale(mu,sigma)",
      col = heat.colors(200)
    )
# Grafico (3) -- help(contour)
contour(mu.seq, sd.seq, fun.ver.rel,
      xlab = "mu", ylab = "sigma",
      main = "Verosimiglianza Relativa \n Distribuzione Normale(mu,sigma)",
    )
# Grafico (4) -- image & contour combinati
image(mu.seq, sd.seq, fun.ver.rel,

```



```

      xlab = "mu", ylab = "sigma",
      main = "Verosimiglianza Relativa \n Distribuzione Normale(mu,sigma)",
      col = topo.colors(200)
    )
contour(mu.seq, sd.seq, fun.ver.rel,
      add = TRUE,
      lwd = 2
    )
# Grafico (5) -- il solito!
persp(mu.seq, sd.seq, fun.ver.rel,
      main = "", xlab = "", ylab = "", zlab = "",
      theta = 30, phi = 30, expand = 0.5,
      col = "orange",
      shade = 0.75
    )

```

2.5 Grafici interattivi

In R è anche possibile “interagire” con un grafico appena creato andando ad esempio ad identificare i punti dato contenuti nel sistema di assi attivo.

2.5.1 Le principali funzioni

identify Registra la posizione del puntatore quando viene premuto il tasto sinistro del mouse. Quindi individua le coordinate del punto-dato più vicino. Se il punto risulta *sufficientemente* prossimo al puntatore, l'indice o l'etichetta del punto-dato viene aggiunta al grafico.

locator Registra la posizione del puntatore quando viene premuto il tasto sinistro del mouse.

2.5.2 Esempio

Partiamo considerando un data set disponibile in MASS

```

library(MASS)  # Pacchetto aggiuntivo che contiene i dati che ci interessano
data(Cars93)   # Carichiamo i dati
help(Cars93)   # Diamo un'occhiata al significato delle variabili
attach(Cars93) # Attacciamo i dati

# Piccolo sommario
summary(Cars93)

# Piccolo sommario
summary(Cars93)

```

```

# Focalizziamoci su alcune variabili
dati.car = Cars93[,c(4,5,6,7,8)]

# ...con nomi
windows()
plot(Weight,MPG.highway,type ="n")
text(Weight,MPG.highway,labels = Manufacturer, lwd = 1, cex = 0.6)

# ...interattivo...
windows()
plot(Weight,MPG.highway)
identify(Weight,MPG.highway, labels = Manufacturer, cex = 0.7)

# ...con simboli diversi
windows()
plot(Weight,MPG.highway,pch=unclass(Type))
legend(3500,45,pch=unclass(Type),legend=levels(Type))

# ...con colori diversi
windows()
plot(Weight,MPG.highway,col=unclass(Type))
legend(3500,45,text.col=1:nlevels(Type),legend=levels(Type))

# legenda pi classica
windows()
plot(Weight,MPG.highway,col=unclass(Type))
legend(3500,45,col=1:nlevels(Type),legend=levels(Type),pch=1)

#-----
# Scatterplot avanzati aggiungendo una
# terza variabile quantitativa: bubbleplot
#-----

windows()
plot(Weight,MPG.highway,pch = 21, bg = "yellow", cex = 1.15)
symbols(Weight,MPG.highway,circles = abs(Price), inches = 0.2, add = T)
identify(Weight,MPG.highway, labels = Manufacturer, cex = 0.7)

```

Parametro	Descrizione
<code>adj</code>	Giustificazione del testo
<code>ann</code>	Disegna etichette e titoli nel grafico?
<code>bg</code>	Colore di "background"
<code>bty</code>	Tipologia del box disegnato da <code>box(...)</code>
<code>cex</code>	Dimensione del testo (proporzione)
<code>cex.axis</code>	Dimensione delle etichette di riferimento sugli assi
<code>cex.lab</code>	Dimensione delle etichette associate agli assi
<code>cex.main</code>	Dimensione del titolo
<code>cex.sub</code>	Dimensione dei sottotitoli
<code>col</code>	Colore delle linee e dei simboli
<code>col.axis</code>	Colore delle etichette di riferimento sugli assi
<code>col.lab</code>	Colore delle etichette associate agli assi
<code>col.main</code>	Colore del titolo
<code>col.sub</code>	Colore dei sottotitoli
<code>fg</code>	Colore di "foreground"
<code>font</code>	Stile del font (bold , <i>italic</i>) del testo
<code>font.axis</code>	Stile del font delle etichette di riferimento sugli assi
<code>font.lab</code>	Stile del font delle etichette associate agli assi
<code>font.main</code>	Stile del font del titolo
<code>font.sub</code>	Stile del font dei sottotitoli
<code>gamma</code>	Correzione <i>gamma</i> per i colori
<code>lab</code>	Numero di punti di riferimento sugli assi
<code>las</code>	Rotazione del testo a margine
<code>lty</code>	Tipo di linea (piena, tratteggiata, ecc.)
<code>lwd</code>	Spessore della linea
<code>mgp</code>	Posizionamento di punti di riferimento e loro etichette sugli assi
<code>pch</code>	Tipo di simbolo utilizzato
<code>srt</code>	Rotazione del testo all'interno del grafico
<code>tck</code>	Lunghezza dei segmenti associati ai punti di riferimento sugli assi (relativo alla dimensione del grafico)
<code>tcl</code>	Lunghezza dei segmenti associati ai punti di riferimento sugli assi (relativo alla dimensione del testo)
<code>tmag</code>	Dimensione del titolo (relativo alle altre etichette)
<code>type</code>	Tipo di grafico (punti, linee, entrambi)
<code>xaxp</code>	Numero di punti di riferimento sull'asse- <i>x</i>
<code>xaxs</code>	Rapporto di riscalatura dell'asse- <i>x</i>
<code>xaxt</code>	Stile dell'asse- <i>x</i> (standard, nessuno)
<code>xpd</code>	Regione di "clipping"
<code>yaxp</code>	Numero di punti di riferimento sull'asse- <i>y</i>
<code>yaxs</code>	Rapporto di riscalatura dell'asse- <i>y</i>
<code>yaxt</code>	Stile dell'asse- <i>y</i> (standard, nessuno)

Tabella 2.1: Parametri grafici di alto-livello. Questo tipo di parametri possono essere impostati e richiamati sia mediante la funzione `par(...)`, sia utilizzando all'interno di comandi grafici come `plot(...)` e `lines(...)`.

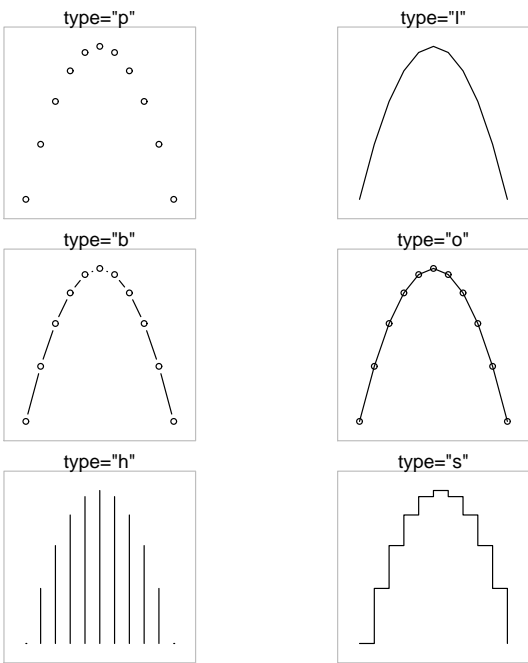


Figura 2.6: Tipi di plot. Il grafico in ciascun riquadro è stato prodotto con una espressione del tipo `plot(x, y, type = qualcosa)`, utilizzando per il parametro `type` l’impostazione mostrata sopra ciascuno di essi (adattato da (1)).

Intero	Esempio	Stringa
<i>Predefiniti</i>		
0		"blank"
1	—————	"solid"
2	- - - - -	"dashed"
3	"dotted"
4	- . - . -	"dotdash"
5	- - - - -	"longdash"
6	- . - . -	"twodash"
<i>Custom</i>		
	"13"
	— — — —	"F8"
	- . - . -	"431313"
	- - - . -	"22848222"

Figura 2.7: Tipi di linea. Il tipo di linea può essere specificato in tre modi distinti: 1. utilizzando un numero intero predefinito 2. mediante una stringa di testo predefinita 3. introducendo una stringa di valori esadecimali nel caso volessimo “customizzare” il tipo di linea (adattato da (1)).

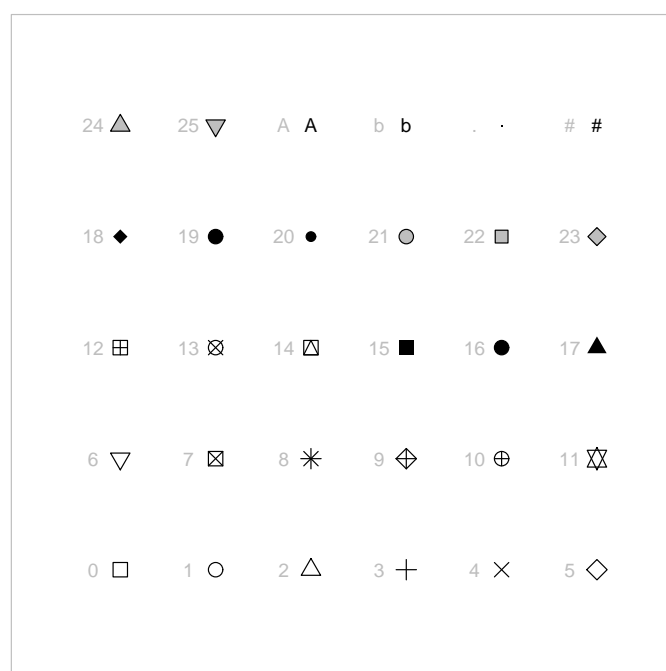


Figura 2.8: Simboli disponibili in R. Un particolare simbolo può essere selezionato specificando un numero intero tra 0 e 25 o, alternativamente, un singolo carattere, per il parametro grafico `pch` (adattato da (1)).

Bibliografia

- [1] Murrell, P. (2005), *R Graphics*. Chapman and HallCRC.

Capitolo 3

Descrizione dei dati

La semplice descrizione dei dati può essere effettuata in R tramite funzioni nel pacchetto base, quindi senza particolari problemi di programmazione e implementazione. Introduciamo queste funzioni seguendo la scaletta di un tipico corso di statistica descrittiva, ovvero prima trattando l'analisi statistica univariata e poi la statistica bivariata.

3.1 Una variabile qualitativa

Ricordiamo che i dati statistici vengono tipicamente rappresentati nella forma di una matrice di tipo **unità × variabili**: ad esempio, consideriamo la distribuzione del numero di arresti per vari tipi di crimine (variabili) in 50 stati degli USA (unità) relativa all'anno 1973.

	Murder	Assault	UrbanPop	Rape	Abbr	Geographic
Alabama	13.2	236	58	21.2	AL	South
Alaska	10.0	263	48	44.5	AK	Pacific
Arizona	8.1	294	80	31.0	AZ	Mountain
Arkansas	8.8	190	50	19.5	AR	South Central
California	9.0	276	91	40.6	CA	Pacific
Colorado	7.9	204	78	38.7	CO	Mountain
Connecticut	3.3	110	77	11.1	CT	Northeast
Delaware	5.9	238	72	15.8	DE	Northeast
Florida	15.4	335	80	31.9	FL	South
Georgia	17.4	211	60	25.8	GA	South
Hawaii	5.3	46	83	20.2	HI	Pacific
Idaho	2.6	120	54	14.2	ID	Mountain
Illinois	10.4	249	83	24.0	IL	Central
.....		

Le variabili Murder, Assault, UrbanPop e Rape sono di tipo quantitativo ed esprimono la percentuale di arresti per ciascun crimine nei vari stati; la variabili Abbr e

Geographic sono un variabili qualitative, in quanto sono rispettivamente un'abbreviazione del nome dello stato e un indicatore della ripartizione geografica.

Più formalmente, una **variabile qualitativa** è una variabile le cui modalità, ovvero le misurazioni effettuate, sono dei valori non numerici.

Una **variabile quantitativa** è una variabile le cui modalità, ovvero le misurazioni effettuate, sono dei valori numerici.

3.1.1 Cosa vedremo

Consideriamo ora il caso di distribuzioni di una sola variabile.

La prima operazione di sintesi su una variabile qualitativa riguarda i conteggi, ovvero la determinazione di una *tabella di frequenza* (*cross-tabulation*) che associa ad ogni modalità il numero di unità che presentano tale modalità della variabile (*frequenze assolute*).

Le *frequenze relative* si ottengono poi riportando il totale delle *frequenze assolute* all'unità, cioè dividendo ciascuna delle frequenze assolute per la numerosità del collettivo.

Le *frequenze percentuali* non sono altro che le *frequenze relative* moltiplicate per 100. Le *frequenze relative* e *percentuali* risultano particolarmente utili quando si vogliono confrontare le distribuzioni di frequenza di collettivi di numerosità diversa.

A partire dalla tabella di frequenza si possono individuare due indici: la moda e l'entropia. La *moda* è un indice di posizione definito come la modalità a cui corrisponde la frequenza più elevata. L'entropia è un indice di eterogeneità della distribuzione di un carattere qualitativo definito attraverso la seguente formula:

$$H = - \sum_{j=1}^k f_j \log(f_j) \quad (3.1)$$

dove f_i sono le frequenze relative e k il numero di modalità distinte del carattere. L'indice di entropia può essere standardizzato (ovvero riportato su una scala di valori compresi tra 0 e 1) dividendolo per il logaritmo del numero di classi.

Tra i grafici vedremo il grafico a barre (Figura 3.1) e il grafico a torta (Figura 3.2).

3.1.2 Le principali funzioni

factor: Le variabili categoriali vengono spesso intese da R come fattori, e quindi trattate in maniera speciale. I fattori sono vettori che hanno internamente codificato quali sono i livelli. Questo risulta comodo per il loro utilizzo, specialmente per i modelli inferenziali, ma rende difficile la modifica.

levels, nlevels: `levels(x)` elenca i livelli di un fattore. `nlevels(x)` riporta il numero di livelli di un fattore.

`table`: `table(x)` costruisce una tabella di frequenza di una variabile di tipo qualunque.
`table(x, exclude=c("A","B"))` elimina dalla tabella le categorie "A" e "B". I valori mancanti (NA e NaN) sono eliminati automaticamente.

`which`: `which(condizione)` riporta l'elemento del vettore che risponde a una certa condizione.

`barplot`: Grafico a barre

`pie`: Grafico a torta

3.1.3 Esempio

Consideriamo una parte dell'indagine multiscopo svolta negli USA nel 1993. Per $n = 1000$ soggetti sono state rilevate le seguenti variabili: la posizione lavorativa, lo stato civile, il segno zodiacale, il titolo di studio, la razza, la tendenza politica e il voto alle ultime elezioni.

```
# importiamo i dati

library(gdata)
indagine=read.xls("indagineUSA.xls",na.strings=" ")

names(indagine)

#[1] "id" "statolav" "sposato" "zodiaco" "titstud" "sesso" "razza"
#[2] "voto92" "tendenzapolitica"

attach(indagine)

is.factor(tendenzapolitica)
nlevels(tendenzapolitica)
levels(tendenzapolitica)

tab=table(tendenzapolitica)
tab

# moda

names(tab)[which(tab==max(tab))]
```

```
#[1] "Moderato "  
  
# entropia  
  
fi=tab/sum(tab)  
entr=-sum(fi*log(fi))  
  
# 1.524463  
  
# entropia relativa  
  
entr/log(length(tab))  
  
#[1] 0.9472022  
  
# grafico a barre  
  
barplot(tab)  
  
# torta  
  
pie(tab)  
  
# ok... ora complichiamoci la vita  
  
# ricodifica in destra centro e sinistra  
  
politica=tendenzapolitica  
levels(politica)=c("Destra","Sinistra","Centro","Destra","Sinistra")  
  
tab2=table(politica)  
  
tab2  
  
# entropia relativa  
  
fi=tab2/sum(tab2)  
  
entr=sum(-fi*log(fi))
```

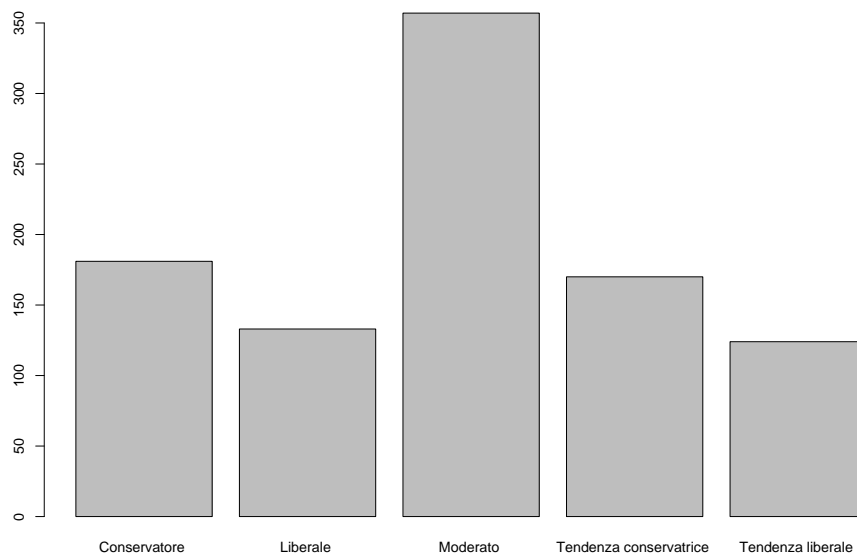


Figura 3.1: **Grafico a barre** della distribuzione per la tendenza politica dei 10000 soggetti

```
entr/nlevels(politica)
```

```
barplot(tab2, main="Distribuzione Votanti USA 1992", col=c("black","blue","red"))
```

```
pie(tab2, main="Distribuzione Votanti USA 1992")
text(0.25,0.5,fi[1])
text(0.25,-0.5,fi[2])
text(-0.25,0,fi[3])
```

3.2 Una variabile quantitativa

3.2.1 Cosa vedremo

Passiamo, ora, alle variabili quantitative. Per cominciare vedremo come dividere in classi la variabile quantitativa, di fatto riportandoci al caso di una variabile qualitativa.

Tratteremo il calcolo dei principali indici di posizione: media, mediana, medie troncate (trimmed). La media è la somma dei valori osservati divisa per la loro numerosità. La mediana è il valore che lascia al di sopra e al di sotto il 50% dei valori

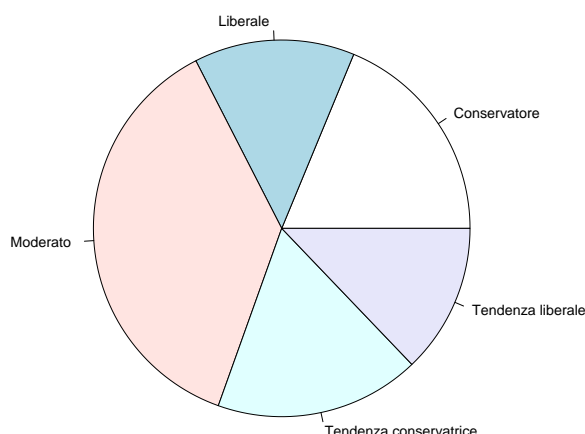


Figura 3.2: **Grafico a torta** della distribuzione per la tendenza politica dei 10000 soggetti

osservati. Le medie troncate sono le medie semplici calcolate sul sottoinsieme dei dati ottenuto scartando una percentuale pre-specificata di valori estremi. Se ad esempio tronchiamo al 5%, la media viene calcolata su tutti i valori, tranne il 2.5% piú piccolo ed il 2.5% piú grande. Vedremo inoltre i quantili, che generalizzano la mediana: il quantile α -mo lascia al di sotto $\alpha \cdot 100\%$ dei valori osservati e $(1 - \alpha) \cdot 100\%$ al di sopra.

Tra gli indici di variabilità vedremo il range (differenza tra minimo e massimo), ed il piú robusto range interquartilico (differenza tra terzo e primo quartile). Alla media va accompagnata la varianza (media degli scarti al quadrato dalla media), o piú opportunamente la sua radice, la deviazione standard (detta anche “scarto quadratico medio”). Alla mediana va accompagnato un indice di variabilità dalla mediana, come ad esempio il MAD: la mediana degli scarti assoluti dalla mediana (*Median Absolute Deviation*).

Tra i grafici, l’istogramma è molto spesso in grado di fornire informazioni quasi esaustive sulla variabile. L’istogramma è un grafico a barre sulla variabile quantitativa divisa in classi, con barre di area proporzionale alla frequenza della classe (e barre adiacenti e non riordinabili, ovviamente). La Figura 3.4 mostra un esempio di istogramma sulla distribuzione del monossido di carbonio relativa a 25 marche di sigarette.

Con tecniche piú complesse, di cui omettiamo i dettagli, è possibile inoltre avere

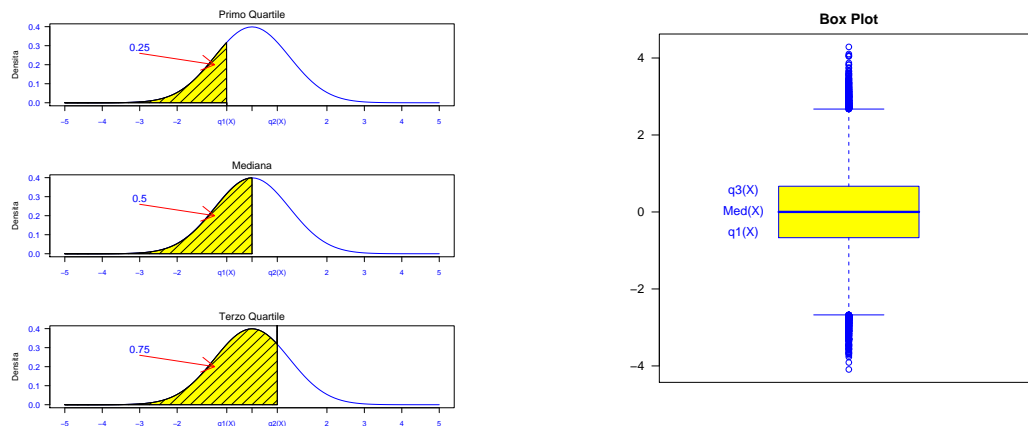


Figura 3.3: Corrispondenza tra **Boxplot** e curva di densità

una versione “continua” dell’istogramma, più leggibile. Un esempio è in Figura 3.5.

Vedremo inoltre il boxplot, una versione diversa di istogramma “visto dall’alto”. Nel boxplot sono rappresentate la mediana, il primo e terzo quartile, e viene spesso utilizzato per identificare in maniera semplice i valori anomali (indicati da alcuni punti isolati), che vengono evidenziati al di fuori da una distanza opportuna dalla mediana (in generale, per distanze superiori al terzo quartile più λ volte il range interquartilico o inferiori al primo quartile meno λ volte il range interquartilico, dove tipicamente $\lambda = 1.5$ o $\lambda = 2$). Un esempio è mostrato in Figura 3.6, dove si evidenzia una distribuzione approssimativamente simmetrica con la presenza di un valore anomalo inferiore.

Concluderemo con la distribuzione empirica, ovvero la stima della funzione di ripartizione: la probabilità, in funzione di una soglia, che la variabile osservata sia inferiore. Per ogni x , la distribuzione empirica è la frequenza relativa delle osservazioni minori o uguali a x .

3.2.2 Le principali funzioni

`cut` Divide in classi.

`apply` Permette di applicare una funzione alle righe o colonne di una matrice, visualizzando così le statistiche per tutte le variabili.

`mean` La media aritmetica. Trimmed al 5% se `mean(x, trim=0.05)`.

`median` La mediana

`quantile` I quantili. `quantile(x, probs = c(0.25, 0.6))` dà il primo quartile e il 60-mo percentile

`range` Calcola il minimo e il massimo

`diff` Calcola la differenza tra elementi successivi del vettore in argomento. Il range è quindi `diff(range(x))`.

`var, sd` Varianza e deviazione standard. Attenzione: versione non distorta.

`mad` Mediana delle deviazioni assolute dalla mediana. Per il calcolo esatto, `mad(x, constant=1)`. Il default è maggiorato di circa il 50% per ottenere un valore approssimativamente pari alla deviazione standard in caso di dati normali.

`boxplot` L'opzione `range` è l'equivalente del parametro λ citato sopra. Il default è $\lambda = 1.5$.

`match` Generalizza `which` a vettori.

`hist` Istogramma

`ecdf` Distribuzione empirica

`density` Funzione per fare una versione "continua" dell'istogramma.

3.2.3 Esempio

```
# carichiamo i dati delle sigarette

load("sigarette.RData")

names(dati)
names(dati)[1]="marca"
names(dati)[3]="monossido"
attach(dati)

# punti estremi

range(monossido)

# chi   che fa pi   male?

marca[which(monossido==max(monossido))]

# e quelle che fumo io?

dati[marca=="Marlboro",]

# istogramma
```



```
hist(monossido)

# voglio 10 barre

hist(monossido, breaks=10)

# voglio classi predefinite

hist(monossido, breaks=c(min(monossido),10,12,15,18,max(monossido)))

# versione "continua"

plot(density(monossido),main="Istogramma")

# ma concordano?

hist(monossido, probability=T)
lines(density(monossido))

### altre statistiche

mean(monossido)
sd(monossido)

median(monossido)
mad(monossido)

mean(monossido, trim=0.05)

# esempio di apply

apply(dati[,2:3],2,median)

# altri indici di variabilit 

diff(range(monossido))
diff(quantile(monossido,c(0.25,0.75))) #IQR

boxplot(monossido)
# etichettiamo i valori anomali
out=boxplot(monossido)$out
```

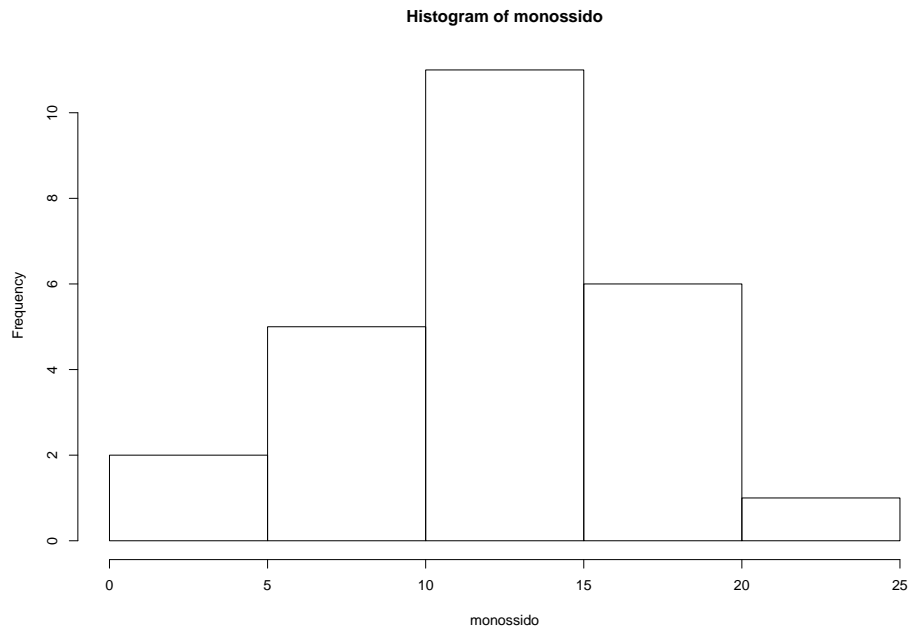


Figura 3.4: Istogramma

```

text(rep(1.1,length(out)),out, marca[match(out,monossido)])

# check normalit

hist(monossido,prob=T)
curve(dnorm(x,mean(monossido),sd(monossido)),add=T)

# dividiamo in classi

cl=cut(monossido,4)
cl1=cut(monossido,breaks=c(min(monossido),10,12,15,18,max(monossido)),
  include.lowest=T)

table(cl)
table(cl1)

# giochiamo con la distribuzione empirica

curve(pnorm, -3,3)
lines(ecdf(rnorm(10)))
lines(ecdf(rnorm(100)),pch=2)

```

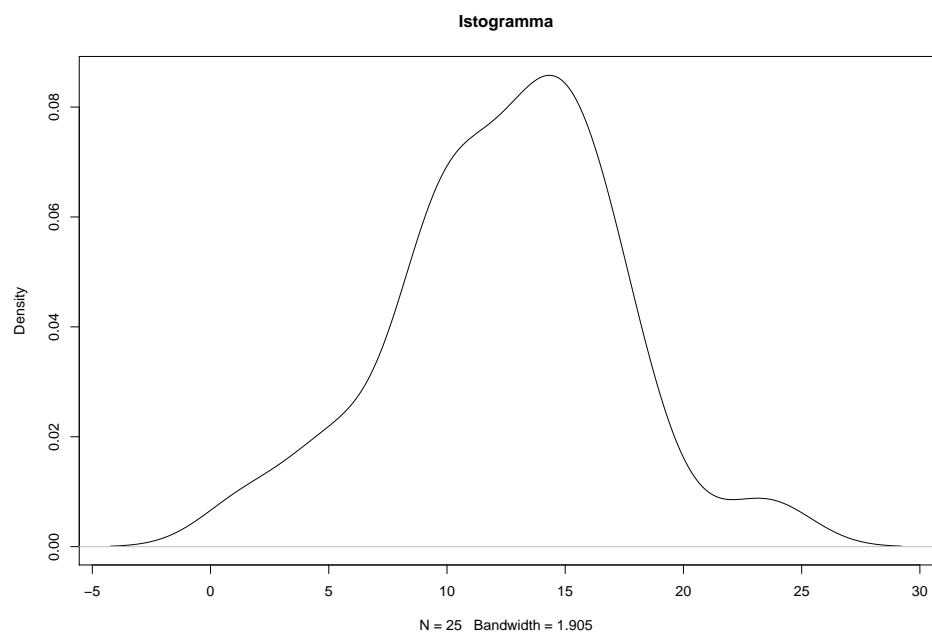


Figura 3.5: Istogramma “continuo” (Stima Kernel)

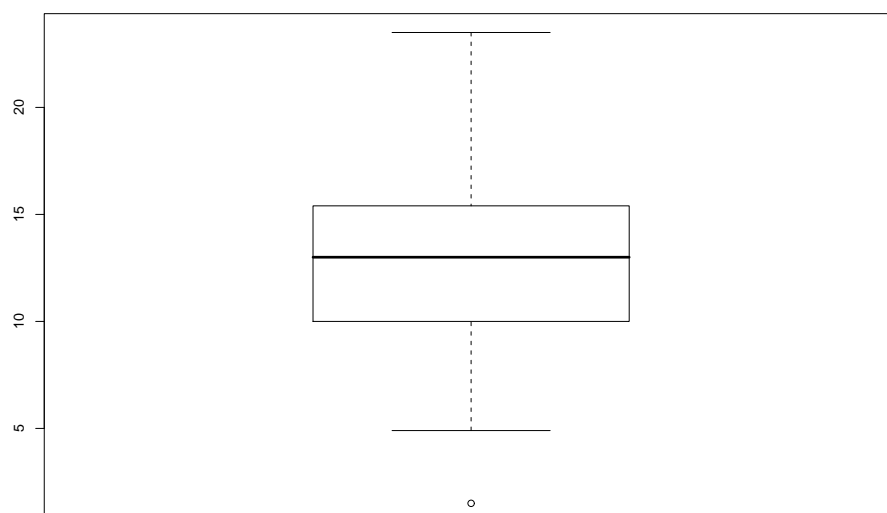


Figura 3.6: Boxplot

3.3 Relazioni tra due variabili

3.3.1 Due variabili qualitative

Passiamo ora all'analisi della relazione tra due variabili, e cominciamo dal caso di due qualitative.

3.3.2 Cosa vedremo

La tabella di frequenza si generalizza nella tabella a doppia entrata, o *tabella di contingenza* come quella riportata qui di seguito. Ciascun conteggio n_{ij} della tabella riporta il numero di soggetti per cui è osservata la i -ma modalità della prima variabile e in contemporanea la j -ma modalità della seconda.

$X \backslash Y$	y_1	y_2	\cdots	y_j	\cdots	y_k	Totale
x_1	n_{11}	n_{12}	\cdots	n_{1j}	\cdots	n_{1k}	$n_{1\cdot}$
x_2	n_{21}	n_{22}	\cdots	n_{2j}	\cdots	n_{2k}	$n_{2\cdot}$
\cdots	\cdots	\cdots	\cdots	\cdots	\cdots	\cdots	\cdots
x_i	n_{i1}	n_{i2}	\cdots	n_{ij}	\cdots	n_{ik}	$n_{i\cdot}$
\cdots	\cdots	\cdots	\cdots	\cdots	\cdots	\cdots	\cdots
x_h	n_{h1}	n_{h2}	\cdots	n_{hj}	\cdots	n_{hk}	$n_{h\cdot}$
Totale	$n_{\cdot 1}$	$n_{\cdot 2}$	\cdots	$n_{\cdot j}$	\cdots	$n_{\cdot k}$	n

Nella tabella di contingenza possiamo inserire frequenze assolute o relative. Sommando le righe possiamo osservare la tabella di frequenza per la variabile sulle colonne, e viceversa. Tali tabelle di frequenza vengono chiamate "tabelle marginali". Chiamiamo $n_{i\cdot} = \sum_j n_{ij}$ le marginali di riga e $n_{\cdot j} = \sum_i n_{ij}$ le marginali di colonna.

Dividendo i conteggi assoluti per il rispettivo totale di riga si ottiene la tabella con le distribuzioni per la variabile sulle colonne, condizionando a ciascuna modalità della variabile sulle righe, e analogamente per i totali di colonna. La relazione tra le due variabili è quasi sempre leggibile più semplicemente utilizzando una delle due condizionate.

La semplice l'analisi dei bar plot relativi alle distribuzioni condizionate può fornirci un'idea della presenza o meno di relazione tra le due variabili in esame: infatti, se le distribuzioni condizionate non variano in modo determinante al variare delle modalità del carattere condizionante (ossia le altezze del bar plot non variano tante) allora possiamo supporre che la relazione tra le due variabili non sia molto rilevante.

Tra gli indici vedremo il χ^2 ("chi quadrato"), una misura di associazione che è anche una misura di distanza tra tabelle. Nel caso più comune, si misura la distanza tra la tabella osservata e la tabella che si avrebbe nel caso di assenza di relazione (indipendenza) tra le due variabili. Questa tabella è nota, ed è il caso teorico in cui le

frequenze siano pari a $\tilde{n}_{ij} = \frac{n_{i.}n_{.j}}{n}$. Le differenze tra le osservate e le teoriche vengono elevate al quadrato per eliminare il segno, e divise per le teoriche per rendere relative (e quindi comparabili) le differenze in ciascuna cella. La distanza totale tra la tabella osservata e quella di indipendenza è pertanto:

$$\chi^2 = \sum_{ij} \frac{(n_{ij} - \tilde{n}_{ij})^2}{\tilde{n}_{ij}}.$$

Se si osserva un χ^2 elevato si è distanti dal caso di indipendenza, e quindi vi è relazione tra le variabili. La *direzione* della relazione non è individuata dall'indice, e va in generale investigata ad esempio utilizzando le condizionate di riga o di colonna.

La statistica χ^2 non è standardizzata, e pertanto risulta complesso giudicare il suo ordine di grandezza. La versione standardizzata è chiamata V di Cramer, ed è pari a $\sqrt{\chi^2/n \min(R-1, C-1)}$, dove R è il numero di modalità della variabile sulle righe e C il numero di modalità della variabile sulle colonne.

Si usa talvolta anche il ϕ^2 , pari a χ^2/n .

Per i grafici vedremo l'estensione dei grafici a barre al caso bivariato: grafici a barre sovrapposti o raggruppati. Vedremo inoltre i grafici a barre sovrapposti e riscalati al 100%, equivalenti alla tabella condizionata, che sono utilizzati per investigare la relazione tra le due variabili. Un esempio di grafico a barre sovrapposto e riscalato è in Figura 3.7: il grafico rappresenta la distribuzione dei sopravvissuti nel naufragio del Titanic per le varie classi di appartenenza. Il confronto delle barre permette di leggere la relazione tra la sopravvivenza e la classe di appartenenza dei passeggeri del Titanic. La percentuale di sopravvissuti per la prima classe, ad esempio, è di molto superiore alla percentuale di sopravvissuti per terza classe o equipaggio.

3.3.3 Le principali funzioni

`table` Calcola la tabella

`margin.table` Calcola le marginali (a partire dalla tabella)

`prop.table` Calcola le condizionate o le frequenze relative (a partire dalla tabella)

`chisq.test` utilizzare l'opzione `correct=F` per il χ^2 .

`legend` Inserisce la legenda in un grafico

3.3.4 Esempio

Lavoriamo sui dati del Titanic. Si hanno i conteggi del numero di superstiti e di dispersi divisi per sesso, classe di appartenenza (o equipaggio), e classe di età (Bambino o Adulto). I dati sono già in forma di tabella di contingenza multipla, in un array a quattro dimensioni.

```
# Consideriamo i dati del Titanic

data(Titanic)
help(Titanic)

#limitiamoci alla relazione tra classe di appartenenza e mortalit 
# per le donne e i bambini

dati=margin.table(Titanic,c(1,4))

# con le 2 marginali

tot=cbind(dati,margin.table(dati,1))
tot=rbind(tot,margin.table(tot,2))

tot

# frequenze relative

prop.table(dati)

# condizionate

prop.table(dati,1)
prop.table(dati,2)

# chi quadro & company

chi2=chisq.test(dati,correct=F)$statistic
chi2

# phi2

phi2=chi2/sum(dati)
phi2

# V Cramer

V=sqrt(phi2/min(ncol(dati)-1,nrow(dati)-1))
V
```

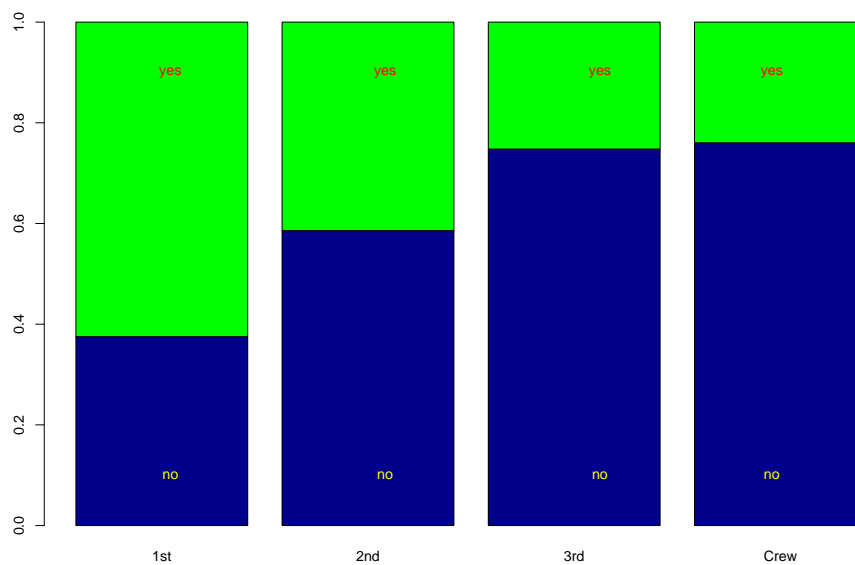


Figura 3.7: Grafico a barre sovrapposto e riscalato

```
# grafico a barre raggruppato
```

```
barplot(t(dati), beside=T, col=c("dark blue","green"))
legend("topleft",c("no","yes") ,fill=c("dark blue","green"))
```

```
# grafico a barre sovrapposto
```

```
barplot(t(dati), col=c("dark blue","green"))
legend("topleft",c("no","yes") ,fill=c("dark blue","green"))
```

```
# uno utile: sovrapposto e riscalato
```

```
barplot(prop.table(t(dati),2), col=c("dark blue","green"))
```

```
for(x in c(0.75,2,3.25,4.25)) {
  text(x,0.9,"yes",col="red")
  text(x,0.1,"no",col="yellow")}
```

```
# in alternativa, molto semplice:
```

```
plot(dati)
```

3.4 Due variabili quantitative

3.4.1 Cosa vedremo

La regressione verrà trattata nel dettaglio nel Capitolo 5 in questa dispensa. Per ora ci limiteremo alla descrizione della relazione tra due variabili quantitative in senso simmetrico.

Lo strumento principale è il diagramma a dispersione o scatterplot (Figura 3.8).

Gli indici che useremo sono la covarianza e la correlazione. La covarianza consente di verificare se tra due variabili statistiche esiste un legame lineare e, in caso affermativo, se vi sia concordanza o discordanza tra di esse; la correlazione fornisce un'informazione aggiuntiva rispetto alla covarianza in quanto oltre a dirci se esiste o meno una relazione lineare tra le variabili, permette di quantificare l'intensità di tale legame lineare.

Notiamo inoltre che in caso di variabili fortemente asimmetriche è talvolta opportuno trasformare la variabile (per esempio considerando la scala logaritmica) e/o in certi casi standardizzare .

3.4.2 Le principali funzioni

scale Standardizza ($Z = \frac{X - \text{Media}(X)}{sd(X)}$)

cov Covarianza

cor Correlazione

3.4.3 Esempio

Consideriamo ancora i dati delle sigarette.

```
load("sigarette.RData")
```

```
# scatterplot
```

```
plot(dati[,2],dati[,3],main="Sigarette",xlab="peso",ylab="monossido")
```

```
w=which(dati[,2]==min(dati[,2]))
```

```
text(dati[w,2]+0.02,dati[w,3],dati[w,1])
```

```
# Fumate le Now, se proprio dovete
```

```
# indici
```



```
cov(dati[,2:3])
cor(dati[,2:3])

# la prova del nove

cov(scale(dati[,2:3]))

# consideriamo ora un data set piu ampio

library(MASS)
attach(Cars93)

# scatterplot avanzati (aggiungendo una terza variabile categoriale)

# con simboli diversi

plot(Weight,MPG.highway,pch=unclass(Type))
legend(3500,45,pch=unclass(Type),legend=levels(Type))

# con colori diversi

plot(Weight,MPG.highway,col=unclass(Type))
legend(3500,45,text.col=1:nlevels(Type),legend=levels(Type))

# legenda pi classica
plot(Weight,MPG.highway,col=unclass(Type))
legend(3500,45,col=1:nlevels(Type),legend=levels(Type),pch=1)

# scatterplot matrix

plot(Cars93[,c(4,5,6,7,8)])
```

3.5 Una variabile qualitativa e una quantitativa

3.5.1 Cosa vedremo

Lo studio della relazione tra due variabili di natura diversa si riduce spesso al confronto dei valori della variabile quantitativa in gruppi indicizzati dalle modalità della qualitativa. Si parla in tal caso di *statistiche stratificate*. È possibile confrontare

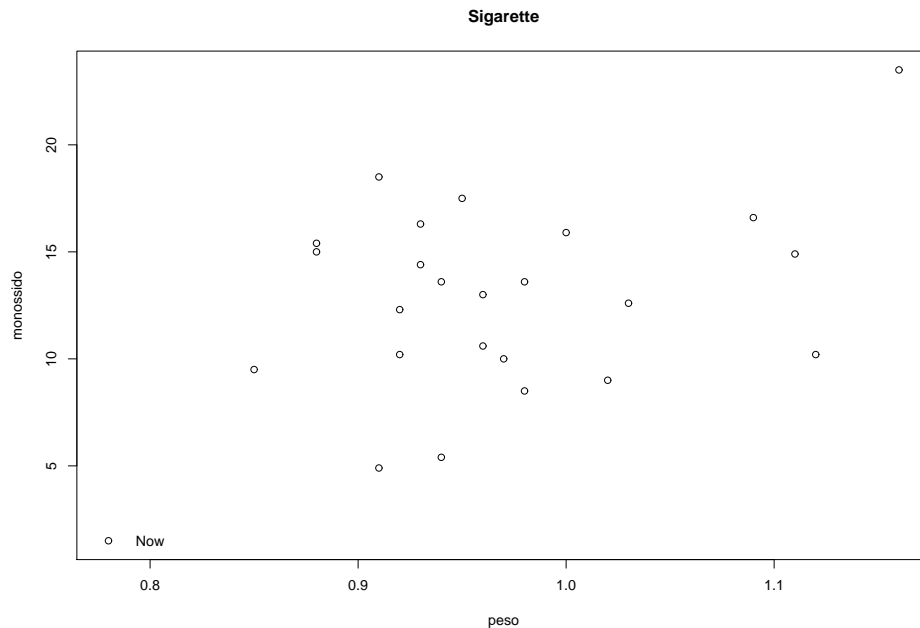


Figura 3.8: Grafico a dispersione

medie, mediane, deviazioni standard, grafici, e in generale tutta l'analisi univariata applicata in maniera distinta nei sottogruppi.

3.5.2 Le principali funzioni

Aggiungeremo alle funzioni già viste la sola `tapply`, che permette di applicare una funzione a gruppi di righe della matrice di dati.

3.5.3 Esempio

```
# riprendiamo i dati delle macchine
```

```
mea=sort(tapply(Price,Type,mean))
med=sort(tapply(Price,Type,median))
```

```
mea
med
```

```
names(mea)
names(med)
```

```
tapply(Price,Type,sd)
```

```
par(mfrow=c(3,2))
tapply(Price,Type,hist,xlim=range(Price),probability=T,xlab="Price")

# se vogliamo anche i titoli e altri orpelli, ciclo for

xli=range(Price)
par(mfrow=c(3,2))

for(i in 1:6) {
  lt=levels(Type)[i]
  hist(Price[Type==lt],xlim=xli,probability=T,xlab="Price",main=NULL)
  title(main=paste("Hist of",lt))}
```

3.6 Esercizi

Esercizio 1 Utilizzando il data set `mondo.xls`:

1. Analizzare la distribuzione dei casi di AIDS per 100.000 abitanti, con particolare attenzione ai casi estremi.
2. Studiare la variabile Regione.
3. Studiare la variabile Clima.
4. Studiare la relazione tra speranza di vita maschile e femminile.
5. Studiare la relazione tra PIL e speranza di vita. Succede quello che vi aspettavate?
6. Analizzare la relazione tra religione e regione
7. Togliersi qualche curiosità.

Capitolo 4

Inferenza Statistica

In questa sezione ci occuperemo di risolvere con R problemi di *inferenza statistica*, definita come il processo attraverso il quale si traggono conclusioni su un'intera popolazione a partire da un campione osservato, che si presume essere estratto da tale popolazione. Stime puntuali, intervalli di confidenza e test di ipotesi costituiscono i tre ingredienti principali dell'inferenza statistica. Anche per questo tipo di analisi, R dispone di una serie di semplici funzioni predefinite: sarà nostro obiettivo, quindi, cercare di capire come utilizzare tali funzioni, quale di queste scegliere per i diversi casi di studio e come interpretarne i risultati.

4.1 Stima puntuale

4.1.1 Cosa vedremo

In questa sezione ci occuperemo di come ottenere stime puntuali di un parametro incognito di una popolazione a partire da un campione osservato.

La prima e più immediata valutazione che si può dare di una caratteristica della popolazione, usando i dati campionari, è quella rappresentata dalla sua stima puntuale, definita come una funzione dei dati campionari che fornisca una buona approssimazione del parametro incognito di interesse.

Diverse metodologie sono state proposte per ottenere stime puntuali: nelle successive sezioni, ci limiteremo ad una trattazione abbastanza intuitiva del problema, illustrando le procedure di R per ottenere tali stime.

4.1.2 Esempio

Per capire cosa è una stima puntuale, consideriamo il seguente semplice esempio: supponiamo di avere rilevato l'altezza di $n = 50$ di ragazzi di età compresa tra i 18 e i 25 anni residenti nel Lazio. Questi 50 ragazzi costituiscono un campione estratto dalla popolazione dei ragazzi di età compresa tra i 18 e i 25 anni residenti nel Lazio. Come utilizzare l'informazione campionaria per ottenere informazioni

sull'intera popolazione? Ad esempio, a partire dalle altezze rilevate, cosa possiamo concludere circa l'altezza media dell'intera popolazione?

L'approccio più naturale sarebbe quello di utilizzare la media del campione \bar{x} come **stima** della media μ della popolazione. In particolare, la variabile aleatoria \bar{X} è detta *stimatore* di μ e il suo valore assunto nel campione, \bar{x} , è detto stima.

Pertanto, nell'esempio considerato, se assumiamo la media campionaria come stima della media della popolazione, in R basterà scrivere:

```
# Leggiamo i dati
altezze = read.csv("Altezze.csv",header=T)
length(altezze)

> 50

# Calcoliamo la media campionaria
stima.media = mean(altezze)

> 175.3
```

Quindi, possiamo concludere che 175.3cm è una stima dell'altezza media dei ragazzi di età compresa tra i 18 ed i 25 anni residenti nel Lazio.

Ovviamente le cose non sono sempre così semplici, nel senso che non sempre lo stimatore "migliore" per un parametro è il suo corrispettivo campionario. Infatti, per uno stesso parametro esistono più stimatori: il "migliore" è quello che meglio approssima il parametro incognito di interesse.

Come detto precedentemente, svariate metodologie sono state proposte al fine di determinare formalmente lo stimatore "migliore", le sue proprietà e la rispettiva distribuzione.

Come si vedrà in seguito, in R possiamo ottenere stime puntuali indirettamente come output delle procedure di test di ipotesi.

4.2 Stima per intervallo

4.2.1 Cosa vedremo

L'informazione derivante dallo stimatore puntuale può, però, non essere esaustiva. Come detto, la media campionaria è spesso utilizzata per stimare la media della popolazione. È probabile, però, che 2 campioni diversi producano medie diverse e che una stima puntuale non fornisca alcuna informazione sulla variabilità dello stimatore, nel senso che non sappiamo quanto \bar{x} sia vicino a μ . Inoltre, la stima puntuale non fornisce informazioni in merito alla dimensione del campione.

Pertanto, in molti casi, si preferisce associare alla stima puntuale anche un intervallo

di valori attorno alla stima puntuale che ci aspettiamo contenga, con un certo livello di fiducia, il valore del parametro incognito. Anche per la stima per intervallo, così come per la stima puntuale, non esistono funzioni di R che ne permettono un calcolo diretto: anche in questo caso, come vedremo nella prossima sezione, gli intervalli di stima sono inseriti come output delle procedure utilizzate per test di ipotesi.

Senza entrare nei dettagli, cerchiamo ora di capire cosa è un intervallo di confidenza e come interpretarlo.

4.2.2 Esempio

Il problema di stima per intervallo consiste nel fornire un range di possibili valori entro i quali si ritiene sia compreso il parametro in esame con un certo grado di confidenza. Ad esempio, se per la media μ otteniamo un intervallo $[a, b]$ a livello di confidenza $1 - \alpha = 0.95$, ciò significa che se selezioniamo 100 campioni casuali dalla popolazione ed utilizziamo questi campioni per calcolare 100 diversi intervalli di confidenza per μ , circa 95 di essi comprenderanno la media reale della popolazione e 5 no. Ovviamente non vi è certezza che questo intervallo contenga il valore vero del parametro; l'unica garanzia è che la procedura produce con probabilità $1 - \alpha$ un intervallo che contiene il valore vero del parametro. $1 - \alpha$ è detto *copertura* dell'intervallo di confidenza.

Consideriamo il seguente esempio: per una popolazione Normale di media μ incognita e varianza σ^2 nota, si dimostra che l'intervallo di confidenza per la media è:

$$\left[\bar{x} - z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \bar{x} + z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}} \right] \quad (4.1)$$

dove con $z_{1-\frac{\alpha}{2}}$ abbiamo indicato il quantile a livello $1 - \frac{\alpha}{2}$ di una distribuzione Normale standard.

Simuliamo ora $M = 1000$ campioni ciascuno composto da 50 unità da una distribuzione Normale standard ($\mu = 0$ e $\sigma = 1$) e, costruiti gli intervalli di confidenza come definiti nell'Equazione 4.1, valutiamo la copertura di questi intervalli, ossia contiamo quanti degli M intervalli includono il valore vero del parametro incognito, cioè 0.

Nella Figura 4.1 riportiamo gli intervalli di confidenza per i primi 15 campioni simulati: come possiamo vedere, tutti e 15 gli intervalli contengono il vero valore di $\mu = 0$, in corrispondenza del quale abbiamo tracciato una linea tratteggiata.

Considerando tutti e 1000 gli intervalli, la copertura di questi intervalli è pari a 0.952. Ciò significa che 952 dei 1000 intervalli generati contengono il vero valore del parametro incognito. Riportiamo di seguito il codice di R utilizzato per questo esempio:

```
M = 1000      # numero di simulazioni che effettueremo
n = 50        # dimensione del campione che genereremo ad ogni sim.
```

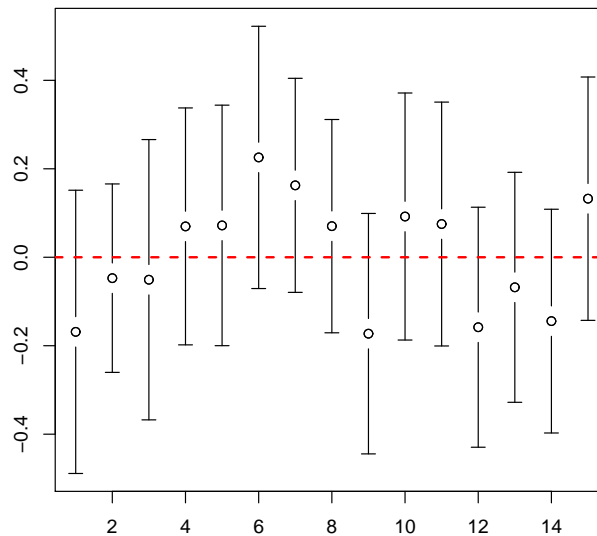


Figura 4.1: Intervalli di confidenza per i primi 15 campioni simulati

```

alpha = 0.05 # fissiamo un livello di confidenza
theta0 = 0    # ...ed un valore VERO del parametro "theta"...
sigma0 = 1    # ... sigma...

# generiamo i nostri M x n dati da una Normale(theta0, sigma0)
sim.dati = rnorm(M*n, theta0, sigma0)

#...ed organizziamoli in una matrice M x n
x.matrice = matrix(sim.dati, nrow = M)

# Calcoliamo medie e deviazioni standard dei campioni
medie = apply(x.matrice, 1, mean)
s.dev = apply(x.matrice, 1, sd)

# Costruiamo gli estremi dell'intervallo
l.inf = medie - qt(1-alpha/2, n-1)*s.dev/sqrt(n)
l.sup = medie + qt(1-alpha/2, n-1)*s.dev/sqrt(n)

# Infine valutiamo la copertura dell'intervallo, cio\`e
# quantevolte (sulle M simulazioni fatte) esso include
# il valore vero del parametro incognito 0
# Graficamente utilizziamo la library gplots

```



```
library(gplots)
centri = (l.inf + l.sup)/2 # centri degli intervalli

# Facciamo un plot dei primi 15 intervalli generati
plotCI(centri[1:15], ui = l.sup[1:15], li = l.inf[1:15], ylab='', xlab='')
abline(h = theta0, col = "red", lwd = 2, lty = 2)

# Ora cerchiamo quelli che NON contengono il vero valore del parametro
good.inf = l.inf[theta0 < l.inf] # casi in cui theta0 "sfora" in basso
good.sup = l.sup[theta0 > l.sup] # casi in cui theta0 "sfora" in alto

# Ed infine contiamo
coperturaIC = 1 - (length(good.inf) + length(good.sup))/M
coperturaIC

> 0.952
```

Come detto in precedenza, diverse metodologie sono state proposte per determinare intervalli di confidenza: come per la stima puntuale, in R possiamo ottenere intervalli di confidenza in modo diretto solo se ne conosciamo l'espressione, come quella riportata nell'Equazione 4.1. Altrimenti, come vedremo nella sezione successiva, intervalli di confidenza per i parametri di interesse possono essere ottenuti come output delle procedure utilizzate per i test di ipotesi.

4.3 Test di ipotesi

4.3.1 Cosa vedremo

La verifica di ipotesi è una delle maggiori aree di interesse dell'inferenza statistica e trova applicazione in svariati problemi reali. Ad esempio, un'industria farmaceutica potrebbe essere interessata a confrontare l'efficacia di un nuovo farmaco con uno già in commercio, oppure si potrebbe essere interessati a verificare se il consumo medio delle famiglie italiane è rimasto invariato rispetto all'anno precedente e così via. In tutti questi casi, il problema può essere ricondotto all'accettazione o rifiuto di un'ipotesi riguardante il valore di un parametro di una o più popolazioni.

Nelle sezioni successive, dopo un breve richiamo teorico, introdurremo i test statistici più utilizzati. Inizialmente, prenderemo in considerazione i cosiddetti test parametrici, ossia test in cui si suppone nota la distribuzione della popolazione. In particolare, ci concentreremo sul caso di popolazioni normali, per poi estendere il discorso a popolazioni per le quali l'assunzione di normalità potrebbe risultare non adeguata.

4.3.2 Qualche cenno teorico

Per **ipotesi statistica** si intende un'affermazione riguardante un parametro della popolazione di interesse.

Supponiamo di voler verificare se la media μ di una popolazione sia uguale ad un valore fissato μ_0 . Questa affermazione sul valore del parametro della popolazione è detta **ipotesi nulla** e scriviamo $H_0 : \mu = \mu_0$. In modo complementare all'ipotesi nulla, definiamo l'**ipotesi alternativa** che può essere di tipo bilaterale, ossia $H_A : \mu \neq \mu_0$ oppure unilaterale $H_A : \mu < \mu_0$ o $H_A : \mu > \mu_0$.

Dopo aver formulato le ipotesi, selezioniamo un campione casuale di dimensione n dalla popolazione in esame. Confrontiamo la media di questo campione, \bar{x} , con la media assunta in H_0 , μ_0 ; vogliamo sapere se la differenza tra la media del campione e la media ipotizzata è troppo grande per essere attribuita solo al caso.

Se c'è evidenza che il campione non può provenire da una popolazione con media μ_0 rifiutiamo l'ipotesi nulla, altrimenti H_0 verrà accettata. Nel prendere questo tipo di decisioni, è possibile commettere degli errori. Nell'ambito della teoria dell'inferenza statistica, si definiscono i seguenti errori:

Errore di I tipo = $\Pr(\text{rifiutare } H_0 \text{ dato che } H_0 \text{ è vera});$

Errore di II tipo = $\Pr(\text{non rifiutare } H_0 \text{ dato che } H_0 \text{ è falsa});$

Potenza del test = $\Pr(\text{rifiutare } H_0 \text{ dato che } H_0 \text{ è falsa}) = 1 - \text{Errore di II tipo}.$

L'ideale sarebbe avere errori di I e di II tipo entrambi piccoli (o equivalentemente elevata potenza): si dimostra che i due errori non possono essere minimizzati contemporaneamente. In generale, si suole fissare l'errore di I tipo per poi minimizzare quello di II tipo (o equivalentemente massimizzare la potenza).

Nell'applicare un test di ipotesi, la massima probabilità con la quale siamo disposti a rischiare di avere un **errore di I tipo** è detto *livello di significatività* ed è indicato con α . I livelli di significatività più spesso usati sono quelli dell'1% e del 5%. Un livello di significatività del 5%, ad esempio, significa che accettiamo al massimo un 5% di probabilità di rifiutare l'ipotesi nulla erroneamente, ossia accettiamo di sbagliare al più 5 volte su 100. In altre parole, si può dire che siamo confidenti di aver accettato correttamente H_0 nel 95% dei casi.

Per spiegare la logica di un test, consideriamo il seguente sistema di ipotesi per la media μ di una popolazione Normale:

$$H_0 : \mu = \mu_0 \qquad H_A : \mu \neq \mu_0 \qquad (4.2)$$

Se l'ipotesi nulla è vera, la probabilità di osservare una media campionaria vicina a μ_0 è 0.95, mentre la probabilità di osservare un valore distante è molto piccola, complessivamente pari a 0.05, livello di significatività α . Pertanto, il campo di variazione della media campionaria può essere suddiviso in due regioni complementari, una

detta **regione di rifiuto** e l'altra **regione di accettazione**. Ovviamente, maggiore sarà il valore stabilito per α , più ampia sarà la regione di rifiuto.

La media campionaria, che nell'esempio che stiamo analizzando è la statistica utilizzata per decidere se un determinato campione porta all'accettazione o al rifiuto di H_0 , viene chiamata **statistica test**. Se il valore osservato della statistica test, nel nostro caso la media campionaria \bar{x} , cade nella regione di rifiuto, allora rifiuteremo H_0 , altrimenti la accetteremo.

Un'altra quantità rilevante ai fini dell'accettazione o del rifiuto dell'ipotesi nulla è il *p-value* o *valore p*, definito come la probabilità che la statistica test Z sia pari o più estrema del suo valore osservato z : il *p-value* viene confrontato con il livello predeterminato di significatività α per decidere se l'ipotesi nulla deve essere rifiutata seguendo la seguente regola:

$$\begin{cases} p\text{-value} \leq \alpha & \text{rifiuto } H_0; \\ p\text{-value} > \alpha & \text{non rifiuto } H_0. \end{cases}$$

Nelle successive sezioni, non ci occuperemo di come determinare le statistiche test e le regioni di rifiuto, ma lasceremo che R lo faccia per noi. Illustreremo, quindi, i test per popolazioni Normali più utilizzati e come implementarli in R.

4.3.3 Test per la media di una popolazione Normale con varianza incognita

Sia X una popolazione con distribuzione Normale, in simboli $X \sim N(\mu, \sigma^2)$, dove sia la media μ che la varianza σ^2 sono incognite. Supponiamo di voler fare inferenza sulla media μ verificando il seguente sistema di ipotesi:

$$H_0 : \mu = \mu_0 \qquad H_A : \mu \neq \mu_0 \qquad (4.3)$$

In questo caso, la statistica test è

$$T = \frac{\bar{X} - \mu_0}{S / \sqrt{n}} \qquad (4.4)$$

dove $S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$ è la varianza campionaria corretta.

Si dimostra che la statistica T si distribuisce come una t di Student con $n - 1$ gradi di libertà; pertanto, il test da utilizzare è detto **test t**. In R, tale test sulla media di una popolazione Normale è implementato nel comando `t.test(...)`.

Facciamo un esempio per vedere come utilizzare questo comando e come interpretare il suo output. Supponiamo di aver misurato la pressione sistolica di un campione casuale di 20 uomini affetti da una particolare patologia. Vogliamo verificare se la pressione media assume valori diversi da 125, livello di pressione tipico per un maschio adulto sano. Definiamo quindi il seguente sistema di ipotesi:

$$H_0 : \mu = 125 \qquad H_A : \mu \neq 125 \qquad (4.5)$$

Supponendo che i dati provengano da una distribuzione Normale, utilizziamo R per applicare il **test t** nel seguente modo:

```
# Inseriamo i valori della pressione sistolica del campione osservato
x=c(98,160,136,128,130,114,123,134,128,107,123,125,129,132,154,115,126,132,136,130)

t.test(x, alternative="two.sided", mu=125, paired=F, var.equal=F, conf.level = 0.95)

# Alcune di queste opzioni sono di default, quindi otteniamo lo
# stesso risultato scrivendo
t.test(x, mu=125)
```

Proviamo ad interpretarne l'output:

```
> One Sample t-test
> t = 0.9648, df = 19, p-value = 0.3468
```

Per prima cosa, R ci ricorda che tipo di test stiamo utilizzando: in questo caso, stiamo applicando il test t per un solo campione. Riporta poi il valore della statistica test, $t=0.9648$ e i gradi di libertà $df = n - 1 = 19$ della distribuzione t di Student della statistica test.

Vediamo subito che, scegliendo un il livello di significatività $\alpha = 0.05$, il *p-value* = 0.3468 assume valori maggiori di α : applicando la regola di decisione illustrata nella precedente sezione, possiamo concludere che i dati non si discostano significativamente dall'ipotesi che la media sia 125. Quindi non possiamo rifiutare l'ipotesi nulla.

```
> alternative hypothesis: true mean is not equal to 125
```

R ci ricorda, inoltre, che il test effettuato è *bilaterale*, ossia l'ipotesi alternativa è $H_A : \mu \neq \mu_0$. Per avere un test *unilaterale* del tipo $H_A : \mu < \mu_0$ o $H_A : \mu > \mu_0$, nella linea di comando di R dovremmo modificare l'opzione *alternative* rispettivamente in *alternative = less* e *alternative = greater*.

```
> 95% confidence interval:
> 121.4919 134.5081
```

Questo è l'intervallo di confidenza al livello 95% per la media vera, ottenuto mediante un metodo detto dell' "inversione del test". In fine, ci viene anche fornito il valore della stima puntuale per la media:

```
> sample estimates: mean of x
> 128
```

In conclusione, abbiamo visto che il comando `t.test(...)`, oltre che ad effettuare il test di ipotesi, permette di calcolare intervallo di confidenza e stima puntuale per il parametro di interesse μ .

4.3.4 Test per la differenza tra due medie (campioni indipendenti)

Consideriamo ora due popolazioni X_1 e X_2 tali che $X_1 \sim N(\mu_1, \sigma_1^2)$ e $X_2 \sim N(\mu_2, \sigma_2^2)$; siamo interessati ad un confronto tra le loro medie, ossia vogliamo verificare se tra le medie delle due popolazioni sussiste una differenza.

Formalizziamo questo problema mediante il seguente sistema di ipotesi:

$$H_0 : \mu_1 = \mu_2 \qquad H_A : \mu_1 \neq \mu_2 \qquad (4.6)$$

In questo caso l'ipotesi *alternativa* è bilaterale; alternativamente, H_A può essere formalizzata in modo unilaterale, ossia $H_A : \mu_1 < \mu_2$ oppure $H_A : \mu_1 > \mu_2$.

In questo caso, indicando con n_1 e n_2 le numerosità dei due campioni, la statistica test è:

$$T = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{S_{X_1}^2(n_1 - 1) + S_{X_2}^2(n_2 - 1)}} \sqrt{\frac{n_1 n_2 (n_1 + n_2 - 2)}{n_1 + n_2}} \qquad (4.7)$$

Si dimostra che questa statistica test si distribuisce come una t di Student con $n_1 + n_2 - 2$, e pertanto il test da utilizzare è sempre un **t test** implementato dal comando R `t.test(...)`. A differenza di quanto fatto prima, però, ora occorre specificare che si ha a che fare con due campioni.

Relativamente alle varianze delle popolazioni, possiamo supporre che le varianze siano incognite ma uguali, cioè $\sigma_1^2 = \sigma_2^2$, oppure che esse siano diverse. Ciò va specificato utilizzando l'opzione `var.equal` che sarà posta uguale a `FALSE` se assumiamo che le varianze siano diverse o posta uguale a `TRUE` se supponiamo siano uguali.

Consideriamo il seguente esempio: supponiamo di voler sperimentare se una dieta a base di sola carne produca un incremento di peso maggiore rispetto a una dieta a base di soli cereali. A tal fine, 40 topi vengono randomizzati in due gruppi sottoposti per lo stesso periodo alle due diverse diete. Vogliamo verificare se le due diete producono uno stesso incremento medio di peso, ossia vogliamo verificare le seguenti ipotesi:

$$H_0 : \mu_{\text{Carne}} = \mu_{\text{Cereali}} \qquad H_A : \mu_{\text{Carne}} \neq \mu_{\text{Cereali}} \qquad (4.8)$$

Supponiamo inoltre di poter assumere che le varianze dei due gruppi siano incognite, ma uguali. In R, utilizziamo i seguenti comandi:

```
# Leggiamo i dati
peso.topi=read.table('weight-Rat-T-test.txt',header=T)

# Vediamo le dimensioni del data set
dim(peso.topi)

> 40 3 #quindi abbiamo 40 righe 3 colonne
```

```
# Vediamo a cosa corrispondono le singole colonne
names(peso.topi)

> "Gain"      "Protein" "Amount"

# Le colonne di interesse sono la prima e la seconda
# La prima colonna relativa all'incremento di peso,
# La seconda specifica la dieta seguita ("Beef" o "Cereal")
# Distinguiamo i due gruppi di dieta
gruppo.dieta.carne=peso.topi[peso.topi$Protein=='Beef'],$Gain
gruppo.dieta.cereali=peso.topi[peso.topi$Protein=='Cereal'],$Gain

# Appliciamo il test t
t.test(x=gruppo.dieta.carne,y=gruppo.dieta.cereali,alternative='two.sided',paired=F,
var.equal=TRUE, conf.level = 0.95)
```

Otteniamo il seguente risultato:

```
>          Two Sample t-test data:
> gruppo.dieta.carne and gruppo.dieta.cereali
> t = 0.9057, df=38, p-value = 0.3708
> alternative hypothesis: true difference in means is not equal to 0
> 95 percent confidence interval:
>      -5.805005 15.205005
> sample estimates: mean of x mean of y
>      89.6      84.9
```

Nella prima riga, R ci ricorda che stiamo facendo un test t per il confronto tra due campioni. Vediamo subito che non possiamo rifiutare l'ipotesi nulla poiché $p\text{-value} = 0.3708 > 0.05$. Quindi possiamo concludere che le due diete non sortiscono effetti diversi in termini di incremento medio di peso.

Anche in questo caso, possiamo optare per ipotesi unilaterali, modificando l'opzione `alternative`.

Come detto, in questo esempio abbiamo assunto che le varianze tra i due gruppi siano incognite ma uguali inserendo nel codice l'opzione `var.equal=TRUE`. Alternativamente, possiamo assumere varianze diverse modificando l'opzione in `var.equal=FALSE`. In questo caso otteniamo il seguente risultato:

```
>          Welch Two Sample t-test
> data:  gruppo.dieta.carne and gruppo.dieta.cereali
> t = 0.9057, df =36.992, p-value = 0.3709
> alternative hypothesis: true difference in means is not equal to 0
> 95 percent confidence interval:
>      -5.814412 15.214412
> sample estimates: mean of x mean of y
>      89.6      84.9
```

Quando assumiamo varianze diverse, il test va sotto il nome di “test di Welch”.

Anche in questo caso poiché $p\text{-value} = 0.3709 > 0.05$ non possiamo rifiutare l'ipotesi nulla di equivalenza delle due diete in termini di incremento medio di peso.

4.3.5 Test sulle varianze di campioni indipendenti

Come abbiamo visto, il test t può essere effettuato sia quando le varianze dei due campioni X e Y sono uguali sia quando sono diverse. Ma come decidere quando possiamo assumere varianze uguali o diverse?

In qualche caso potremmo essere interessati a verificare l'ipotesi di uguaglianza delle varianze, ossia potremmo essere interessati a verificare il seguente sistema di ipotesi:

$$H_0 : \sigma_1^2 = \sigma_2^2 \quad H_A : \sigma_1^2 \neq \sigma_2^2 \quad (4.9)$$

In questo caso la statistica test è:

$$T = \frac{S_1^2}{S_2^2} \quad (4.10)$$

dove $S_1^2 = \frac{1}{n_1-1} \sum_{i=1}^{n_1} (X_i - \bar{X})^2$ e $S_2^2 = \frac{1}{n_2-1} \sum_{i=1}^{n_2} (Y_i - \bar{Y})^2$ sono, rispettivamente, gli stimatori corretti di σ_1^2 e σ_2^2 . Sotto l'ipotesi nulla H_0 , la statistica test si distribuisce come una F di Fisher con $n_1 - 1$ e $n_2 - 1$ gradi di libertà, dove n_1 e n_2 sono le numerosità dei campioni X e Y .

In R, questo test può essere effettuato utilizzando il comando `var.test(...)`.

Nell'esempio precedente, per verificare:

$$H_0 : \sigma_{\text{Carne}}^2 = \sigma_{\text{Cereali}}^2 \quad H_A : \sigma_{\text{Carne}}^2 \neq \sigma_{\text{Cereali}}^2 \quad (4.11)$$

utilizziamo il seguente codice:

```
var.test(gruppo.dieta.carne, gruppo.dieta.cereali, conf.level = 0.95)
```

```
>      F test to compare two variances
> data:  gruppo.dieta.carne and gruppo.dieta.cereali
> F = 1.3954, numdf = 19, denom df = 19, p-value = 0.4746
> alternative hypothesis:true ratio of variances is not equal to 1
> 95 percent confidence interval:
>      0.5523098 3.5253684
> sample estimates: ratio of variances
>      1.395384
```

Visto il valore del p -value, non possiamo rifiutare l'ipotesi di uguaglianza delle varianze nei due gruppi.

4.3.6 Test per la differenza tra due medie per campioni appaiati

In alcuni casi i due campioni di osservazioni X e Y non possono essere considerati indipendenti. In questo caso si parla di *campioni appaiati*, ossia campioni in cui ad ogni osservazione nel primo gruppo corrisponde un'osservazione nel secondo gruppo.

Avremmo campioni appaiati, ad esempio, quando ogni soggetto è esaminato prima e dopo un trattamento, oppure quando un ricercatore appaia i soggetti di un gruppo con quello di un secondo gruppo così che i membri di una coppia siano il più possibile simili in relazione ad importanti caratteristiche come, ad esempio, l'età e il sesso. Pertanto, non si tratta di campioni indipendenti, perché X_i e Y_i sono variabili casuali associate alla stessa unità statistica. Assumendo X e Y con distribuzione Normale, poiché le variabili casuali differenze $D_i = X_i - Y_i$ sono tra loro indipendenti, l'analisi del campione casuale D_1, D_2, \dots, D_n si riconduce a quello sul valore medio di una variabile casuale Normale con varianza incognita, e la cui regione critica è determinata dalla statistica test riportata nell'Equazione 4.4 che, sotto H_0 , si distribuisce come una t di Student con $n - 1$ gradi di libertà. Pertanto, anche in questo caso, in R useremo il comando `t.test(...)` modificando l'opzione `paired=TRUE`.

Consideriamo il seguente esempio: supponiamo di misurare la pressione sistolica di un gruppo di pazienti prima e dopo un trattamento. Vogliamo verificare se il trattamento influisce sul livello medio di pressione, ossia vogliamo verificare il seguente sistema di ipotesi:

$$H_0 : \mu_{\text{After}} = \mu_{\text{Before}} \qquad H_A : \mu_{\text{After}} \neq \mu_{\text{Before}} \qquad (4.12)$$

In R, effettuiamo un test per campioni appaiati nel seguente modo:

```
#Leggiamo i dati
blood.data= read.table('blood-Paired-T-test.txt',header=T)

# Leggiamo i nomi delle colonne
names(blood.data)

# Vediamo le dimensioni del dataset
dim(blood.data)

# Distinguiamo i due gruppi
pressure.before=blood.data$Before
pressure.after=blood.data$After

# Applichiamo il test t per campioni appaiati
t.test(x=pressure.before,y=pressure.after,
       alternative='two.sided',paired=T)

>      Paired t-test
> data:  pressure.before and pressure.after
> t = 7.7487, df = 29, p-value = 1.519e-08
> alternative hypothesis: true difference in means is not equal to 0
> 95 percent confidence interval:
```



```
>          10.37838 17.82162
> sample estimates: mean of the differences
>          14.1
```

Come possiamo vedere, poiché il $p\text{-value} < 0.05$, possiamo rifiutare l'ipotesi nulla in favore dell'ipotesi alternativa: ciò significa che c'è abbastanza evidenza sperimentale per poter concludere che il livello medio di pressione varia dopo aver somministrato il trattamento.

È da notare che questo test resta inalterato anche se i campioni sono indipendenti: l'unica differenza sta nel fatto che mentre per campioni appaiati si considera una distribuzione t di Student con $n - 1$ gradi di libertà, per campioni indipendenti si ha un guadagno in termini di gradi di libertà che diventano $2n - 2$.

4.4 Test non Parametrici

4.4.1 Cosa vedremo

Tutti i test riportati nelle precedenti sezioni possono essere applicati quando si ha a che fare con popolazioni Normali.

Ma come verificare quando l'assunzione di normalità è valida o no?

In generale, per risolvere questo problema, possiamo effettuare sia un'analisi informale, basata ad esempio su alcuni opportuni grafici, sia un'analisi più formale mediante l'uso di test che verifichino quanto la distribuzione dei dati in esame si discosti da quella Normale.

Nella sezione successiva illustreremo alcune procedure che ci permettono di dare un giudizio sulla normalità o meno della distribuzione.

4.4.2 Normale o non Normale?

Un primo approccio sulla valutazione della normalità è fornito dall'analisi grafica dei nostri dati: in particolare, in prima battuta l'istogramma dei dati e la corrispondente *stima kernel* (vedi Paragrafo 3.2) possono aiutarci ad avere un'idea della distribuzione dei dati e a verificare quanto i nostri dati si discostano dalla Normale. Ad esempio, se alcune caratteristiche salienti della distribuzione Normale, come la simmetria e l'unimodalità non sembrano rispettate, possiamo cominciare ad avere qualche legittimo dubbio.

Per chiarire le idee, consideriamo la seguente Figura 4.2 in cui, per due dataset simulati, plottiamo gli istogrammi delle due distribuzioni, sovrapponiamo le stime kernel e le densità della distribuzione teoriche corrispondenti:

In questo caso, poiché vogliamo valutare la normalità dei dati, come distribuzione teorica abbiamo scelto la distribuzione Normale. La curva teorica è stata quindi

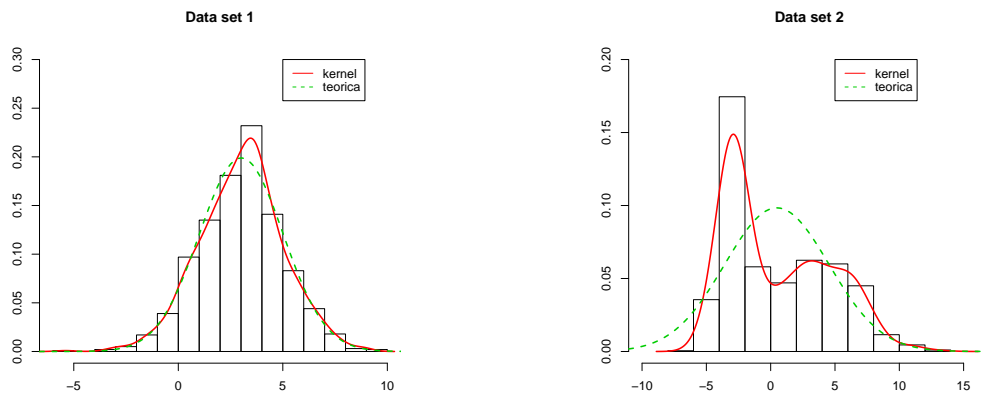


Figura 4.2: Istogrammi, stime kernel e curva teorica per due dataset simulati

costruita considerando media e deviazione standard del campione osservato: pertanto, ad esempio, per il dataset 1 la curva teorica è una curva Normale con media e deviazione standard pari alla media e alla deviazione standard del dataset 1.

Quanto più la stima kernel è vicina a quella teorica, tanto più saremo sicuri che l'assunzione di normalità può essere valida.

Nell'esempio riportato in Figura 4.2, per il dataset 1 le due curve (teorica ed empirica) sono abbastanza vicine e la forma della distribuzione dei dati sembra rispettare le proprietà della distribuzione Normale. Per il dataset 2, al contrario, la bimodalità accentuata della curva e la discrepanza tra curva teorica ed empirica non fanno ben sperare.

Un'analisi grafica più approfondita può basarsi sull'uso del `qqplot` che, per la distribuzione Normale, è ottenuto in R mediante il comando `qqnorm(...)`: questo plot confronta i quantili del campione in esame con quelli della distribuzione Normale standard. Se i quantili della distribuzione in esame sono molto diversi da quelli della distribuzione Normale standard, i punti del plot non sono allineati. In questo caso, possiamo avere dei dubbi sulla normalità dei dati.

Costruiamo i `qqplot` per i due data set simulati.

Come possiamo vedere, questo plot concorda con quanto concluso dalla semplice analisi grafica delle curve empiriche e teoriche: mentre il dataset 1 sembra seguire una distribuzione Normale, il dataset 2 si discosta in modo significativo da essa.

Al fine di valutare l'ipotesi di normalità in modo più formale, sono stati introdotto svariati test di normalità. Senza entrare nei dettagli, nel seguito considereremo alcuni dei test di normalità più utilizzati, come il test di *Kolmogorov-Smirnov*, di *Lilliefors* e di *Shapiro-Wilk*.

Il test di **Kolmogorov-Smirnov** viene utilizzato per capire se un campione proviene da una popolazione con una specifica distribuzione (non necessariamente Normale).

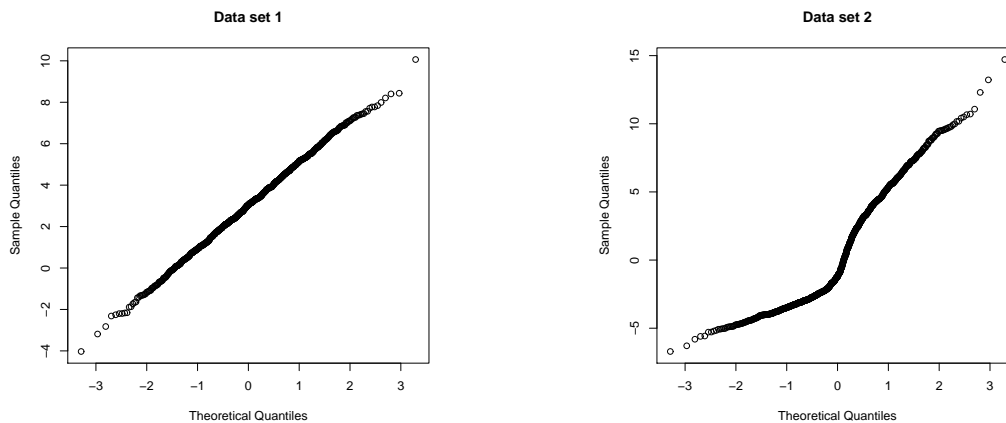


Figura 4.3: qqnorm dei data set simulati

In particolare, il test viene utilizzato per verificare ipotesi nulle del tipo

H_0 : Il campione proviene dalla distribuzione \mathcal{P} , con \mathcal{P} distribuzione di probabilità.

Il test è basato sul confronto tra la *funzione di ripartizione empirica* $F_n(x)$ e quella teorica $F(x)$ associata alla distribuzione di probabilità \mathcal{P} specificata in H_0 . Più in particolare, esso è basato sulla statistica test:

$$D_n = \sup_{1 \leq i \leq n} |F_n(x_i) - F(x_i)|$$

ossia sull'estremo superiore delle differenze in valore assoluto tra la funzione di ripartizione empirica e quella teorica. La distribuzione di probabilità di questa statistica, sotto l'ipotesi nulla, non dipende dal tipo di distribuzione ipotizzata in H_0 (assumendo che quest'ultima sia continua). Rifiuteremo H_0 se $D_n > d_\alpha$ dove il valore critico d_α è riportato su apposite tavole. È da notare che il test di Kolmogorov-Smirnov, nella sua formulazione standard, assume che la distribuzione $F(x)$, sotto H_0 sia *completamente specificata* (vedi Babu e Padmanabhan, Sankhyā 2002, per una recente discussione).

In R, il test di Kolmogorov-Smirnov è implementato nel comando `ks.test(...)`. Appliciamo tale test a dati simulati estratti da una popolazione Weibull e confrontiamoli con una distribuzione Weibull di parametri 2 e 1. In prima battuta, tracciamo il grafico per confrontare le due funzioni di ripartizione, quella empirica e quella teorica:

```
> x.wei=rweibull(100,2,1)
> x=seq(0,2,0.1)
> plot(x,pweibull(x,2,1),main='FDC empirica e teorica Weibull',type='l',col=2,lwd=2,ylab='',xlab='')
> plot(ecdf(x.wei),add=T)
```

Le due funzioni di ripartizione riportate in Figura 4.4 sono molto vicine: ci aspettiamo quindi che il test di Kolmogorov-Smirnov ci permetta di concludere circa

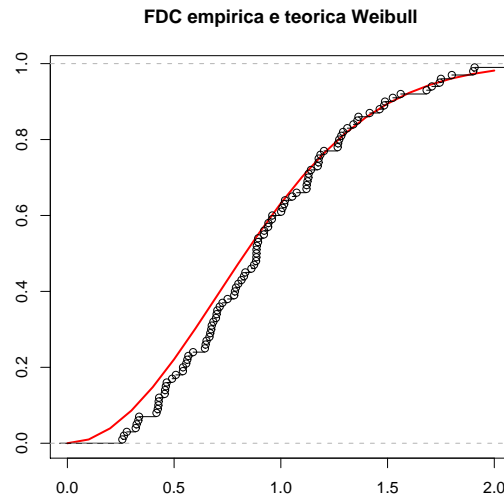


Figura 4.4: Funzione di ripartizione empirica e teorica per una distribuzione Weibull di parametri 2 e 1

la conformità della distribuzione empirica con quella teorica. Questo risultato è, in questo caso, abbastanza ovvio visto che i dati sono stati simulati dalla distribuzione teorica scelta. Effettuiamo quindi il test, utilizzando le seguenti righe di codice:

```
ks.test(x.wei,"pweibull",2,1)
```

```
>          One-sample Kolmogorov-Smirnov test
> data:    x.wei
> D = 0.0998, p-value = 0.2722
> alternative hypothesis: two.sided
```

Ovviamente, si accetta l'ipotesi di conformità alla distribuzione teorica scelta in quanto il *p-value* ottenuto è superiore al livello di significatività scelto $\alpha = 0.05$.

Come detto precedentemente, il test di Kolmogorov-Smirnov può essere applicato per confrontare la distribuzione di un campione con una qualsiasi distribuzione di probabilità. Tuttavia, per la distribuzione Normale, è stato dimostrato che quando i parametri della popolazione μ e σ non sono noti ma devono essere stimati a partire dai dati, il test risulta essere abbastanza conservativo, nel senso che la sua potenza è inferiore a quella ottenibile con μ e σ noti a priori.

Per ovviare a questo problema, un'estensione di questo test per la distribuzione Normale con parametri μ e σ incogniti, è stata introdotta da Lilliefors e va sotto il nome di **test di Lilliefors**.

Il test è identico a quello di Kolmogorov-Smirnov eccetto per il fatto che la distribuzione teorica avrà media e varianza pari alla media e alla varianza campionaria. Pertanto, la statistica test che ne deriva ha una particolare distribuzione, detta "di Lilliefors".

Per effettuare questo test in R, basterà utilizzare il comando `lillie.test(...)`, dopo

	Es. 1: Pressione	Es. 2: Diete topi	Es. 3: Trt Pressione
Lilliefors	0.1825 ($p = 0.0794$)	0.1471 ($p = 0.3080$)	0.1218 ($p = 0.3069$)
Shapiro-Wilk	0.9353 ($p = 0.1955$)	0.9275 ($p = 0.1381$)	0.9615 ($p = 0.3389$)

Tabella 4.1: Test di normalità, Lielliefors e Shapiro-Wilk, per due dataset: pressione, diete topi e trattamenti per la pressione

aver installato e caricato l'apposito pacchetto `nortest`. Consideriamo nuovamente l'esempio sui valori della pressione sistolica:

```
# Carichiamo il pacchetto nortest
library(nortest)

# Inseriamo i valori della pressione sistolica del campione osservato
x=c(98,160,136,128,130,114,123,134,128,107,123,125,129,132,154,115,126,132,136,130)
lillie.test(x)

>      Lilliefors (Kolmogorov-Smirnov) normality test
> data:  x D = 0.1825, p-value = 0.07941
```

Poiché il $p\text{-value} > 0.05$, non possiamo rifiutare l'ipotesi nulla di normalità.

Un altro test di normalità largamente utilizzato è quello di **Shapiro-Wilk**. In questo test, la verifica della normalità avviene confrontando due stimatori alternativi della varianza s^2 : uno stimatore non parametrico, basato sulla combinazione lineare ottimale della statistica d'ordine di una variabile aleatoria Normale, e il suo consueto stimatore parametrico, ossia la varianza campionaria. In R, questo test può essere facilmente implementato utilizzando il comando `shapiro.test(...)`. Per i dati sui valori della pressione sistolica si ha:

```
# Inseriamo i valori della pressione sistolica del campione osservato
x=c(98,160,136,128,130,114,123,134,128,107,123,125,129,132,154,115,126,132,136,130)

shapiro.test(x)

>      Shapiro-Wilk normality test
> data:  x W = 0.9353, p-value = 0.1955
```

Poiché il $p\text{-value} > 0.05$, non possiamo rifiutare l'ipotesi di normalità concordando con quanto concluso dal test di Kolmogorov-Smirnov.

Riportiamo ora i grafici (Figura 4.5) e i test di Lilliefors e di Shapiro-Wilk (Tavola 4.1) per i tre dataset utilizzati nei paragrafi precedenti, ossia l'esempio della pressione sanguigna, delle diete dei topi e della pressione prima e dopo il trattamento.

Come possiamo vedere dalla Tabella 4.1, in tutti i casi non possiamo rifiutare l'ipotesi nulla di normalità e quindi ci possiamo fidare dei risultati dei test parametrici precedentemente applicati.

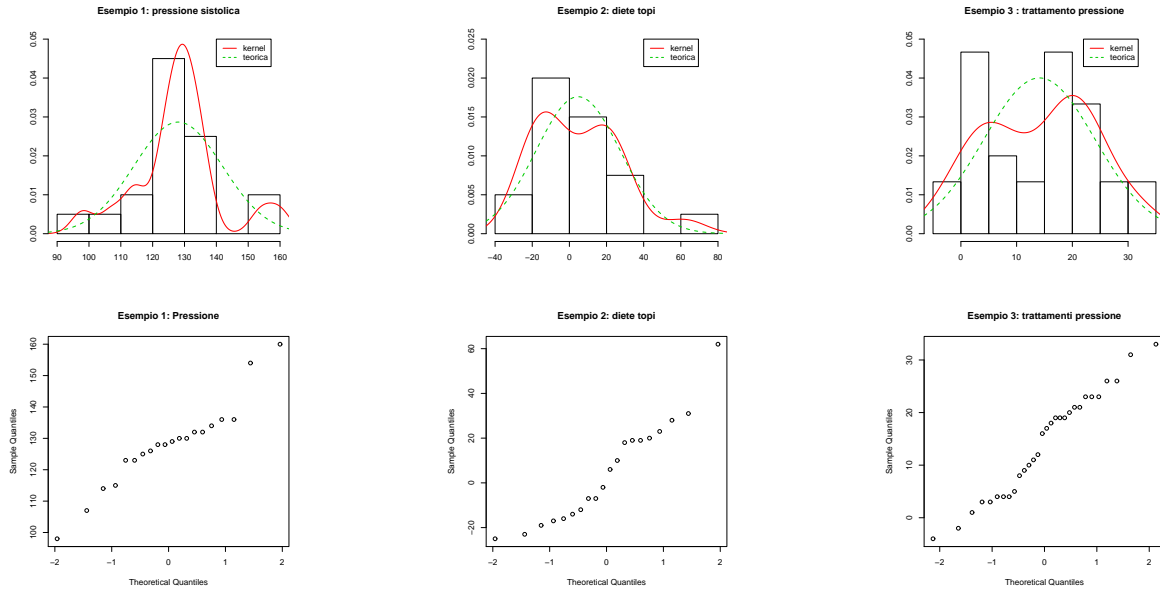


Figura 4.5: Istogrammi, stime kernel, curva teorica e qqnorm per i due dataset: pressione sistolica, diete topi e trattamenti per la pressione

Sebbene i test *parametrici* siano abbastanza robusti rispetto all'assunzione di normalità, talvolta tale ipotesi potrebbe non essere soddisfatta oppure, più semplicemente, vorremmo evitare di fare ipotesi sulla distribuzione, specialmente quando abbiamo a che fare con campioni di bassa numerosità. A tal fine sono stati sviluppati alcuni test detti *non parametrici* che permettono di effettuare confronti tra campioni senza richiedere particolari assunzioni sulla loro distribuzione.

4.4.3 Test dei ranghi con segno di Wilcoxon

In questa sezione, consideriamo il **test di Wilcoxon**, che potrebbe essere considerato come l'equivalente non parametrico del **test t**. Questo test è basato sul confronto delle mediane di due campioni, richiedendo come unica assunzione la continuità delle variabili di interesse.

In particolare, per un campione casuale di osservazioni *appaiate*, $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$, indichiamo con $D_i = (Y_i - X_i)$ le corrispondenti differenze. Se si tratta di un solo campione, indichiamo con $D_i = (X_i - Med)$ le differenze rispetto alla mediana Med . In entrambi i casi, il sistema di ipotesi da verificare è del tipo:

$$H_0 : Med_{D_i} = 0 \quad H_A : Med_{D_i} \neq 0 \quad (4.13)$$

Anche in questo caso, l'ipotesi alternativa può essere specificata in modo unilaterale come $H_A : Med_{D_i} < 0$ oppure $H_A : Med_{D_i} > 0$.

La statistica test per il test dei ranghi con segno di Wilcoxon è la somma dei ranghi

$r(|D_i|)$ corrispondenti alle differenze $D_i > 0$:

$$T_n = \sum_{i=1}^n r(|D_i|) \mathbb{I}(D_i > 0) \quad (4.14)$$

dove $\mathbb{I}(D_i > 0)$ è una funzione indicatrice, che vale 1 se $D_i > 0$ e 0 altrimenti. I valori critici di questa statistica test sono disponibili su apposite tavole.

Per questo test, la sintassi di R è simile a quella dei test precedentemente considerati e il comando da utilizzare è `wilcox.test(...)`.

Applichiamo questo test al data set delle diete dei topi, sebbene, come visto nella precedente sezione, per questi dati l'assunzione di normalità non risulti implausibile. In particolare il sistema di ipotesi che vogliamo verificare è:

$$H_0 : Med_{Carne} = Med_{Cereali} \quad H_A : Med_{Carne} \neq Med_{Cereali} \quad (4.15)$$

In R utilizziamo le seguenti righe di comando:

```
peso.topi=read.table('weight-Rat-T-test.txt',header=T)
gruppo.dieta.carne=peso.topi[peso.topi$Protein=='Beef',]$Gain
gruppo.dieta.cereali=peso.topi[peso.topi$Protein=='Cereal',]$Gain
wilcox.test(gruppo.dieta.carne,gruppo.dieta.cereali,paired=F)

>      Wilcoxon rank sum test with continuity correction
> data:  gruppo.dieta.carne and gruppo.dieta.cereali
>  W = 230.5, p-value = 0.4167
> alternative hypothesis: true mu is not equal to 0
```

Poichè il $p\text{-value} > 0.05$, non c'è abbastanza evidenza sperimentale per rifiutare l'ipotesi nulla di uguaglianza delle mediane. Ovviamente questo risultato è concorde a quello ottenuto per le medie con il corrispettivo test parametrico t: infatti, essendo rispettata l'ipotesi di normalità e per le proprietà della distribuzione Normale, non potevamo aspettarci conclusioni discordanti.

In generale, i test non parametrici sono basati su delle assunzioni molto meno restrittive rispetto a quelli parametrici: verrebbe spontaneo chiedere perché non utilizzare sempre i test non parametrici, qualora le assunzioni sulle distribuzioni su cui si basano risultino verificate. Il motivo fondamentale è legato alla cosiddetta potenza del test: si dimostra che i test parametrici sono, in generale, più potenti rispetto a quelli non parametrici, ossia sono in grado di rilevare ipotesi nulle false più frequentemente rispetto a quelli non parametrici.

4.5 Test per variabili qualitative

4.5.1 Test di omogeneità per due variabili qualitative

I test illustrati fino ad ora, parametrici e non parametrici, possono essere applicati solo a variabili di tipo quantitativo. Molto spesso però le variabili di interesse sono

di tipo qualitativo o categorico (vedi Capitolo 3). Consideriamo il seguente esempio: in una prova clinica si vuole verificare l'efficacia di due tipi diversi di chemioterapia. Il trattamento classico, detto sequenziale, consiste nel somministrare la stessa combinazione di agenti chemioterapici ad ogni ciclo; il nuovo trattamento, detto alternante, consiste nel somministrare tre differenti combinazioni, alternando nei vari cicli. Vogliamo verificare se la distribuzione del numero di peggiorati, stazionari, parzialmente migliorati e migliorati nei due gruppi di terapia, sequenziale o alternata, è la stessa: ciò significa che vogliamo verificare se tra la terapia e la situazione dei pazienti ci sia una relazione. Costruiamo una tabella di contingenza del tipo

	Peggiorati	Stazionari	Parzialmente Migliorati	Migliorati
Sequenziale	28	45	29	26
Alternato	41	44	20	20

Per avere le distribuzioni marginali basterà scrivere:

```
#Leggiamo i dati
dati=read.table("contingenza.txt")

#Definiamo la tabella di contigenza
table.con = table(dati[,2],dati[,1])
table.con

>
>      migliorati  parz.migliorati  peggiorati  stazionari
> alternating      20             20         41         44
> sequenziale      26             29         28         45

# Distribuzione marginale per la ‘variabile trattamento’
margin.table(table.con,1)

> alternating sequenziale
>      125      128

# Distribuzione marginale per la variabile ‘condizione paziente’
margin.table(table.con,2)

> migliorati parz.migliorati  peggiorati  stazionari
>      46      49             69             89

In prima battuta, al fine di avere un'idea della dipendenza che lega le due variabili, guardiamo alle frequenze relative e alle distribuzioni relative condizionate:

# Tabella delle frequenze relative
```



```
prop.table(table.con)
```

```
>               migliorati   parz.migliorati   peggiorati   stazionari
> alternating 0.07905138      0.07905138      0.16205534      0.17391304
> sequenziale 0.10276680      0.11462451      0.11067194      0.17786561
```

```
# Distribuzione relative condizionate
```

```
prop.table(table.con,1)
```

```
>               migliorati   parz.migliorati   peggiorati   stazionari
> alternating 0.16000000      0.16000000      0.32800000      0.35200000
> sequenziale 0.20312500      0.22656250      0.21875000      0.35156250
```

```
prop.table(table.con,2)
```

```
>               migliorati   parz.migliorati   peggiorati   stazionari
> alternating 0.4347826      0.4081633      0.5942029      0.4943820
> sequenziale 0.5652174      0.5918367      0.4057971      0.5056180
```

Le distribuzioni condizionate possono essere rappresentate tramite barplot:

```
listacol=rainbow(12)
par(mfrow=c(1,1))
barplot(prop.table(table.con,1),beside=T,legend=T,ylim=c(0,1),col=c(listacol[6],listacol[8]))

barplot(t(prop.table(table.con,2)),beside=T,legend=T,ylim=c(0,1),
        col=c(listacol[6],listacol[4],listacol[5],listacol[3]))
```

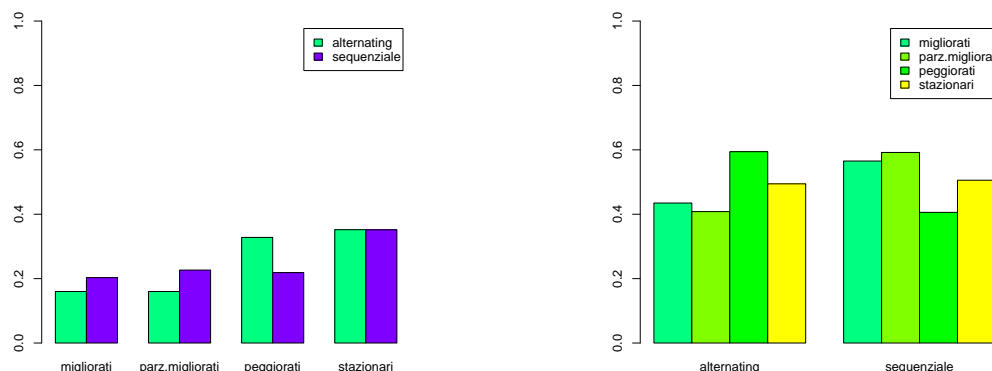


Figura 4.6: Grafici a barre per le distribuzioni condizionate

Guardando la Figura 4.5.1, non sembra esserci grande dipendenza tra le due variabili, in quanto le distribuzioni condizionate non cambiano molto al variare della

modalità condizionante.

Al fine di verificare formalmente la presenza o meno di associazione tra due variabili X e Y , rispettivamente con H e K modalità, possiamo utilizzare il **test Chi-quadrato** le cui ipotesi sono:

H_0 : Le due distribuzioni coincidono H_A : le due distribuzioni sono differenti

Alternativamente, indicate con π_{ij} le probabilità congiunte e con $\pi_{i\cdot}$ e $\pi_{\cdot j}$ quelle marginali, il sistema di ipotesi può essere scritto in modo equivalente come:

H_0 : $\pi_{ij} = \pi_{i\cdot}\pi_{\cdot j}$ H_A : le π_{ij} non rispettano i vincoli previsti da H_0

Nel nostro esempio, se le distribuzioni nei due gruppi coincidessero, ossia se H_0 fosse vera, potremmo concludere che le due terapie sortiscono lo stesso effetto.

La statistica test è basata sulla differenza tra le frequenze osservate n_{ij} e quelle teoriche ottenibili in caso di indipendenza, $n'_{ij} = np_{i\cdot}p_{\cdot j}$:

$$\chi^2 = \frac{(n_{ij} - n'_{ij})^2}{n} \quad (4.16)$$

Si può dimostrare che tale statistica si distribuisce asintoticamente come un Chi-Quadrato con $(H - 1)(K - 1)$ gradi di libertà.

In R, il test Chi-quadrato è implementato nel seguente modo usando il comando `chisq.test(...)`.

```
chisq.test(table.con)
```

```
>      Pearson's Chi-squared test
> data:  x X-squared = 4.8613, df = 3, p-value = 0.1822
```

o, alternativamente,

```
summary(table.con)
```

```
> Number of cases in table: 253 Number of factors: 2
> Test for independence of all factors:
>      Chisq = 4.861, df = 3, p-value = 0.1822
```

Il *p-value* è maggiore del livello di significatività fissato 0.05, pertanto non possiamo rifiutare l'ipotesi nulla: quindi possiamo concludere che l'evidenza sperimentale non è sufficiente a dimostrare una significativa differenza tra le due terapie.

È da notare che questo test presenta dei problemi quando il numero di osservazioni è piuttosto basso (in genere inferiore a 5): in questi casi appare un messaggio di warning del tipo

```
> Chi-squared approximation may be incorrect in: chisq.test(x).
```

In genere, questo problema può essere risolto mediante l'uso di metodi *simulativi*: in R basterà aggiungere nella linea di comando `chisq.test(...)`, l'opzione `simulate.p.value=TRUE`.

4.5.2 Test per proporzioni

Il test per le proporzioni viene utilizzato per valutare la frequenza di un particolare evento o carattere.

Indicata con π la frequenza del carattere di interesse, supponiamo di voler verificare il seguente sistema di ipotesi:

$$H_0 : \pi = \pi_0 \qquad H_A : \pi \neq \pi_0$$

Ipotizzando che nella popolazione un carattere sia presente nel π percento dei casi, allora il numero di volte x che esso compare in un campione bernoulliano di ampiezza n è distribuito come una variabile casuale binomiale.

Per convenienza denotiamo il risultato $X = 1$ come presenza (o successo), e con $X = 0$ indichiamo assenza (o insuccesso) e sia $Y = \sum_{i=1}^n X_i$ il numero di successi (ovvero il numero di risultati in cui $X = 1$). Si dimostra che la statistica test Y ha distribuzione binomiale di parametri n e p , ossia $Y \sim \text{Binomiale}(n, p)$.

Questo test può essere facilmente effettuato in R mediante l'uso del comando `binom.test(...)`.

Consideriamo il seguente esempio: l'ISTAT ha rilevato che nel 1991 l'8.4% degli occupati in Italia è impiegato nel settore agricolo. Si vuole verificare se ad oggi tale proporzione è rimasta invariata o invece è diminuita, avendo osservato un campione casuale di $n = 1000$ occupati di cui 53 sono impiegati nel settore agricolo.

Indicando con π la proporzione di occupati, l'ipotesi nulla e quella alternativa possono essere scritte come:

$$H_0 : \pi = 0.084 \qquad H_A : \pi < 0.084. \qquad (4.17)$$

Questo test è detto **test esatto binomiale** e per verificare questo sistema di ipotesi, basterà a scrivere in R:

```
binom.test(53,1000,0.084, alternative="less")

>      Exact binomial test
> data:  53 and 1000 number of successes = 53, number of trials = 1000, p-value = 0.0001128
> alternative hypothesis: true probability of success is less than 0.084
> 95 percent confidence interval:
>  0.00000000 0.06617064
> sample estimates: probability of success
>                0.053
```

Il *p-value* conferma una forte evidenza contro l'ipotesi nulla di invarianza della percentuale di occupati.

Nel caso di campioni di elevata numerosità, si può fare ricorso all'approssimazione Normale: ossia si può dimostrare che la statistica test, sotto l'ipotesi nulla H_0 , si distribuisce approssimativamente come una distribuzione Normale standardizzata. Questo test è detto **test asintotico per proporzioni**. Per applicare questo test ai dati del precedente esempio, basterà utilizzare il comando `prop.test(...)` scrivere:

```
prop.test(53,1000,0.084,alternative='less')
```

```
> data: 53 out of 1000, null probability 0.084 X-squared = 12.09, df= 1, p-value = 0.0002535
> alternative hypothesis: true p is less than 0.084
> 95 percent confidence interval:
>  0.0000000 0.0664557
> sample estimates:
>  p
> 0.053
```

Il risultato è equivalente a quello ottenuto con il test esatto, ossia possiamo rifiutare l'ipotesi nulla di invarianza della proporzione di impiegati nel settore agricolo. E' da notare che l'approssimazione Normale è lecita quando la dimensione campionaria è elevata: infatti, la bontà dell'approssimazione, garantita dal teorema del limite centrale, dipende da n e da π_0 . L'approssimazione migliora per valori crescenti di n e all'avvicinarsi di π_0 a 0.5. Come regola indicativa, l'approssimazione Normale può essere utilizzata quando $n \cdot \pi_0 \geq 5$ e $n \cdot (1 - \pi_0) \geq 5$.

Lo stesso test può essere effettuato quando si vogliono confrontare due o più proporzioni: confrontiamo ad esempio la proporzione di fumatori in quattro popolazioni. Vogliamo verificare se la proporzione di fumatori nelle popolazioni considerate è la stessa (l'ipotesi alternativa è che tale proporzione differisca in almeno una delle popolazioni). In R scriviamo:

```
smokers <- c( 83, 90, 129, 70 )
patients <- c( 86, 93, 136, 82 )
prop.test(smokers, patients)

>      4-sample test for equality of proportions without continuity correction
> data:  smokers out of patients
> X-squared = 12.6004, df = 3, p-value = 0.005585
> alternative hypothesis: two.sided sample estimates:
>  prop 1    prop 2    prop 3    prop 4
> 0.9651163 0.9677419 0.9485294 0.8536585
```

Il valore del *p-value* permette di concludere che c'è abbastanza evidenza sperimentale per rifiutare l'ipotesi nulla in favore di quella alternativa.

Vediamo, ora, come l'applicazione di questo semplice test può fornirci informazioni interessanti su problemi reali, come ad esempio la correttezza o meno di alcuni giochi a premi. In questo ultimo esempio consideriamo il Gioco a premi Affari Fuori. Il gioco consiste nell'eliminazione, uno dopo l'altro, di premi di diversa entità.

E' stato osservato che in molte puntate i premi più ricchi sono arrivati alla fine più frequentemente, attirando l'interesse del pubblico con conseguente incremento dello share televisivo. Questo ha fatto emergere dubbi sulla regolarità del gioco, nel senso che in qualche modo l'organizzazione del gioco potesse fornire indicazioni ai concorrenti, in modo da far sì che i premi più ricchi rimanessero fino alla fine.

Il gioco consiste nelle seguenti fasi:

- Si inizia con 20 pacchi, ognuno dei quali contiene un premio. I premi hanno un valore variabile: da pochi centesimi o fino a 250mila o 500 mila euro.

- Un giocatore viene sorteggiato e viene richiesto di scegliere una per volta un pacco. In questo modo, con il procedere del gioco, aumenta l'informazione che il giocatore ha rispetto al contenuto del pacco.
- Durante il gioco viene offerta al giocatore la possibilità di cambiare il pacco o fermare il gioco accettando una somma di denaro

Se tutto procedesse regolarmente, ossia se i giocatori non avessero informazioni sul contenuto del pacco, tutti i pacchi avrebbero la stessa probabilità di arrivare alla fine del gioco. Si parla, quindi, di *condizione di equiprobabilità*. In particolare, possiamo formalizzare il problema nel seguente modo: vogliamo valutare se **la frequenza con cui almeno uno dei due pacchi più ricchi (250mila e 500mila) arriva fino alla fine del gioco è compatibile con l'ipotesi di equiprobabilità**.

Indichiamo, quindi, come Successo il seguente evento:

Successo= almeno uno dei due pacchi che arrivano alla fine del gioco contiene 250mila o 500mila euro.

In condizioni di equiprobabilità,

$$Pr(\text{successo}) = 1 - \frac{C_{18,2}}{C_{20,2}} = 0.1947$$

La legge di probabilità che governa questo tipo di eventi è quella Binomiale (numero di successi **S** in **N** prove).

Calcoliamo, quindi, la probabilità che in 10 puntate si verifichino da 0 a 10 successi in condizioni di equiprobabilità.

Come farlo in R?

```
dbinom(0:10, 10, 0.1947)*100
```

Ora, guardiamo ai dati relativi alle puntate di Affari Tuoi e verifichiamo quanto la frequenza osservata dei successi si discosta da quella teorica in condizioni di equiprobabilità.

```
### Leggiamo i dati pacchi=read.csv('Pacchi.csv',header=T)
dim(pacchi)
names(pacchi)
### Contiamo quante volte i 2 premi
maggiori sono arrivati fino alle ### ultime 6, 5, 4, 3 e 2 chiamate
sum(pacchi$ultimi6)
sum(pacchi$ultimi5)
sum(pacchi$ultimi4)
sum(pacchi$ultimi3)
sum(pacchi$ultimi2)
```

```
### Definiamo "successo = almeno uno dei 2 pacchi contenenti i 2 ##
premi maggiori arrivano fino alle ultime due estrazioni
```

```
### Calcoliamo la probabilit di successo in caso di equiprobabilit
```

```
p.equiprob= 1- choose(18,2)/choose(20,2) p.equiprob
```

```
### Applichiamo il test
```

```
n.successi= sum(pacchi$ultimi2)
```

```
n.prove= nrow(pacchi)
```

```
test.esatto= binom.test(x=n.successi, n=n.prove, p=p.equiprob,
alternative="greater")
```

Cosa possiamo concludere?

4.6 Esercizi

Esercizio 1 Caricare il data set Burt con i comandi:

```
library(car)
data(Burt)
```

1. Come è fatto il data set? Cosa rappresentano le righe? Cosa le colonne?
2. Costruire una funzione che per ogni vettore di numeri calcoli le seguenti quantità: media, mediana, differenza interquartilica e deviazione standard. Usarla per calcolare queste quantità separatamente sui due diversi quozienti intellettivi.
3. Calcolare le quantità su tutti i gemelli insieme (sia allevati dai genitori naturali che adottivi).
4. Per ciascuna delle due variabili quantitative, consideriamo le seguenti trasformazioni: la trasformazione logaritmica, la radice quadrata e la seconda potenza. Fare alcuni grafici per ciascuna delle trasformazioni e valutarne la normalità mediante opportuni grafici e test di normalità.
5. Consideriamo la trasformazione logaritmica di entrambe le variabili. Verificare tramite un test al 10% se vi è una differenza significativa nel quoziente di intelligenza tra gemelli allevati da genitori biologici e adottivi.

6. Fondere le due variabili IQ in una sola variabile Y e farne la trasformazione logaritmica. Sia X questa nuova variabile:

1. Calcoliamo l'intervallo di confidenza per la media di X , sapendo che l'intervallo di confidenza per la media di X è dato da

$$\left[\bar{x} - t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{\text{sd}(x)}{\sqrt{n}}, \bar{x} + t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{\text{sd}(x)}{\sqrt{n}} \right],$$

dove $t_{1-\frac{\alpha}{2}}$ è il quantile a livello $1 - \frac{\alpha}{2}$ della distribuzione t di Student, mentre $\text{sd}(\cdot)$ denota la deviazione standard dei dati.

2. Calcolare l'intervallo di confidenza per Y , sapendo che l'intervallo di confidenza per $f(X)$ è pari a $f^{-1}(\text{IC}_X)$ dove IC_X è l'intervallo di confidenza per X .

Esercizio 2 In uno studio sugli effetti sulla salute degli inquinanti atmosferici contenenti zolfo, agli abitanti nei pressi di una fabbrica di pasta di legno in Finlandia, sono state formulate delle domande sulla sintomatologia dopo una forte emissione avvenuta nel settembre del 1987. Gli stessi soggetti sono stati intervistati dopo quattro mesi, durante un periodo di bassa esposizione. I dati sono riportati nel file `cefalea-Finlandia.csv`.

1. Quali variabili sono contenute nel data set? Che tipo di variabili sono?
2. Costruire la tabella di contingenza.
2. Costruire le distribuzioni marginali per ciascuna variabile.
3. Al fine di avere un'idea della dipendenza tra esposizione ad inquinanti atmosferici contenenti zolfo e insorgenza di cefalea, calcolare le distribuzioni relative condizionate e rappresentarle mediante opportuni grafici.
4. Testare l'ipotesi nulla che non esiste alcuna associazione tra esposizione ad inquinanti atmosferici contenenti zolfo e insorgenza di cefalea.

Capitolo 5

Correlazione e regressione lineare

5.1 Cosa vedremo

Come si è visto nel Capitolo 3, l'analisi della interdipendenza (associazione) tra due caratteri non necessita di particolari assunzioni, visto che gli unici *ingredienti* utilizzati nel calcolo di indici di tipo χ^2 sono le frequenze della tabella doppia $\{n_{ij}\}$, $i = 1, \dots, r$, $j = 1, \dots, s$. Tra l'altro, la costruzione di indici di questo tipo non prevede la presenza di una relazione asimmetrica tra le variabili considerate. In questo capitolo, ci interesseremo dell'estensione dell'analisi della interdipendenza (associazione) al caso in cui si osservino due (o più) caratteri quantitativi. Tale analisi sarà essenzialmente basata sul *coefficiente di correlazione* di Bravais Pearson e sulle sue estensioni.

Il concetto di associazione implica una relazione di tipo simmetrico, nel senso che le variabili analizzate rivestono lo stesso ruolo, ed il ricercatore è interessato a verificare la presenza di un legame, piuttosto che di una relazione *casuale*.

In alcuni casi è però possibile pensare ad una relazione di tipo asimmetrico tra le variabili, ossia, in particolare, una relazione che specifichi un *nesso causale*. In questo caso, alcune delle variabili rivestono il ruolo di *causa* ed agiscono su altre variabili che rivestono il ruolo di *effetto*. Il primo ambito in cui si incontrano tali relazioni è probabilmente quello della *dipendenza in media* di un carattere quantitativo da un carattere che, a sua volta, può essere di tipo qualunque. Tra le possibili specificazioni del legame tra la media della variabile quantitativa e le informazioni aggiuntive riassunte da ulteriori variabili, di particolare rilevanza è il modello di regressione lineare.

Tale modello permette di descrivere il comportamento della media (condizionata) di una variabile quantitativa *risposta* (o *dipendente*), Y , al variare di un insieme di variabili quantitative *indipendenti* (o *covariate*), $\{X_1, \dots, X_p\}$. In particolare, la definizione di un modello di regressione lineare implica un meccanismo di tipo *causale* tra vettore aleatorio (che agisce da causa) e variabile Y (il cui comportamento è l'effetto prodotto dall'azione della causa \mathbf{x}).

5.2 Esempio: EAEF data

Il dataset di esempio che tratteremo in queste pagine si chiama `eaef21.csv` e proviene da una indagine denominata *Educational Attainment and Earnings Functions* (EAEF), costituita da 22 insiemi paralleli, ciascuno costituito da un campione di 540 osservazioni estratte dal *US National Longitudinal Survey of Youth*, con l'intento di descrivere le distribuzioni dei salari (orari) e dei livelli di educazione scolastica raggiunti (Dougherty, 2002).

Per la sua rilevanza in termini di politica sociale, l'analisi delle determinanti dei profili educativi e dei salari è stata per lungo tempo un obiettivo in campo econometrico. Di particolare interesse per la scolarità sono le potenziali differenze riconducibili al gruppo etnico di appartenenza, al genere, a fattori sociali ed alla loro interazione, così come i cambiamenti nel tempo. Il dataset presentato permette di esplorare alcuni di questi punti utilizzando dei dati provenienti da uno dei maggiori database statunitensi, il National Longitudinal Survey of Youth 1979 (NLSY79). Il NLSY79 costituisce un'indagine con interviste ripetute ad un campione rappresentativo (a livello nazionale) di giovani uomini e giovani donne con una età compresa tra 14 e 21 anni (nel 1979). L'indagine è di tipo longitudinale, e le interviste sono state effettuate annualmente dal 1979 al 1994; a partire dal 1994 sono invece state effettuate ogni due anni. Il campione di partenza era composto, originariamente, da 3,003 maschi e 3,108 femmine. In aggiunta, sono stati costruiti campioni supplementari, alcuni non più attivi, di minoranze etniche, di persone in situazione di povertà e di arruolati nelle forze armate. Le variabili di base sono state rilevate nel 1979 e, da allora, le informazioni su variabili di interesse quali il livello scolastico, l'occupazione, lo stato civile, la fertilità, la salute, la proprietà, il reddito, sono state aggiornate con cadenza annuale. Maggiori informazioni sul dataset sono riportate in Appendice.

Il codice R che utilizziamo per leggere il file è basato sulla funzione `read.table` ed è riportato di seguito, insieme alla funzione `names` che permette di stampare a video il nome delle variabili contenute nel dataset.

```
> eaef21=read.table('D:/Documenti/CorsoR/eaef21.csv', sep=',', header=T)
> names(eaef21)
[1] "ID"      "FEMALE"  "MALE"    "AGE"     "S"       "ASVAB02" "ASVAB03"
[8] "ASVAB04" "ASVAB05" "ASVAB06" "ASVABC"  "SM"      "SF"      "EARNINGS"
```

5.3 Notazione

Supponiamo, dunque, che i dati a nostra disposizione consistano di n osservazioni (y_i, \mathbf{x}_i) , dove y_i rappresenta il valore assunto dalla variabile risposta nella i -esima unità statistica, ed $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})^T$ il corrispondente vettore (di dimensione p) delle covariate. I dati osservati possono essere scritti come:

$$(y_1, \mathbf{x}_1), \dots, (y_n, \mathbf{x}_n)$$

dove \mathbf{x}_i rappresenta il vettore delle covariate di generico elemento x_{ij} , che indica il valore assunto dalla j -esima covariata in corrispondenza della i -esima unità statistica.

5.4 Le funzioni `cor(...)` e `cor.test(...)`

Nel Capitolo 3, abbiamo presentato alcuni indici per calcolare l'associazione tra caratteri qualitativi, ossia caratteri le cui modalità possono essere espresse in termini di attributi. In questo capitolo, ci interessiamo, invece, delle misure di associazione tra caratteri quantitativi. La misura della interdipendenza tra due caratteri quantitativi più comunemente utilizzata è certamente il coefficiente di correlazione lineare di Bravais-Pearson, definito come segue:

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

Tale indice misura il legame *lineare* tra le due variabili e presenta le seguenti caratteristiche:

- $-1 \leq r \leq 1$;
- $r = 0$ se le due variabili sono *linearmente* indipendenti;
- $r = 1$ o $r = -1$ se esiste una relazione *lineare* perfetta di segno crescente (rispettivamente decrescente).

Come primo spunto, è necessario notare che la nozione di interdipendenza lineare è più debole rispetto all'interdipendenza in una tabella a doppia entrata, poiché misura un aspetto particolare (la linearità) della relazione tra le variabili. Quindi, $\chi^2 = 0 \Rightarrow \eta^2 = 0 \Rightarrow r = 0$.

La funzione `cor(...)` può essere utilizzata per calcolare il valore del coefficiente di correlazione tra due o più caratteri; nel caso di due variabili, il comando ed il risultato prodotto sono riportati di seguito:

```
> cor(EARNINGS, S)
[1] 0.415329
```

Ovviamente la funzione può anche essere utilizzata sull'intero data frame `eaef21`, producendo il risultato seguente:

	ID	FEMALE	MALE	AGE	S
ID	1.00000000000	0.0006719482	-0.0006719482	-0.065477566	-0.026780603
FEMALE	0.0006719482	1.00000000000	-1.00000000000	0.033513843	-0.020523621
MALE	-0.0006719482	-1.00000000000	1.00000000000	-0.033513843	0.020523621
AGE	-0.0654775659	0.0335138434	-0.0335138434	1.00000000000	0.004774746
S	-0.0267806031	-0.0205236215	0.0205236215	0.004774746	1.00000000000
.....					

Se indichiamo con R_{XY} il coefficiente di correlazione nella popolazione di riferimento, tramite la funzione `cor.test(...)` è possibile sottoporre a verifica l'ipotesi nulla che il campione osservato provenga da una popolazione con correlazione nulla, ossia il sistema di ipotesi:

$$\begin{cases} H_0 : R_{XY} = 0 \\ H_A : R_{XY} \neq 0 \end{cases}$$

Questa funzione ci permette, quindi, di verificare se i dati raccolti indicano una differenza significativa del valore campionario dal valore ipotizzato sotto l'ipotesi nulla. Supponiamo, ad esempio, che ci interessi verificare se il coefficiente di correlazione tra le variabili EARNINGS e S è significativamente diverso da 0. Il comando corrispondente è:

```
> cor.test(EARNINGS, S)
```

Pearson's product-moment correlation

```
data: EARNINGS and S
t = 10.5901, df = 538, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3429706 0.4827875
sample estimates:
      cor
0.415329
```

che mostra il valore della statistica test t ed il p -value (i.e. $\Pr(|t_{n-2}| > t_{obs})$) calcolato in corrispondenza dell'ipotesi nulla adottata. È possibile utilizzare il risultato della funzione per costruire l'intervallo di confidenza (al livello $\alpha = 0.95$) per il coefficiente di correlazione nella popolazione.

Ovviamente, la relazione tra le variabili potrebbe non essere lineare; risulta, quindi, necessario definire degli indici di associazione che misurino la relazione tra le variabili in modo più generale. Una scelta semplice è rappresentata, in questo contesto, dal coefficiente ρ di Spearman, che si ottiene sostituendo ai valori osservati (x_i, y_i) i ranghi corrispondenti e calcolando il coefficiente di correlazione lineare tra tali ranghi. Questo indice può essere ottenuto utilizzando la funzione `cor(...)` nel modo seguente:

```
> cor(EARNINGS, S, method="spearman")
[1] 0.454674
```

Anche in questo caso si può sottoporre a verifica l'ipotesi $H_0 : \rho = 0$, contro un'alternativa bidirezionale, utilizzando la funzione `cor.test(...)` vista prima:

```
> cor.test(EARNINGS, S, method="spearman")
```

Spearman's rank correlation rho

```
data: EARNINGS and S
S = 14311487, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.454674
```

```
Warning message:
Impossibile calcolare p-value esatti in presenza di ties in:
cor.test.default(EARNINGS, S, method = "spearman")
```

In questo caso, viene visualizzato un messaggio di warning derivante dalla presenza di *ties*, ossia di code di valori coincidenti cui corrispondono ranghi coincidenti. In questo caso, il *p-value* calcolato rappresenta un valore approssimato. È interessante notare che il ρ di Spearman può essere utilizzato anche se le variabili sono qualitative ordinabili (vedi Capitolo 3).

Una ulteriore alternativa è rappresentata dall'indice τ di Kendall che rappresenta un indice di interdipendenza basato non sui valori osservati ma sul conteggio delle coppie concordi e discordi. Una coppia è definita concorde se le differenze rispetto alle media del carattere X ed Y presentano lo stesso segno. Altrimenti, la coppia è definita discorde. Anche in questo caso, l'indice può essere ottenuto utilizzando le funzioni `cor(...)` e `cor.test(...)`, specificando l'opzione `method=kendall`.

5.5 Regressione lineare: la funzione `lm(...)`

Per **modello di regressione lineare** si intende una relazione tra y_i ed x_i della forma:

$$y_i = \beta_0 + \sum_{j=1}^p \beta_j x_{ij} + u_i \quad i = 1, \dots, n$$

La quantità

$$\mu_i = \beta_0 + \sum_{j=1}^p \beta_j x_{ij} \quad ,$$

rappresenta la componente *strutturale* o *sistematica* del modello, cui ci si riferirà nel seguito con il termine di *predittore lineare*; la quantità u_i rappresenta invece un termine residuo, che tiene conto della non perfetta aderenza del modello (o meglio della parte strutturale del modello) alla realtà osservata, per motivi di pura casualità.

Le quantità β_0, \dots, β_p rappresentano dei parametri incogniti che devono essere stimati a partire dai dati osservati. L'interpretazione del loro ruolo è piuttosto immediata: β_0 rappresenta l'ordinata all'origine, ossia il valore che il predittore lineare assume quando tutte le covariate sono pari a zero. I termini β_j , $j = 1, \dots, p$ quantificano, invece, l'influenza delle singole covariate sulla variabile risposta e, in particolare, misurano il segno della relazione e l'incremento che si suppone la variabile risposta presenti in *risposta* ad un incremento unitario

della singola covariata, se si ipotizza che tutte le altre covariate siano fissate ad un valore costante. L'aggettivo *lineare* non si riferisce alle X_j che possono essere variabili trasformate (ad es. potenze, logaritmi, etc.) ma piuttosto al fatto che la forma funzionale che lega la variabile risposta alle covariate è lineare nei parametri del modello $\beta_j, j = 0, \dots, p$.

Nota È importante ricordare che il modello di regressione lineare è stato, almeno inizialmente, sviluppato per descrivere dati sperimentali e che, invece, è molto spesso utilizzato su dati osservazionali. In quest'ultimo caso, sorgono alcuni problemi di interpretazione, legati a

- presenza di effetti di confondimento, dovuti alla mancata osservazione di alcune delle variabili indipendenti;
- fenomeni di causalità simultanea: ad esempio una (o più) variabili indipendenti sono a loro volta influenzate da altre variabili indipendenti;
- il fatto che esista un legame tra Y ed alcune $X_j, j = 1, \dots, p$ non implica necessariamente la presenza di un effetto casuale.

Le ipotesi su cui si fonda il modello di regressione lineare, gli approcci alla stima dei parametri del modello, e le eventuali estensioni sono riportati in Appendice.

La funzione fondamentale per la stima dei parametri di un modello lineare in R è la funzione `lm(...)`. Tale funzione presenta una sintassi piuttosto semplice, ma allo stesso tempo molto flessibile. La sintassi corrispondente è la seguente:

```
lm(formula, data, subset, weights, na.action, ...
    model = TRUE, ..., offset, ...)
```

le cui componenti hanno il significato riportato in appendice.

L'output della funzione è molto ricco: in particolare, la funzione produce un oggetto di classe `lm`, ossia una lista che contiene numerose informazioni e diagnostiche che permettono una analisi approfondita del modello stimato. Questa opzione, a dire la verità, non è immediatamente visibile all'utente. Ad esempio il comando:

```
lm(EARNINGS ~ S)
```

produce l'output seguente

```
> lm(EARNINGS ~ S)
```

```
Call: lm(formula = EARNINGS ~ S)
```

```
Coefficients: (Intercept)          S
          -13.933           2.455
```

che risulta piuttosto scarno e, almeno all'inizio, provoca un certo imbarazzo nell'utente. Abbiamo però detto che la funzione `lm(...)` può produrre, in output, un oggetto ricco di informazioni. Ad esempio il comando:

```
summary(lm(EARNINGS $ \sim$ S))
```

produce l'output seguente:

```
Call: lm(formula = EARNINGS $ \sim$ S)
```

Residuals:

Min	1Q	Median	3Q	Max
-24.112	-7.142	-2.476	3.625	95.974

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-13.9335	3.2199	-4.327	1.8e-05 ***
S	2.4553	0.2319	10.590	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.13 on 538 degrees of freedom

Multiple R-Squared: 0.1725, Adjusted R-squared: 0.171

F-statistic: 112.1 on 1 and 538 DF, p-value: < 2.2e-16

che permette di ottenere un livello di approfondimento decisamente maggiore. In particolare, sono riconoscibili le stime dei parametri (Estimate), le stime degli errori standard corrispondenti (Std. Err.), il valore della statistica test (t value) calcolata come rapporto tra le due quantità precedenti, ed utilizzata per la verifica del sistema di ipotesi:

$$\begin{cases} H_0 : \beta_j = 0 \\ H_A : \beta_j \neq 0 \end{cases} \quad j = 1, \dots, p$$

per ciascuno dei coefficienti di regressione. La statistica test t rappresenta quindi uno strumento per determinare quali parametri del modello stimato possono essere considerati, sulla base dell'evidenza empirica, significativamente diversi da zero. La decisione si basa sul p -value corrispondente ($\Pr(>|t|)$) e sul confronto tra questo ed il livello di significatività α utilizzato nel test, anche attraverso la rappresentazione "per asterischi" che R graziosamente fornisce.

Oltre ai test *locali* che è possibile condurre utilizzando la statistica test t , la funzione `lm(...)` fornisce anche il valore della statistica test

$$F = \frac{\text{Dev}_M/p}{\text{Dev}_E/(n-p-1)}$$

per la verifica dell'ipotesi *globale*:

$$\begin{cases} H_0 : & \beta_1 = \dots = \beta_p = 0 \\ H_A : & \text{almeno un } \beta_j \neq 0 \end{cases} \quad j = 1, \dots, p$$

Nel caso in cui tutti i coefficienti siano pari a 0, infatti, la devianza spiegata del modello e la devianza residua (divise per i corrispondenti gradi di libertà) sono stimatori della stessa quantità σ^2 , la varianza residua. Più il valore osservato della statistica test F si discosta da 1, minore sarà il *p-value* corrispondente, più ci si allontana dall'ipotesi nulla.

È anche possibile misurare la bontà di adattamento del modello in termini di proporzione della variabilità complessiva della variabile risposta che risulta spiegata dal modello adottato, sintetizzata nell'indice R^2 (R-squared) e, rispettivamente $\text{adj-}R^2$ (Adjusted R-squared). Quest'ultimo tiene conto del fatto che l'inserimento di una ulteriore covariata, per quanto questa possa essere inutile, sicuramente non provoca una diminuzione della varianza spiegata dal modello. Per questo motivo è necessario definire un indice che non sia influenzato dal numero di covariate inserite dal modello, ma solo dalla capacità esplicativa delle stesse.

Una rappresentazione grafica interessante può essere ottenuta utilizzando la serie di comandi seguente:

```
> modello=lm(EARNINGS ~ S )
> plot(S, EARNINGS)
> abline(lm(EARNINGS ~ S ))
> segments(S, fitted(modello), S, EARNINGS)
```

in cui si sovrappongono, sulla stessa rappresentazione grafica (indotta dal comando `plot(...)`), il diagramma a dispersione della nuvola dei punti, la retta di regressione stimata (prodotta dal comando `abline(...)`) ed il grafico dei residui definiti come differenza tra valori osservati e valori predetti (prodotto dal comando `segments(...)`). Il risultato è riportato in Figura 5.1.

Ovviamente, i risultati sono in linea con quanto ci aspettavamo: il salario aumenta all'aumentare del numero di anni di frequenza scolastica. Oltre alla funzione `summary(...)` è possibile fare uso anche della funzione `plot(...)` per sintetizzare il risultati prodotti dalla funzione `lm(...)`, utilizzando, ad esempio, il comando

```
plot(lm(EARNINGS ~ S), i), i = 1, ..., 6
```

I grafici prodotti sono riportati in Figura 5.2. Il primo grafico rappresenta il diagramma a dispersione dei residui e dei valori previsti, con indicate le unità che corrispondono a potenziali valori anomali (ad esempio le unità numero 135, 177, 439); tale diagramma può essere utilizzato per verificare la presenza di eventuali tendenze nei residui al crescere dei valori predetti. Segue un `qqplot` (vedi Capitolo 4) per l'ispezione visuale dell'ipotesi di normalità sui residui. Viene poi proposto un diagramma a dispersione dei valori della radice quadrata dei residui standardizzati secondo i valori predetti, al fine di ispezionare le ipotesi di omoschedasticità dei residui (vedi Equazione C.1). Segue poi un plot della distribuzione delle osservazioni secondo la *distanza di Cook*, che misura l'effetto della presenza di ciascuna unità statistica sul modello stimato. Ulteriori grafici mostrano poi i diagrammi a dispersione dei valori di *leverage* secondo i residui standardizzati e, rispettivamente, secondo la distanza di Cook. Il *leverage* può essere considerato come l'effetto della singola osservazione nel formare

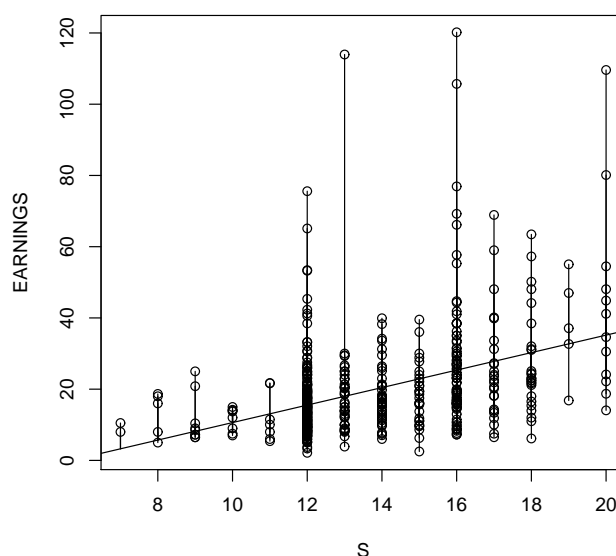


Figura 5.1: Plot dei residui e dei valori predetti

i valori predetti dal modello; può anche essere interpretato, quindi, come distanza del vettore \mathbf{x}_i corrispondente a ciascuna unità statistica, dal baricentro $\bar{\mathbf{x}}$. Per ulteriori approfondimenti si veda (1).

Dal punto di vista della capacità esplicativa del modello, può risultare importante prendere in considerazione, oltre ad S , anche altre caratteristiche delle unità statistiche osservate. Ad esempio, quelle che possono essere derivate dai test della batteria ASVAB. In particolare, inseriremo nel predittore lineare un test di velocità (ASVAB05) ed la media dei test cognitivi (ASVABC). A questo riguardo possiamo definire il seguente modello lineare:

$$\text{EARNINGS} \sim S + \text{ASVABC} + \text{ASVAB05}$$

che produce l'output seguente:

```
> summary(lm(EARNINGS ~ S + ASVABC + ASVAB05))
```

Call:

```
lm(formula = EARNINGS ~ S + ASVABC + ASVAB05)
```

Residuals:

Min	1Q	Median	3Q	Max
-21.296	-6.722	-2.205	3.496	93.191

Coefficients:

Estimate	Std. Error	t value	Pr(> t)
----------	------------	---------	----------

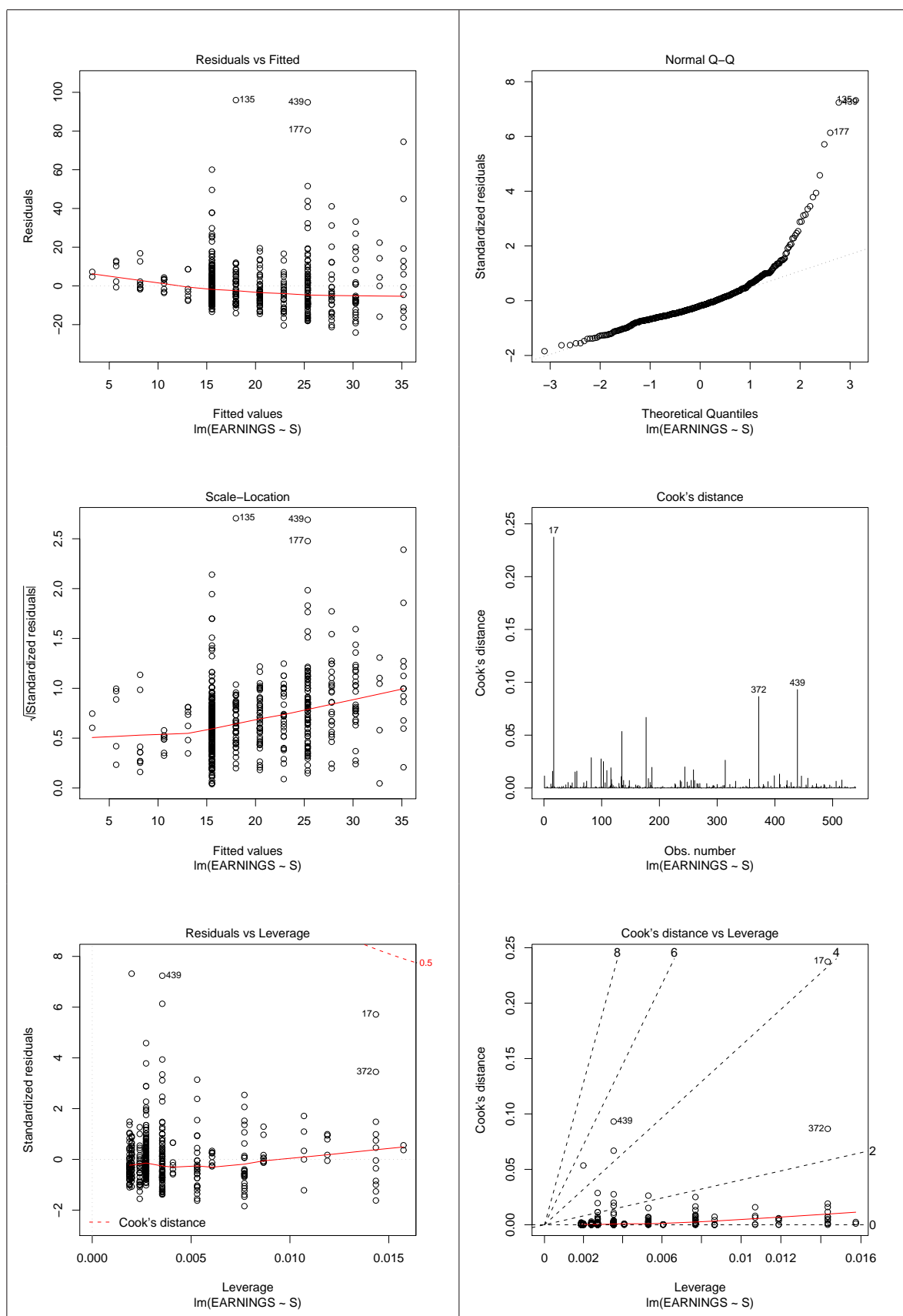


Figura 5.2: Risultati di $\text{plot}(\text{lm}(\text{EARNINGS} \sim S), i) \ i = 1, \dots, 6$

```
(Intercept) -22.64469    3.73894   -6.056 2.62e-09 ***
S            1.70182     0.27995    6.079 2.30e-09 ***
ASVABC       0.25198     0.08247    3.056 0.00236 **
ASVAB05      0.11955     0.07350    1.627 0.10442
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 12.88 on 536 degrees of freedom
Multiple R-Squared: 0.2062,    Adjusted R-squared: 0.2018
F-statistic: 46.41 on 3 and 536 DF,  p-value: < 2.2e-16
```

Utilizzando, come già fatto in precedenza, la flessibilità dell'output della funzione `lm(...)`, possiamo produrre i grafici in Figura 5.3.

Dall'output precedente, risulta chiaro che la variabile `ASVAB05` può essere rimossa dal modello, e che la variabile `ASVABC`, invece, ha una influenza di segno positivo sulla variabile risposta, anche se gli anni di scolarità sembrano rappresentare la variabile più discriminante tra quelle disponibili. Come ultimo passo, è interessante verificare se la distribuzione del salario dipende, in qualche modo, dal ceto sociale, approssimato dagli anni di scolarità della madre (`SM`) e del padre (`SF`), rispettivamente.

Si utilizza, quindi, il modello:

$$\text{EARNINGS} \sim S + \text{ASVABC} + \text{ASVAB05} + \text{SF} + \text{SM}$$

che produce il seguente output testuale

```
> summary(lm(EARNINGS ~ S + ASVABC + SF + SM))
```

```
Call: lm(formula = EARNINGS ~ S + ASVABC + SF + SM)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-22.552  -6.656  -2.198   3.336  94.598
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -20.28048    3.58735  -5.653 2.56e-08 ***
S             1.65731    0.28667   5.781 1.26e-08 ***
ASVABC        0.30509    0.07467   4.086 5.06e-05 ***
SF            0.31464    0.20773   1.515  0.130
SM           -0.18455    0.25982  -0.710  0.478
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 12.9 on 535 degrees of freedom
Multiple R-Squared: 0.2057,    Adjusted R-squared: 0.1998
```

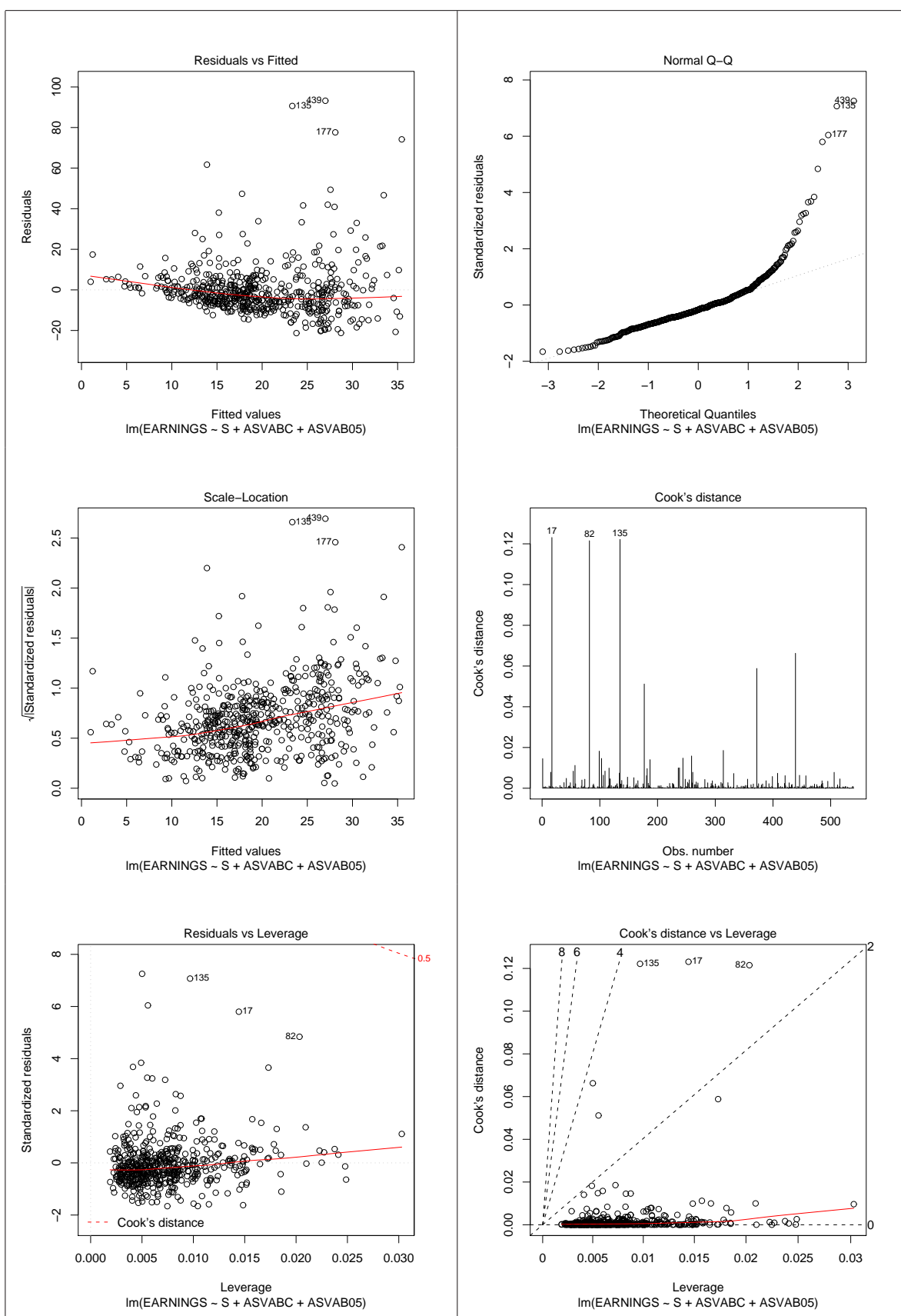


Figura 5.3: Risultati di `plot(lm(EARNINGS ~ S + ASVABC + ASVAB05),i)`, $i = 1, \dots, 6$

F-statistic: 34.64 on 4 and 535 DF, p-value: < 2.2e-16

ed i grafici corrispondenti, riportati in Figura 5.4.

Dall'analisi dell'adattamento del modello, nonché dai grafici e dalle statistiche di influenza, si può concludere che gli anni di scolarità dei genitori non rivestono alcun ruolo nel predire il valore del salario medio del rispondente, e che quindi possono essere rimossi dal modello.

5.6 Esercizi guidati

5.6.1 Valori predetti e bande di confidenza

Fino a questo punto, la linea di regressione è sempre stata presentata senza alcun riferimento all'incertezza sui valori predetti. In generale, però, è buona regola di comportamento associare alla linea di regressione una misura di incertezza, rappresentata dalle regioni di confidenza dei valori predetti in corrispondenza di valori effettivamente osservati e dei valori previsti in corrispondenza di osservazioni future.

Le regioni di confidenza possono essere costruite utilizzando il procedimento seguente. Inizialmente, si possono creare i valori predetti utilizzando il comando `predict(...)`:

```
> predict(regsemp)[1:20]
```

1	2	3	4	5	6	7	8
15.530381	15.530381	22.896342	17.985701	30.262304	25.351663	20.441022	15.530381
9	10	11	12	13	14	15	16
15.530381	15.530381	27.806983	5.709099	25.351663	30.262304	32.717624	15.530381
17	18	19	20				
35.172945	15.530381	25.351663	15.530381				

I limiti delle regioni di confidenza possono anche essi essere costruiti, per ciascuna unità statistica, utilizzando il comando `predict(...)`:

```
> predict(regsemp, int="c")
```

	fit	lwr	upr
1	15.530381	14.18458393	16.876177
2	15.530381	14.18458393	16.876177
3	22.896342	21.63268688	24.159998
4	17.985701	16.83467496	19.136727
5	30.262304	28.00039837	32.524209
6	25.351663	23.81703062	26.886295
7	20.441022	19.32146225	21.560581
8	15.530381	14.18458393	16.876177

che produce il valore della stima puntuale (`fit`) del limite superiore (`upr`) ed inferiore (`lwr`) dell'intervallo di confidenza per ciascuna unità statistica. Il parametro `int` può assumere i valori `c` e `p`, i quali rappresentano, rispettivamente, intervalli di confidenza per la retta di regressione e per i valori previsti (i cosiddetti *intervalli di predizione o predittivi*).

Ovviamente, questi limiti di confidenza possono essere rappresentati graficamente in un diagramma a dispersione, avendo però alcune cautele. Queste sono dovute al fatto che

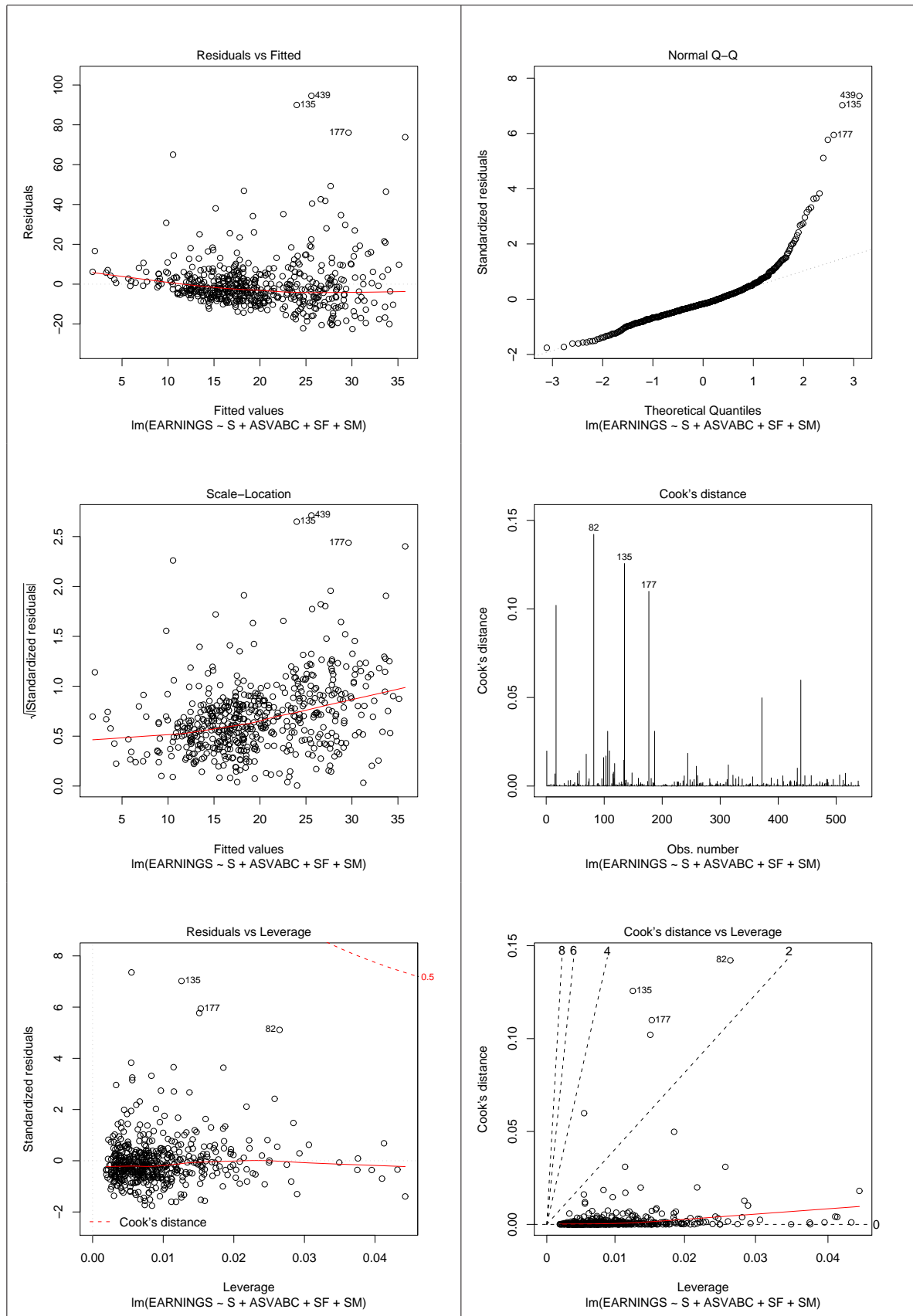


Figura 5.4: Risultati di $\text{plot}(\text{lm}(\text{EARNINGS} \sim \text{S} + \text{ASVABC} + \text{ASVAB05} + \text{SM} + \text{SF}), i), i = 1, \dots, 6$

- i valori della variabile esplicativa, S , non sono ordinati, e non ci interessa definire dei segmenti che connettano punti del piano disposti a caso,
- i limiti delle regioni di confidenza potrebbero essere esterni all'area del grafico
- dobbiamo evitare di creare cicli all'interno del grafico (utilizzeremo per questo il comando `matlines(...)` su un data frame opportunamente modificato).

A questo proposito, si crea un nuovo dataset (detto di *predizione*) con alcuni valori della variabile esplicativa (ad es. $S \in 7, \dots, 20$), ordinati in modo crescente. Poi, si costruiscono i valori dei limiti superiore ed inferiore per gli intervalli di predizione e di previsione (li chiameremo rispettivamente pc e pp). A questo punto, sfruttando il comando `matlines(...)` aggiungiamo al grafico aperto con il comando `plot(...)` le spezzate corrispondenti agli intervalli di predizione e, successivamente, le spezzate corrispondenti agli intervalli di previsione.

```
> pred.frame=data.frame(S=7:20)
> pc=predict(regsemp, int="c", newdata=pred.frame)
> pp=predict(regsemp, int="p", newdata=pred.frame)
> plot(S, EARNINGS, ylim=range(EARNINGS, pp))
> pred.S=pred.frame$S
> matlines(pred.S, pc, lty=c(1,2,2), col="black")
> matlines(pred.S, pp, lty=c(1,3,3), col="black")
```

Il grafico prodotto è riportato nella Figura 5.5.

5.6.2 Approfondimenti sulla forma funzionale

Ciò che lascia perplessi circa l'adattamento del modello lineare fin qui discusso, è la coda destra piuttosto lunga, che può essere osservata nel `qqplot` per quanto riguarda i residui, e che corrisponde ad un noto modello economico. Si osservi al proposito il plot seguente:

```
hist(EARNINGS)
```

che produce l'istogramma riportato in Figura 5.6:

Come è semplice notare, la distribuzione appare fortemente asimmetrica, con una *coda* lunga a destra, che rappresenta la presenza di una ridotta proporzione di salari orari molto più elevati della media. Proviamo a calcolare il logaritmo naturale del salario orario, utilizzando la funzione `log`:

```
LNEARN = log(EARNINGS)
```

e rappresentiamo con `hist(LNEARN)` (vedi Figura 5.7) la distribuzione corrispondente. L'istogramma mostra una distribuzione decisamente più simmetrica rispetto a quella di partenza, sicuramente più coerente con la eventuale ipotesi di normalità dei residui. Procediamo, quindi, a definire il modello di regressione lineare per la variabile trasformata $LNEARN$ ricordando che in questo caso, l'interpretazione dei coefficienti deve essere mutata in

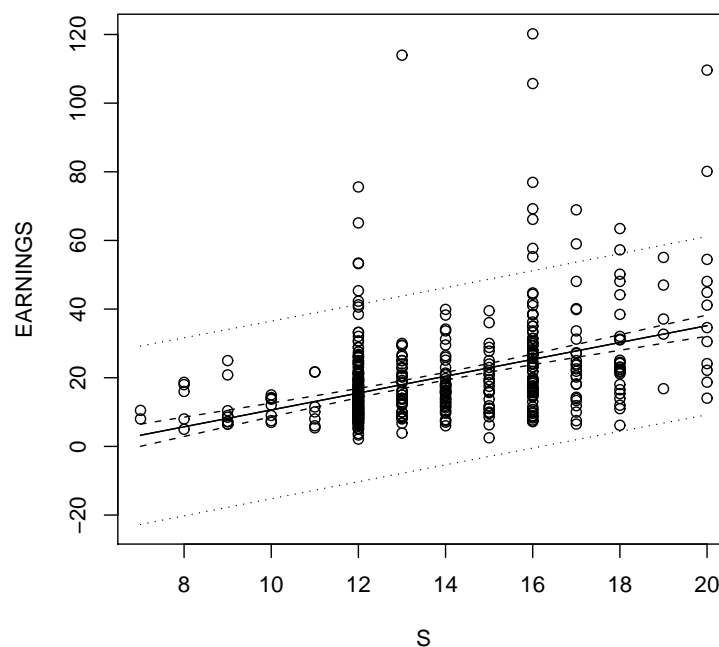


Figura 5.5: Plot della relazione lineare e dei limiti di confidenza per valori predetti e previsti

modo corrispondente. Il comando corrispondente:

```
summary(lm(EARNINGS ~ S + ASVABC + ASVAB05 + SF + SM))
```

produce l'output seguente:

Call:

```
lm(formula = llearn ~ S + ASVABC + ASVAB05 + SM + SF)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-2.080395	-0.301869	-0.004153	0.284802	1.886532

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.856051	0.151569	5.648	2.64e-08	***
S	0.067042	0.011293	5.937	5.24e-09	***
ASVABC	0.013709	0.003362	4.078	5.25e-05	***
ASVAB05	0.004678	0.002897	1.615	0.107	
SM	-0.004614	0.010225	-0.451	0.652	
SF	0.011075	0.008177	1.354	0.176	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5073 on 534 degrees of freedom

Multiple R-Squared: 0.2638, Adjusted R-squared: 0.2569

F-statistic: 38.27 on 5 and 534 DF, p-value: < 2.2e-16

ed i grafici riportati in Figura 5.8:

5.7 Estensioni del modello di regressione lineare

La regressione polinomiale fornisce uno strumento di base per descrivere forme semplici di non linearità nella relazione tra la variabile risposta e le variabili indipendenti. Casi piuttosto semplici di non linearità possono essere osservati utilizzando il diagramma a dispersione della variabile risposta secondo le covariate, se le corrispondenti nuvole dei punti tendono a mostrare un andamento che può ricordare una forma funzionale di grado superiore al primo. In questo caso, è possibile ricorrere sempre al comando `lm(...)`, specificando quali potenze della variabile indipendente devono essere prese in considerazione.

È bene ricordare che i coefficienti dei polinomi ortogonali possono, se richiesto, essere trasformati nei coefficienti delle potenze delle variabili da cui sono definiti.

```
> reg=lm(EARNINGS ~ S + I(S^2))
```

```
> summary(reg)
```

Call:

```
lm(formula = EARNINGS ~ S + I(S^2))
```

Residuals:

Min	1Q	Median	3Q	Max
-25.944	-6.944	-2.493	3.502	96.827

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	22.25089	14.92883	1.490	0.1367
S	-2.77232	2.11913	-1.308	0.1914
I(S^2)	0.18297	0.07373	2.482	0.0134 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.06 on 537 degrees of freedom

Multiple R-Squared: 0.1819, Adjusted R-squared: 0.1788

F-statistic: 59.69 on 2 and 537 DF, p-value: < 2.2e-16

Per evitare problemi di collinearità, è bene utilizzare polinomi ortogonali, in cui i termini di ordine superiore rappresentano il contributo *addizionale* delle potenze rispetto alla semplice forma lineare, quadratica, etc.

```
> reg=lm(EARNINGS ~ poly(S,2))
> summary(reg)
```

Call:

```
lm(formula = EARNINGS ~ poly(S, 2))
```

Residuals:

Min	1Q	Median	3Q	Max
-25.944	-6.944	-2.493	3.502	96.827

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	19.6362	0.5622	34.930	<2e-16 ***
poly(S, 2)1	139.0020	13.0632	10.641	<2e-16 ***
poly(S, 2)2	32.4182	13.0632	2.482	0.0134 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.06 on 537 degrees of freedom

Multiple R-Squared: 0.1819, Adjusted R-squared: 0.1788

F-statistic: 59.69 on 2 and 537 DF, p-value: < 2.2e-16

L'utilizzo della funzione *wrapper* `I(...)` assicura che la variabile S^2 sia trattata come una variabile ulteriore nel senso più proprio del termine. Invece, l'utilizzo dell'opzione `poly(...)` permette di costruire polinomi ortogonali di grado specificato. Come si può notare dall'output precedente, la significatività globale, i valori predetti ed i valori residui sono identici nei due modelli. L'interpretazione dei coefficienti è però differente.

5.8 Esercizi

Esercizio 1 Nel 1609, Galileo dimostrò matematicamente che la distanza in linea retta percorsa da un oggetto avente una data velocità iniziale è descritta da una parabola. Egli basò le sue conclusioni su di un famoso esperimento consistente nel posizionare e quindi rilasciare una sfera posta ad una certa altezza su di una rampa. Tale esperimento, come noto, fu scelto per minimizzare gli effetti dell'attrito sui risultati finali. I dati consistono delle seguenti due variabili:

- y = distanza percorsa = [253, 337, 395, 451, 495, 534, 573]
- x = altezza iniziale della sfera = [600, 700, 800, 950, 1100, 1300, 1500]

Con questi dati alla mano, e con i comandi R finora introdotti, provare che Galileo aveva visto giusto! [Hint: in altre parole, far vedere che adattando tre regressioni polinomiali – lineare, quadratica e cubica – il modello migliore è proprio quello quadratico]. Anche se non ne abbiamo parlato nel testo, per confrontare formalmente due modelli (...il quadratico ed il cubico, ad esempio...), possiamo utilizzare il comando `anova(...)`.

Esercizio 2 Il dataset `Cars93` del pacchetto `MASS` contiene dati relativi ad auto vendute negli Stati Uniti nel 1993. Fitta un modello di regressione che tenti di spiegare la variabile `MPG.city` attraverso le variabili `EngineSize`, `Weight`, `Passengers`, e `price`. Quale di queste variabili è statisticamente significativa?

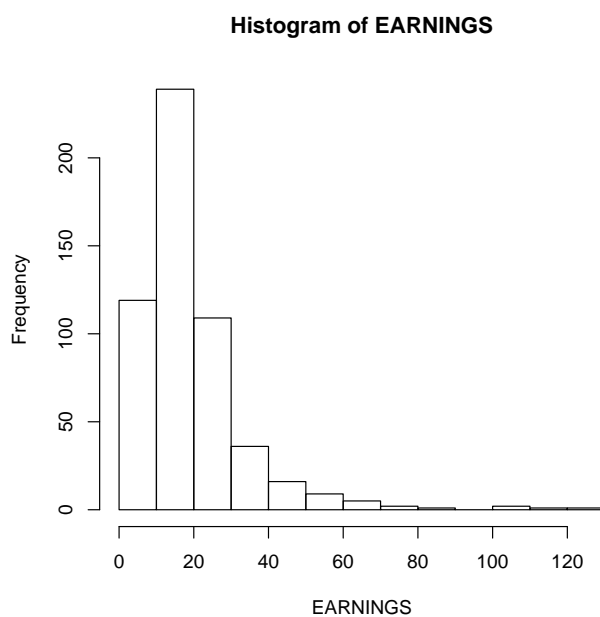


Figura 5.6: Istogramma di EARNINGS

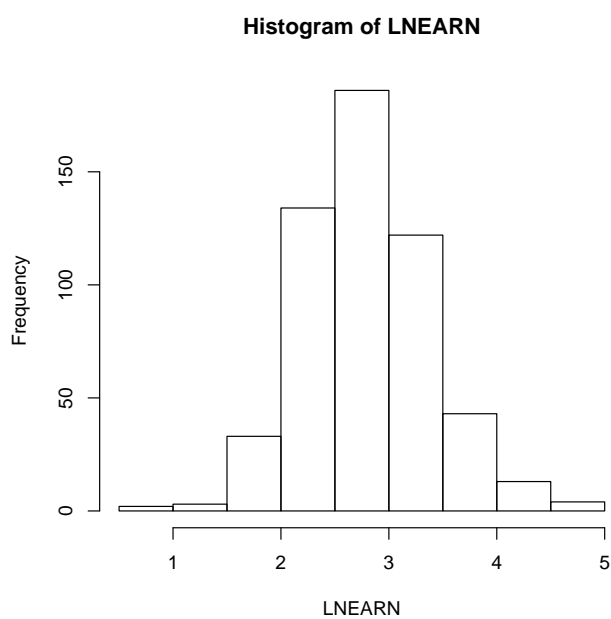


Figura 5.7: Istogramma di LNEARN

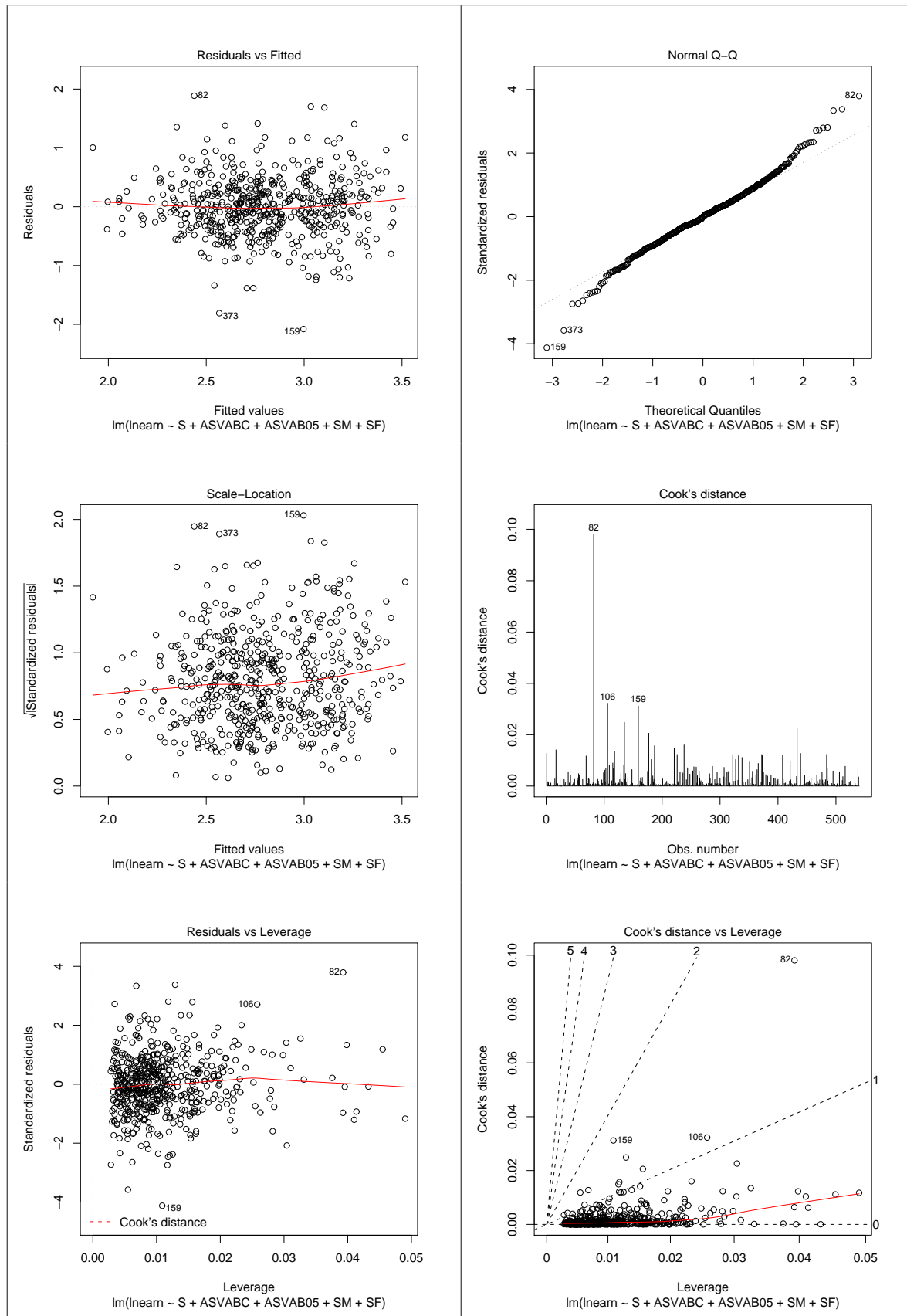


Figura 5.8: Risultati di $\text{plot}(\text{lm}(\text{EARNINGS} \sim \text{S} + \text{ASVABC} + \text{ASVAB05} + \text{SM} + \text{SF}), i), i = 1, \dots, 6$

Appendice A

Opzioni della funzione `lm(...)`

<code>formula</code>	Descrizione simbolica della relazione tra variabile risposta e covariate. Assume forme del tipo: <code>EARNINGS ~ S</code> nel caso di regressione semplice, oppure: <code>EARNINGS ~ S + ASVABC + ASVAB05</code> nel caso di regressione multipla. La variabile risposta è posta a sinistra del simbolo <code>'~'</code> , mentre le covariate sono inserite alla destra, separate da un <code>'+'</code>
<code>data</code>	(opzionale) nome del dataset che contiene le variabili specificate nel modello
<code>subset</code>	(opzionale) vettore (condizione) che specifica quali osservazione devono essere considerate
<code>weights</code>	(opzionale) vettore che fornisce i pesi da associare (in sede di stima) ad ogni unità statistica analizzata
<code>na.action</code>	(opzionale) funzione che specifica cosa accade se i dati contengono valori mancanti (o meglio <code>'NA's'</code>); il default è settato sul valore <code>'na.fail'</code> che provoca una interruzione della funzione. Ulteriori valori possibili sono <code>'na.omit'</code> che esclude le righe cui corrispondono i valori mancanti ed <code>'na.exclude'</code> simile alla precedente, a parte il fatto che i vettori dei valori predetti o residui avranno la stessa lunghezza del dataset originale con valori mancanti fissati a <code>'NA'</code> .
<code>offset</code>	(opzionale) specifica una componente numerica costante nota a priori nel predittore lineare.

Appendice B

Il dataset `eaef21.csv`

La descrizione delle variabili presenti nel dataset è riportata di seguito:

Nome variabile	Descrizione
ID	identificativo dell'unità statistica
FEMALE	genere femminile: 1 se femmina, altrimenti 0
MALE	genere maschile: 1 se maschio, altrimenti 0
AGE	età del rispondente nel 2002
EARNINGS	salario orario nel 2002
S	anni di scolarità del rispondente (completati nel 2002)
ASVAB02	ragionamento matematico (punteggio)
ASVAB03	conoscenza lessicale (punteggio)
ASVAB04	comprensione testi (punteggio)
ASVAB05	capacità di calcolo (test di velocità, punteggio)
ASVAB06	capacità di codifica (test di velocità, punteggio)
ASVABC	media di ASVAB2 (peso doppio), ASVAB3 e ASVAB4
SF	anni di scolarità del padre del rispondente
SM	anni di scolarità della madre del rispondente

Nota La batteria ASVAB (*Armed Services Vocational Aptitude Battery*) è una serie di 10 test proposti a potenziali reclute militari. Praticamente tutti i soggetti intervistati nell'ambito del NLSY79 sono stati sottoposti a questa batteria di test all'interno di un progetto sponsorizzato dal Dipartimento della Difesa con lo scopo di ottenere informazioni sulla distribuzione dei punteggi grezzi (numero di risposte corrette) ottenuti in ciascun test. In otto test dei dieci proposti, le domande partono da un livello molto semplice e procedono con livelli di difficoltà via via maggiori, con un ampio intervallo di tempo a disposizione in modo che la variabile tempo non sia discriminante. Tre di questi otto test sono test cognitivi (legati a caratteristiche di base dell'intelligenza), mentre cinque sono test di conoscenza. Le variabili ASVAB2 – ASVAB4 rappresentano i punteggi dei test cognitivi. Anche gli item più difficili di un test sono comunque piuttosto semplici, visto che la finalità della batteria ASVAB è quella di discriminare tra soggetti la cui scolarità è limitata alla "high school, che rappresenta la maggiore fonte di reclutamento delle forze armate.

Appendice C

Approfondimenti teorici

Se si definiscono le seguenti quantità:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix}$$

il modello di regressione lineare può essere scritto in forma compatta come segue:

$$\mathbf{y} = \boldsymbol{\iota}_n \beta_0 + \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$

dove $\boldsymbol{\iota}_n$ rappresenta un vettore di lunghezza n con elementi pari all'unità. Se definiamo la *matrice del disegno*

$$\mathbf{Z} = [\boldsymbol{\iota}_n, \mathbf{X}] = \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_n^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1p} \\ \vdots & & & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{bmatrix}$$

il modello di regressione lineare può essere scritto in modo compatto come:

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\beta} + \mathbf{u}$$

Per omogeneità di trattazione con i testi più utilizzati in questo ambito, nel seguito continueremo ad utilizzare la forma lineare:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$

con l'implicita ipotesi che la prima colonna della matrice \mathbf{X} è composta da elementi unitari.

Oltre alla forma funzionale adottata, che è quella lineare, il modello di regressione è caratterizzato da ulteriori ipotesi di fondo che ne accompagnano lo sviluppo. Le più importanti possono essere così riassunte:

1. $\forall i = 1, \dots, n$

$$\mathbb{E}(Y_i | \mathbf{x}_i) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$$

L'espressione precedente implica che i residui u_i sono variabili casuali a media nulla, linearmente indipendenti dalle componenti di \mathbf{X} . Tale ipotesi si basa sulla distinzione tra componente *sistemica*, rappresentata dal predittore lineare, e componente erratica ortogonale alla precedente, rappresentata dal residuo $U_i, i = 1, \dots, n$.

Quindi, l'informazione circa la variabile risposta è decomposta in due parti che, dal punto di vista concettuale, non si sovrappongono. In questo caso, le covariate X_j , $j = 1, \dots, p$ sono considerate quantità fisse *predeterminate* oppure quantità casuali *strettamente esogene*, ossia quantità che variano ma non in funzione di variabili interne al modello che stiamo analizzando.

2. $\text{Cov}(\mathbf{U}) = \sigma^2 \mathbb{I}_n$, i.e.

$$\begin{aligned} \text{Var}(u_i) &= \sigma^2, & \forall i = 1, \dots, n \\ \text{Cov}(u_i, u_l) &= 0, & \forall i \neq l = 1, \dots, n \end{aligned} \quad (\text{C.1})$$

3. la matrice \mathbf{X} ha rango pieno, pari a $p + 1$, ossia nessuna delle covariate può essere vista come combinazione lineare delle altre.
4. (opzionale) $u_i \sim N(0, \sigma^2)$, $\forall i = 1, \dots, n \Rightarrow Y_i \sim N(\mu_i, \sigma^2)$

Una volta definita la forma funzionale utilizzata per descrivere il legame tra la variabile risposta e le covariate, è necessario definire un metodo per la stima dei parametri del modello. In questo contesto, sembra ragionevole ricercare, tra tutti i possibili stimatori di $\boldsymbol{\beta}$, il vettore \mathbf{b} più *coerente* rispetto alla nuvola dei punti osservata (\mathbf{Y}, \mathbf{X}) . E' quindi necessario definire un criterio per misurare la *coerenza*, ossia la *distanza*, tra i punti osservati, di coordinate (y_i, \mathbf{x}_i) , ed i valori *predetti* dal modello, di coordinate $(\widehat{y}_i, \mathbf{x}_i)$, dove:

$$\widehat{y}_i = b_0 + \sum_{j=1}^p x_{ij} b_j = \mathbf{x}_i^T \mathbf{b}$$

La *misura* che viene utilizzata per calcolare la *coerenza* tra dati osservati e dati predetti dal modello è rappresentata dalla somma dei quadrati dei residui:

$$Q(\mathbf{b}) = \sum_{i=1}^n (y_i - \widehat{y}_i)^2 = (\mathbf{y} - \mathbf{X}\mathbf{b})^T (\mathbf{y} - \mathbf{X}\mathbf{b}) = \sum_{i=1}^n \widehat{u}_i^2.$$

La scelta di questa misura può essere motivata dal fatto che tra tutte le possibili rette (iperpiani) di regressione scegliamo quella che passa più vicino (nel senso della distanza euclidea) ai punti osservati. Inoltre, residui positivi o negativi non devono in alcun modo controbilanciarsi; ciò che interessa non è tanto (almeno in questo contesto) il segno del residuo, quanto la sua dimensione relativa. Lo stimatore \mathbf{b} dei *minimi quadrati ordinari* (ordinary least squares, OLS) rappresenta la soluzione del problema di minimo:

$$\begin{cases} \min_{\mathbf{b}} Q(\mathbf{b}) \\ \mathbf{b} \in \mathbb{R}^{(p+1)} \end{cases} \quad (\text{C.2})$$

cui corrisponde il sistema di $(p + 1)$ equazioni lineari:

$$\sum_{i=1}^n \mathbf{x}_i (Y_i - \mathbf{x}_i^T \mathbf{b}) = 0$$

note anche come *equazioni normali dei minimi quadrati ordinari*. Se la matrice del disegno \mathbf{X} ha rango pieno (pari a $p + 1$), allora $Q(\cdot)$ è una funzione strettamente *convessa* e limitata; la soluzione al problema è quindi unica:

$$\mathbf{b} = \left[\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \right]^{-1} \left[\sum_{i=1}^n \mathbf{x}_i Y_i \right] = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

Nel caso particolare di un modello di regressione lineare semplice:

$$\widehat{y}_i = b_0 + x_{i1}b_1$$

tale soluzione può essere esplicitata come:

$$\widehat{b}_1 = \frac{\sigma_{xy}}{\sigma_x^2} \quad b_0 = \bar{y} - b_1\bar{x}$$

Sfruttando la soluzione precedente, il modello di regressione lineare può quindi essere scritto come:

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{u} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} + \mathbf{u} = \mathbf{H}_X\mathbf{y} + \mathbf{U}$$

La matrice \mathbf{H}_X (spesso denominata *Hat matrix*) proietta i valori osservati \mathbf{y} sullo spazio generato dalle colonne della matrice \mathbf{X} . Tale matrice è *simmetrica* ed *idempotente* (cioè $\mathbf{H}_X\mathbf{H}_X^T = \mathbf{H}_X^T\mathbf{H}_X = \mathbf{H}_X$). I residui *stimati* sono quindi definiti da:

$$\widehat{\mathbf{u}} = \mathbf{y} - \widehat{\mathbf{y}} = [\mathbf{y} - (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}] = [\mathbf{I}_n - (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]\mathbf{y} = \mathbf{M}\mathbf{y}$$

A sua volta, la matrice $\mathbf{M} = \mathbf{I}_n - \mathbf{H}_X = [\mathbf{I}_n - (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]$ rappresenta la proiezione dei valori osservati \mathbf{y} sullo spazio ortogonale a quello generato dalle colonne della matrice \mathbf{X} . Anche questa matrice, così come la matrice \mathbf{H}_X è simmetrica ed idempotente, ossia $\mathbf{M}^T\mathbf{M} = \mathbf{M}\mathbf{M}^T$. Supponendo che i vettori \mathbf{y} siano *centrati* (ossia definiti come scarti dalla media corrispondente), le espressioni precedenti implicano la seguente scomposizione della *devianza* totale:

$$\begin{aligned} \text{Dev}(Y) &= \sum_{i=1}^n (y_i - \bar{y})^2 = \mathbf{y}^T\mathbf{y} = \mathbf{y}^T\mathbf{I}_n\mathbf{y} = \mathbf{y}^T(\mathbf{I}_n + \mathbf{H}_X - \mathbf{H}_X)\mathbf{y} = \\ &= \mathbf{y}^T\mathbf{H}_X\mathbf{y} + \mathbf{y}^T(\mathbf{I}_n - \mathbf{H}_X)\mathbf{y} = \mathbf{y}^T\mathbf{H}_X^T\mathbf{H}_X\mathbf{y} + \mathbf{y}^T\mathbf{M}^T\mathbf{M}\mathbf{y} = \\ &= \widehat{\mathbf{y}}^T\widehat{\mathbf{y}} + \widehat{\mathbf{u}}^T\widehat{\mathbf{u}} = \sum_{i=1}^n (y_i - \widehat{y}_i)^2 + \sum_{i=1}^n (\widehat{y}_i - \bar{y})^2 = \text{Dev}_M + \text{Dev}_E \end{aligned}$$

Questa uguaglianza ha un notevole significato, teorico ed applicativo; innanzitutto, notiamo che le devianze sono, per costruzione, quantità non negative. Questo implica che:

$$\text{Dev}(Y) \geq \max \{\text{Dev}_M, \text{Dev}_E\} \quad (\text{C.3})$$

Inoltre, l'informazione contenuta nella nuvola dei punti (y_i, x_i) , riassunta da $\text{Dev}(Y)$, può essere scomposta in due parti ad intersezione nulla. Una parte, Dev_M , è trattenuta, ossia *spiegata*, dal modello lineare stimato, ossia dalle informazioni disponibili sotto forma di variabili aggiuntive (x_i) . La parte residua, Dev_E , rappresenta l'informazione *persa* se si sostituiscono le informazioni originali, y_i , con i valori predetti $\widehat{y}_i, i = 1, \dots, n$, ossia se si sintetizzano le n osservazioni utilizzando le $(p+1)$ informazioni corrispondenti agli elementi del vettore \mathbf{b} . È quindi chiaro che l'adozione di un modello lineare induce una riduzione della complessità del problema da n coppie (y_i, x_i) al vettore $(p+1)$ -dimensionale \mathbf{b} . Tale decomposizione può anche essere utilizzata per costruire un indice per la valutazione della *bontà di adattamento* del modello ipotizzato ai dati osservati:

$$R^2 = \frac{\text{Dev}_M}{\text{Dev}(Y)} = 1 - \frac{\text{Dev}_E}{\text{Dev}(Y)}$$

Tale indice è noto con l'acronimo di *coefficiente di determinazione* e rappresenta la *quota* della variabilità originale spiegata dal modello stimato. Ricordando le disuguaglianze descritte nell'Equazione C.3, si può notare che il coefficiente di determinazione assume valori nell'intervallo $[0, 1]$; il valore $R^2 = 0$ corrisponde alla situazione di indipendenza *lineare* tra la variabile risposta e le covariate, mentre il

valore $R^2 = 1$ corrisponde ad una *perfetta* dipendenza lineare tra queste. Infatti, in questo ultimo caso, la distribuzione della variabile risposta è completamente *determinata* dalle covariate, e la quota di devianza spiegata dal modello è pari ad 1.

Per quanto riguarda le statistiche di influenza, la distanza di Cook e la misura di leverage sono rispettivamente definiti da:

$$D_i^{\text{Cook}} = \frac{e_{i,\text{std}}}{p} \frac{h_i}{(1 - h_i)}$$

dove $e_{i,\text{std}}$ rappresenta il residuo standardizzato, e da:

$$DF_i = \sqrt{\frac{h_{ii}}{(1 - h_{ii})}}$$

dove:

$$h_{ii} = \mathbf{x}_i(\mathbf{X}\mathbf{X})^{-1}\mathbf{x}_i$$

rappresenta l' i -esimo elemento della diagonale della matrice HAT, tale che $\widehat{y}_i = \mathbf{H}_X y_i$.

Bibliografia

- [1] Belsey, D.A., Kuh, E., Welsch, R.E. (2004), *Regression Diagnostics. Identifying Influential Data and Sources of Collinearity*, New York: John Wiley and sons.
- [2] Cook R.D., Weisberg S. (1982), *Residuals and Influence in Regression*. New York: Chapman and Hall.
- [3] Dougherty, C. (2002), *Introduction to Econometrics*. Oxford: Oxford University Press.
- [4] Maindonald, J., Braun, J. (2003), *Data Analysis and Graphics Using R. An example-based Approach*. Cambridge: Cambridge University Press.
- [5] Peracchi, F. (2001), *Econometrics*. Chichester: Wiley.