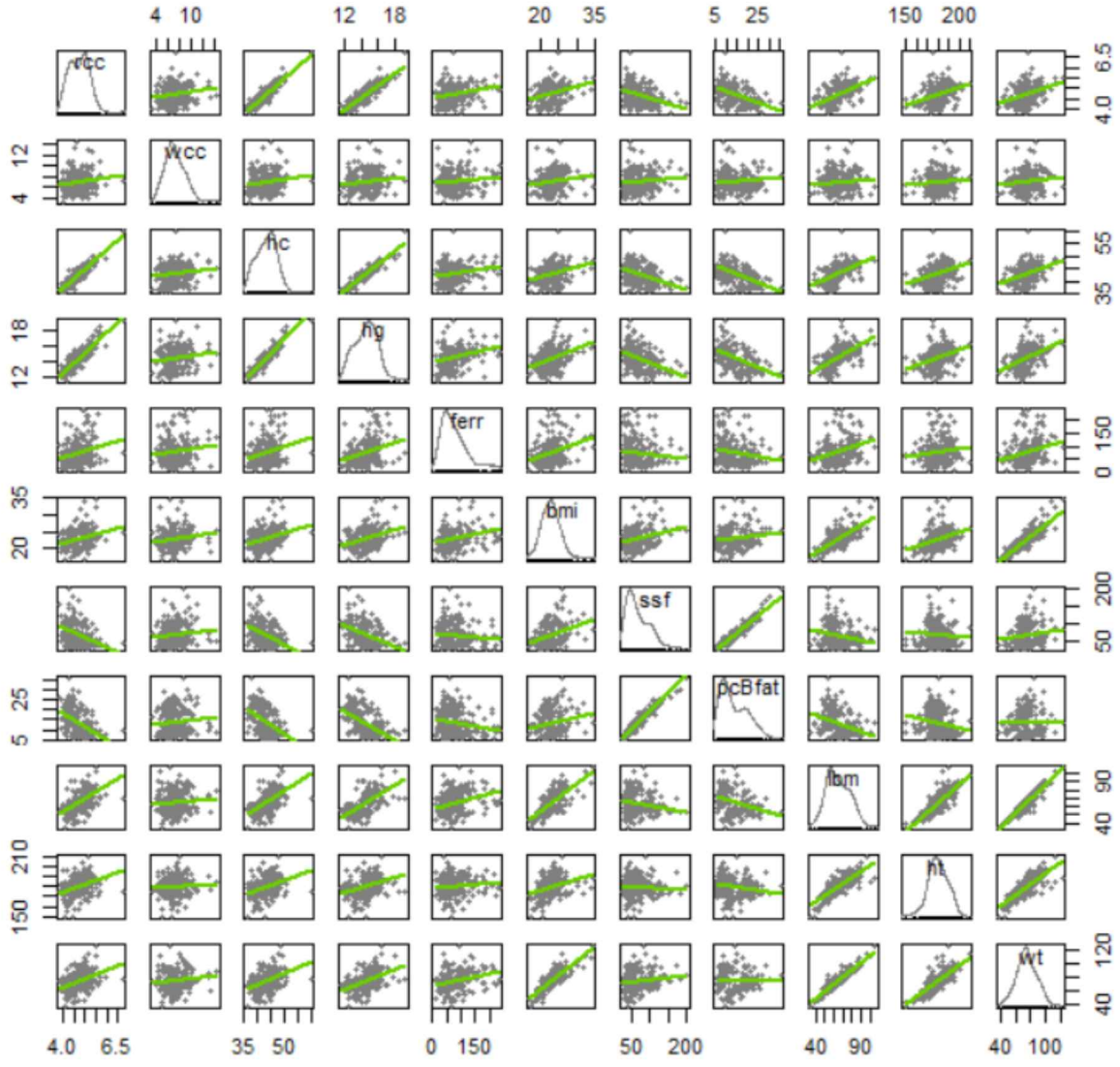
M. Besozzi

# STATISTICA
# E GRAFICA CON R

## Breve guida per imparaR facendo



Questa guida presuppone una formazione statistica di base, e non intende e non può in alcun modo sostituirla.

Le informazioni di tipo statistico contenute in questa guida possono essere utilizzate - sotto la responsabilità personale di chi le utilizza, sia per sé stesso sia per terzi - **esclusivamente a scopo didattico** per educare e formare alle tecniche di base per l'utilizzo del programma di statistica / ambiente per lo sviluppo dell'analisi statistica e grafica dei dati **R**[1].

Le informazioni di tipo biologico/medico contenute in questa guida sono inserita al solo scopo di offrire degli esempi computazionali, e non possono e non devono essere utilizzate né a scopo diagnostico, né a scopo prognostico, né a scopo terapeutico, né per qualsiasi attività che abbia un impatto diretto o indiretto sullo stato di salute di un individuo.

Nessuna responsabilità può essere imputata all'autore per danni diretti o indiretti e di qualsivoglia natura che potrebbero essere causati a sé stessi o a terzi in seguito all'utilizzo dei contenuti di questa guida o delle fonti cui esso fa riferimento a causa di errori, imprecisioni, omissioni, errate interpretazioni o qualsivoglia altro tipo di problema nei contenuti della guida.

~~Ver. 1.0 del 12/11/2018~~
~~Ver. 1.1 del 25/11/2018~~
Ver. 1.2 del 19/12/2018

**NOTA BENE**

In questo volume è presentato un estratto dei post pubblicati sul sito

https://impararfacendo.blogspot.com/

al quale si rimanda per gli argomenti che sono continuamente aggiunti e per i loro aggiornamenti.

Il download dei dati impiegati negli esempi può essere effettuato dal sito alla pagina Dati o seguendo le indicazioni riportate di seguito al punto 1.3.



[1] *The R Project for Statistical Computing*. URL consultato il 07/08/2018: https://goo.gl/MW66w1

*“Se ascolto dimentico, se vedo ricordo, se faccio capisco.”*
*(Proverbio)*

# Indice

# 1. Per iniziare con R

*"Ma a cosa serve saper calcolare le radici quadrate?"*
*(Fabio Fazio)*

*"Quando non sai fare niente, bisogna essere almeno ambiziosi."*
*(Georges Wolinsky)*

---

**R** è un ambiente di sviluppo specifico per l'analisi statistica e grafica dei dati che impiega un linguaggio di programmazione orientato agli oggetti[2]. **R** è quindi molto tecnico, e per questo è difficile superare le difficoltà iniziali, al punto che molti rinunciano ad utilizzarlo.

Vi sono tuttavia almeno quattro buone ragioni che giustificano lo sforzo necessario per imparare ad usarlo:
→ **R** è gratuito, e questo fatto è un fatto significativo, visto il costo di un pacchetto di statistica di ampiezza e profondità paragonabile;
→ **R** è disponibile per tutti e tre i sistemi operativi più diffusi, cioè per Windows, per MacOS X e per Linux, e viene costantemente aggiornato;
→ una volta installato il programma **R** (o più semplicemente **R**) con le sue funzioni statistiche e grafiche di base è possibile accedere a una raccolta di migliaia di pacchetti aggiuntivi, che offrono soluzioni di analisi statistica e rappresentazione grafica dei dati per (praticamente) qualsiasi problema;
→ se non si dovesse trovare quello che serve tra i pacchetti già disponibili (cosa oltremodo poco probabile) **R**, che è un software libero (distribuito con la licenza GNU GPL) ed è open-source[3], permette comunque di riutilizzare e adattare migliaia di funzioni già scritte e rese disponibili dai rispettivi autori e di scriverne di nuove per personalizzare al bisogno un pacchetto già esistente ovvero per creare ex-novo un pacchetto per l'analisi statistica e/o la rappresentazione grafica dei dati orientato al proprio specifico problema.

L'**obiettivo** di questa **guida** è di aiutare a superare le non poche difficoltà iniziali incontrate da coloro che intendono utilizzare **R** e lo affrontano per la prima volta, fornendo un **percorso pratico** che consente di **imparare facendo**.

Il **risultato atteso** al termine degli esercizi contenuti nella **guida** è rappresentato dalla capacità di impiegare nel giro di pochi giorni in autonomia le **tecniche di base** per l'analisi statistica e grafica dei propri dati con **R**.

Chi avrà la pazienza di seguire questo percorso, si troverà a disporre delle basi per affrontare successivamente, se lo vorrà, lo studio di **R** in modo sistematico e strutturato, secondo le proprie esigenze e necessità (va ricordato che **R** è un linguaggio di programmazione, cosa che va ben al di la dei contenuti di questa guida). E non solo troverà in **R** uno strumento straordinario, ma soprattutto scoprirà con il tempo di

---

[2] Cioè un linguaggio di programmazione basato su *oggetti software* in grado di interagire gli uni con gli altri attraverso lo scambio di *messaggi*.
[3] Per il significato di *Software libero*, di *Software open source* e di *Software di dominio pubblico* vedere: *The Free Software Foundation. GNU Operating System. Categories of free and nonfree software*. URL consultato il 18/10/2018: https://goo.gl/MqwM2Q

avere fatto un investimento strategico per il proprio sviluppo personale e professionale.

Per raggiungere l'obiettivo, oltre ovviamente a una formazione statistica di base, che questa guida non può in alcun modo fornire né sostituire, e a un interesse personale per la soluzione di problemi per i quali sono utili la elaborazione statistica e grafica dei dati, sono necessari:
→ il programma **R**;
→ gli **script** con cui elaborare i dati;
→ i **dati** da elaborare;
→ la **documentazione**.

Il **programma** e la **documentazione** di **R** sono liberamente disponibili (vedremo subito come) sul web.

Gli **script** con cui elaborare i dati e i **dati** da elaborare sono forniti con questa guida e sono il materiale che verrà impiegato per **imparare facendo**.

**Nota bene**: le indicazioni e l'iconografia riportate in questa guida fanno riferimento alla versione di **R** per Windows[4].

## 1.1. Il programma

Il programma è disponibile sul sito ufficiale del *"The R Project for Statistical Computing"*[5].

Nel capitolo
→ **2. R - il programma e gli script**
trovate:
→ come installare il programma **R** (o più semplicemente **R**) con le sue funzioni statistiche e grafiche di base
→ come iniziare ad usare la `Console di R` e per interagire con l'`Interprete di comandi di R`
→ come eseguire il vostro primo script in linguaggio **R**.

## 1.2. Gli script

Uno **script** in linguaggio **R** è costituito semplicemente da una serie di righe di testo scritte seguendo le regole previste per il linguaggio **R**, righe che sono eseguite in sequenza dal **programma R** allo scopo di elaborare i dati forniti, realizzando l'analisi statistica o l'elaborazione grafica desiderate.

In questa guida sono forniti numerosi script che possono essere eseguiti immediatamente con un semplice **copia** (copiandoli da questo testo) e **incolla** (incollandoli nella `Console di R` dove lo script verrà eseguito). Una serie di brevi note esplicative consentirà di collegare il codice **R** eseguito al risultato statistico o grafico ottenuto, al fine di **imparare facendo**.

---

[4] Per MacOS e Linux le differenze sono limitate ad aspetti ai quali le informazioni riportate possono essere facilmente adattate. Vedere a questo proposito il capitolo **2. Installing R under Unix-alikes** e il capitolo **4. Installing R under macOS** del manuale *R Installation and Administration* in versione .pdf (URL consultato il 5/11/2018: https://goo.gl/kzzyXY) o in versione .html (URL consultato il 5/11/2018: https://goo.gl/bAxtxE).

[5] *The R Project for Statistical Computing*. URL consultato il 07/08/2018: https://goo.gl/MW66w1

Oltre a potere essere eseguiti, gli script possono essere salvati sotto forma di file di testo. In questo modo è possibile realizzare una propria personale raccolta di script da riutilizzare e soprattutto da adattare al bisogno onde evitare di dovere continuamente ridigitare lo stesso codice **R** (faccenda piuttosto noiosa) ogni volta che dovete eseguire la stessa elaborazione statistica o grafica su dati differenti.

Nei capitoli
→ **4. R - analisi statistica dei dati**
→ **5. R - rappresentazione grafica dei dati**
trovate gli script con le tecniche di base, rispettivamente, per l'elaborazione statistica e per l'elaborazione grafica dei dati.

## 1.3. I dati

I **dati** relativi agli specifici problemi trattati e che sono elaborati mediante gli script forniti, possono di volta in volta essere:
→ dati immessi direttamente nel programma tramite righe di codice **R**
→ dati inclusi nei pacchetti di **R** e con questi caricati nel programma
→ dati contenuti in file predisposti appositamente e dei quali dovete fare il `download`.

Nell'ultimo caso i dati (salvo un paio di eccezioni) sono contenuti in file con **estensione** `.csv` di cui è necessario fare il `download` come segue:
→ creare sul proprio PC o notebook la cartella `C:\Rdati\`
→ effettuare il `download` dei file di dati da GitHub a questo link[6] in un unico file compresso `.zip` (fate click su `Code` quindi su `Download ZIP`);
→ salvare il file scaricato e scompattare i file in esso contenuti in `C:\Rdati\`

Gli script forniti in questa guida quando fanno riferimento a dati contenuti in file assumono sia il nome del file fornito, sia che i file si trovino nella posizione `C:\Rdati`. È ovviamente possibile cambiare tanto il nome dei file quanto la posizione nella quale i file sono stati salvati correggendo opportunamente gli script.

Nel capitolo
→ **3. R - gestione dei dati**
trovate gli script con le tecniche di base per **salvare** i dati sotto forma di file, **importare** i dati in **R**, **esportare** i dati da **R**.

## 1.4. La documentazione

La **documentazione** necessaria per raggiungere l'obiettivo, cioè per **imparare facendo**, e finalizzata all'apprendimento delle basi di **R**, è contenuta in questa stessa guida.

Ma vi sono almeno quattro altri livelli di documentazione che vi sarà necessario consultare volendo proseguire autonomamente nell'apprendimento di **R**:

[6] https://github.com/marbesoz/imparaRfacendo

→ la **documentazione generale** di **R** contenuta nel sistema di **aiuto** di **R** cui si può accedere direttamente dalla `Console di R` mentre si lavora e che viene trattata al paragrafo **2.5. Il sistema di aiuto di R**;
→ la **documentazione generale** e la **documentazione aggiuntiva** contenute nella sezione `Documentation` del sito ufficiale di **R**[7]. Qui è possibile alla voce `The R manuals`[8] scaricare, in vari formati, i manuali relativi alla versione di **R** installata. Sempre dalla voce `The R manuals` è possibile accedere alla sezione `Contributed documentation`[9] nella quale si trovano anche testi in lingua italiana. Questa documentazione vi consentirà di affrontare molte delle domande e delle esigenze di approfondimento che inevitabilmente sorgeranno durante l'uso di **R**;
→ la **documentazione specifica** fornita con ciascuno delle svariate migliaia di pacchetti aggiuntivi rispetto al programma base che è possibile scaricare dal sito ufficiale di **R**;
→ i **manuali** e i **siti** dedicati a **R**.

Va da sé che il ricorso all'uno o all'altro o a tutti questi livelli di documentazione sarà legato alle esigenze di approfondimento di volta in volta richieste dalle proprie specifiche necessità.

**Nota bene**: in **R** il separatore delle cifre decimali è il punto (.) e questa convenzione per ragioni di omogeneità viene adottata, oltre che nei file di **dati**, anche in tutto il **testo** e negli **script** riportati.

---

[7] *The R Project for Statistical Computing*. URL consultato il 07/08/2018: https://goo.gl/MW66w1
[8] URL consultato il 16/10/2018: https://goo.gl/CG7dzh
[9] Questa sezione non era più attivamente manutenuta alla data in cui la pagina è stata consultata, ma comprende comunque documentazione utile soprattutto agli inizi. URL consultato il 16/10/2018: https://goo.gl/sXq5Uc

# 2. R - il programma e gli script

*"È più facile modificare le esigenze in funzione del programma che viceversa."*
*(Arthur Bloch)*

*"La nuova versione di un programma si bloccherà non appena la vecchia è stata cancellata."*
*(Arthur Bloch)*

---

**R** è un **ambiente di sviluppo** che impiega un linguaggio di programmazione (il linguaggio **R**) orientato agli oggetti nel quale il **programma R** svolge una duplice funzione:
→ è un programma di statistica vero e proprio, autonomo e con una serie di funzioni statistiche e di rappresentazioni grafiche di base che possono essere integrate e ampliate mediante migliaia di pacchetti aggiuntivi, che offrono soluzioni di analisi statistica e rappresentazione grafica dei dati per (praticamente) qualsiasi problema;
→ è lo strumento mediante il quale si possono sviluppare, nel linguaggio di programmazione **R**, nuovi script e nuove funzioni per realizzare le elaborazioni statistiche e grafiche desiderate[10].

Il programma R è strettamente integrato con gli **script**, cioè con i blocchi di linee di codice **R** in grado di effettuare specifiche elaborazioni statistiche e grafiche, che sono letti riga per riga ed eseguiti dall'`Interprete di comandi` contenuto nel programma stesso.

[10] Quindi quando si parla genericamente di **R** si parla contemporaneamente di un *ambiente di sviluppo*, di un *linguaggio di programmazione* e di un *programma* di statistica di base che può essere enormemente ampliato con i pacchetti aggiuntivi.

## 2.1. Download e installazione del programma

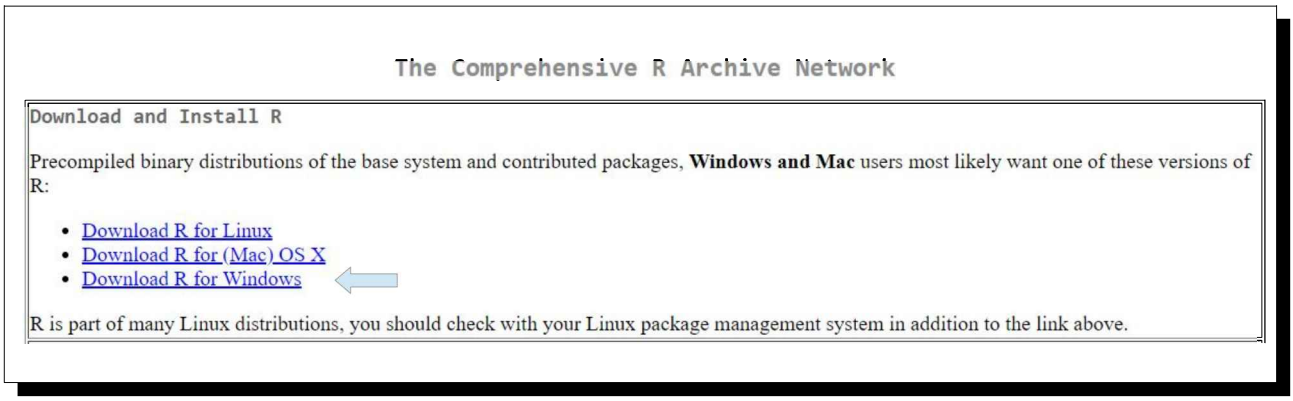
Per effettuare il `download` di **R** collegatevi alla pagina del sito ufficiale di **R**[11] facendo `click` sul link della nota a fondo pagina o sul link che trovate digitando semplicemente la lettera dell'alfabeto **R** sul motore di ricerca Google, quindi fate click su CRAN o su **download R** oppure su CRAN mirror (è indifferente):


Selezionate il server del **CRAN** (**C**omprehensive **R** **A**rchive **N**etwork) dal quale fare il `download`. Per l'Italia al momento in cui stavo componendo questo testo erano disponibili i server del GARR (**G**ruppo per l'**A**rmonizzazione delle **R**eti della **R**icerca) di Milano e dell'Università di Padova. Io ho scelto il sito sicuro (sito https[12]) dell'Università di Padova:

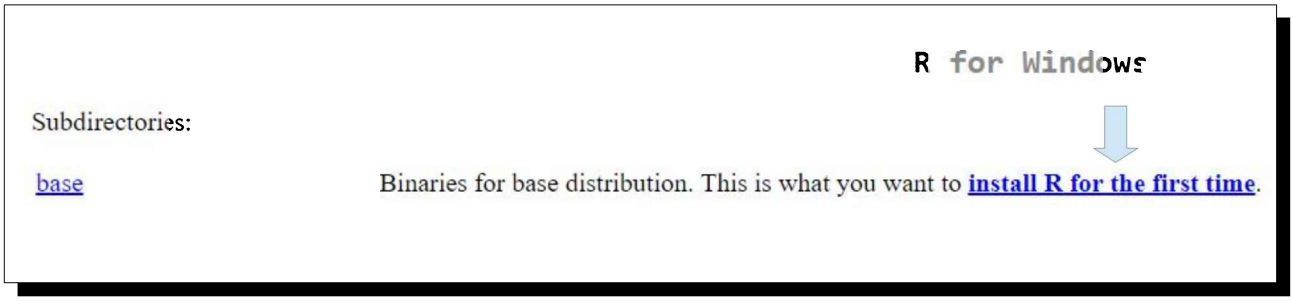

[11] *The R Project for Statistical Computing*. URL consultato il 07/08/2018: https://goo.gl/MW66w1

[12] HTTPS (Hypertext Transfer Protocol Secure) è il protocollo per la comunicazione su Internet che protegge integrità e riservatezza dei dati scambiati tra i computer e i siti. I dati inviati tramite HTTPS vengono protetti tramite il protocollo TLS (Transport Layer Security ) che fornisce tre livelli di protezione:
→ *crittografia*;
→ *integrità dei dati*;
→ *autenticazione*.
Con la *crittografia* i dati scambiati vengono criptati per proteggerli dalle intercettazioni. L'*integrità dei dati* assicura che sia rilevata qualsiasi modifica dei dati che dovesse intercorrere nel percorso tra il sito e l'utilizzatore. L'*autenticazione* dimostra che gli utenti comunicano con il sito web previsto. Un sito HTTPS viene garantito da un certificato emesso da un'autorità di certificazione: al momento dell'accesso al sito il browser valida il certificato del server controllando che la firma digitale dei certificati del server sia valida e riconosciuta da una autorità di certificazione nota, utilizzando una cifratura a chiave pubblica. Dopo questa autenticazione il browser indica la connessione come sicura mostrando generalmente l'icona di un lucchetto.

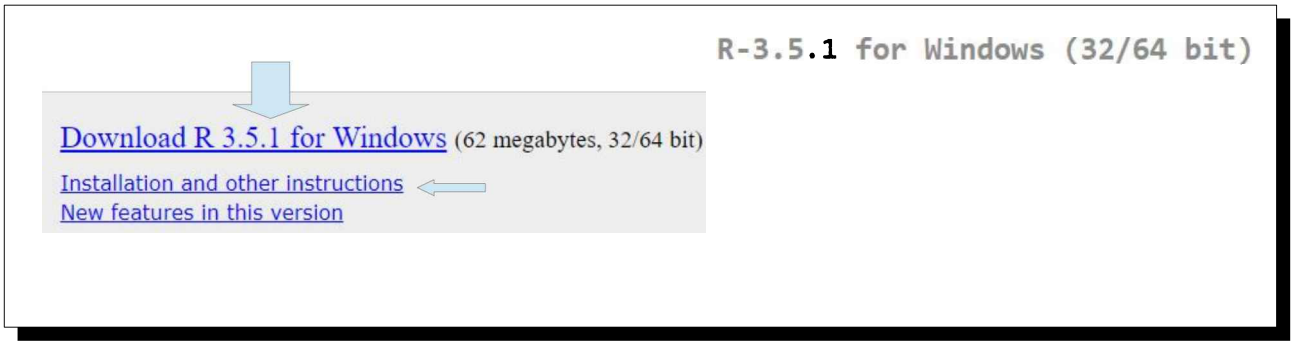
A questo punto è necessario scegliere la versione di **R** per il sistema operativo del vostro PC o notebook. Se avete Windows scegliete Download R for Windows:




Scegliete di installare R per la prima volta (install R for the first time):




A questo punto compare finalmente la pagina con il link sul quale fare `click` per effettuare il `download`, pagina che include anche una serie di informazioni dettagliate che è opportuno consultare:




Salvate sul vostro disco in un posto sicuro il file che avrà un nome del tipo `R-3.5.1-win.exe` o similare, dove invece di 3.5.1 (che era la versione di **R** disponibile al momento in cui stavo componendo questo testo e che vedete nell'immagine sopra) troverete la sigla dell'ultima versione aggiornata che avete appena scaricato. Quindi fate `doppio click` sul file scaricato e seguite le istruzioni fornite per installare **R** sul vostro PC o notebook.

**Nota bene**: si consiglia di non modificare i parametri di default proposti dal programma di installazione.

Al termine dell'installazione se avete un PC con sistema operativo a 32 bit sul desktop vi comparirà l'icona con il collegamento a **R** con la sigla `i386` che precede il numero di versione di **R**. Se avete un PC con sistema operativo a 64 bit vi comparirà l'icona con il collegamento a **R** con la sigla `x64`. Vi compariranno entrambe le icone se al momento dell'installazione avete scelto di installare entrambe le versioni di **R**, sia quella a 32 bit sia quella a 64 bit. E questa è la scelta migliore, dato che potrebbero esservi dei pacchetti che non funzionano con la versione a 64 bit e funzionano con quella a 32 bit e quindi potrete ripiegare su quest'ultima. La differenza tra le due versioni è nella velocità di elaborazione, che è maggiore per la versione a 64 bit. Ma per il tipo di script e di dati impiegati in questa guida noterete raramente, nei tempi di esecuzione, differenze importanti tra le due versioni.


Per gli approfondimenti sull'installazione, incluse le procedure per i sistemi operativi macOS e Linux, consultare il manuale **R Installation and Administration** in versione .pdf[13] o in versione html[14].

E ora possiamo finalmente iniziare ad usare **R**.

[13] R Core Team. *R Installation and Administration*. URL consultato il 5/11/2018: https://goo.gl/kzzyXY
[14] R Core Team. *R Installation and Administration*. URL consultato il 5/11/2018: https://goo.gl/bAxtxE

## 2.2. La console di **R** e l'interprete di comandi

Quando fate doppio click sull'icona di **R** per avviare il programma vi appare l'interfaccia grafica (RGui) di **R** (**Fig. 2.1**) con in alto la classica `Barra dei menù` e sotto questa una finestra aperta, la `Console di R` nella quale si svolgerà per intero l'attività pratica prevista in questa guida.

Il simbolo > nella `Console di R` è il `prompt` che indica che l'`Interprete di comandi di R` attende che scriviamo qualcosa sulla tastiera, nel suo linguaggio (del quale cerchiamo seguendo questa guida di apprendere le basi).

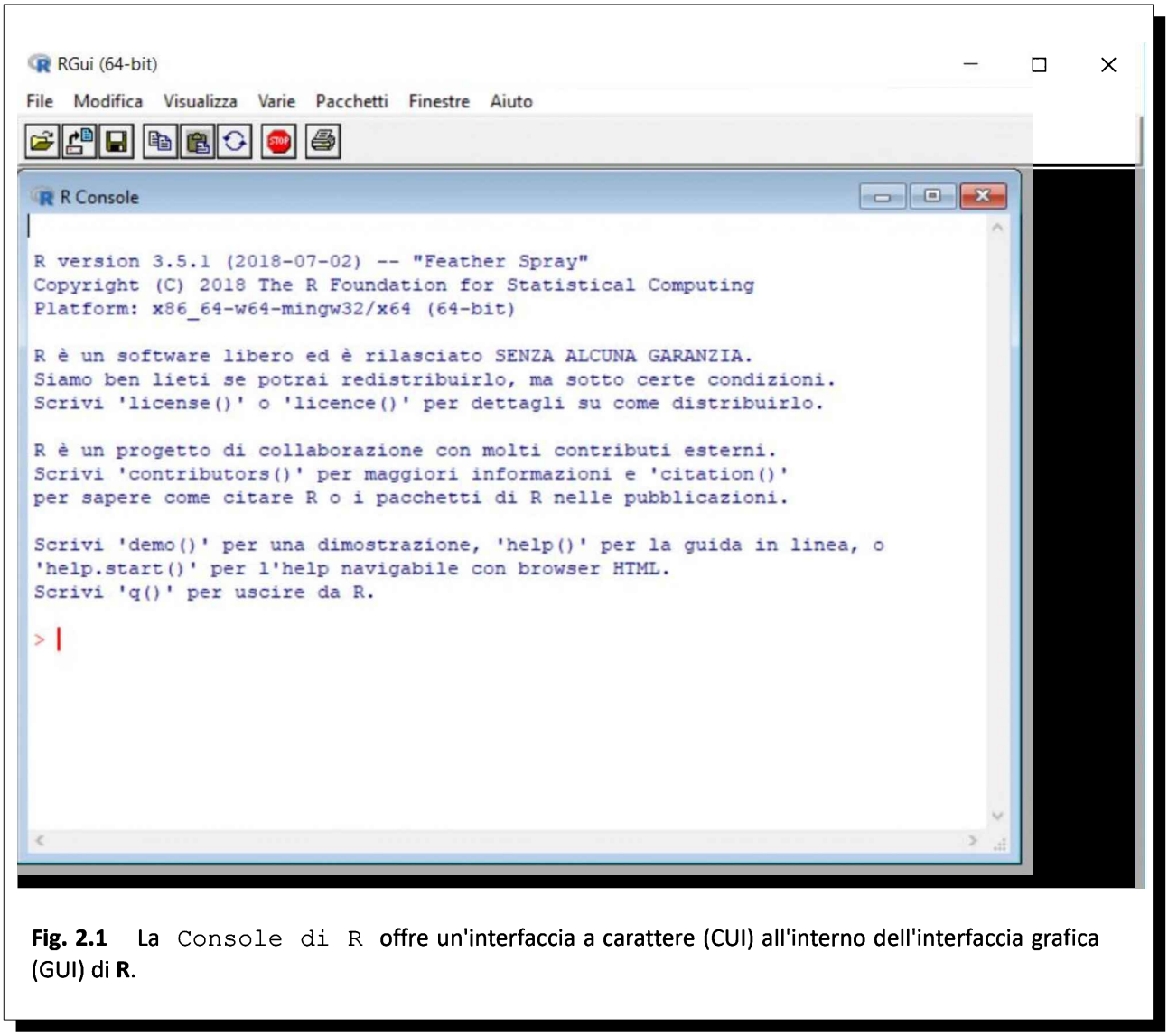



**Fig. 2.1** La `Console di R` offre un'interfaccia a carattere (CUI) all'interno dell'interfaccia grafica (GUI) di **R**.

La prima cosa che facciamo con **R** è molto semplice:

→ al `prompt` di **R** scrivete nella `Console di R`
**demo()**
quindi premete ↵`Invio`.

Si aprirà una finestra con un elenco delle dimostrazioni disponibili nel pacchetto base appena installato (**Fig. 2.2**):

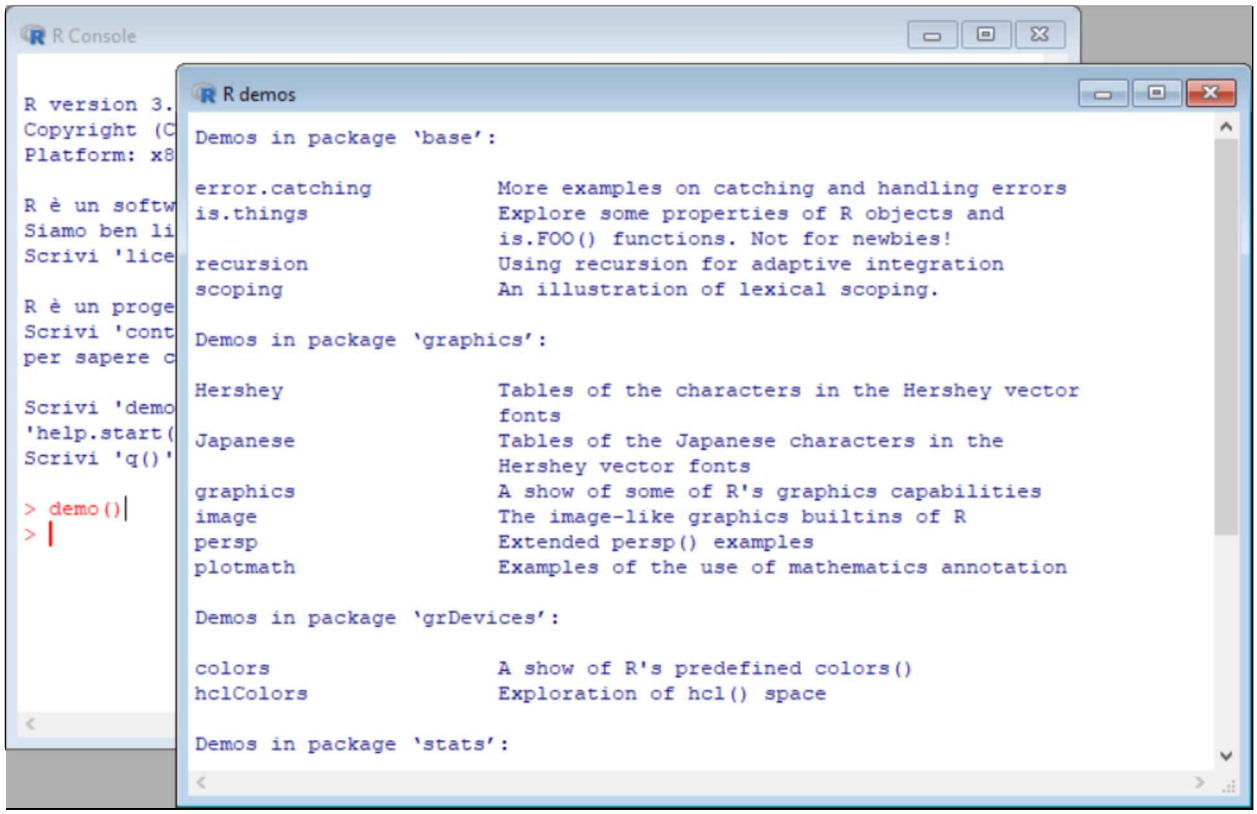



**Fig. 2.2** Digitando il comando **demo()** nella `Console di R` compare un elenco delle dimostrazioni disponibili nel pacchetto base.

Ora chiudete la finestra che si è aperta quindi:

→ al `prompt` di **R** scrivete nella `Console di R`
**demo(graphics)**
e premete ↵`Invio`.

Se seguite le istruzioni che compaiono nella `Console di R` potrete apprezzare una dimostrazione delle capacità grafiche della installazione base. Come vedremo nei pacchetti aggiuntivi disponibili per **R**[15] troverete una infinità di altre possibilità di elaborazione grafica. Per terminare chiudete l'ultima finestra che è rimasta aperta e tornate con un `click` alla finestra della `Console di R` e all'`Interprete di comandi di R`.

**R** fornisce una interfaccia grafica o GUI (`Graphical User Interface`) limitata alla gestione delle operazioni generiche, con i menù `File`, `Modifica`, `Visualizza`, `Varie`, `Pacchetti`, `Finestre`, `Aiuto` (**Fig. 2.1**), mentre la `Console di R`, la finestra nella quale si svolge tutto il lavoro, fornisce solo una semplice interfaccia a carattere o CUI (`Character User Interface`) nella quale le istruzioni per effettuare le elaborazioni statistiche e grafiche dei **dati** sono inserite digitandole a mano lettera per lettera, parola per parola e riga per riga (in realtà vedremo come aiutandosi con gli **script** la cosa

[15] Vedere: **2.6. Pacchetti aggiuntivi di statistica e grafica**.

non sarà tragica come potrebbe sembrare ora).

L'interfaccia a carattere della `Console di R` ci riporta agli interpreti di comandi dei primordi dell'informatica. E potrebbe apparire obsoleta oggi che ci siamo abituati ad utilizzare esclusivamente interfacce grafiche (ma non lo è, viene ancora ampiamente utilizzati dai professionisti informatici, come programmatori e amministratori dei sistemi informativi, anche perché molte funzioni dei sistemi operativi sono gestibili molto meglio o addirittura solamente con una interfaccia a carattere, che non solo è più leggera ma è anche molto più affidabile. Se viaggiate in aereo, sappiate che in avionica sono impiegati, per la vostra sicurezza, computer che utilizzano esclusivamente interfacce a carattere).

Ovviamente anche per **R** sono disponibili programmi che consentono all'utente di lavorare impiegando una GUI[16] nella fase di elaborazione statistica e grafica dei dati, un'interfaccia "evoluta" simile a quella degli altri programmi di statistica in ambiente Windows, macOS e Linux. Ma non li utilizzeremo: lavorare direttamente sulla formulazione in chiaro delle istruzioni impiegando la `Console di R` consente di esercitarsi con sintassi e regole del linguaggio **R**, e risulta essenziale per **imparare facendo**.

**Nota bene**: in questa guida viene utilizzata esclusivamente l'interfaccia a carattere della `Console di R`.

---

[16] Sono due i programmi che forniscono per **R** una GUI anche nella fase di elaborazione statistica e grafica del dati, rispettivamente **Rstudio** e **R Commander**, e ad essi si rimandano gli interessati. **R Commander** è in genere ritenuto il programma più stabile.

## 2.3. Come eseguire uno script

Uno **script** in linguaggio **R** è costituito semplicemente da una serie di righe di testo scritte seguendo le regole sintattiche previste per il linguaggio, righe che sono eseguite in sequenza dall'`Interprete di comandi di R` allo scopo di elaborare dei **dati**.

Gli script forniti con questa guida sono riportati con queste convenzioni:

→ il codice **R** in carattere normale e colore nero e preceduto dal simbolo # è un semplice promemoria, un commento, e non viene eseguito dall'`Interprete di comandi`;
→ il codice **R in grassetto e colore rosso** è il codice che viene eseguito.

Per il testo che compare nella finestra della `Console di R` valgono queste convenzioni:

→ il codice **R** viene riportato, commenti inclusi, in `colore rosso`;
→ i **risultati** dell'elaborazione da parte del programma sono riportati in `colore blu`.

Ed eccoci al nostro primo script, riportato qui sotto. Per eseguirlo dovete selezionate il testo dello script dal primo all'ultimo # (inclusi) e fare click su `Modifica >> Copia` quindi:
→ nella `Console di R` fate click[17] su `Modifica >> Incolla` e premete ↵`Invio`;
→ oppure in alternativa fate click sulla finestra della `Console di R` quindi nella finestra che si apre facendo click con il tasto destro del mouse selezionate `Incolla` e infine premete ↵`Invio`.

Potete anche fare in quest'alto (e terzo) modo:
→ selezionate il testo dello script dal primo all'ultimo # (inclusi) e fate `ctrl-C` (tenendo premuto il tasto `ctrl` premete il tasto con la lettera `C`);
→ fate click sulla finestra della `Console di R` quindi fate `ctrl-V` (tenendo premuto il tasto `ctrl` premete il tasto con la lettera `V`) e infine premete ↵`Invio`.

```
# CALCOLA LA MEDIA E LA DEVIAZIONE STANDARD DEI VALORI 8, 6, 11, 7
#
# viene generato l'oggetto mydata che contiene i valori numerici 8,6,11,7
mydata <- c(8,6,11,7)
# calcola la media dei valori numerici contenuti nell'oggetto mydata
mean(mydata)
# calcola la deviazione standard dei valori numerici contenuti nell'oggetto mydata
sd(mydata)
#
```

Le righe di codice che avete copiato e che avete incollato nella `Console di R`, dopo avere digitato ↵`Invio` sono eseguite una ad una in sequenza dall'`Interprete di comandi di R`.

Nella finestra della `Console di R` il codice inserito viene riportato, commenti inclusi, in `colore rosso` e i risultati dell'elaborazione da parte del programma sono riportati in `colore blu`. E questo è il risultato che compare nella `Console di R` del nostro primo script:

[17] Per semplicità **click** qui e altrove sta per **click con il tasto sinistro del mouse**.

```
> # CALCOLA LA MEDIA E LA DEVIAZIONE STANDARD DEI VALORI 8, 6, 11, 7
> #
> # viene generato l'oggetto mydata che contiene i valori numerici
8,6,11,7
> mydata <- c(8,6,11,7)
> # calcola la media dei valori numerici contenuti nell'oggetto mydata
> mean(mydata)
[1] 8
> # calcola la deviazione standard dei valori numerici contenuti
nell'oggetto mydata
> sd(mydata)
[1] 2.160247
```

Già dai commenti inseriti nello **script** è possibile capire cosa accade:
→ la funzione **mean()** calcola sui dati (**8,6,11,7**) contenuti nell'oggetto **mydata** la media dei valori, che risulta uguale a 8
→ la funzione **sd()** calcola sui dati (8,6,11,7) contenuti nell'oggetto **mydata** la deviazione standard dei valori, che risulta uguale a 2.160247

Aggiungo qui peraltro alcuni chiarimenti importanti sugli elementi del linguaggio **R** che compaiono in questo breve script per consentire a chi è completamente digiuno di **R** di affrontare con maggior consapevolezza gli altri script che seguiranno e che formano il nucleo pratico della guida:
→ **mydata** è un **oggetto**;
→ **<-** è l'**operatore di assegnamento**;
→ **c()**, **mean()** e **sd()** sono **funzioni;**
→ il **mydata** contenuto tra parentesi è l'**argomento** delle funzioni **mean()** e **sd()** che qui assumono la forma **mean(mydata)** e **sd(mydata).**

Il nome **mydata** attribuito all'**oggetto** è arbitrario, avremmo potuto chiamarlo anche **imieidati**, o in qualsiasi altro modo purché rimanga un identificativo **univoco** dell'oggetto di cui ci vogliamo servire.

La **funzione c()** combina gli elementi numerici (**8,6,11,7**) in essa inseriti come **argomento** in un **array** (o **vettore**), cioè in un contenitore unico formato da celle in sequenza ciascuna della quali contiene un dato. In informatica si parla di **array** nel caso di dati monodimensionali disposti su una sola riga come questi:

| | | | |
|---|---|---|---|
| 8 | 6 | 11 | 18 |

Nel caso di dati bidimensionali disposti su più righe e colonne si parla di **matrice** quando i dati sono solo numerici

| | | | |
|---|---|---|---|
| 8 | 6 | 11 | 18 |
| 6 | 9 | 18 | 15 |
| 7 | 8 | 15 | 19 |

e si parla invece di **tabella** quando i dati oltre ai numeri includono testo e/o operatori logici:

| | | | |
|---|---|---|---|
| M | 7 | 9 | VERO |
| F | 3 | 12 | VERO |
| F | 5 | 10 | FALSO |

L'**operatore di assegnamento** **<-** assegna all'**oggetto** **mydata** il contenuto dell'**array** generato dalla funzione **c()**. Se nella finestra della `Console di R` digitate **help(c)** si apre la finestra di aiuto con la documentazione della funzione **c()**. All'interno della parentesi graffa {} compare il nome del pacchetto nel quale è inclusa la funzione, nel caso specifico la funzione **c()** è inclusa nel pacchetto **base** che viene installato al momento dell'installazione di **R**. Queste regole valgono per tutte le funzioni di **R**.

c {base} R Documentation

**Combine Values into a Vector or List**

**Description**

This is a generic function which combines its arguments.

The default method combines its arguments to form a vector ...

Nell'espressione **mean(mydata)** la funzione **mean()** utilizza come argomento l'oggetto **mydata** che contiene i valori **8,6,11,7** per calcolarne la media. Per aprire la finestra di aiuto nella quale viene riportata la documentazione della funzione **mean()**, digitate **help(mean)** nella `Console di R`.

Nell'espressione **sd(mydata)** la funzione **sd()** utilizza come argomento l'oggetto **mydata** che contiene i valori **8,6,11,7** per calcolarne la deviazione standard. Per aprire la finestra di aiuto nella quale viene riportata la documentazione della funzione **sd()**, digitate **help(sd)** nella `Console di R`.

Semplice no? Per il resto della **guida** il lavoro sarà di portare alle estreme conseguenze questa logica impiegando **funzioni** via via più complesse, ciascuna con i propri specifici **argomenti**, all'interno di **script** sempre più articolati, e **imparare facendo**.

**Nota bene**: per aprire la finestra di aiuto nella quale viene riportata la documentazione di una qualsiasi funzione digitare **help(***nomedellafunzione***)** nella `Console di R.`

## 2.4. Come salvare uno script

Riprendiamo ora il nostro primo script

```
# CALCOLA LA MEDIA E LA DEVIAZIONE STANDARD DEI VALORI 8, 6, 11, 7
#
# viene generato l'oggetto mydata che contiene i valori numerici 8,6,11,7
mydata <- c(8,6,11,7)
# calcola la media dei valori numerici contenuti nell'oggetto mydata
mean(mydata)
# calcola la deviazione standard dei valori numerici contenuti nell'oggetto mydata
sd(mydata)
#
```

con l'obiettivo di salvarlo, commenti inclusi, e poterlo riutilizzare in un secondo tempo, eventualmente modificandolo per adattarlo a nuove esigenze. Ci sono praticamente quattro modi per farlo. Ma tutti prevedono di salvare il codice **R** in un file di testo[18].

Il primo modo prevede di utilizzare in Windows il `Blocco note` (**Fig. 2.3**) che si trova in `Accessori Windows`.

Dopo avere incollato il codice nel `Blocco note` selezionate `File >> Salva con nome` e salvatelo (per esempio) con il nome `Media e ds.txt` (l'estensione `.txt` viene aggiunta automaticamente dal `Blocco note`).

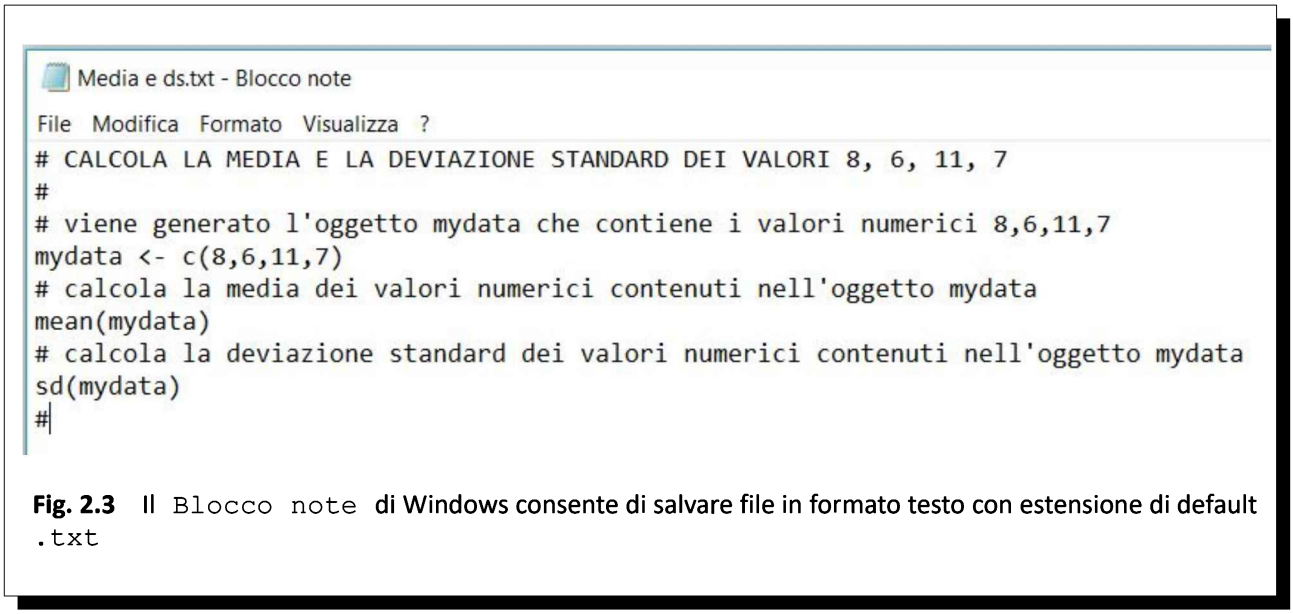



**Fig. 2.3** Il `Blocco note` di Windows consente di salvare file in formato testo con estensione di default `.txt`

[18] Cioè in un file nel quale i caratteri possono essere scritti (e successivamente essere riletti) in chiaro nello stesso modo in cui sono scritti su (e possono essere riletti da) un foglio di carta scritto con una macchina da scrivere, impiegando un set di caratteri/codice universalmente riconosciuto (vedere: **A1. Codifica dei caratteri ASCII, ANSI, Unicode e UTF**).

Il secondo modo prevede di utilizzare l'`Editor di R` (**Fig. 2.4**) che si apre dalla `Console di R` selezionando `File >> Nuovo script`.

Incollate il codice copiato nell'`Editor di R` quindi selezionate `File >> Salva con nome` e salvatelo (per esempio) con il nome `Media e ds.R` (l'estensione `.R` viene aggiunta automaticamente dall'`Editor di R`).

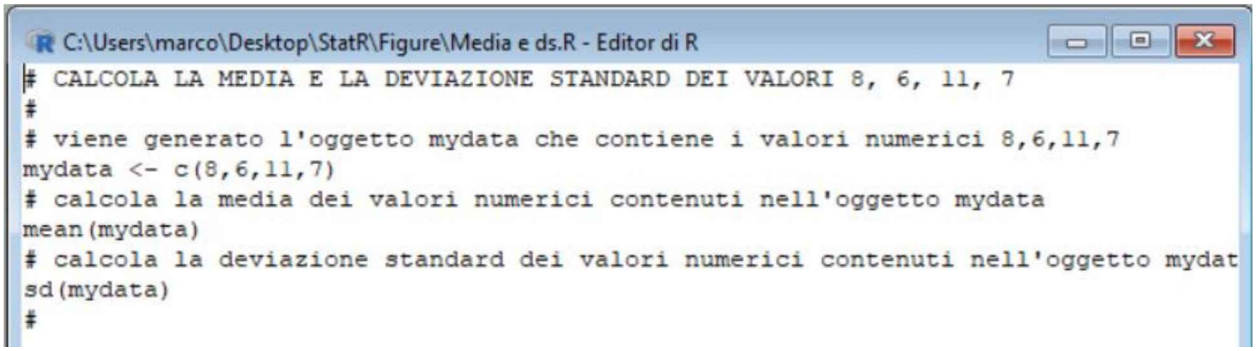



**Fig. 2.4** L'`Editor di R` consente di salvare file in formato testo con estensione di default `.R`

Notate quindi che l'`Editor di R` aggiunge automaticamente al nome del file l'estensione `.R` mentre il `Blocco note` di Windows aggiunge automaticamente al nome del file l'estensione `.txt`: ma si tratta di file di testo perfettamente identici tanto che è possibile aprire i file `.R` con il `Blocco note` di Windows e aprire i file `.txt` con l'`Editor di R`.

Il terzo modo prevede di impiegare un qualsiasi altro editor di test, mentre il quarto modo prevede di impiegare il wordprocessor al quale siete abituati e salvare i file, invece che nel suo formato specifico, in formato testo (tutti i wordprocessor hanno questa opzione per il salvataggio dei file).

Quale di questi metodi usare allora? Personalmente uso il `Blocco note di Windows` ma in realtà non fa nessuna differenza: è possibile salvare gli script con il programma che si trova più comodo e più pratico utilizzare. La sola cosa che conta è salvarli in formato testo perché questo è il solo formato con cui lo script può essere riconosciuto ed essere eseguito in **R**.

**Nota bene**: per la gestione degli script riportati nella guida è possibile utilizzare indifferentemente non solo un qualsiasi editor di testo ma anche un qualsiasi wordprocessor, purché i file siano salvati in formato testo.

## 2.5. Il sistema di aiuto di **R**

Se nella `Barra dei menù` di **R** selezionate l'opzione `Aiuto` compare un menù a tendina (**Fig. 2.5**) nel quale trovate varie opzioni per accedere alla **documentazione** di **R**.

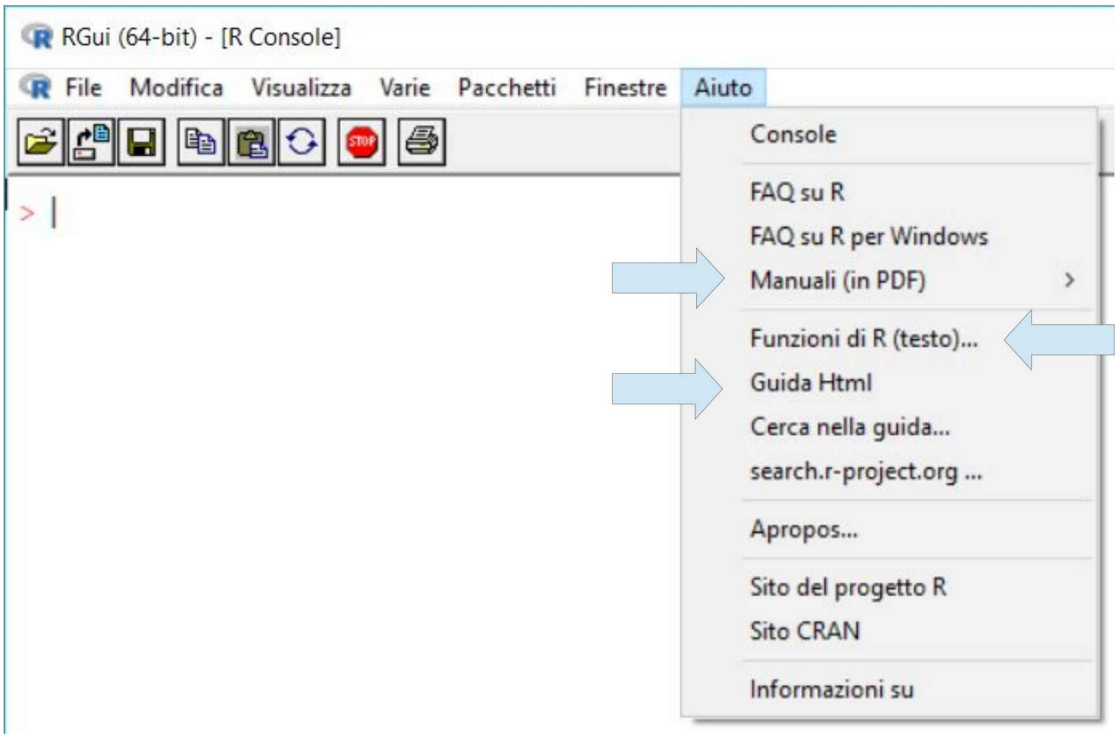



**Fig. 2.5** Il menù a tendina che compare selezionando `Aiuto` dalla barra dei menù di **R** offre la possibilità di accedere alla documentazione di **R** sia in formato `.pdf` selezionando `Manuali (in PDF)` sia in modalità interattiva selezionando `Guida Html`.

Se continuate a preferire la carta stampata alla voce `Manuali (in PDF)` del menù `Aiuto` (**Fig. 2.6**) trovate nelle prime tre posizioni i tre manuali più importanti
→ **An Introduction to** R[19]
→ **R Reference**[20]
→ **R Data Import/Export**[21]
che potete scaricare e al bisogno stampare (attenzione, il manuale di riferimento è molto voluminoso).

Molte delle informazioni che interessano, come ad esempio la documentazione delle funzioni **c()**, **mean()**, **sd()** riportate sopra, si possono trovare proprio nel manuale di riferimento di **R** (**Fig. 2.6**).

[19] *An Introduction to R.* URL consultato il 17/10/2018: https://goo.gl/PnjsBW
[20] *R: A Language and Environment for Statistical Computing, Reference Index*. URL consultato il 17/10/2018: https://goo.gl/kes91X
[21] *R Data Import/Export*. URL consultato il 17/10/2018: https://goo.gl/8mzH4L

Un metodo più generale per trovare la documentazione delle funzioni è quello di selezionare l'opzione `Funzioni di R (testo)...` (**Fig. 2.5**) che apre una finestra (**Fig. 2.7**) nella quale è possibile inserire (senza le parentesi) il nome della funzione della quale si cerca la documentazione. Nell'esempio riportato la ricerca riguarda la documentazione della funzione **mean()**.

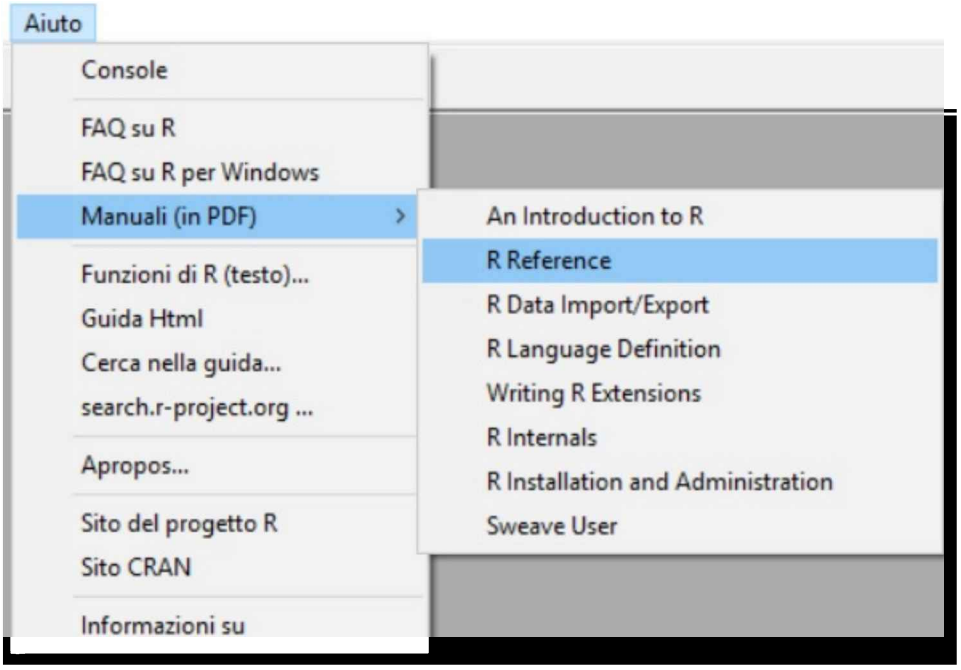



**Fig. 2.6** Nel manuale di riferimento di **R** in formato `.pdf` (`R Reference`) si trova tra le altre cose la documentazione completa delle funzioni di **R** incluse nell'installazione base.

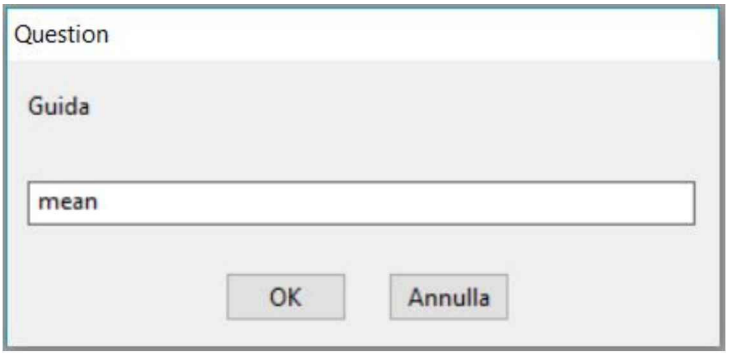



**Fig. 2.7** La finestra che consente di ricercare la documentazione delle funzioni di **R**. Nell'esempio viene effettuata la ricerca della documentazione della funzione **mean()**.

Come già anticipato si può trovare la documentazione delle funzioni anche digitando **help(*nomedellafunzione*)** nella `Console di R`.

Se ai manuali in PDF preferite i manuali interattivi potete trovare il loro equivalente nel menù `Aiuto` selezionando l'opzione `Guida Html` (**Fig. 2.5**) che apre una pagina contenente varie opzioni (**Fig. 2.8**).

Nella sezione `Manuals` di questa pagina vi saranno di aiuto principalmente la guida introduttiva `An Introduction to R` e il manuale con tutte le modalità per importare e esportare i dati `R Data Import/Export`.

Nella sezione `Reference` alla voce `Packages` trovate la documentazione dei pacchetti che avete installato sul vostro PC o notebook, mentre alla voce `Search Engine & Keywords` si apre una finestra nella quale potete ricercare con varie modalità l'argomento di vostro interesse all'interno della documentazione di **R**.

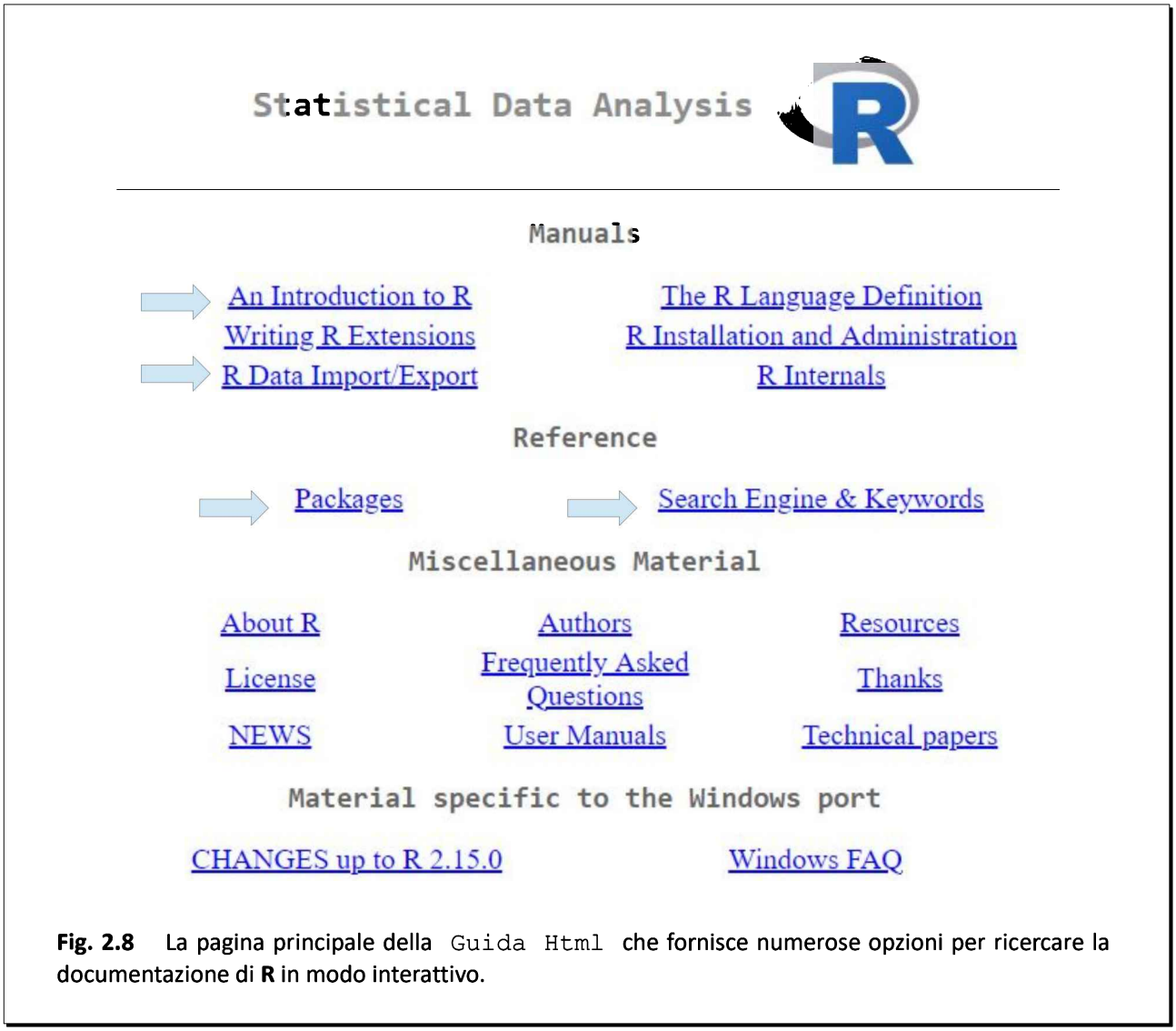



**Fig. 2.8** La pagina principale della `Guida Html` che fornisce numerose opzioni per ricercare la documentazione di **R** in modo interattivo.

## 2.6. Pacchetti aggiuntivi di statistica e grafica

Come detto **R** include una serie di funzioni statistiche e grafiche di base. Se vi collegate al **CRAN**[22] nella sezione `Software` alla voce `Packages` (**Fig. 2.9**) si apre una pagina dalla quale è possibile accedere all'elenco dei pacchetti di statistica e grafica aggiuntivi disponibili (13035 nel momento in cui la pagina è stata consultata), ciascuno orientato alla risoluzione di un problema specifico.

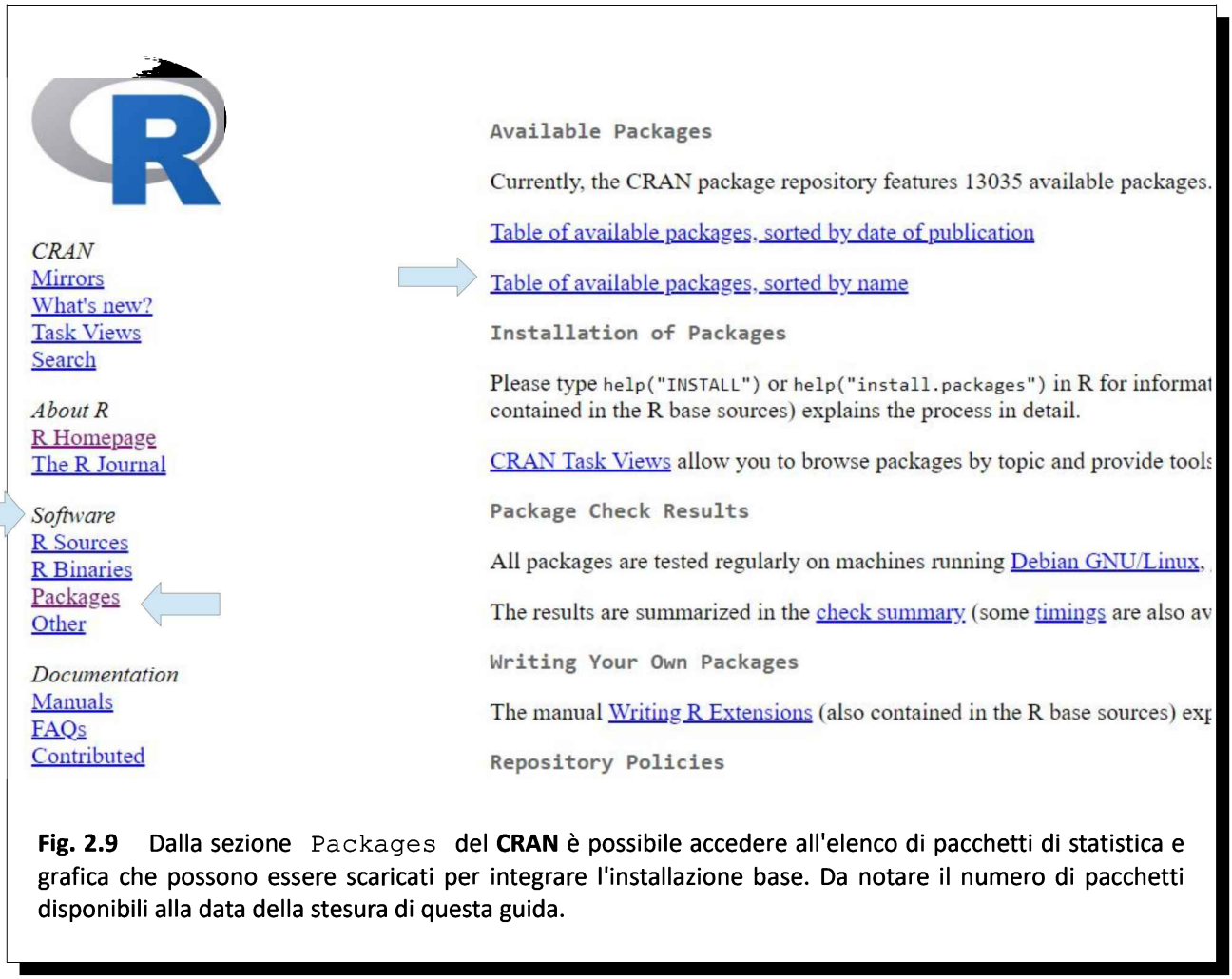



**Fig. 2.9** Dalla sezione `Packages` del **CRAN** è possibile accedere all'elenco di pacchetti di statistica e grafica che possono essere scaricati per integrare l'installazione base. Da notare il numero di pacchetti disponibili alla data della stesura di questa guida.

Se ora fate click sul link alla pagina contenente la `Table of available packages, sorted by name` (**Fig. 2.9**) potete cercare il pacchetto che vi interessa e accedere al suo `Reference manual` l'unico documento che contiene la descrizione completa di tutte le funzioni fornite con il pacchetto e degli argomenti che consentono di utilizzarle in modo appropriato.

Per scaricare un pacchetto e installarlo sul vostro PC o notebook dovete (**Fig. 2.10**):
→ selezionare dalla `Barra dei menù di R` l'opzione `Pacchetti`
→ selezionare dal menù a tendina che compare `Installa pacchetti...`

[22] **C**omprehensive **R** **A**rchive **N**etwork. Nell'esempio (**Fig. 2.11**) è stato utilizzato il sito https dell'Università di Padova. URL consultato il 10/08/2018: https://goo.gl/pn7ecp

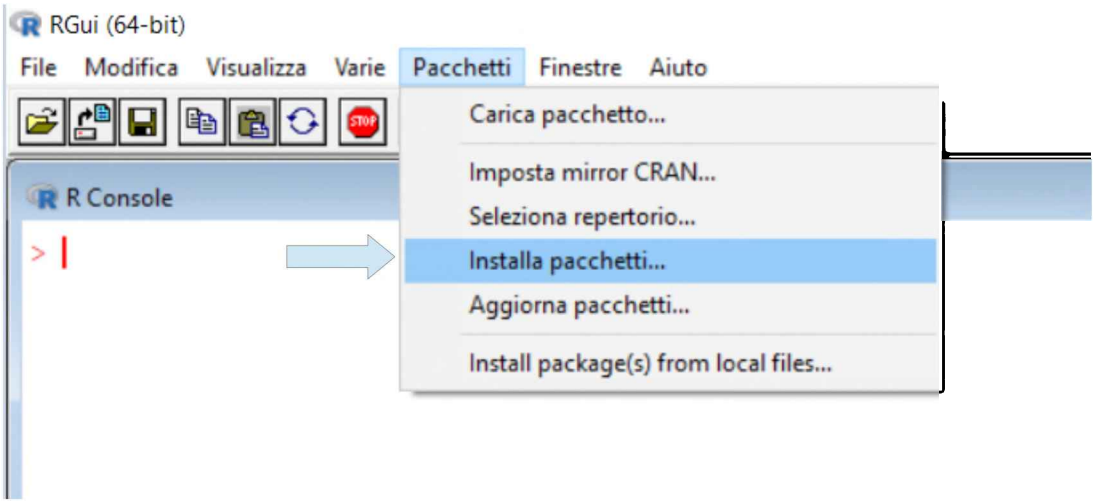



**Fig. 2.10** Per utilizzare le funzioni e i dati contenuti in uno specifico pacchetto questo deve essere installato con l'apposita funzione.

Si aprirà prima una finestra per la selezione del **CRAN** dal quale scaricare il pacchetto (**Fig. 2.11**). Poi una volta selezionato il **CRAN** si aprirà la finestra con l'elenco dei pacchetti disponibili (**Fig. 2.12**).

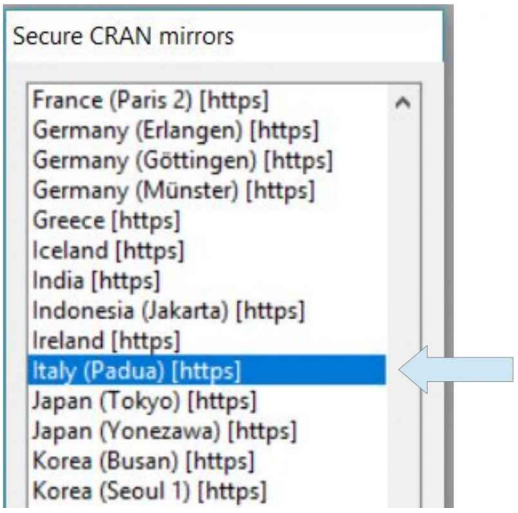



**Fig. 2.11** La finestra per la selezione del **CRAN**.

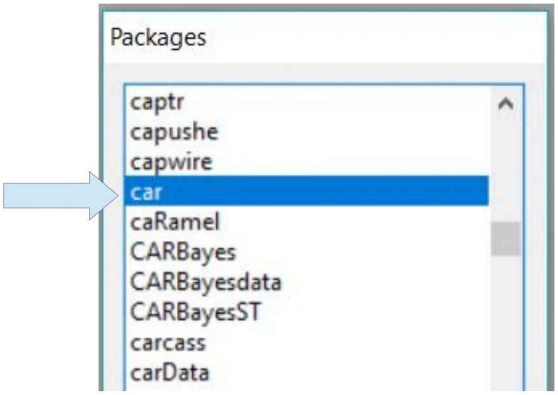



**Fig. 2.12** La finestra per la selezione del pacchetto da scaricare.

Ora scaricate il pacchetto `car` (ci servirà più avanti). Il pacchetto richiede che sul PC o notebook siano installati altri pacchetti, cui esso si appoggia. Se questi non sono installati (e se è la prima volta che usate **R** ovviamente non lo saranno) verrà effettuato automaticamente il `download` e l'installazione di più pacchetti, tutti quelli tra loro concatenati e quindi necessariamente e contemporaneamente richiesti per garantire la funzionalità dell'unico pacchetto che si desiderava installare (in questo caso `car`).

In **R** un "**pacchetto**" è una raccolta di funzioni, di set di dati e di codice **R** compilato. Quando si scarica e si installa un **pacchetto**, questo viene immagazzinato nella locale **libreria**. Ecco perché se nella `Console di R` digitate

**library()**

potete vedere l'elenco dei pacchetti che avete installato e che sono disponibili in **R**. Per ora compariranno solamente il pacchetto `car` e gli altri pacchetti installati in quanto richiesti dal pacchetto `car`.

Per utilizzare funzioni e dati contenuti in un pacchetto è necessario caricarlo, e questo si può fare dalla `Barra dei menù di R` facendo click su `Pacchetti`, selezionando `Carica pacchetto...` (**Fig. 2.13**) e infine selezionando dalla finestra che compare il pacchetto da caricare .

In alternativa è possibile, ed è più pratico, tanto che impiegheremo sempre questa modalità, caricare i pacchetti impiegando all'interno degli script la funzione **library()** o la funzione **require()** e riportando come argomento all'interno della parentesi **( )** il nome del pacchetto.

Quindi digiteremo **library(car)** oppure in alternativa **require(car)** quando dovremo caricare in **R** le funzioni e i dati contenuti nel pacchetto `car` e così via per gli altri pacchetti che utilizzeremo.

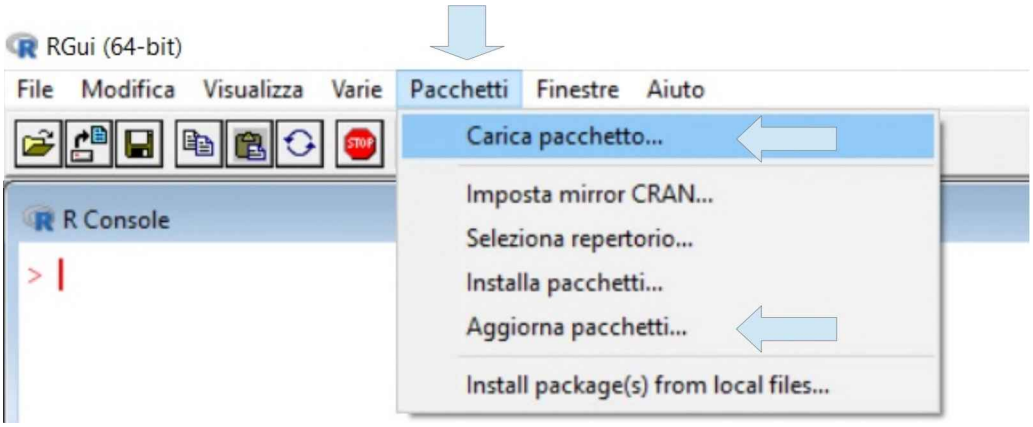



**Fig. 2.13** Per utilizzare le funzioni e i dati contenuti in uno specifico pacchetto questo deve essere caricato impiegando le apposite funzioni (vedi testo) oppure dal menù `Pacchetti`, nel quale è inclusa anche la funzione `Aggiorna pacchetti...`

Una volta caricato un pacchetto se nella `Console di R` digitate

**data()**

vedete aprirsi una finestra nella quale sono elencati tutti i set di dati che sono stati caricati con quel pacchetto[23]. Se nella `Console di R` digitate il nome di uno di questi set di dati potete vederne il contenuto[24].

Se nella funzione **data()** specificate come argomento il nome di un pacchetto che non avete precedentemente caricato come ad esempio

[23] Digitate **help(data)** nella `Console di R` per la documentazione della funzione **data()**.
[24] In questa guida per illustrare le funzioni statistiche e le funzioni grafiche di **R** verranno impiegati sia set di dati inseriti manualmente, sia set di dati ricavati da testi di statistica, sia set di dati contenuti in pacchetti di **R**.

**data(package="DAAG")**
si apre una finestra nella quale sono elencati tutti i set di dati che verrebbero caricati con questo pacchetto.

Sempre mediante la funzione **data()** possono essere caricati set di dati senza caricare il pacchetto che li contiene. Così ad esempio è possibile caricare selettivamente il set di dati **ais** senza caricare il pacchetto **DAAG** digitando
**data(ais, package="DAAG")**
quindi digitando
**ais**
potete vedere i dati caricati (che sono illustrati in **A3. Il set di dati ais**).

Infine se nella `Console di R` digitate
**help(package=*nomedelpacchetto*)**
potete accedere via web alla documentazione del pacchetto.

**Nota bene**: ricordatevi periodicamente di aggiornare i pacchetti che avete installato selezionando dalla `Barra dei menù di R` l'opzione `Pacchetti` quindi selezionando dal menù a tendina che compare `Aggiorna pacchetti...` (**Fig. 2.13**).

## 2.7. Pulizia dell'area di lavoro e uscita dal programma

Quando vi sono troppi dati può essere utile nella `Barra dei menù` del programma selezionare `Modifica` e nel menù a tendina che compare selezionare `Pulisci console` per ripulire la finestra della `Console di R` (**Fig. 2.14**).

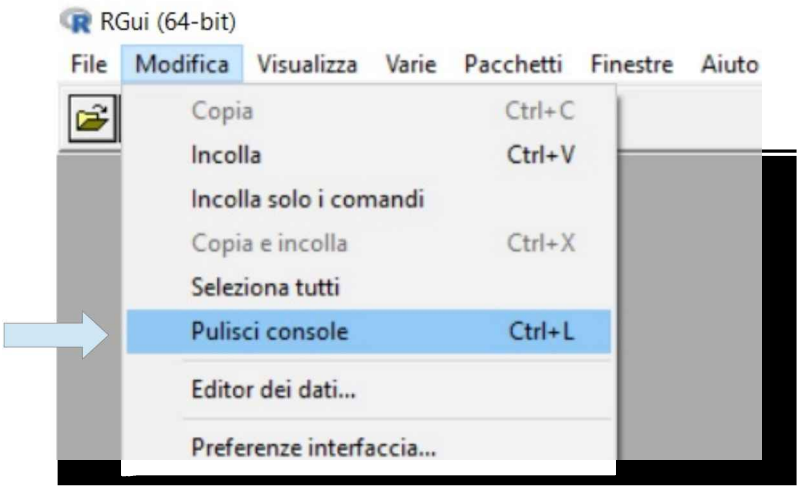



**Fig. 2.14** Pulizia della console di **R**.

Questo tuttavia non elimina gli oggetti generati nella sessione corrente di **R**. Per avere un elenco degli oggetti che **R** ha nell'area di lavoro potete digitare **ls()**[25] nella `Console di R`. Per eliminare il contenuto di questi oggetti nella `Barra dei menù` selezionate `Varie` e nel menù a tendina che compare selezionate `Rimuovi tutti gli oggetti`. (**Fig. 2.15**).

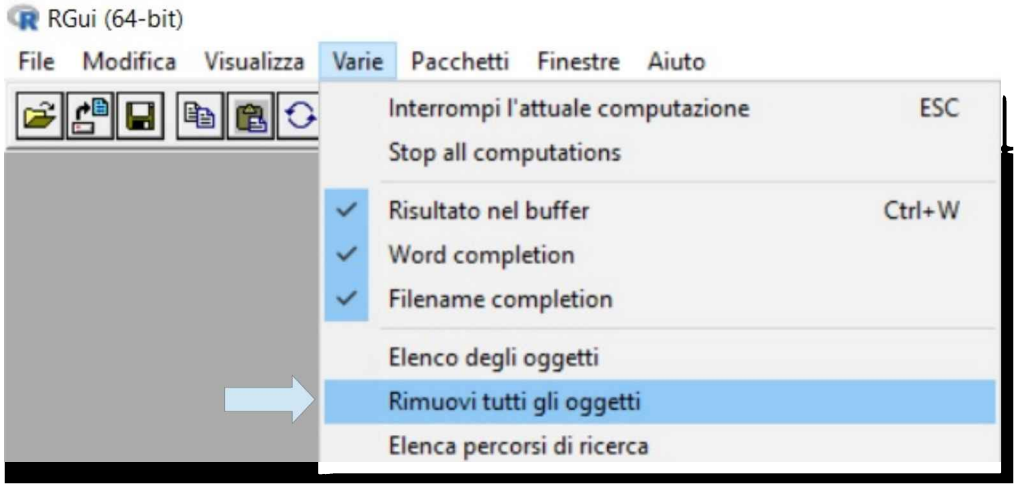



**Fig. 2.15** Rimozione degli oggetto dall'area di lavoro.

[25] Digitate **help(ls)** nella `Console di R` per la documentazione della funzione **ls()**.

In alternativa si può impiegare la funzione **rm()**[26] per eliminare uno specifico oggetto digitando nella `Console di R`
**rm(***nomedell'oggetto***)**
oppure eliminare tutti gli oggetti della sessione di lavoro corrente digitando
**rm(list=ls(all=TRUE))**

Si può effettuare contemporaneamente la pulizia della `Console di R` e la rimozione di tutti gli oggetti generati durante la sessione uscendo dal programma. Per effettuare questa radicale pulizia è sufficiente fare click su `File` e selezionare `Esci` (o in alternativa nella `Console di R` digitare **q()**[27]) e rispondere **No** alla domanda `Salva area di lavoro?` che compare (**Fig. 2.16**).


**Fig. 2.16** Uscita dal programma e salvataggio dell'area di lavoro.

In **R** salvare l'**area di lavoro** significa salvare gli oggetti sui quali avete lavorato. Gli oggetti salvati vengono poi ripristinati da **R** nella sessione successiva. Dato che si tratta di un'operazione delicata, è importante che rispondiate sempre **No** alla domanda `Salva area di lavoro?` In questo modo non vi ritroverete nulla della sessione precedente che possa interferire con le elaborazioni effettuate all'apertura della successiva sessione di **R**. Questo almeno fino a quando non avrete acquisito con **R** una maggiore dimestichezza[28].

Per evitare di dovere rispondere ogni volta alla domanda potete uscire da **R** senza salvare l'area di lavoro con questo script, che consiste in una sola riga di comando:

```
q(save="no") # esce da R senza salvare l'area di lavoro che al rientro in R risulterà ripulita
```

**Nota bene**: al termine di una sessione di **R** effettuate una pulizia dell'area di lavoro per evitare potenziali interferenze con la sessione successiva.

---

[26] Digitate **help(rm)** nella `Console di R` per la documentazione della funzione **rm()**.
[27] Digitate **help(q)** nella `Console di R` per la documentazione della funzione **q()**.
[28] In Windows il salvataggio dell'area di lavoro avviene in due file denominati `.Rdata` e `.rhistory` che si trovano nella cartella `Documenti`. Se doveste per errore salvare l'area di lavoro, per ripulirla è sufficiente eliminare questi due file.

# 3. R - gestione dei dati

*"Non esiste vento favorevole per il marinaio che non sa dove andare."*
*(Lucio Anneo Seneca)*

*"Il meteorologo non sbaglia mai. Se c'è l'80 % di probabilità di pioggia, e non piove, vuol dire che siamo nel 20 %."*
*(Saul Barron)*

---

**R** nella versione base non prevede strumenti evoluti per la gestione dei dati in forma tabellare. Si tratta di una scelta voluta e comprensibile visto che ciascuno di noi sarà portato a continuare a gestirli con lo strumento al quale è abituato. Strumento che nella maggior parte dei casi sarà un foglio elettronico (spreadsheet) in grado di salvare i file in formato `.xls o .xlsx` ma anche `.csv` e quindi tipicamente `Excel` o meglio ancora `OpenOffice calc` oppure `LibreOffice calc` che sono software libero[29]. Per questa ragione in **R** non potevano mancare pacchetti e funzioni per importare i dati.

Se dal menù `Aiuto` di **R** selezionate `Guida Html` alla voce `R Data Import/Export` trovate la guida ufficiale di **R** alle modalità con le quali i dati possono essere importati in **R**[30], che è ovviamente l'aspetto che più ci interessa visto che vogliamo elaborare i dati con **R** (ma ci trovate anche come possono essere esportati).

L'argomento è piuttosto tecnico, e pertanto in una guida come questa di livello introduttivo ci limiteremo a capire:
→ come **inserire manualmente** in **R** i dati da elaborare (modalità che vi sarà utile in varie occasioni);
→ come **importare** in **R** i dati da file in formato `.csv`, che è il formato dati raccomandato per **R** in quanto il più semplice e quello che fornisce maggiori certezze sui dati, che sono leggibili in chiaro e quindi direttamente verificabili (non è così per la maggior parte degli altri file, nei quali i dati sono codificati in formato binario);
→ come si possono in alternativa **importare i dati da un foglio elettronic**o nei due formati codificati in formato binario ma molto diffusi `.xls` e `.xlsx`.

Con la seconda e con la terza di queste modalità vi sarà possibile importare in **R** dati esterni di qualsiasi provenienza.

---

[29] Per il significato di *Software libero*, di *Software open source* e di *Software di dominio pubblico* vedere: *The Free Software Foundation. GNU Operating System. Categories of free and nonfree software*. URL consultato il 18/10/2018: https://goo.gl/MqwM2Q
[30] La guida *R Data import/Export* in formato `.pdf` la trovate anche nel menù `Aiuto` di **R** alla voce `Manuali (in PDF)` e sul sito di **R**. URL consultato il 6/11/2018: https://goo.gl/hR9gyd

## 3.1. Inserimento manuale dei dati

Se normalmente i dati sono importati in **R** dall'esterno, in alcuni casi è utile saperli gestire direttamente dalla `Console di R`. Per questo ho predisposto tre esempi, che illustrano la sintassi da utilizzare per inserire direttamente da tastiera `array` (vettori) numerici e combinarli in `dataset` (tabelle) assegnando i nomi alle `variabili`.

Il primo esempio genera due `array` contenenti ciascuno dieci numeri, li combina in una `matrice` di 2 colonne per 10 righe, poi assegna i nomi alle variabili/colonne e infine assegna un descrittore ai casi/righe. Trattandosi solamente del vostro secondo script ricordo ancora un volta come fare per eseguirlo. Dovete selezionate il testo dello script dal primo all'ultimo # (inclusi) e fare click su `Modifica >> Copia` quindi:
→ nella `Console di R` fate click su `Modifica >> Incolla` e premete ↵`Invio`;
→ oppure in alternativa fate click sulla finestra della `Console di R`, quindi nella finestra che si apre con il tasto destro del mouse selezionate `Incolla` e infine premete ↵`Invio`.

Potete anche fare in quest'alto (e terzo) modo:
→ selezionate il testo dello script dal primo all'ultimo # (inclusi) e fate `ctrl-C` (tenendo premuto il tasto `ctrl` premete il tasto con la lettera `C`);
→ nella `Console di R` fate `ctrl-V` (tenendo premuto il tasto `ctrl` premete il tasto con la lettera `V`) e infine premete ↵`Invio`.

```
# GENERA  DUE ARRAY E LI COMBINA IN UNA MATRICE
#
# genera l'array x con gli interi da 101 a 110
x <- 101:110
# mostra l'array x
x
# genera l'array y con dieci valori di deviata normale standardizzata z
y <- rnorm(10)
# mostra l'array y
y
# combina gli array x e y nella matrice mymatrix
mymatrix <- data.frame(x,y)
# mostra la matrice mymatrix nel quale le righe sono numerate automaticamente da 1 a 10
mymatrix
# assegna i nomi alle variabili/colonne
names(mymatrix) <- c("Caso", "Deviata normale standardizzata z")
# mostra mymatrix con i nomi delle variabili/colonne
mymatrix
# assegna un descrittore ai casi/righe
row.names(mymatrix) <- c("Riga uno","Riga due","Riga tre","Riga quattro","Riga cinque","Riga sei", "Riga
sette","Riga otto","Riga nove","Riga dieci")
# mostra mymatrix con i nomi delle variabili/colonne e i descrittori dei casi/righe
mymatrix
#
```

Dopo avere eseguito l'esempio utilizzate i tasti `Pag-su` e `Pag-giù` per scorrere nella finestra della `Console di R` quanto è accaduto, che viene spiegato nei commenti inseriti riga per riga. Da notare

come una volta combinati i due array **x** e **y** nella matrice **mymatrix** mediante la funzione **data.frame()**[31] **R** numera automaticamente le righe da 1 a 10. A questo punto viene impiegata la funzione **names()**[32] per assegnare i nomi alle variabili delle due colonne. Successivamente i numeri delle righe sono sostituiti con dei descrittori ("Riga uno", "Riga due", eccetera) impiegando la funzione **row.names()**[33].

Questo sarà quindi il contenuto dell'oggetto **mymatrix** definitivo riportato da **R** al termine dell'elaborazione dopo l'ultimo comando **mymatrix** dell'ultima riga di codice (i valori di `z` saranno diversi da quelli qui mostrati in quanto vengono generati al momento ogni volta)[34]:

```
> mymatrix
             Caso Deviata normale standardizzata z
Riga uno      101                       -0.6215375
Riga due      102                        0.2438526
Riga tre      103                       -0.3301488
Riga quattro  104                        0.3720381
Riga cinque   105                        1.2564957
Riga sei      106                        0.6085911
Riga sette    107                        0.2938775
Riga otto     108                       -0.8904089
Riga nove     109                        0.1231109
Riga dieci    110                       -0.4504350
```

Il secondo esempio genera un matrice 2x2 (due righe per due colonne) e assegna i nomi alle righe e i nomi alle colonne. Copiate per intero lo **script** riportato qui sotto quindi incollatelo nella `Console di R` in uno dei tre modi che ora dovreste conoscere e premete ↵`Invio`:

```
# DAI VALORI 1, 26, 24, 68 GENERA UNA MATRICE 2x2
#
# genera l'array cells con i valori numerici contenuti nelle celle
cells <- c(1,26,24,68)
# mostra l'array cells
cells
# genera l'array rnames con i nomi delle righe
rnames <- c("Riga 1", "Riga 2")
# mostra l'array rnames
rnames
# genera l'array cnames con i nomi delle colonne
cnames <- c("Colonna 1", "Colonna 2")
# mostra l'array cnames
cnames
# costruisce la matrice mymatrix a partire dagli array cells, rnames, cnames
mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames))
# mostra la matrice mymatrix
mymatrix
#
```

Dopo avere eseguito l'esempio utilizzate i tasti `Pag-su` e `Pag-giù` per scorrere nella finestra della

[31] Digitate **help(data.frame)** nella `Console di R` per la documentazione della funzione **data.frame()**.
[32] Digitate **help(names)** nella `Console di R` per la documentazione della funzione **names()**.
[33] Digitate **help(row.names)** nella `Console di R` per la documentazione della funzione **row.names()**.
[34] I valori di z vengono generati al momento ogni volta che si esegue lo script, digitate **help(rnorm)** nella Console di R per la documentazione della funzione **rnorm()**.

Console di R quanto è accaduto.

Con la funzione **c()** sono generati gli array che mediante l'operatore di assegnamento **<-** vengono denominati **cells**, **rnames**, **cnames** e che ora contengono rispettivamente i dati (**cells**), i nomi delle righe (**rnames**) e i nomi delle colonne (**cnames**). Quindi mediante la funzione **matrix()**[35] gli array sono combinati nella matrice **mymatrix** di 2 righe (**nrow=2**) per due colonne (**ncol=2**) inserendo i valori per riga (**byrow=TRUE**) e quindi da sinistra a destra e dall'alto in basso ottenendo quindi la matrice desiderata alla quale con l'ultimo argomento (**dimnames=list(rnames, cnames)**) sono assegnati i nomi alle righe e alle colonne.

Questo è il contenuto dell'oggetto **mymatrix** definitivo riportato da **R** al termine dell'elaborazione dopo l'ultimo comando **mymatrix**:

```
> mymatrix
       Colonna 1 Colonna 2
Riga 1         1        26
Riga 2        24        68
```

Il terzo esempio genera una tabella (dataframe o dataset) che contiene valori numerici, alfanumerici e logici, e assegna i nomi alle variabili (colonne). Copiate per intero lo **script** riportato qui sotto quindi incollatelo nella Console di R e premete ↵Invio:

```
# GENERA UN DATAFRAME (sinonimo di DATASET in SPSS e in SAS) DI 3 COLONNE PER 4 RIGHE
#
# genera l'array d con i quattro valori numerici riportati nella parentesi
d <- c(9901,9902,9903,9904)
# mostra l'oggetto d
d
# genera l'array e con i quattro valori alfanumerici riportati nella parentesi, NA=Not Available
e <- c("rosso", "bianco", "blu", NA)
# mostra l'oggetto e
e
# genera l'array f con i quattro valori logici riportati nella parentesi
f <- c(TRUE,TRUE,TRUE,FALSE)
# mostra l'oggetto f
f
# genera la tabella (dataframe/dataset) mydataset a partire dagli array d, e, f
mydataset <- data.frame(d,e,f)
# mostra il contenuto della tabella mydataset
mydataset
# assegna i nomi alle variabili/colonne
names(mydataset) <- c("Identificativo","Colore","Dato valido")
# mostra il contenuto della tabella mydataset
mydataset
#
```

Dopo avere eseguito l'esempio utilizzate casi i tasti Pag-su e Pag-giù per scorrere nella finestra della Console di R quanto è accaduto.

[35] Digitate **help(c)** nella Console di R per la documentazione della funzione **c()** e **help(matrix)** per la documentazione della funzione **matrix**.

Qui di nuovo con la funzione **c()** sono generati gli array che mediante l'operatore di assegnamento **<-** vengono denominati **d**, **e**, **f** e che subito dopo sono combinati mediante la funzione **data.frame()** nella tabella **mydataset**. Successivamente mediante l'onnipresente funzione **c()** sono generati i nomi che vengono assegnati alle colonne della tabella mediante la funzione **names(mydataset)**.

Questo sarà il contenuto dell'oggetto **mydataset** definitivo riportato da **R** al termine dell'elaborazione dopo l'ultimo comando **mydataset**:

```
> mydataset
  Identificativo Colore Dato valido
1           9901  rosso         TRUE
2           9902 bianco         TRUE
3           9903    blu         TRUE
4           9904   <NA>        FALSE
```

Anche se il significato dei vari passaggi è illustrato dai commenti inseriti negli script, vale la pena di notare alcune regole generali:
→ i nomi assegnati agli oggetti che vengono generati, come **x**, **y**, **mymatrix** nel primo esempio, come **cells**, **rnames**, **cnames**, **mymatrix** nel secondo esempio, e come **d**, **e**, **f** e **mydataset** nel terzo esempio, sono arbitrari, e possono essere cambiati a piacere (ovviamente nell'ambito dello stesso **script** il vecchio nome deve essere sostituito con il nuovo tutte le volte che ricorre);
→ per vedere il contenuto di un oggetto è sufficiente digitare nella `Console di R` il nome dell'oggetto;
→ il nome assegnato a un oggetto può essere riutilizzato in script differenti (vedasi ad esempio il nome di oggetto **mymatrix** che è utilizzato sia nel primo sia nel secondo esempio). In questo caso il vecchio contenuto dell'oggetto verrà perso, e verrà sostituito dal nuovo;
→ lo stesso oggetto può contenere sia valori numerici, sia valori alfanumerici (stringhe di testo), sia valori logici, come si vede nel terzo esempio riportato sopra.
→ è possibile inserire `NA` (`Not Available`) nel caso in cui il dato non sia disponibile.

Inserire a mano i dati in **R** non accade di frequente, ma è utile quando i dati da inserire sono pochi, come accade ad esempio quando si vuole effettuare un test chi-quadrato ($\chi^2$), impiegare il teorema di Bayes o costruire un grafico a torta.

**Nota bene**: si consiglia di salvare i tre script riportati sopra che potranno essere facilmente riutilizzati riadattandoli per costruire rapidamente, al bisogno, piccole tabelle con i vostri dati da analizzare statisticamente o da rappresentare graficamente con **R**.

## 3.2. Struttura dei dati e file .csv

Una struttura tipica[36] dei dati estratti da un database è quella riportata nella **Tab. 3.1** nella quale le cose da notare in relazione a **R** sono abbastanza semplici:
→ le **righe** corrispondono ai **casi** (e ai **record** di un database);
→ le **colonne** corrispondono alle **variabili** (e ai **campi** di un record);
→ i **nomi delle variabili** sono riportati nella prima riga (in **R** sono facoltativi ma fortemente raccomandati per mantenere ordine e chiarezza nel proprio lavoro);
→ le **variabili** possono essere sia numeriche, sia qualitative (per esempio, qui, la variabile `Sesso`);
→ un **identificativo univoco** di ciascun dei **casi** può essere riportato nella (in genere, ma non necessariamente) prima colonna ed è facoltativo, ma può essere utile in casi specifici. Se l'identificativo non è presente nei dati originali **R** numera automaticamente i **casi** in ordine crescente per identificare ciascun caso in modo univoco;
→ è ammessa la **mancanza di dati**, per esempio qui manca il valore di `Altezza` nel caso `GF`, e il campo è quindi vuoto. **R** al momento di importare i dati riconoscerà e classificherà automaticamente riportando al posto del dato mancante la sigla `NA` ovvero `Not Available`.

In **R** le variabili qualitative, non numeriche, sono importanti in quanto consentono di raggruppare i dati. Così nella **Tab. 3.1** i dati potranno essere elaborati o tutti insieme o suddivisi in due gruppi in base al valore assunto dalla variabile `Sesso`. Poiché **R** riconosce lettere maiuscole e lettere minuscole, è indispensabile che la variabile in base alla quale i dati possono essere raggruppati sia codificata in modo rigoroso, così il sesso maschile deve essere espresso sempre con `M` (o con `m`) e non si possono usare `M` o `m` indifferentemente.

| | id | sesso | anni | peso_kg | altezza_cm |
|---|---|---|---|---|---|
| | MT | M | 69 | 76 | 178 |
| I casi | GF | F | 56 | 63 | |
| | MC | F | 53 | 71 | 160 |
| I casi | SB | M | 28 | 73 | 178 |
| | FE | F | 61 | 54 | 154 |
| | AB | M | 46 | 92 | 184 |
| | RF | F | 31 | 81 | 156 |

Le variabili

**Tab. 3.1** La struttura dati tipica di un database con le variabili nelle colonne e i casi nelle righe.

Il tema che si pone con **R** è ora questo: in quale formato salvare (e leggere) i dati. Abbiamo già visto in precedenza[37] che il modo migliore per salvare uno script è farlo in un **file di testo** nel quale i caratteri possono essere scritti (e successivamente essere riletti) **in chiaro** nello stesso modo in cui sono scritti su (e possono essere riletti da) un foglio di carta scritto con una macchina da scrivere[38], impiegando un set di

[36] Questa è la struttura dati più tipica, che risulta dalla estrazione di dati da un database, ma non è la sola possibile in **R**. Altre strutture dati che i pacchetti di **R** sono in grado di utilizzare, e che in taluni casi sono specificamente richieste da alcuni pacchetti perché i dati possano essere elaborati, saranno illustrate con gli esempi trattati di volta in volta.
[37] Vedere: **2.4. Come salvare uno script**.
[38] Oramai da tempo soppiantata da PC e stampante, ma è da qui che è originato il concetto.

caratteri limitato, ma universalmente riconosciuto (il codice **ASCII**).

La stessa identica soluzione, salvare i dati in un **file di testo** nel quale i dati sono codificati **in chiaro**, può essere applicata ai **dati** impiegando il formato `.csv`[39] che è il formato dati raccomandato per **R**.

Ecco ora come appaiono i dati della **Tab. 3.1** salvati in un file denominato `peso_altezza.csv` quando apriamo il file con un editor di testo come il `Blocco note` di Windows:

```
id;sesso;anni;peso_kg;altezza_cm
MT;M;69;76;178
GF;F;56;63;
MC;F;53;71;160
SB;M;28;73;178
FE;F;61;54;154
AB;M;46;92;184
RF;F;31;81;156
```

In un tipico file `.csv` come questo:
→ nella prima riga sono riportati i nomi dei **campi** che compongono i **record**;
→ nelle righe successive alla prima sono riportati, uno per riga, i **record** che compongono il **database**;
→ il punto e virgola (;)[40] è il **separatore di campo** che indica la fine di un campo e il passaggio al campo successivo.

Il file `peso_altezza.csv` per ora non esiste. Ma lo potete realizzare semplicemente copiando le otto righe riportate sopra e salvandole con un qualsiasi editor di testo in un file denominato appunto `peso_altezza.csv`. (**Fig. 3.1**).

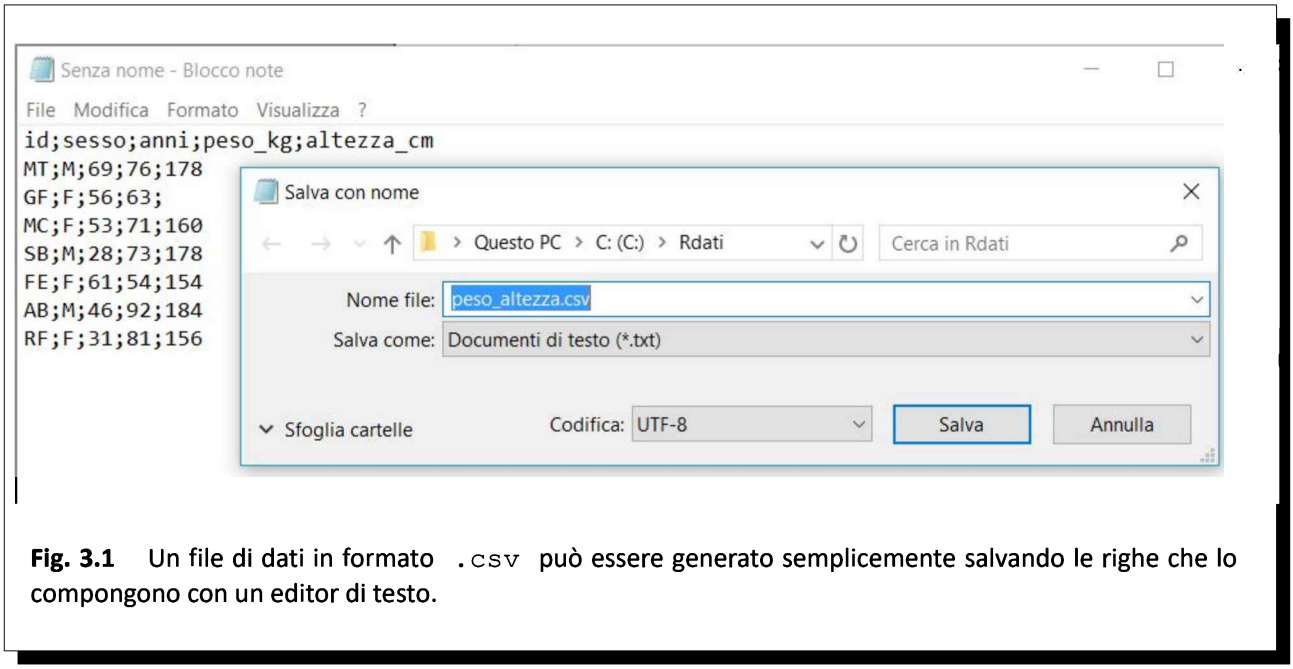



**Fig. 3.1** Un file di dati in formato `.csv` può essere generato semplicemente salvando le righe che lo compongono con un editor di testo.

[39] L'acronimo `csv` sta per `comma-separated values` cioè per valori separati dalla virgola (,). Tuttavia come vedremo subito il separatore di campo può essere anche un altro carattere.
[40] Il punto e virgola (;) in questo caso. Ma è possibile impiegare come separatore di campo un altro carattere come ad esempio la virgola (;), lo spazio ( ) o altro.

Così facendo avete creato il vostro primo file `.csv` che quindi in definitiva è, come i file `.txt`, un file di testo nel quale i dati sono scritti in chiaro e sono organizzati con regole molto semplici e immediatamente riconoscibili semplicemente aprendo il file con un qualsiasi editor di testo.

Da notare che in questo caso viene specificato (**Fig. 3.1**) che il salvataggio deve essere effettuato impiegando la codifica **UTF-8**. Si tratta di un metodo che codifica un insieme vastissimo di set di caratteri, ma questo comporta uno svantaggio in **R**. Se per salvare un file vengono impiegati caratteri al di fuori di quelli previsti nel set di caratteri **ASCII di base**[41] è necessario che chi legge il file con **R** sappia:
→ quale set di caratteri è stata impiegato;
→ come fare a importare correttamente i caratteri.

Dato che la prima cosa non è evidente dal contenuto del file e che la seconda non è così banale, nei file `.csv` da importare in **R**, anche se salvati mediante l'editor di file di testo specificando la codifica **UTF-8**, consiglio di impiegare esclusivamente il set di caratteri **ASCII di base**, che è quello che viene impiegato anche nei file di dati questa guida, quindi di impiegare esclusivamente i seguenti caratteri

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

che sono più che adeguati per compilare i nomi dei campi, i descrittori delle righe e gli eventuali campi di testo di un database in formato testo salvato come file `.csv`. Il salvataggio di un file di testo impiegando la codifica **UTF-8** garantisce ovviamente la compatibilità con il set di caratteri **ASCII di base** nel percorso inverso, quando si importano i dati del file in **R**.

Con questa premessa ora possiamo vedere come importare in **R** i dati di un file `.csv`.

**Nota bene**: il formato dei dati raccomandato per **R** è il formato `.csv`. Si consiglia di impiegare nei file di testo `.csv` da importare in **R** esclusivamente il set di caratteri **ASCII di base**.

---

[41] Vedere: **A1. Codifica dei caratteri ASCII, ANSI, Unicode e UTF**

## 3.3. Importare in **R** i dati di un file .csv

**R** è un programma per la analisi statistica e grafica dei dati e quindi la cosa che accade con maggior frequenza è ovviamente quella di avere dei dati gestiti esternamente a **R**, in un database o in un foglio elettronico, e di avere la necessità di importare in **R** i dati da elaborare.

Ripartiamo dalla **Tab. 3.2** che riporta gli stessi dati della **Tab. 3.1** questa volta con l'altezza espressa in metri, cosa che ci consente di affrontare sia il tema del **separatore di campo** sia il tema del **separatore dei decimali**:

| id | sesso | anni | peso_kg | altezza_m |
|---|---|---|---|---|
| MT | M | 69 | 76 | 1,78 |
| GF | F | 56 | 63 | |
| MC | F | 53 | 71 | 1,60 |
| SB | M | 28 | 73 | 1,78 |
| FE | F | 61 | 54 | 1,54 |
| AB | M | 46 | 92 | 1,84 |
| RF | F | 31 | 81 | 1,56 |

**Tab. 3.2** Gli stessi dati della tabella 3.1 con l'altezza riportata in metri.

Ora per continuare dovete:
→ effettuare il `download` del file `importa_csv.csv`
→ salvare il file nella cartella `C:\Rdati\`

Il file, qui aperto con un editor di testo, contiene i dati della **Tab. 3.2**:

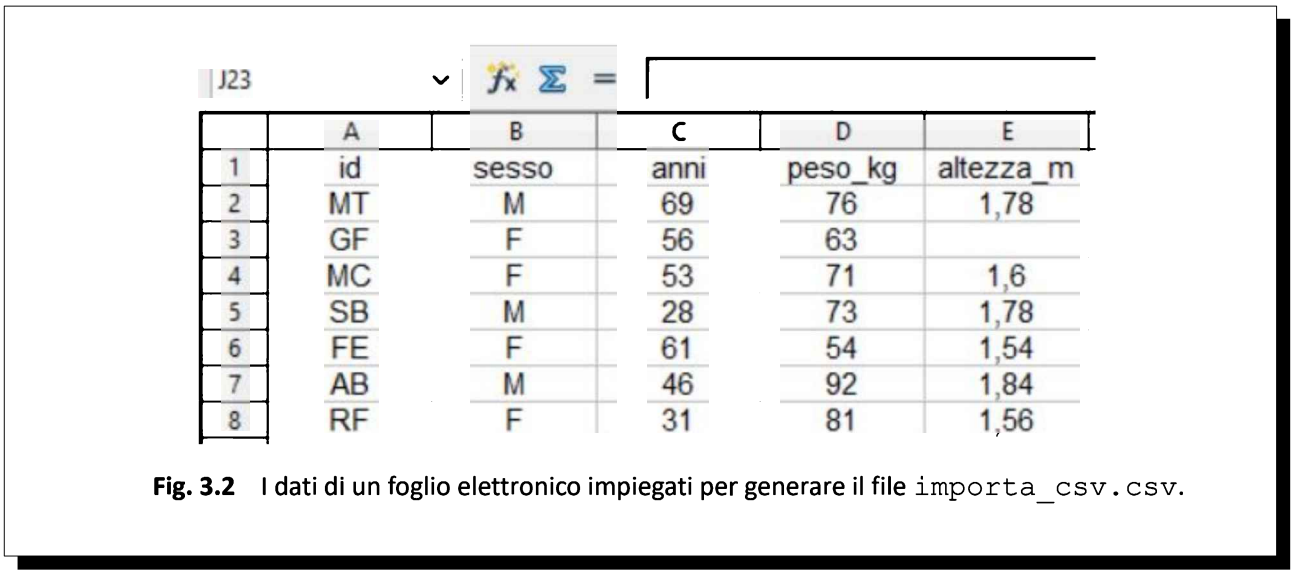
```
id;sesso;anni;peso_kg;altezza_m
MT;M;69;76;1,78
GF;F;56;63;
MC;F;53;71;1,60
SB;M;28;73;1,78
FE;F;61;54;1,54
AB;M;46;92;1,84
RF;F;31;81;1,56
```

In alternativa al `download` potete:
→ copiare con un editor di testo le otto righe di testo riportate qui sopra;
→ salvarle come file di testo avente il nome `importa_csv.csv` nella cartella `C:\Rdati\`

Come vedete nel file sono stati impiegati come **separatore di campo** il punto e virgola (**;**) e come **separatore dei decimali** la virgola (**,**).

Non è un caso: il file `importa_csv.csv` è stato generato salvando in formato `.csv` i dati contenuti

in un foglio elettronico (**Fig. 3.2**), che è lo strumento più frequentemente impiegato per gestire i propri dati. E il punto e virgola (;) come separatore di campo e la virgola (,) come separatore dei decimali sono impostati di default in Windows nei paesi come Spagna, Italia e Francia.




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | id | sesso | anni | peso_kg | altezza_m |
| 2 | MT | M | 69 | 76 | 1,78 |
| 3 | GF | F | 56 | 63 | |
| 4 | MC | F | 53 | 71 | 1,6 |
| 5 | SB | M | 28 | 73 | 1,78 |
| 6 | FE | F | 61 | 54 | 1,54 |
| 7 | AB | M | 46 | 92 | 1,84 |
| 8 | RF | F | 31 | 81 | 1,56 |

**Fig. 3.2** I dati di un foglio elettronico impiegati per generare il file `importa_csv.csv`.

Ora copiate e incollate nella `Console di R` questo **script** e premete ↵`Invio`:

```
# IMPORTA I DATI DI UN FILE CSV
#  notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",")
#
```

All'oggetto **mydata** viene assegnato (**<-**) il contenuto importato dalla funzione **read.table()**. Gli argomenti della funzione, racchiusi nella parentesi, specificano che:
→ il file dal quale importare i dati è **C:/R/importa_csv.csv**;
→ la prima riga nel file è una riga di intestazione con i nomi delle variabili (**header=TRUE**);
→ nel file è impiegato come separatore di campo il punto e virgola (**sep=";"**);
→ nel file è impiegato come separatore dei decimali la virgola (**dec=","**).

Se ora nella `Console di R` digitate
**mydata**
vedete comparire i dati che avete appena importato:

```
> mydata
  id sesso anni peso_kg altezza_m
1 MT     M   69      76      1.78
2 GF     F   56      63        NA
3 MC     F   53      71      1.60
4 SB     M   28      73      1.78
5 FE     F   61      54      1.54
6 AB     M   46      92      1.84
7 RF     F   31      81      1.56
```

Notate che la virgola (,) presente come separatore dei decimali nei dati originali è stata trasformata da **R** in un punto (.)

**Nota bene**: si rammenta che in **R** il separatore delle cifre decimale è il punto (.)

Notare anche che **R** ha aggiunto un identificativo dei casi sotto forma di un numero progressivo (`1` per la prima riga, `2` per la seconda riga, eccetera). Se ritenete che come identificativo dei casi possa essere impiegato il valore del campo "id" potete eseguire quest'altro **script**:

```
# IMPORTA I DATI DI UN FILE CSV CON IDENTIFICATIVO UNIVOCO DEI CASI
#  notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",", row.names="id")
#
```

Rispetto al precedente script la sola modifica che noterete è rappresentato dell'aggiunta alla funzione **read.table()** dell'argomento **row.names="id"** mediante il quale viene specificato che come identificativo dei casi deve essere utilizzata la variabile `id`.

Se ora nella `Console di R` digitate
**mydata**
vedete che con questa modifica i dati questa volta sono stati importati assegnando ai casi l'identificativo specificato, e che conseguentemente la numerazione dei casi non viene più effettuata da **R**:

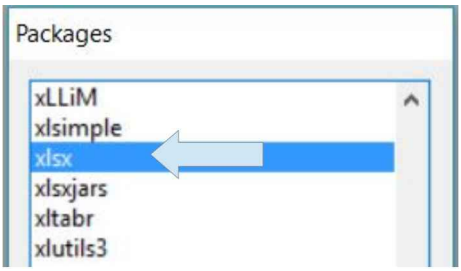
```
> mydata
   sesso anni peso_kg altezza_m
MT     M   69      76      1.78
GF     F   56      63        NA
MC     F   53      71      1.60
SB     M   28      73      1.78
FE     F   61      54      1.54
AB     M   46      92      1.84
RF     F   31      81      1.56
```

**Nota bene**: dato che l'importazione dei dati è un punto di svolta nell'apprendimento di **R**, si consiglia di familiarizzare adeguatamente con questo aspetto adattando gli script riportati a file `.csv` contenenti propri dati.

## 3.4. Importare in **R** i dati di un file .xls o .xlsx

I formati `.xls` e `.xlsx` non sono formati standard, e la struttura dei file potrebbe cambiare senza preavviso nelle nuove versioni di `Excel` causando errori imprevedibili e pertanto non gestibili nell'importazione dei dati in **R**. Questo perché i file sono in formato binario, e se provate ad aprirli con un editor di testo, non riuscite a trovare i dati, come vedete nella **Fig. 3.3** nella quale è presentato, aperto con un editor di testo, il contenuto del file `importa_xls.xls` che contiene i dati (id, sesso, anni, peso_kg, altezza_m) della **Tab. 3.2**, che risultano illeggibili.


**Fig. 3.3** In un file `.xls` i dati sono salvati in codice binario e pertanto non sono direttamente accessibili, come invece accade in un file di testo `.csv`.




Nonostante questo in **R** si trovano pacchetti che consentono di importare i dati direttamente da file `.xls` e `.xlsx`. come ad esempio il pacchetto `xlsx` che si può scaricate dal **CRAN** selezionando nel menù `Pacchetti di R` l'opzione `Installa pacchetti…` e quindi selezionando `xlsx` dall'elenco dei pacchetti (`Packages`) disponibili (figura accanto).

Per la documentazione completa del pacchetto `xlsx` incluso il suo fondamentale `Reference manual` si rimanda al **CRAN**[42]. Se nel menù `Aiuto` selezionate `Guida Html` nella sezione `Reference` alla voce `Packages` trovate la documentazione dei pacchetti che avete installato sul vostro PC o notebook, che include, però in una versione meno completa, la

[42] *xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files*. URL consultato il 12/08/2018: https://goo.gl/DJCHTx

documentazione del pacchetto `xlsx`. Potete anche digitare **help(read.xlsx)** nella `Console di R` per una documentazione concisa ed essenziale della funzione **read.xlsx()**.

Per proseguire ora:
→ effettuate il `download` del file `importa_xls.xls`
→ effettuate il `download` del file `importa_xlsx.xlsx`
→ salvate i file nella cartella `C:\Rdati\`

Copiate e incollate nella `Console di R` questo **script** e premete `↵Invio`:

```
# IMPORTA I DATI DI UN FILE XLS
#
# carica il pacchetto xlsx
require(xlsx)
# carica nell'oggetto mydata i dati del file e del foglio specificati,  notare / invece di \ su windows
mydata <- read.xlsx("C:/Rdati/importa_xls.xls", sheetName="peso_altezza")
#
```

Gli unici argomenti richiesti dalla funzione **read.xlsx** sono il nome del file con il percorso completo (**"C:/Rdati/importa_xls.xls"**) e il nome del foglio che contiene i dati (**sheetName="peso_altezza"**) all'interno del file[43].

Se ora digitate **mydata** e premete `↵Invio` potete scorrere nella `Console di R` i dati che sono stati importati:

```
> mydata
  id sesso anni peso_kg altezza_m
1 MT     M   69      76      1.78
2 GF     F   56      63        NA
3 MC     F   53      71      1.60
4 SB     M   28      73      1.78
5 FE     F   61      54      1.54
6 AB     M   46      92      1.84
7 RF     F   31      81      1.56
```

Per avere la conferma del fatto che i dati sono stati importati correttamente confrontateli con l'originale aprendo il file con `Excel` o con `OpenOffice calc` o con `LibreOffice calc`. La sola differenza che riscontrerete risiederà nel separatore dei decimali, che nel foglio elettronico vedrete essere (a meno che abbiate modificato la configurazione di Windows) la virgola (,) mentre qui è il punto (.) come previsto in **R**.

Copiate e incollate nella `Console di R` questo **script** e premete `↵Invio`:

```
# IMPORTA I DATI DI UN FILE XLS
#
# carica il pacchetto xlsx
require(xlsx)
# carica nell'oggetto mydata i dati del file e del foglio specificati,  notare / invece di \ su windows
mydata <- read.xlsx("C:/Rdati/importa_xls.xls", sheetName="peso_altezza", row.names="id")
#
```

[43] Questo argomento permette di gestire più fogli all'interno di un unico file `.xls` o `xlsx`.

Questa volta l'argomento **row.names="id"** specifica che gli identificativi dei casi sono contenuti nel campo `id` del file, onde evitare che **R** li numeri automaticamente. In effetti se ora digitate **mydata** e premete ↵`Invio` potete scorrere nella `Console di R` i dati che sono stati importati:

```
> mydata
   sesso anni peso_kg altezza_m
MT     M   69      76      1.78
GF     F   56      63        NA
MC     F   53      71      1.60
SB     M   28      73      1.78
FE     F   61      54      1.54
AB     M   46      92      1.84
RF     F   31      81      1.56
```

A questo punto sono lasciati come esercizio la modifica dei due script riportati sopra, sostituendo **C:/Rdati/importa_xls.xls** con **C:/Rdati/importa_xlsx.xlsx**, e la verifica che l'importazione avvenga correttamente anche con il file `.xlsx`.

**Nota bene**: se si ha l'esigenza di importare i dati da un foglio elettronico, si consiglia di familiarizzare adeguatamente con questo aspetto adattando gli script riportati a file `.xls` e `.xlsx` contenenti propri dati.

## 3.5. Esportare i dati da **R**

Anche se a prevalere nettamente sarà sempre l'esigenza di importare i dati in **R**, talora può essere necessario esportare i dati da **R**.

Di nuovo la soluzione migliore è quella di salvarle i dati in un file `.csv` che potrà essere poi importato facilmente in qualsiasi software esterno. La tecnica viene illustrata nella sesta e ultima riga di codice di questo script (le prime cinque righe servono a generare e a mostrare la matrice dati da esportare):

```
# GENERA DUE ARRAY LI COMBINA IN UNA MATRICE ED ESPORTA I DATI
#
x <- 101:110 # genera un array x con gli interi da 101 a 110
y <- rnorm(10) # genera un array y con dieci valori di deviata normale standardizzata z
mymatrix <- data.frame(x,y) # combina gli array x e y nella matrice mymatrix
names(mymatrix) <- c("Caso", "Deviata normale standardizzata z") # assegna i nomi alle colonne
mymatrix # mostra mymatrix nella quale è impiegato il punto (.) come separatore dei decimali
#
# esporta i dati in un file di testo .csv sostituendo il punto (.) con la virgola (,)
#
write.table(mymatrix,file="C:/Rdati/esporta_csv.csv",sep=";", dec=",", quote=FALSE, row.names=FALSE)
#
```

In questo script gli argomenti della funzione **write.table()** consentono di:
→ esportare i dati dell'oggetto **mymatrix**;
→ generare in uscita il file **C:/esporta_csv.csv**;
→ impiegare come separatore di campo il punto e vigola (**sep=";"**)
→ impiegare come separatore dei decimali la virgola (**dec=","**)
→ non chiudere tra le virgolette "" i campi non numerici (**quote=FALSE**)
→ non esportare i nomi delle righe (**row.names=FALSE**)

Questo è il contenuto della matrice generata con lo script[44], con il punto (.) come separatore dei decimali e con gli identificativi/nomi di riga (1, 2, eccetera) generati automaticamente da **R**:

```
> mymatrix # mostra mymatrix nella quale è impiegato il punto (.) come
separatore dei decimali
   Caso Deviata normale standardizzata z
1   101                         0.23293784
2   102                        -0.90141720
3   103                        -1.07984939
4   104                        -1.18306833
5   105                        -0.30620847
6   106                         0.72325028
7   107                        -1.07569947
8   108                         1.25625169
9   109                        -0.25972486
10  110                        -0.41700472
```

[44] I valori di deviata normale standardizzata **z** cambiano di volta in volta essendo generati al momento ogni volta che si esegue lo script, digitate **help(rnorm)** nella `Console di R` per la documentazione della funzione **rnorm()**.

Questo è il contenuto del file `esporta_csv.csv` generato, nel quale compaiono come previsto nello script il punto e virgola (;) come separatore di campo e la virgola (,) come separatore dei decimali. Inoltre grazie all'argomento **row.names=FALSE** non compaiono gli identificativi/nomi di riga di **R**:

```
Caso;Deviata normale standardizzata z
101;0,232937843286289
102;-0,901417196246198
103;-1,07984938772418
104;-1,1830683314683
105;-0,306208467439374
106;0,723250277017055
107;-1,075699474921302
108;1,256251691518791
109;-0,259724860701385
110;-0,417004727242312
```

Qualsiasi foglio elettronico è in grado di riconoscere molti formati incluso il formato `.csv`.

Se aprite il file `esporta_csv.csv` con `Excel` o con `OpenOffice calc` oppure con `LibreOffice calc` i dati si presenteranno come nella **Fig. 3.4** (a meno di minime differenze nell'aspetto della grafica rispetto ad `OpenOffice calc` che è quello dal quale è stata tratta l'immagine):

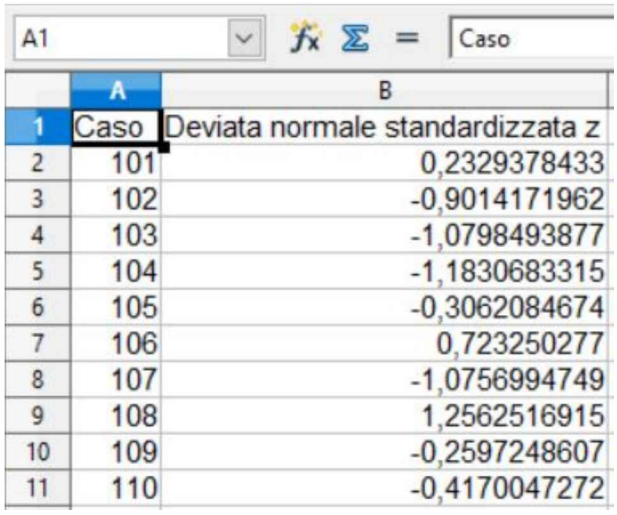



**Fig. 3.4** I dati esportati da **R** e correttamente importati in un foglio elettronico.

L'unica nota importante è che nel momento in cui si apre un file `.csv` con un foglio elettronico, perché questo riconosca correttamente i dati, è necessario specificare il separatore di campo impiegato che in questo caso nel file `esporta_csv.csv` è il punto e virgola (;)[45].

[45] Con alcuni fogli elettronici come ad esempio in `OpenOffice calc` è necessario specificare anche il set di caratteri, che con Windows nei paesi occidentali è UNICODE (UTF-8).

Nella **Fig. 3.5** viene riportato un esempio delle indicazioni che è necessario fornire prima che i dati del file possano essere importati nel foglio elettronico, in questo caso `OpenOffice calc`, con una anteprima (in basso) di come verranno importati i dati, che fornisce un valido aiuto in questa fase.

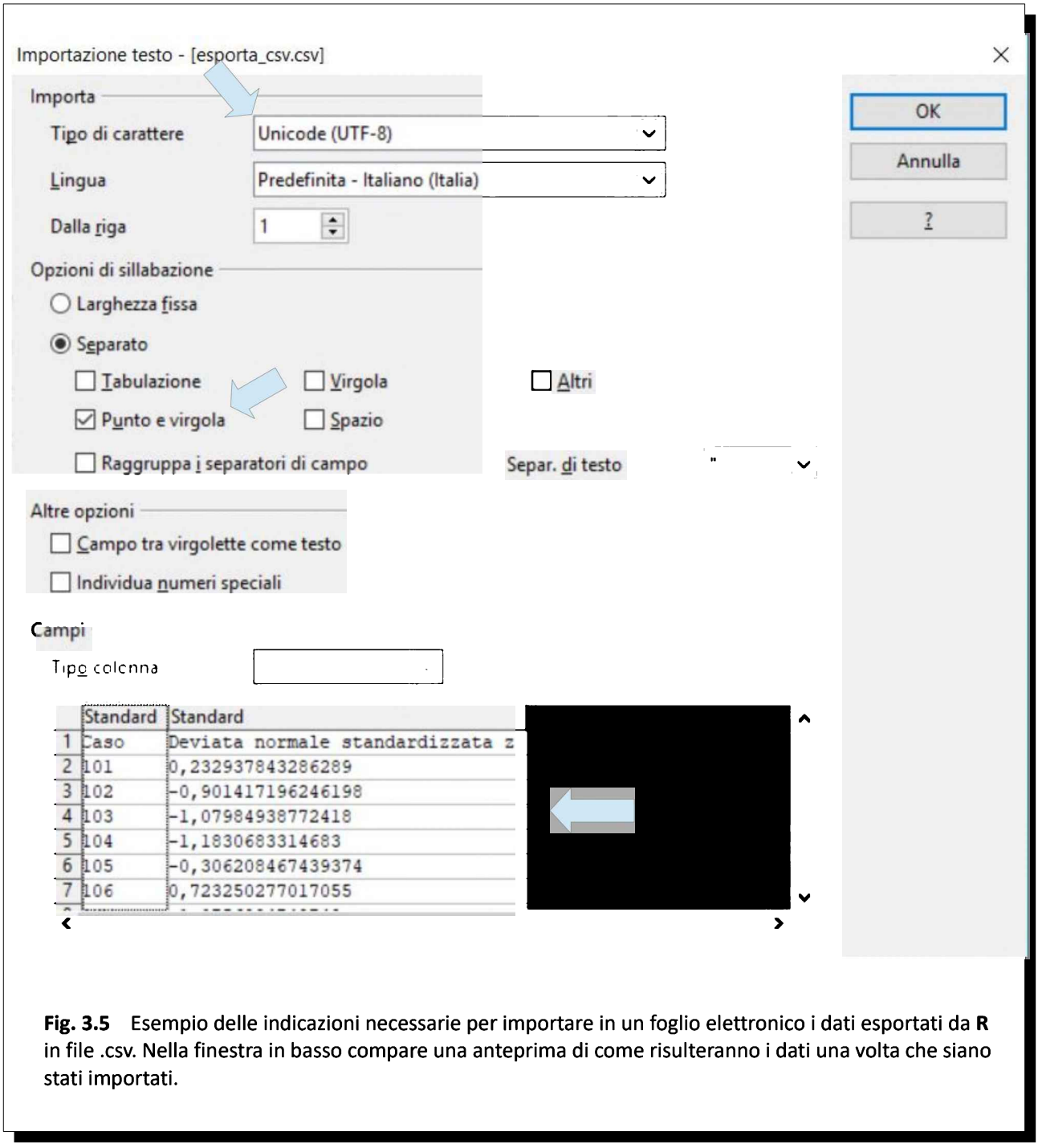



**Fig. 3.5** Esempio delle indicazioni necessarie per importare in un foglio elettronico i dati esportati da **R** in file .csv. Nella finestra in basso compare una anteprima di come risulteranno i dati una volta che siano stati importati.

Un esempio di come impiegare la funzione **write.table()** per esportare in un file `.csv` uno dei numerosi set di dati forniti con **R** e con i suoi pacchetti aggiuntivi è riportato in appendice[46].

Ma in realtà **R** oltre alla funzione **write.table()** fornisce altre due funzioni per esportare i dati in formato `.csv`:

[46] Vedere: **A3. Il set di dati ais.**

→ la funzione **write.csv()** che consente di esportare i dati impiegando la virgola come separatore di campo (,) e il punto (.) come separatore dei decimali;
→ la funzione **write.csv2()** che consente di esportare i dati impiegando il punto e virgola (;) come separatore di campo e la virgola (,) come separatore dei decimali.

In pratica la funzione **write.csv()** impiega i separatori secondo le convenzioni vigenti nel mondo anglosassone, mentre la funzione **write.csv2()** impiega i separatori secondo le convenzioni vigenti in Italia, Spagna, Francia e in altri Paesi europei.

Nello script che segue, i dati della **Tab. 3.2** sono prima ricostruiti (impiegando il punto come separatore dei decimali) e poi salvati impiegando le tre funzioni **write.table()**, **write.csv()** e **write.csv2()**:

```
# COSTRUISCE UNA TABELLA E LA ESPORTA IMPIEGANDO TRE DIVERSE FUNZIONI
#
# genera cinque array
id <- c("MT","GF","MC","SB","FE","AB","RF")
sesso <- c("M","F","F","M","F","M","F")
anni <- c(69,56,53,28,61,46,31)
peso_kg <- c(76,63,71,73,54,92,81)
altezza_m <- c(1.78,NA,1.60,1.78,1.54,1.84,1.56)
#
mytable <- data.frame(id,sesso,anni,peso_kg,altezza_m) # combina gli array nell'oggetto mytable
mytable # mostra l'oggetto mytable
# (a)
write.table(mytable, file="C:/Rdati/write_table_p.csv", quote=FALSE, sep=";", dec=".", na="",
col.names=TRUE, row.names=FALSE) # write.table consente sia (.) sia (,) come separatore dei decimali
# (b)
write.table(mytable, file="C:/Rdati/write_table_v.csv", quote=FALSE, sep=";", dec=",", na="",
col.names=TRUE, row.names=FALSE) # write.table consente sia (.) sia (,) come separatore dei decimali
# (c)
write.csv(mytable, file="C:/Rdati/write_csv.csv", quote=FALSE, na="", row.names=FALSE) # write.csv
# usa (,) come separatore di campo e (.) come separatore dei decimali
# (d)
write.csv2(mytable, file="C:/Rdati/write_csv2.csv", quote=FALSE, na="", row.names=FALSE) # write.csv2
# usa (;) come separatore di campo e (,) come separatore dei decimali
#
```

Nelle prime cinque righe dello script, a scopo di esercizio, sono generati con la funzione **c()** gli array che contengono le cinque variabili/colonne della **Tab. 3.2** che vengono poi riuniti nella tabella definitiva **mytable** mediante la funzione **data.frame()**.

Dopo avere mostrato il contenuto (**mytable**) della tabella:
→ in **(a)** la funzione **write.table()** viene impiegata una prima volta per esportare i dati nel file `write_table_p.csv` nel quale è previsto il punto (.) come separatore dei decimali;
→ in **(b)** la funzione **write.table()** viene impiegata una seconda volta per esportare i dati nel file `write_table_v.csv` nel quale è prevista la virgola (,) come separatore dei decimali;
→ in **(c)** la funzione **write.csv()** viene impiegata per esportare i dati nel file `write_csv.csv` impiegando i separatori secondo le convenzioni vigenti nel mondo anglosassone, cioè la virgola come separatore di campo (,) e il punto (.) come separatore dei decimali;
→ in **(d)** la funzione **write2.csv()** viene impiegata per esportare i dati nel file `write_csv2.csv` impiegando i separatori secondo le convenzioni vigenti in Europa, cioè il punto e virgola (;) come separatore

di campo e la virgola (,) come separatore dei decimali.

Da notare che il separatore dei decimali nelle funzioni **write.csv()** e **write.csv2()** è fissato all'interno delle funzioni stesse e non può essere cambiato.

I quattro file generati sono salvati nella cartella `C:\Rdati\` e possono essere aperti con un editor di testo per verificarne il contenuto.

---

## 3.6. Gestione dei dati mancanti

Se avete eseguito quanto previsto al punto **3.3. Importare in R i dati di un file .csv** potete passare immediatamente ad eseguire lo **script** riportato qui sotto in quanto nella cartella `C:\Rdati\` avete già il file `importa_csv.csv`.

Altrimenti per continuare dovete:
→ effettuare il `download` del file `importa_csv.csv`
→ salvare il file nella cartella `C:\Rdati\`

In alternativa potete:
→ copiare con un editor di testo le otto righe di testo riportate qui sopra;
→ salvarle come file di testo con il nome `importa_csv.csv` nella cartella `C:\Rdati\`

```
id;sesso;anni;peso_kg;altezza_m
MT;M;69;76;1,78
GF;F;56;63;
MC;F;53;71;1,60
SB;M;28;73;1,78
FE;F;61;54;1,54
AB;M;46;92;1,84
RF;F;31;81;1,56
```

Copiate e incollate nella `Console di R` questo **script** e premete ↵`Invio`:

```
# ESEMPIO DI MEDIA NON COMPUTABILE A CAUSA DI DATI MANCANTI                    1/2.a
#
# importa i dati, notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",", row.names="id")
#
mydata # mostra i dati importati
peso <- mydata[c("peso_kg")] # carica i valori della colonna peso_kg nell'array peso
colMeans(peso) # calcola la media aritmetica dei valori di peso
altezza <- mydata[c("altezza_m")] # carica i valori della colonna altezza_m nell'array altezza
colMeans(altezza) # calcola la media aritmetica dei valori di altezza
#
```

Il codice **R** fa riferimento ai dati della **Tab. 3.2** contenuti nel file `importa_csv.csv` sui quali ora abbiamo voluto calcolare la media del peso e la media dell'altezza.

Il risultato che compare nella `Console di R` è il seguente:

```
# ESEMPIO DI MEDIA NON COMPUTABILE A CAUSA DI DATI MANCANTI        1/2.a
#
# importa i dati, notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";",
dec=",", row.names="id")
```

```
#
> mydata # mostra i dati importati
   sesso anni peso_kg altezza_m
MT     M   69      76      1.78
GF     F   56      63        NA
MC     F   53      71      1.60
SB     M   28      73      1.78
FE     F   61      54      1.54
AB     M   46      92      1.84
RF     F   31      81      1.56
> peso <- mydata[c("peso_kg")] # carica i valori della colonna peso_kg
nell'array peso
> colMeans(peso) # calcola la media aritmetica dei valori di peso
 peso_kg
72.85714
> altezza <- mydata[c("altezza_m")] # carica i valori della colonna
altezza_m nell'array altezza
> colMeans(altezza) # calcola la media aritmetica dei valori di altezza
altezza_m
       NA
```

Come potete vedere il calcolo della media effettuato con la funzione **colMeans()** ha avuto successo solamente per il peso[47]. Per l'altezza in luogo del valore della media è stata riportata la sigla `NA`, evidente conseguenza del valore mancante (`NA = Not Available`) nel file importato.

Al solo scopo di esercitarsi con **R** in luogo della funzione **colMeans()** possiamo impiegare la funzione **mean()** ed effettuare gli stessi calcoli con un secondo **script** più compatto (quattro righe di codice invece di sei):

```
# ESEMPIO DI MEDIA NON COMPUTABILE A CAUSA DI DATI MANCANTI                1/2.b
#
# importa i dati, notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",", row.names="id")
#
mydata # mostra i dati importati
mean(mydata$peso_kg) # calcola la media aritmetica dei valori di peso
mean(mydata$altezza_m) # calcola la media aritmetica dei valori di altezza
#
```

Anche in questo caso la media non viene calcolata.

Ora copiate e incollate nella `Console di R` la seconda parte della script .a e premete ↵`Invio` (da notare che riutilizziamo gli oggetti **peso** e **altezza** appena creati con la parte 1/2.a dello scrip):

```
# la corretta gestione dei dati mancanti consente di calcolare la media dei valori di altezza   2/2.a
#
colMeans(peso) # ricalcola la media aritmetica dei valori di peso
colMeans(altezza, na.rm=TRUE) # ricalcola la media aritmetica dei valori di altezza
#
```

[47] Digitate **help(colMeans)** nella `Console di R` per la documentazione della funzione **colMeans()**.

Il risultato dei calcoli della media che compare nella `Console di R` è ora questo:

```
# la corretta gestione dei dati mancanti consente di calcolare la media
dei valori di altezza       2/2.a
> #
> colMeans(peso) # ricalcola la media aritmetica dei valori di peso
 peso_kg
72.85714
>
> colMeans(altezza, na.rm=TRUE) # ricalcola la media aritmetica dei
valori di altezza
altezza_m
 1.683333
```

Quindi con l'aggiunta nella funzione **colMeans()** dell'argomento **na.rm=TRUE**, che fa rimuovere i valori mancanti, è stato reso possibile il calcolo della media anche per l'altezza.

**Nota bene**: ora potete salvare il primo script combinando la parte 1/2.a con la parte 2/2.a.

Lo stesso ricalcolo può essere effettuato per lo script .b con questa seconda parte:

```
# la corretta gestione dei dati mancanti consente di calcolare la media dei valori di altezza          2/2.b
#
mean(mydata$peso_kg) # ricalcola la media aritmetica dei valori di peso
mean(mydata$altezza_m, na.rm=TRUE) # ricalcola la media aritmetica dei valori di altezza
#
```

Vedete che anche in questo caso, con l'aggiunta nella funzione **mean()** dell'argomento **na.rm=TRUE**, la media dell'altezza viene calcolata correttamente.

**Nota bene**: ora potete salvare anche il secondo script combinando la parte 1/2.b con la parte 2/2.b.

Con la funzione **na.omit()** è possibile eliminare definitivamente da una tabella i casi con dati mancanti, anche un solo dato mancante comporta la cancellazione dell'intera riga[48]:

```
# ELIMINA I CASI CON DATI MANCANTI
#
# importa i dati, notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",", row.names="id")
#
mydata # mostra i dati importati
newdata <- na.omit(mydata) # elimina i casi con dati mancanti
newdata # mostra i dati dopo eliminazione dei casi con dati mancanti
#
mean(newdata$peso_kg) # calcola la media aritmetica dei valori di peso
mean(newdata$altezza_m, na.rm=TRUE) # calcola la media aritmetica dei valori di altezza
#
```

Dai dati importati nella tabella **mydata** mediante la funzione **na.omit()** viene generata una nuova tabella

[48] Digitate **help(na.omit)** nella Console di R per la documentazione della funzione **na.omit()**.

**newdata** dalla quale sono esclusi i casi con dati mancanti.

```
> mydata # mostra i dati importati
   sesso anni peso_kg altezza_m
MT     M   69      76      1.78
GF     F   56      63        NA
MC     F   53      71      1.60
SB     M   28      73      1.78
FE     F   61      54      1.54
AB     M   46      92      1.84
RF     F   31      81      1.56
> newdata <- na.omit(mydata) # elimina i casi con dati mancanti
> newdata # mostra i dati dopo eliminazione dei casi con dati mancanti
   sesso anni peso_kg altezza_m
MT     M   69      76      1.78
MC     F   53      71      1.60
SB     M   28      73      1.78
FE     F   61      54      1.54
AB     M   46      92      1.84
RF     F   31      81      1.56
> #
> mean(newdata$peso_kg) # calcola la media aritmetica dei valori di peso
[1] 74.5
> mean(newdata$altezza_m, na.rm=TRUE) # calcola la media aritmetica dei
valori di altezza
[1] 1.683333
```

Da notare che impiegando la funzione **na.omit()** la media dei valori di peso, che prima era `72.85714`, ora cambia, in quanto in seguito all'eliminazione del caso GF è stato eliminato anche il suo valore di peso (`63`).

---

# 4. R - analisi statistica dei dati

*"Sai ched'è la statistica? È 'na cosa / che serve pe' fa' un conto in generale / de la gente che nasce, che sta male, / che more, che va in carcere e che spósa. / Ma pe' me la statistica curiosa / è dove c'entra la percentuale, / pe' via che, lì, la media è sempre eguale / puro co' la persona bisognosa. / Me spiego: da li conti che se fanno / secondo le statistiche d'adesso / risurta che te tocca un pollo all'anno: / e, se nun entra ne le spese tue, / t'entra ne la statistica lo stesso / perché c'è un antro che ne magna due."*
*(Trilussa)*

*"Nei tempi antichi non c'erano le statistiche, perciò era necessario ripiegare sulle menzogne."*
*(Stephen Leacock)*

Per gli argomenti concernenti **R** trattati qui e nei capitoli successivi si assume che abbiate familiarizzato adeguatamente con gli argomenti generali contenuti nei primi tre capitoli della guida e che abbiate preso nota delle indicazioni fornite. Per questa ragione in questo e nei successivi capitoli l'esposizione sarà più concisa, e verranno omesse le informazioni ripetitive e ridondanti.

**Nota bene**: per gli argomenti di statistica si assume un livello di preparazione statistica di base. I dati impiegati negli esempi sono di tipo biomedico ma sono presentati in modo da essere di immediata comprensione e non richiedono una competenza specifica.

## 4.1. Test chi-quadrato ($\chi^2$)

Nel caso delle **scale nominali** e delle **scale ordinali**[49] esiste un solo modo per esprimere le osservazioni in modo quantitativo (numerico): contare gli oggetti/eventi. Per verificare se questi si presentano con la frequenza attesa, sono impiegati il **test chi-quadrato** o una delle sue varianti, il **test di Fisher** e il **test di McNemar**.

### 4.1.1. 1 riga · *n* colonne

Le leggi della genetica applicate alla riproduzione umana consentono di stabilire che la **probabilità** che un nuovo nato sia di sesso maschile o di sesso femminile è la stessa ed è pari a `p = 0.5` per entrambi i sessi. Nel Dipartimento materno-infantile di un ospedale in un anno sono nati 817 femmine e 756 maschi. I dati sono organizzati in una tabella di una riga per due colonne

| Femmine | Maschi |
|---|---|
| 817 | 756 |

Apparentemente sono nate più femmine. Ci si domanda se la differenza tra i casi osservati e le frequenze attese previste dalle leggi della genetica sia significativa.

Ora con **R** eseguite questo script nel quale come vedete i dati per il **test chi-quadrato** sono inseriti manualmente:

```
# TEST CHI-QUADRATO - 1 riga · n colonne
#
casi.osservati <- c(817,756) # sono immessi i casi osservati
prob.teor <- c(0.5,0.5) # sono immesse le probabilità previste dalla teoria
chiquad <- chisq.test(casi.osservati, p = prob.teor) # risultati del test sono salvati nell'oggetto chiquad
chiquad$observed # mostra i casi osservati
chiquad$expected # mostra le frequenze attese
chiquad #  mostra i risultati del test chi-quadrato
#
```

Come prima cosa mediante la funzione **c()** costruiamo il vettore che contiene i casi osservati (**817,756**), che per comodità e leggibilità del codice che seguirà salviamo nell'oggetto **casi.osservati**. Quindi nella seconda riga di codice facciamo la stessa cosa per la probabilità teorica, che di nuovo per comodità e leggibilità del codice che seguirà salviamo nell'oggetto **prob.teor**.

Nella terza riga viene eseguito il test chi-quadrato impiegando la funzione **chisq.test()**[50] che ha come argomento i casi osservati e le frequenze attese, che devono essere inserite con l'argomento **p=**. Quindi il risultato del test viene salvato (**<-**) nell'oggetto **chiquad**, dal quale sono prima estratti i e mostrati dati osservati (**chiquad$observed**), poi le frequenze attese sulla base dei casi osservati e delle relative probabilità (**chiquad$expected**). Infine sono mostrati i risultati del test chi-quadrato (**chiquad**).

[49] Vedere: **A9. Scale nominali, scale ordinali e scale numeriche**.
[50] Digitate **help(chisq.test)** nella `Console di R` per la documentazione della funzione **chisq.test()**.

Questi sono i risultati dello script:

```
> chiquad$observed # mostra i casi osservati
[1] 817 756
> chiquad$expected # mostra le frequenze attese
[1] 786.5 786.5
> chiquad #  mostra i risultati del test chi-quadrato

        Chi-squared test for given probabilities

data:  casi.osservati
X-squared = 2.3655, df = 1, p-value = 0.124
```

La probabilità `p` riportata qui e in generale negli altri test statistici inclusi nella guida, rappresenta la probabilità che la differenza osservata sia attribuibile al caso. Se tale probabilità è sufficientemente bassa, si assume che la differenza non sia attribuibile al caso, ovvero che la differenza sia significativa. In genere la soglia viene posta ad un valore di `p = 0.05` quindi per valori di `p < 0.05` la differenza osservata viene ritenuta significativa. Nel caso del nostro test chi-quadrato la probabilità `p` che la differenza sia dovuta al caso è elevata (`p-value = 0.124`) e pertanto il numero di femmine e di maschi osservato non è significativamente diverso da quello atteso che prevede femmine e maschi in ugual numero.

Un altro esempio è fornito da Marubini[51]. In una ricerca sull'ibridazione di specie vegetali ci si attende, in base alle leggi delle genetica, la produzione di individui appartenenti alle varietà AB, Ab, aB, ab nel rapporto di 9:3:3:1. Questo rapporto, espresso in termini di probabilità `p` diventa 0.5625:0.1875:0.1875:0.0625 essendo 9+3+3+1 = 16 e 9/16 = 0.5625, 3/16 = 0.1875, 3/16 = 0.1875, 1/16 = 0.0625 (notare che la somma di queste probabilità è necessariamente uguale a 1).

Il numero di individui prodotti per ciascuna delle varietà in un esperimento di ibridazione è riportato in una tabella di una riga per quattro colonne:

| AB | Ab | aB | ab |
|---|---|---|---|
| 72 | 29 | 36 | 12 |

Per verificare se la differenza tra i casi osservati e le probabilità previste dalle leggi della genetica sia significativa, eseguite con **R** questo script che esegue il test chi-quadrato e nel quale di nuovo per semplicità i casi osservati e le probabilità teoriche sono inseriti manualmente:

```
# TEST CHI-QUADRATO - 1 riga · n colonne
#
casi.osservati <- c(72,29,36,12) # sono immessi i casi osservati
prob.teorica <- c(0.5625,0.1875,0.1875,0.0625) # sono immesse le probabilità previste dalla teoria
chiquad <- chisq.test(casi.osservati, p = prob.teorica) # risultati del test salvati nell'oggetto chiquad
chiquad$observed # mostra i casi osservati
chiquad$expected # mostra le frequenze attese
chiquad #  mostra i risultati del test chi-quadrato
#
```

[51] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp. 293-295.

Questi sono i risultati del test chi-quadrato:

```
> chiquad$observed # mostra i casi osservati
[1] 72 29 36 12
> chiquad$expected # mostra le frequenze attese
[1] 83.8125 27.9375 27.9375  9.3125
> chiquad #  mostra i risultati del test chi-quadrato

        Chi-squared test for given probabilities

data:  casi.osservati
X-squared = 4.8076, df = 3, p-value = 0.1864
```

La probabilità `p` che la differenza tra i casi osservati e le frequenze attese sia dovuta al caso è elevata (`p-value = 0.1864`) e pertanto se ne deduce che il rapporto tra individui appartenenti alle varietà AB, Ab, aB, ab ottenuto nell'esperimento non è significativamente diverso da quello atteso di 9:3:3:1 previsto dalle leggi della genetica.

### 4.1.2. 2 righe · 2 colonne

Il principio che vale per le tabelle nelle quali le osservazioni sono organizzate in una tabella di 2 righe x 2 colonne è il seguente:
→ si utilizza il **test chi-quadrato** quando le osservazioni sono numerose e sono indipendenti;
→ si utilizza il **test di Fisher** quando sono le osservazioni non sono numerose e sono indipendenti;
→ si utilizza il **test di McNemar** quando le osservazioni non sono indipendenti (dati appaiati).

Il **test chi-quadrato** lo vediamo ora applicato a un ampio studio canadese sulla relazione tra fumo e mortalità[52]. Nell'arco di sei anni venne registrato il numero di decessi avvenuti in un gruppo di non fumatori e in un gruppo di fumatori di pipa con i seguenti risultati:

| *Esito* | *Non fumatori* | *Fumatori di pipa* |
|---|---|---|
| Deceduti | 117 | 54 |
| Viventi | 950 | 348 |

Di 1067 non fumatori 117 (il 10,97%) erano deceduti. Dei 402 fumatori di pipa 54 (il 13.43%) erano deceduti. La domanda è se esiste una differenza reale tra le mortalità nei due gruppi, o se la differenza può ancora essere attribuita al caso.

Fate il `download` del file `chi_2x2.csv` quindi copiatelo in `C:\Rdati\`

In alternativa potete anche copiare le tre righe riportate qui sotto e salvarle in `C:\Rdati\` nel file di testo denominato `chi_2x2.csv` (attenzione all'estensione al momento del salvataggio del file).

```
Esito;Non_fumatori;Fumatori_di_pipa
Deceduti;117;54
Viventi;950;348
```

[52] Best EWR et al. *A Canadian Study on Smoking and Health (Final Report)*. Dept. Natl. Health and Welfare, Canada, 1966. Citato in: Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6, p. 124.

In entrambi i casi eseguite questo script che impiega sempre la funzione **chisq.test()**[53]:

```
# TEST CHI-QUADRATO - 2 righe · 2 colonne                                    .a
#
mydata <- read.table("c:/Rdati/chi_2x2.csv", header=TRUE, sep=";", row.names="Esito") # importa i dati
mydata # mostra i dati
chisq.test(mydata, correct=TRUE) # test chi quadrato con la correzione di Yates per la continuità
#
```

Questi sono i risultati:

```
> mydata # mostra i dati
          Non_fumatori Fumatori_di_pipa
Deceduti           117               54
Viventi            950              348
> chisq.test(mydata, correct=TRUE) # test chi quadrato con la correzione
di Yates per la continuità

        Pearson's Chi-squared test with Yates' continuity correction

data:  mydata
X-squared = 1.4969, df = 1, p-value = 0.2212
```

Il test chi-quadrato con 1 grado di libertà è esatto solo asintoticamente per dimensioni molto grandi dei campioni per cui si consiglia di applicare sempre la correzione di Yates per la continuità[54]. Il valore così ottenuto è `p = 0.2212`. La conclusione è che la mortalità osservata nel gruppo dei fumatori di pipa non differisce significativamente dalla mortalità osservata nel gruppo dei non fumatori.

In alternativa con quest'altro script potete eseguire il test inserendo i dati manualmente:

```
# TEST CHI-QUADRATO - 2 righe · 2 colonne                                    .b
#
cells <- c(117,54,950,348) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Deceduti", "Viventi") # genera l'array rnames con i nomi delle righe
cnames <- c("Non_fumatori", "Fumatori_di_pipa") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
mydata # mostra i dati
chisq.test(mydata, correct=TRUE) # test chi quadrato con la correzione di Yates per la continuità
#
```

Se aggiungete allo fine dei due script quest'altra riga

```
chisq.test(mydata, correct=FALSE) # test chi quadrato senza la correzione di Yates
```

potete calcolare il chi-quadrato senza la correzione di Yates per la continuità (`p = 0.1886`). Notate che la correzione di Yates per la continuità aumenta il valore di `p` ovvero rende la differenza meno significativa.

---

[53] Digitate **help(chisq.test)** nella `Console di R` per la documentazione della funzione **chisq.test()**.
[54] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, MIlano, 1979, p.137.

Il **test di Fisher** (detto anche **test esatto di Fisher**) deve essere impiegato quando le osservazioni sono poche. Armitage[55] riporta le stesse indicazioni di Snedecor[56] che a sua volta indica questa necessità quando:
→ le osservazioni sono meno di 20;
→ le osservazioni sono comprese tra 20 e 40 e la frequenza attesa più piccola è inferiore a 5.

Il test di Fisher lo applichiamo all'analisi delle relazione che potrebbe intercorrere tra il tipo di allattamento (naturale o artificiale) e la presenza in età successiva nel bambino di malocclusione dentaria (la suzione da una tettarella in gomma non è del tutto fisiologica e potrebbe determinare un qualche effetto sulla morfogenesi delle arcate dentarie che è in atto nel lattante). I dati[57] sono riportati in questa tabella:

| *Allattamento* | *Denti normali* | *Malocclusione* |
|---|---|---|
| Allattamento naturale | 4 | 16 |
| Allattamento artificiale | 1 | 21 |

Fate il `download` del file `chi_Fisher.csv` quindi copiatelo in `C:\Rdati\`

In alternativa potete copiare le tre righe riportate qui sotto e salvarle in `C:\Rdati\` nel file di testo denominato `chi_Fisher.csv` (attenzione all'estensione al momento del salvataggio del file).

```
Allattamento;Denti_normali;Malocclusione
Allattamento_naturale;4;16
Allattamento_artificiale;10;21
```

Questo è lo script per eseguire il test di Fisher impiegando la funzione **fisher.test()**[58]:

```
# TEST DI FISHER - 2 righe · 2 colonne                                    .a
#
mydata <- read.table("c:/Rdati/chi_Fisher.csv", header=TRUE, sep=";", row.names="Allattamento") #
# importa i dati
mydata # mostra i dati
fisher.test(mydata) # esegue il test di Fisher
#
```

Questi sono i risultati:

```
> mydata # mostra i dati
                        Denti_normali Malocclusione
Allattamento_naturale               4            16
Allattamento_artificiale            1            21
> fisher.test(mydata) # esegue il test di Fisher

        Fisher's Exact Test for Count Data

data:  mydata
p-value = 0.1745
alternative hypothesis: true odds ratio is not equal to 1
```

[55] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, MIlano, 1979, p.140.
[56] Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6, p. 127.
[57] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, MIlano, 1979, pp. 138-140.
[58] Digitate **help(fisher.test)** nella `Console di R` per la documentazione della funzione **fisher.test()**.

```
95 percent confidence interval:
   0.4440411 270.1636947
sample estimates:
odds ratio
  5.059936
```

Il test di Fisher fornisce come risultato un valore di `p = 0.1745` che non è significativo e pertanto la conclusione è che non esiste evidenza che i due tipi di allattamento comportino differenze nell'incidenza di malocclusione.

La funzione **fisher.test()** fornisce una analisi aggiuntiva della tabella calcolando anche l'**odds ratio**. In questo caso un risultato diverso da `1` depone per una associazione tra i fattori presenti nella tabella. Un risultato diverso da `1` si avrebbe se i limiti di confidenza dell'odds ratio trovato (che è uguale a `5.059936`) non includessero il valore `1`. Dato che lo includono (vanno infatti da `0.4440411` a `270.1636947`) il risultato non è significativo, conferma la mancanza di associazione tra i fattori presenti nella tabella e conferma il valore di `p` ottenuto con il test di Fisher.

Per il tema odds ratio, che esula dai limiti di questa guida, si rimanda ai test di statistica, lo trovate tra gli altri in Marubini[59], Campbell[60] e Ingelfinger[61].

In alternativa potete eseguire il test immettendo i dati manualmente con questo secondo script, che fornisce ovviamente gli stessi risultati del precedente:

```
# TEST DI FISHER - 2 righe · 2 colonne                                    .b
#
cells <- c(4,16,1,21) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Allattamento_naturale", "Allattamento_artificiale") # genera l'array rnames con i nomi
# delle righe
cnames <- c("Denti_normali", "Malocclusione") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
mydata # mostra i dati
fisher.test(mydata) # esegue il test di Fisher
#
```

Il **test di McNemar** viene impiegato quando le osservazioni non sono indipendenti. Un esempio tipico è rappresentato da uno studio clinico cross-over nel quale lo stesso soggetto viene esposto in tempi diversi a due trattamenti differenti, secondo una sequenza casuale (in modo che né il paziente né l'operatore sappiano quale è il trattamento effettuato in quella fase). In questo modo tutti i soggetti, ignorando ogni volta di quale trattamento si tratti, ricevono tutti e due i trattamenti.

Campbell[62] riporta i risultati di uno studio nel quale 250 pazienti sofferenti di artrite erano sottoposti al trattamento con il farmaco A e al trattamento con il farmaco B. Era stato poi rilevato il grado di

[59] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp. 282-285.
[60] Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, pp. 121-123 e 154-156.
[61] Ingelfinger JA, Mosteller F, Thibodeau LA, Ware JH. Biostatistica in medicina. Macmillan Publishing Co., Inc.; New York, 1983, ISBN 88-7078-065.1, pp. 30-33.
[62] Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, pp. 148-150.

soddisfazione di ciascuno di essi rispetto all'uno e all'altro trattamento, risultati qui riportati:

| *Esito* | *Soddisfatto da A* | *Non soddisfatto da A* |
|---|---|---|
| Soddisfatto da B | 150 | 20 |
| Non soddisfatto da B | 30 | 50 |

In totale 150 soggetti si sono mostrati soddisfatti di entrambi i trattamenti, 50 hanno espresso insoddisfazione per entrambi, mente altri 50 si sono mostrati insoddisfatti chi dell'uno e chi dell'altro.

Fate il `download` del file `chi_McNemar.csv` quindi copiatelo in `C:\Rdati\`

In alternativa potete anche copiare le tre righe riportate qui sotto e salvarle in `C:\Rdati\` nel file di testo denominato `chi_McNemar.csv` (attenzione all'estensione al momento del salvataggio del file).

```
Esito;Soddisfatto_da_A;Non_soddisfatto_da_A
Soddisfatto_da_B;150;20
Non_Soddisfatto_da_B;30;50
```

Questo è lo script per eseguire il test di McNemar impiegando la funzione **mcnemar.test()**[63]

```
# TEST DI MCNEMAR - 2 righe · 2 colonne                                    .a
#
mydata <- read.table("c:/Rdati/chi_McNemar.csv", header=TRUE, sep=";", row.names="Esito") #
# importa i dati
mydata # mostra i dati
matrix <- data.matrix(mydata) # viene generato l'oggetto matrice richiesto dalla fasi successive
mcnemar.test(matrix, correct=TRUE) # esegue il test di McNemar
#
```

I risultati del test di McNemar sono i seguenti

```
> mydata # mostra i dati
                  Soddisfatto_da_A Non_soddisfatto_da_A
Soddisfatto_da_B                150                  20
Non_soddisfatto_da_B             30                  50
> matrix <- data.matrix(mydata) # viene generato l'oggetto matrice
richiesto dalla fase successiva
> mcnemar.test(matrix, correct=TRUE) # esegue il test di McNemar

        McNemar's Chi-squared test with continuity correction

data:  matrix
McNemar's chi-squared = 1.62, df = 1, p-value = 0.2031
```

e dimostrano che la differenza tra i gradi di soddisfazione espressi dai pazienti in merito ai due trattamenti non è significativa.

In alternativa potete immettere i dati manualmente e calcolare il test di McNemar con quest'altro script,

[63] Digitate **help(mcnemar.test)** nella `Console di R` per la documentazione della funzione **mcnemar.test()**.

ottenendo ovviamente gli stessi risultati:

```
# TEST DI MCNEMAR  - 2 righe · 2 colonne                                    .b
#
cells <- c(150,20,30,50) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Soddisfatto_da_B", "Non_soddisfatto_da_B") # genera l'array rnames con i nomi delle
# righe
cnames <- c("Soddisfatto_da_A", "Non_soddisfatto_da_A") # genera l'array cnames con i nomi delle
# colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
mydata # mostra i dati
mcnemar.test(mydata, correct=TRUE) # esegue il test di McNemar
#
```

### 4.1.3. *n* righe · *n* colonne

Il **test chi-quadrato** può essere esteso anche a più (*n*) righe e più (*n*) colonne. Il problema è che più sono le righe e le colonne e più è difficile individuare il valore (o i valori) che contribuiscono a rendere eventualmente significative le differenze. Per questa ragione in genere le tabelle con più righe e più colonne sono scomposte in tabelle più piccole al fine di valutare meglio i contributi dei valori osservati. Per questo tipo di analisi, molto delicata, e che esula dai limiti di questa guida, si rimanda ai testi di statistica.

Qui riporto un esempio di una tabella di 2 righe per 5 colonne tratto da Marubini[64] nel quale si riesce ancora a individuare con facilità il risultato che contribuisce a rendere significativa la differenza tra le frequenze osservate.

Si trattava di valutare l'efficacia di cinque differenti vaccini influenzali. I vaccini vennero inoculati a novembre in 1080 soggetti sani volontari e nel mese di marzo dell'anno successivo vennero contati i casi di influenza tra i vaccinati.

Questi erano i risultati della sperimentazione. Da notare che viene riportato il numero dei casi cioè il conteggio dei soggetti affetti da influenza e non affetti da influenza, come richiesto dal **test chi-quadrato**, nel quale invece non devono mai essere riportati i dati espressi in percentuale:

| *Esito* | *Vaccino_1* | *Vaccino_2* | *Vaccino_3* | *Vaccino_4* | *Vaccino_5* |
|---|---|---|---|---|---|
| Influenza_si | 39 | 44 | 20 | 41 | 42 |
| Influenza_no | 177 | 166 | 200 | 183 | 168 |

Fate il `download` del file `chi_rxc.csv` quindi copiatelo in `C:\Rdati\`

In alternativa potete anche copiare le tre righe riportate qui sotto e salvarle in `C:\Rdati\` nel file di testo denominato `chi_rxc.csv` (attenzione all'estensione al momento del salvataggio del file).

Esito;Vaccino_1;Vaccino_2;Vaccino_3;Vaccino_4;Vaccino_5

[64] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp. 290-291.

```
Influenza_si;39;44;20;41;42
Influenza_no;177;166;200;183;168
```

Il test chi-quadrato viene calcolato con questo script:

```
# TEST CHI-QUADRATO - n righe · n colonne                                    .a
#
mydata <- read.table("c:/Rdati/Chi_rxc.csv", header=TRUE, sep=";", row.names="Esito") # importa i dati
mydata # mostra i dati
chisq.test(mydata) # calcola il chi-quadrato
#
```

In alternativa potete immettere i dati manualmente e calcolare il test con questo secondo script:

```
# TEST CHI-QUADRATO - n righe · n colonne                                    .b
#
cells <- c(39,44,20,41,42,177,166,200,183,168) # genera l'array cells con il numero dei casi
rnames <- c("Influenza_si", "Influenza_no") # genera l'array rnames con i nomi delle righe
cnames <- c("Vaccino_1", "Vaccino_2", "Vaccino_3", "Vaccino_4", "Vaccino_5") # genera l'array cnames
# con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=5, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
mydata # mostra i dati
chisq.test(mydata) # calcola il chi-quadrato
#
```

Questi sono in entrambi i casi i risultati del test

```
> mydata # mostra i dati
             Vaccino_1 Vaccino_2 Vaccino_3 Vaccino_4 Vaccino_5
Influenza_si        39        44        20        41        42
Influenza_no       177       166       200       183       168
> chisq.test(mydata) # calcola il chi-quadrato

        Pearson's Chi-squared test

data:  mydata
X-squared = 13.678, df = 4, p-value = 0.008395
```

che riporta un valore molto elevato della statistica chi-quadrato (13.678) cui corrisponde un valore di probabilità `p = 0.008395` che conferma quindi la presenza di una differenza significativa nell'efficacia dei vaccini.

Si tratta ora di capire a quale/i vaccino/i debba essere attribuita la differente efficacia senza entrare in analisi troppo complesse. In casi di questo genere può essere di aiuto integrare il risultato numerico con una rappresentazione grafica dei dati impiegati per calcolare il chi-quadrato.

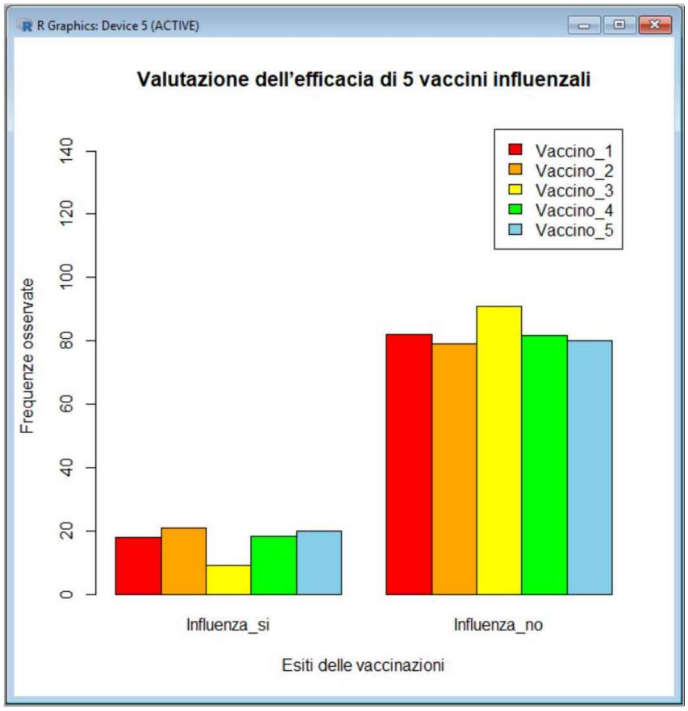



**Fig. 4.1** Un grafico a barre aiuta a identificare nel vaccino 3 (in colore giallo) il vaccino i cui risultati determinano la significatività del test chi-quadrato.

Ripartiamo dai casi osservati (Influenza_si / Influenza_no) espressi in percentuale

| *Esito (%)* | *Vaccino_1* | *Vaccino_2* | *Vaccino_3* | *Vaccino_4* | *Vaccino_5* |
|---|---|---|---|---|---|
| Influenza_si | 18.1 | 21.0 | 9.1 | 18.3 | 20.0 |
| Influenza_no | 81.9 | 79.0 | 90.8 | 81.7 | 80.0 |

per generare la nuova matrice dati **mydata** e per rappresentare i dati sotto forma di un grafico a barre (**Fig. 4.1**) realizzato mediante la funzione **barplot()**[65]:

```
# GRAFICO A BARRE PER L'EFFICACIA DI 5 VACCINI
#
cells <- c(18.1,21,9.1,18.3,20,81.9,79,90.9,81.7,80) # genera l'array cells con i valori di percentuale
rnames <- c("Influenza_si", "Influenza_no") # genera l'array rnames con i nomi delle righe
cnames <- c("Vaccino_1", "Vaccino_2", "Vaccino_3", "Vaccino_4", "Vaccino_5") # genera l'array cnames
# con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=5, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
mydata # mostra i dati
windows() # apre e inizializza una nuova finestra grafica
barplot(t(mydata), beside=TRUE, legend=TRUE, ylim=c(0,150), col=c("red","orange", "yellow", "green",
"skyblue"), ylab="Frequenze osservate", xlab="Esiti delle vaccinazioni", main="Valutazione dell'efficacia
di 5 vaccini influenzali") # traccia il grafico a barre verticali
#
```

[65] La funzione viene illustrata nel capitolo **5. R - rappresentazione grafica dei dati**. Digitate **help(barplot)** nella `Console di R` per la documentazione della funzione **barplot()**.

In effetti il grafico a barre ci aiuta a individuare la causa della differenza significativa rilevata dal test chi-quadrato nel vaccino 3 che appare quello con il quale si sono verificati meno casi di influenza nei soggetti vaccinati, mentre tutti gli altri vaccini mostrano percentuali di influenza tutte attorno al 20%, molto simili tra loro e praticamente doppie rispetto al vaccino 3.

**Nota bene**: effettuate una pulizia dell'area di lavoro per evitare interferenze con la sessione successiva[66].

Potete farlo anche con questa riga di codice:

```
q(save="no") # esce da R senza salvare l'area di lavoro che al rientro in R risulterà ripulita
```

---

[66] Vedere: **2.7 Pulizia dell'area di lavoro e uscita dal programma**. Al bisogno cancellate i due file denominati `.Rdata` e `.rhistory` che si trovano nella cartella `Documenti`.

## 4.2. Statistiche elementari

I dati che ora impiegheremo sono contenuti nel set di dati **ais** del pacchetto **DAAG**. Scaricate il pacchetto **DAAG** dal **CRAN** selezionando `Pacchetti >> Installa pacchetti...`[67]

Potete accedere alla documentazione del pacchetto **DAAG** selezionando `Aiuto >> Guida Html >> Packages >> DAAG`. Per scaricare il manuale di riferimento collegarsi al **CRAN**[68].

Nell'oggetto/set di dati **ais** sono contenuti **dati rilevati su 202 atleti australiani**. I dati sono riportati sotto forma di tabella nell'appendice alla quale si rimanda per la descrizione delle variabili che lo compongono[69].

Dopo avere installato il pacchetto **DAAG** basterà caricarlo dalla libreria con questa riga di codice

```
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
```

per caricare in **R** il set di dati **ais**. Da notare che se nella `Console di R` digitate
```
data()
```
si apre una finestra nella quale sono elencati anche gli altri set di dati caricati con il pacchetto[70].

A questo punto per presentare il minimo essenziale delle funzioni di **R** applicabili ad una analisi elementare di dati campionari, cercheremo di seguire un percorso logico, che prevede per ciascuna delle variabili in esame questi punti:
→ **esplorazione** preliminare dei dati;
→ esecuzione dei **test** per valutare se i dati seguono una distribuzione gaussiana (**test di normalità**);
→ calcolo delle **statistiche elementari parametriche** (media, deviazione standard, varianza) nei casi in cui la distribuzione dei dati è una distribuzione gaussiana;
→ calcolo delle **statistiche elementari non parametriche** (mediana, deviazione assoluta mediana, quartili, percentili) nei casi in cui la distribuzione dei dati non è una distribuzione gaussiana.

I punti sono sviluppati in più parti, mediante uno script (.a) che analizza tutte le variabili del set di dati e uno script (.b) che approfondisce l'analisi della sola variabile `ferr` (ferritina). La numerazione fornita consente al termine di salvare separatamente i due script.

Prima di proseguire scaricate dal **CRAN** anche i pacchetti **nortest**, **Hmisc**, **pastecs**, **psych** selezionandoli da `Barra dei menù >> Pacchetti >> Installa pacchetti...` Trovate la documentazione dei pacchetti sul repository della documentazione di **R**[71].

### 4.2.1. Esplorazione preliminare dei dati

Con queste prime righe di codice viene caricato il pacchetto **DAAG**, sono mostrati il contenuto dell'oggetto/set di dati **ais** e la sua struttura, sono mostrati i casi/righe per i quali eventualmente manca qualche dato e infine è presentato un riepilogo con il calcolo delle statistiche elementari di tutte le variabili,

[67] Come indicato al paragrafo **2.6. Pacchetti aggiuntivi di statistica e grafica**.
[68] *Package 'DAAG'*. URL consultato il 20/10/2018: https://goo.gl/gzKabK
[69] Vedere: **A3. Il set di dati ais**.
[70] Vedere: **2.6. Pacchetti aggiuntivi di statistica e grafica**. Per la documentazione della funzione **data()** digitate **help(data)** nella `Console di R`.
[71] *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

sia quelle numeriche sia quelle non numeriche:

```
# STATISTICHE ELEMENTARI                                                    1/5.a
#
# esplorazione preliminare dei dati, tutte le variabili
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
ais # mostra ais
str(ais) # mostra la struttura di ais
head(ais, n=5) # lista dei primi 5 casi di ais
tail(ais, n=5) # lista degli ultimi 5 casi di ais
ais[!complete.cases(ais),] # mostra i casi con dati NA (Not Available), ma qui non ce ne sono
#
summary(ais) # statistiche elementari per tutte le variabili (numeriche e non) di ais
#
```

La funzione **str(ais)** ci dice che l'oggetto consiste in una tabella (`dataframe` o `dataset`) che contiene 202 osservazioni di 13 variabili, che sono poi elencate nelle righe successive, 11 variabili di tipo numerico, e due variabili qualitative, sostanzialmente delle etichette, che in **R** sono denominate "fattori" e che sono molto importanti in quanto come vedremo consentono di aggregare i record/casi in sottoinsiemi, e per esempio consentono in base al valore della variabile/fattore `sesso` di separare i risultati dei soggetti di sesso femminile (`f`) da quelli di sesso maschile (`m`).

```
> str(ais) # mostra la struttura di ais
'data.frame':   202 obs. of  13 variables:
 $ rcc   : num  3.96 4.41 4.14 4.11 4.45 4.1 4.31 4.42 4.3 4.51 ...
 $ wcc   : num  7.5 8.3 5 5.3 6.8 4.4 5.3 5.7 8.9 4.4 ...
 $ hc    : num  37.5 38.2 36.4 37.3 41.5 37.4 39.6 39.9 41.1 41.6 ...
 $ hg    : num  12.3 12.7 11.6 12.6 14 12.5 12.8 13.2 13.5 12.7 ...
 $ ferr  : num  60 68 21 69 29 42 73 44 41 44 ...
 $ bmi   : num  20.6 20.7 21.9 21.9 19 ...
 $ ssf   : num  109.1 102.8 104.6 126.4 80.3 ...
 $ pcBfat: num  19.8 21.3 19.9 23.7 17.6 ...
 $ lbm   : num  63.3 58.5 55.4 57.2 53.2 ...
 $ ht    : num  196 190 178 185 185 ...
 $ wt    : num  78.9 74.4 69.1 74.9 64.6 63.7 75.2 62.3 66.5 62.9 ...
 $ sex   : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
 $ sport : Factor w/ 10 levels "B_Ball","Field",..: 1 1 1 1 1 1 1 1 1
1 ...
```

Non risultano casi di dati `NA` (`Not Available`).

```
> ais[!complete.cases(ais),] # mostra i casi con dati NA (Not Available),
ma qui non ce ne sono
 [1] rcc    wcc    hc     hg     ferr   bmi    ssf    pcBfat lbm    ht
[11] wt     sex    sport
<0 rows> (or 0-length row.names)
```

Se ora fissiamo la nostra attenzione sulle righe che riportano media e mediana, le statistiche fornite dalla funzione **summary()** forniscono già qualche informazione significativa:

```
      rcc              wcc              hc              hg
 Min.   :3.800   Min.   : 3.300   Min.   :35.90   Min.   :11.60
 1st Qu.:4.372   1st Qu.: 5.900   1st Qu.:40.60   1st Qu.:13.50
 Median :4.755   Median : 6.850   Median :43.50   Median :14.70
 Mean   :4.719   Mean   : 7.109   Mean   :43.09   Mean   :14.57
 3rd Qu.:5.030   3rd Qu.: 8.275   3rd Qu.:45.58   3rd Qu.:15.57
 Max.   :6.720   Max.   :14.300   Max.   :59.70   Max.   :19.20

      ferr             bmi              ssf             pcBfat
 Min.   :  8.00   Min.   :16.75   Min.   : 28.00   Min.   : 5.630
 1st Qu.: 41.25   1st Qu.:21.08   1st Qu.: 43.85   1st Qu.: 8.545
 Median : 65.50   Median :22.72   Median : 58.60   Median :11.650
 Mean   : 76.88   Mean   :22.96   Mean   : 69.02   Mean   :13.507
 3rd Qu.: 97.00   3rd Qu.:24.46   3rd Qu.: 90.35   3rd Qu.:18.080
 Max.   :234.00   Max.   :34.42   Max.   :200.80   Max.   :35.520

      lbm              ht              wt         sex          sport
 Min.   : 34.36   Min.   :148.9   Min.   : 37.80   f:100   Row    :37
 1st Qu.: 54.67   1st Qu.:174.0   1st Qu.: 66.53   m:102   T_400m :29
 Median : 63.03   Median :179.7   Median : 74.40           B_Ball :25
 Mean   : 64.87   Mean   :180.1   Mean   : 75.01           Netball:23
 3rd Qu.: 74.75   3rd Qu.:186.2   3rd Qu.: 84.12           Swim   :22
 Max.   :106.00   Max.   :209.4   Max.   :123.20           Field  :19
                                                           (Other):47
```

In una distribuzione gaussiana media e mediana sono identiche. Qui si osservano tra le due in genere valori abbastanza simili, tranne che nel caso della ferritina (`ferr`), della somma dello spessore delle pliche cutanee (`ssf`) e della percentuale di grasso corporeo (`pcBfat`). Si presenta quindi la necessità di procedere ad un approfondimento con la seconda parte dello script riportata qui sotto.

### 4.2.2. Test di normalità

```
# test di normalità, test di Lilliefors, tutte le variabili                    2/5.a
#
mydata <- ais[c(1,2,3,4,5,6,7,8,9,10,11)] # salva le colonne con le sole variabili numeriche in mydata
str(mydata) # mostra la struttura di mydata
#
library(nortest) # carica il pacchetto
lillie.test(mydata$rcc) # test di Lilliefors (Kolmogorov-Smirnov) per la normalità su eritrociti
lillie.test(mydata$wcc) # idem su leucociti
lillie.test(mydata$hc) # idem su ematocrito
lillie.test(mydata$hg) # idem su emoglobina
lillie.test(mydata$ferr) # idem su ferritina
lillie.test(mydata$bmi) # idem su indice di massa corporea
lillie.test(mydata$ssf) # idem su stima dello spessore delle pliche cutanee
lillie.test(mydata$pcBfat) # idem su percentuale di grasso corporeo
lillie.test(mydata$lbm) # idem su massa magra
lillie.test(mydata$ht) # idem su altezza
lillie.test(mydata$wt) # idem su peso
#
```

L'approfondimento consiste nell'esecuzione dei **test di normalità** per rendere oggettive le impressioni

ricavate dalla esplorazione preliminare dei dati. Viene generato il nuovo oggetto **mydata** che include le sole variabili numeriche (`sesso` e `sport praticato` non lo sono) e ne viene mostrata la struttura. Quindi viene caricato il pacchetto **nortest** e viene eseguito il test di Lilliefors (Kolmogorov-Smirnov)[72] su tutte le variabili numeriche:

Potete constatare che le variabili che con il test di Lilliefors in temini di `p` si discostano maggiormente da una distribuzione gaussiana sono in effetti proprio (i) la ferritina, (ii) la somma dello spessore delle pliche cutanee e (iii) la percentuale di grasso corporeo. Limitiamoci d'ora in poi per semplicità alla sola ferritina, per la quale viene riportato un valore di `p = 0.0000001091` (`1.091e-07`), valore quindi altamente significativo:

```
> lillie.test(mydata$ferr) # idem su ferritina

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  mydata$ferr
D = 0.12163, p-value = 1.091e-07
```

Possiamo passare all'analisi specifica dei valori della ferritina aggiungendo al test di Lilliefors quattro ulteriori test di normalità[73], anch'essi inclusi nel pacchetto **nortest:**

```
# STATISTICHE ELEMENTARI                                          1/5.b
#
# analisi specifica sui dati della ferritina
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
mydata <- ais[c(1,2,3,4,5,6,7,8,9,10,11)] # salva le colonne con le sole variabili numeriche in mydata
library(nortest) # carica il pacchetto
ad.test(mydata$ferr) # test di Anderson-Darling per la normalità
cvm.test(mydata$ferr) # test di Cramer-von Mises per la normalità
pearson.test(mydata$ferr) # test chi-quadrato di Pearson per la normalità
sf.test(mydata$ferr) # test di Shapiro-Francia per la normalità
#
```

In effetti tutti e quattro questi test di normalità confermano l'importante scostamento della distribuzione dei valori della ferritina da una distribuzione gaussiana:

```
> ad.test(mydata$ferr) # test di Anderson-Darling per la normalità

        Anderson-Darling normality test

data:  mydata$ferr
A = 6.097, p-value = 4.904e-15

> cvm.test(mydata$ferr) # test di Cramer-von Mises per la normalità

        Cramer-von Mises normality test
```

[72] Il test di Lilliefors è un test di normalità basato sul test di normalità di Kolmogorov-Smirnov, digitate **help(lillie.test)** nella `Console di R` per la documentazione della funzione **lillie.test()**.
[73] Per la loro documentazione digitate come al solito **help(*nomedellafunzione*)** nella `Console di R`.

```
data:  mydata$ferr
W = 0.94473, p-value = 2.495e-09

> pearson.test(mydata$ferr) # test chi-quadrato di Pearson per la
normalità

        Pearson chi-square normality test

data:  mydata$ferr
P = 63.772, p-value = 2.531e-08

> sf.test(mydata$ferr) # test di Shapiro-Francia per la normalità

        Shapiro-Francia normality test

data:  mydata$ferr
W = 0.89138, p-value = 1.251e-09
```

Ai valori numerici forniti dai cinque test di normalità effettuati, dal test di Lilliefors al test di Shapiro-Francia, già di per sé decisivi, aggiungiamo ora quattro grafici con questo script:

```
# approfondimenti grafici sulla distribuzione della ferritina                    2/5.b
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
hist(mydata$ferr, main="Istogramma", xlab="Ferritina in µg/L", ylab = "Frequenza", xlim = c(0,250)) #
# traccia l'istogramma
#
plot(density(mydata$ferr), main="Kernel density plot", xlab="Ferritina in µg/L", ylab = "Frequenza", xlim
= c(0,250)) # traccia il kernel density plot
#
zetavector<-(mydata$ferr-mean(mydata$ferr))/sd(mydata$ferr) # crea un vettore con le deviate normali
# standardizzate dei dati campionari
qqnorm((zetavector), main="Quantili campionari vs. teorici", xlab="Quantili teorici", ylab = "Quantili
campionari", xlim = c(-1.62,3.25), ylim = c(-1.62,3.25)) # traccia il grafico quantili campionari vs. quantili
# teorici
abline (0,1) # la gaussiana in questo caso è una semplice retta
#
plot(ecdf(mydata$ferr), main="Cumulativa campionaria vs. teorica", xlab="Ferritina in µg/L", ylab =
"Frequenza cumulativa", xlog = FALSE, ylog = FALSE, xlim = c(0,250), ylim = c(0,1), xaxp = c(0, 250, 5), yaxp
= c(0,1,5)) # traccia il grafico della distribuzione cumulativa empirica (del campione)
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
plot(ecdf(rnorm(10000,mean=76.87624,sd=47.50124)), xlog = FALSE, ylog = FALSE, xlim = c(0,250), ylim =
c(0,1), xaxp = c(0, 250, 5), yaxp = c(0,1,5)) # traccia il grafico della distribuzione cumulativa teorica
#
```

I grafici sono:
→ un semplice istogramma;
→ un kernel density plot;
→ un grafico che riporta i quantili campionari in funzione dei quantili teorici e nel quale i dati, se distribuiti in modo gaussiano, sono allineati con la retta teorica;
→ un grafico che riporta la distribuzione cumulativa campionaria e nel quale i dati, se distribuiti in modo

gaussiano, sono allineati con la curva della distribuzione cumulativa teorica.

Questi grafici rappresentano un anticipo di quello che verrà trattato nel capitolo successivo dedicato alle funzioni grafiche di **R**[74]**.** Il risultato del codice della pagina precedente è riportato nella **Fig. 4.2**.

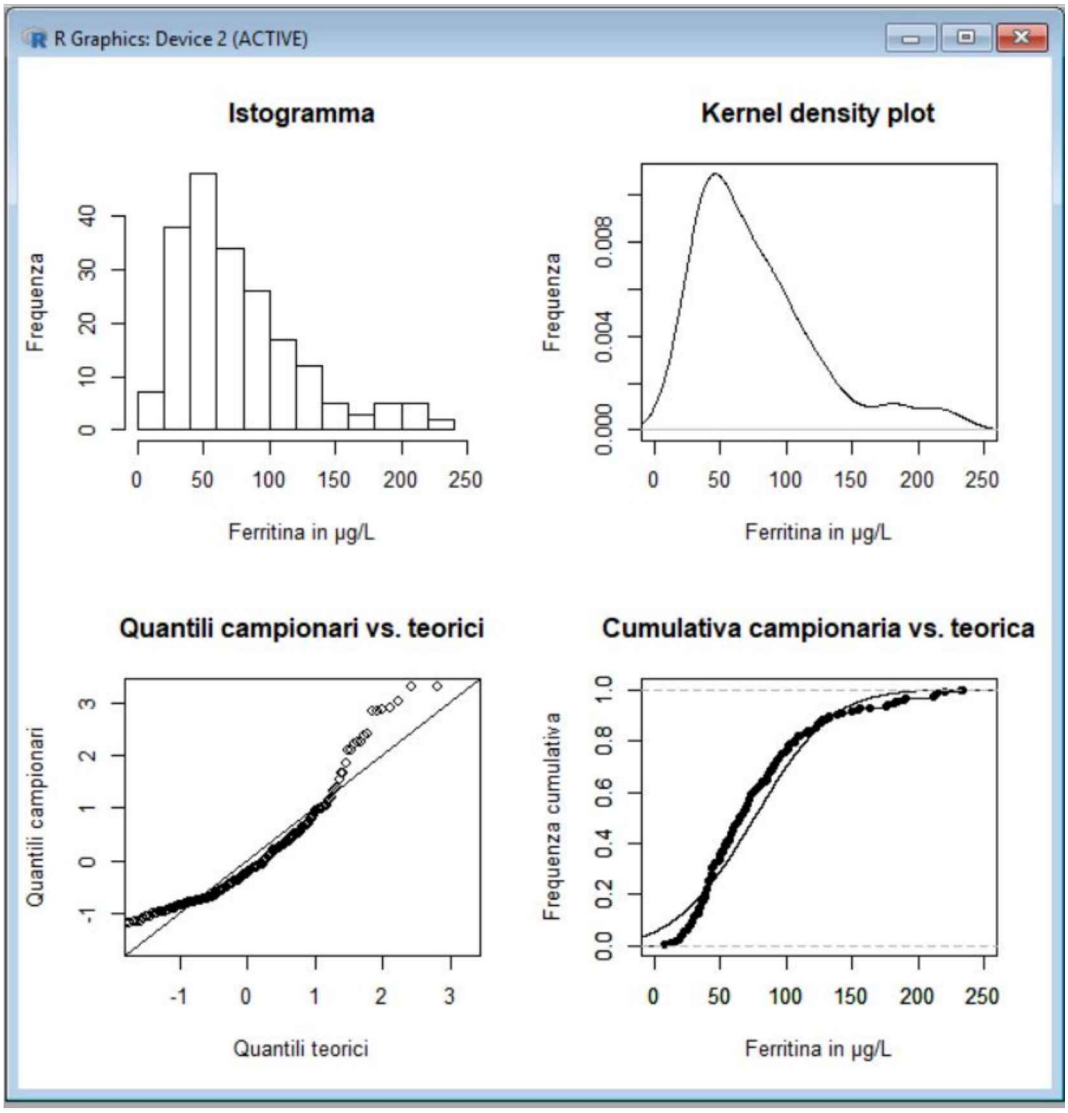



**Fig. 4.2** Quattro grafici confermano visivamente il risultato dei test di normalità, che hanno indicato un significativo scostamento della distribuzione dei valori della transferrina da una distribuzione gaussiana.

Vale la pena di aggiungere alcune considerazioni sul codice che ha generato la **Fig. 4.2**.

Innanzitutto con la funzione **par(mfrow=c(2,2))** la finestra grafica viene suddivisa in quattro quadranti, che verranno riempiti da sinistra a destra e dall'alto verso il basso dai quattro grafici generati con i successivi blocchi di codice.

[74] Vedere: **5. R - rappresentazione grafica dei dati**.

Gli argomenti della funzione **hist()** qui riportata pur non essendo esaustivi (ve ne sono moltissimi altri possibili, per questo si rimanda alla documentazione di R) servono a illustrare alcuni elementi che in **R** sono comuni a tutte le funzioni grafiche:
→ **mydata$ferr** è ia varibile da analizzare, che varierà ovviamente di caso in caso;
→ **main=""** è i titolo del grafico che compare in alto;
→ **xlab=""** è l'etichetta riportata per l'asse delle ascisse x;
→ **ylab=""** è l'etichetta riportata per l'asse delle ordinate y;
→ **xlim = c(0,250)** indica limite inferiore e limite superiore della scala applicata all'asse delle ascisse, trattandosi di valori di concentrazione della ferritina espressi in µg/L questo significa che l'asse delle ascisse riporterà i valori compresi tra 0 e 250 µg/L.

Nella funzione **plot()** dell'ultimo blocco di codice compaio alcuni altri argomenti[75] utili per strutturare il grafico:
→ **xlog = FALSE, ylog = FALSE** ci dicono che la scala non sarà logaritmica, bensì sarà lineare;
→ **ylim = c(0,1)** indica limite inferiore e limite superiore della scala applicata all'asse delle ordinate, trattandosi di valori di probabilità varieranno ovviamente tra 0 e 1;
→ **xaxp = c(0, 250, 5)** dice che sull'asse delle ascisse la scala che va da 0 a 250 µg/L deve essere suddivisa in cinque intervalli, quindi sull'asse delle ascisse compariranno sei tacche con i valori 0, 50, 100, 150, 200, 250;
→ **yaxp = c(0,1,5)** dice che sull'asse delle ordinate la scala che va da 0 a 1L deve essere suddivisa in cinque intervalli, quindi sull'asse delle ordinate compariranno sei tacche con i valori 0, 0.2, 0.4, 0.6, 0.8, 1.0.

Ritornando alle osservazioni che si ricavano dall'analisi dei dati mediante i grafici della **Fig. 4.2**, istogramma e kernel density plot confermano la presenza di una asimmetria nella distribuzione dei dati e di una estesa coda sulla destra, nei valori alti. Confermano la asimmetria i risultati della differenza tra quantili campionari e quantili teorici (questi ultimi rappresentati dalla retta che si estende dall'angolo inferiore sinistro all'angolo superiore destro, sulla quale dovrebbero essere allineati i valori campionari se fossero distribuiti in modo gaussiano) e della differenza tra distribuzione cumulativa campionaria e distribuzione cumulativa teorica (quest'ultima rappresentata dalla curva sulla quale dovrebbero essere allineati i valori campionari se fossero distribuiti in modo gaussiano).

Al termine di questa analisi abbiamo la conferma del fatto che per descrivere la variabile ferritina dovremo utilizzare le **statistiche non parametriche** (mediana, deviazione assoluta mediana (MAD), quartili, percentili). Lo stesso vale per la somma dello spessore delle pliche cutanee e la percentuale di grasso corporeo, per le quali l'approfondimento viene lasciato come esercizio. Per altre variabili, come ad esempio ematocrito ed emoglobina, che non si discostano significativamente da una distribuzione gaussiana, può essere appropriato l'impiego delle **statistiche parametriche** (media, deviazione standard, varianza) che vediamo ora.

### 4.2.3. Statistiche elementari parametriche

Con la funzione **sapply()** le singole statistiche elementari parametriche possono essere calcolate contemporaneamente per tutte le variabili della tabella **mydata** come riportato nel codice che segue:

```
# statistiche elementari parametriche, tutte le variabili                3/5.a
#
sapply(mydata, mean) # media su tutte le variabili
sapply(mydata, sd) # deviazione standard
sapply(mydata, var) # varianza
```

[75] Per gli argomenti **xlog**, **ylog**, **xlim**, **ylim**, **xaxp**, **yaxp** vedere anche la funzione **par()** con **help(par)**.

```
sapply(mydata, min) # valore minimo
sapply(mydata, max) # valore massimo
sapply(mydata, range) # range
#
```

con questi risultati:

```
> sapply(mydata, mean) # media su tutte le variabili
       rcc        wcc         hc         hg       ferr        bmi        ssf
  4.718614   7.108911  43.091584  14.566337  76.876238  22.955891  69.021782
    pcBfat        lbm         ht         wt
 13.507426  64.873713 180.103960  75.007921
> sapply(mydata, sd) # deviazione standard
       rcc        wcc         hc         hg       ferr        bmi        ssf
 0.4579764  1.8003371  3.6629894  1.3624515 47.5012388  2.8639328 32.5653330
    pcBfat        lbm         ht         wt
 6.1898260 13.0701972  9.7344945 13.9251995
> sapply(mydata, var) # varianza
         rcc          wcc           hc           hg         ferr          bmi
   0.2097423    3.2412137   13.4174910    1.8562741 2256.3676912    8.2021109
         ssf       pcBfat          lbm           ht           wt
1060.5009162   38.3139456  170.8300553   94.7603822  193.9111807
> sapply(mydata, min) # valore minimo
   rcc    wcc     hc     hg   ferr    bmi    ssf pcBfat    lbm     ht     wt
  3.80   3.30  35.90  11.60   8.00  16.75  28.00   5.63  34.36 148.90  37.80
> sapply(mydata, max) # valore massimo
   rcc    wcc     hc     hg   ferr    bmi    ssf pcBfat    lbm     ht     wt
  6.72  14.30  59.70  19.20 234.00  34.42 200.80  35.52 106.00 209.40 123.20
> sapply(mydata, range) # range
      rcc  wcc   hc   hg ferr   bmi   ssf pcBfat    lbm    ht    wt
[1,] 3.80  3.3 35.9 11.6    8 16.75  28.0   5.63  34.36 148.9  37.8
[2,] 6.72 14.3 59.7 19.2  234 34.42 200.8  35.52 106.00 209.4 123.2
```

Va da sé che **media**, deviazione standard (**ds**) e varianza (**var**) sono statistiche non applicabili alle variabili che non sono distribuite in modo gaussiano. Questo, come vedremo tra poco, è un passo fondamentale al fine di trarre conclusioni statistiche corrette.

Alcuni altri pacchetti come **Hmisc, pastecs** e **psych** consentono di effettuare una analisi riepilogativa dei dati:

```
# analisi riepilogativa dei dati, tutte le variabili                    4/5.a
#
library(Hmisc) # carica il pacchetto
describe(mydata) # statistiche del pacchetto Hmisc
#
library(pastecs) # carica il pacchetto
stat.desc(mydata) # statistiche del pacchetto pastecs
#
library(psych) # carica il pacchetto
describe(mydata) # statistiche del pacchetto psych
describeBy(mydata, ais$sex) # statistiche del pacchetto psych separate per sesso
describeBy(mydata, ais$sport) # statistiche del pacchetto psych separate per sport
#
```

Sembrano essere superiori per chiarezza nella presentazione i risultati forniti dalla funzione **describe(mydata)** del pacchetto **psych**

```
> describe(mydata) # statistiche del pacchetto psych
        vars   n   mean    sd median trimmed   mad    min    max  range  skew kurtosis   se
rcc        1 202   4.72  0.46   4.75    4.71  0.51   3.80   6.72   2.92  0.41     0.63 0.03
wcc        2 202   7.11  1.80   6.85    7.00  1.56   3.30  14.30  11.00  0.83     1.41 0.13
hc         3 202  43.09  3.66  43.50   43.09  3.71  35.90  59.70  23.80  0.27     0.85 0.26
hg         4 202  14.57  1.36  14.70   14.55  1.48  11.60  19.20   7.60  0.17    -0.02 0.10
ferr       5 202  76.88 47.50  65.50   70.14 37.81   8.00 234.00 226.00  1.27     1.38 3.34
bmi        6 202  22.96  2.86  22.72   22.77  2.49  16.75  34.42  17.67  0.94     2.13 0.20
ssf        7 202  69.02 32.57  58.60   65.20 26.02  28.00 200.80 172.80  1.17     1.32 2.29
pcBfat     8 202  13.51  6.19  11.65   12.92  6.52   5.63  35.52  29.89  0.75    -0.20 0.44
lbm        9 202  64.87 13.07  63.03   64.47 13.78  34.36 106.00  71.64  0.36    -0.27 0.92
ht        10 202 180.10  9.73 179.70  180.28  8.97 148.90 209.40  60.50 -0.20     0.49 0.68
wt        11 202  75.01 13.93  74.40   74.84 13.57  37.80 123.20  85.40  0.24     0.35 0.98
```

che includono il risultato della **MAD**, una statistica interessante in quanto rappresenta l'equivalente non parametrico della deviazione standard[76].

La funzione **describe()** del pacchetto **psych** consente non solo di calcolare le statistiche elementari per tutte le variabili, ma consente anche di calcolarle per ciascuna di esse aggregando i casi/righe in sottoinsiemi specificati da variabili qualitative. Nel nostro caso nella penultima riga dello script 4/5.a con
**describeBy(mydata, ais$sex)**
abbiamo aggregato i dati di tutte le variabili nei due sottoinsiemi `m` e `f` specificati dalla variabile `sex` ottenendo le statistiche separate per uomini e donne, mentre nell'ultima riga con
**describeBy(mydata, ais$sport)**
li abbiamo aggregati nei dieci sottoinsiemi specificati dalla variabile `sport` ottenendo le statistiche separate per ciascuno di essi. Il tutto viene ricordato in quanto di grande utilità pratica (per brevità i risultati aggregati per sesso e per sport sono omessi[77]).

### 4.2.4. Statistiche elementari non parametriche

Applicare statistiche parametriche a una variabile che non è distribuita in modo gaussiano è inappropriato e porta a conclusioni sbagliate. In base alle proprietà della distribuzione gaussiana sappiamo che tra la media - 1.96 · ds e la media + 1.96 · ds si deve trovare il 95% dei dati campionari. Prendiamo ora il valore della media della ferritina che (arrotondato) è 76.9 µg/L, prendiamo il valore della sua deviazione standard che è 47.5 µg/L e lo moltiplichiamo per 1.96 ottenendo 93.1. Da questo dovremmo dedurre che il 95% dei valori misurati negli atleti australiani cadeva nell'intervallo

$$76.9 \pm 93.1 \ \mu g/L$$

e pertanto concludere che in una certa quota degli atleti la concentrazione nel sangue della ferritina aveva valori inferiori a zero, visto che 76.9 - 93.1.4 = - 16.2. Questo evidente controsenso consente di individuare un grave errore nella metodologia statistica impiegata: non si possono applicare media e deviazione standard a dati che non sono distribuiti in modo gaussiano.

Poiché media e deviazione standard sono i "parametri" che descrivono una distribuzione gaussiana, i

---

[76] "*Median Absolute Deviation (about the median)*" o MAD, cioè "*deviazione assoluta mediana (attorno alla mediana)*". Viene calcolata come mediana delle differenze (prese in valore assoluto) tra ciascuno dei valori osservato e la loro mediana, moltiplicata per il fattore b = 1.4826. Digitate **help(mad)** nella Console di R per la documentazione della funzione **mad()**. Vedere anche Rousseeuw PJ, Croux C. *Alternatives to the Median Absolute Deviation*. Journal of the American Statistical Association 88 (424), 1273-1283, 1993. URL consultato il 30/08/2018: https://goo.gl/4Rh53b
[77] Digitate **help(psych)** nella Console di R per la documentazione del pacchetto e accedete al suo manuale di riferimento da *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

metodi statistici che fanno ricorso ad assunti preliminari di gaussianità dei dati sono detti **metodi parametrici**, in contrapposizione a quelli che non sono basati su assunti di gaussianità, che sono detti **metodi non parametrici**. Se i dati non sono distribuiti in modo gaussiano è necessario utilizzare test statistici/metodi non parametrici. Nel caso delle statistiche elementari, l'alternativa non parametrica a media e deviazione standard è rappresentata dalla mediana e dalla deviazione assoluta mediana (**MAD**), dalla distanza interquartile e dai quantili (o frattili) non parametrici: quartili, decili, percentili.

Questo codice

```
# statistiche elementari non parametriche, tutte le variabili                5/5.a
#
sapply(mydata, median) # mediana
sapply(mydata, mad) # deviazione assoluta mediana (MAD)
sapply(mydata, min) # valore minimo
sapply(mydata, max) # valore massimo
sapply(mydata, range) # range
sapply(mydata, quantile) # quartili
#
```

calcola le statistiche elementari, su tutte le variabili, con metodi non parametrici che possono essere applicati in modo appropriato sia a distribuzioni gaussiane sia a distribuzioni non gaussiane.

Questo codice invece calcola le stesse statistiche non parametriche ma le applica alla sola ferritina

```
# statistiche elementari non parametriche della ferritina                    3/5.b
#
median(mydata$ferr) # mediana
mad(mydata$ferr) # deviazione assoluta mediana (MAD)
min(mydata$ferr) # valore minimo
max(mydata$ferr) # valore massimo
range(mydata$ferr) # range
quantile(mydata$ferr) # quartili
#
```

con questi risultati

```
> median(mydata$ferr) # mediana della ferritina
[1] 65.5
> mad(mydata$ferr) # deviazione assoluta mediana (MAD)
[1] 37.8063
> min(mydata$ferr) # valore minimo
[1] 8
> max(mydata$ferr) # valore massimo
[1] 234
> range(mydata$ferr) # range
[1]   8 234
> quantile(mydata$ferr) # quartili
    0%    25%    50%    75%   100%
  8.00  41.25  65.50  97.00 234.00
```

Nella funzione **quantile()** può essere specificato l'argomento **probs** che richiede un vettore contenente i

valori di probabilità ai quali calcolare i valori di concentrazione della ferritina corrispondenti. Onde evitare di digitarli uno per uno (nel caso dei percentili bisognerebbe digitale uno per uno i centouno valori 0, 0.01, 0.02, 0.03 … 0.97, 0.98, 0.99, 1.0) si fa ricorso alla funzione **seq()** che li genera dati il valore iniziale (0), il valore finale (1) e l'incremento da applicare, 0.25 per i quartili, 0.1 per i decili, e 0.01 per i percentili. Questo è il codice

```
# quartili, decili e percentili della ferritina                                    4/5.b
#
quantile(mydata$ferr, probs = seq (0, 1, 0.25)) # con 0.25 calcola i quartili della ferritina
quantile(mydata$ferr, probs = seq (0, 1, 0.1)) # con 0.1 calcola i decili
quantile(mydata$ferr, probs = seq (0, 1, 0.01)) # con 0.01 calcola i percentili
#
```

e questi sono i risultati

```
> quantile(mydata$ferr, probs = seq (0, 1, 0.25)) # con 0.25 calcola i quartili della ferritina
    0%    25%    50%    75%   100%
  8.00  41.25  65.50  97.00 234.00
> quantile(mydata$ferr, probs = seq (0, 1, 0.1)) # con 0.10 calcola i decili
   0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
  8.0  30.0  39.2  44.0  55.0  65.5  76.0  90.7 107.0 138.4 234.0
> quantile(mydata$ferr, probs = seq (0, 1, 0.01)) # con 0.01 calcola i percentili
    0%     1%     2%     3%     4%     5%     6%     7%     8%     9%    10%    11%    12%    13%
  8.00  13.03  19.02  20.03  21.04  22.00  25.06  26.07  29.00  29.09  30.00  30.22  32.12  34.00
   14%    15%    16%    17%    18%    19%    20%    21%    22%    23%    24%    25%    26%    27%
 34.14  35.15  36.00  36.17  38.00  39.00  39.20  40.00  40.22  41.00  41.00  41.25  43.00  43.00
   28%    29%    30%    31%    32%    33%    34%    35%    36%    37%    38%    39%    40%    41%
 43.28  44.00  44.00  45.31  46.64  48.33  50.00  50.00  51.00  52.37  53.00  53.39  55.00  56.41
   42%    43%    44%    45%    46%    47%    48%    49%    50%    51%    52%    53%    54%    55%
 58.00  58.00  58.44  59.45  60.46  61.47  63.48  64.00  65.50  66.51  68.00  69.00  70.54  71.00
   56%    57%    58%    59%    60%    61%    62%    63%    64%    65%    66%    67%    68%    69%
 72.00  72.57  73.00  73.59  76.00  77.61  79.24  81.26  83.28  85.65  86.66  87.00  88.00  89.69
   70%    71%    72%    73%    74%    75%    76%    77%    78%    79%    80%    81%    82%    83%
 90.70  91.71  93.00  94.00  97.00  97.00 100.52 101.77 102.00 105.37 107.00 109.00 109.82 115.00
   84%    85%    86%    87%    88%    89%    90%    91%    92%    93%    94%    95%    96%    97%
117.84 122.00 124.00 124.87 126.88 131.78 138.40 142.82 154.60 163.44 176.94 182.95 188.80 211.37
   98%    99%   100%
212.98 219.94 234.00
```

Proseguendo nell'analisi della distribuzione della ferritina possiamo, con una sola riga di codice, generare i boxplot[78] (**Fig. 4.3**) della ferritina per ciascuno degli sport praticati. Questo è il codice:

```
# boxplot della ferritina per sport praticato                                      5/5.b
#
windows() # apre e inizializza una nuova finestra grafica
boxplot(ferr~sport, data=ais, main="Ferritina nei vari sport", xlab="Sport praticato", ylab="Ferritina in
µg/L", notch=FALSE, col="yellow") # traccia i boxplot
#
```

[78] O grafici a scatola con i baffi o box and whiskers plot. Vedere: **5.4. Grafici a scatola con i baffi**.

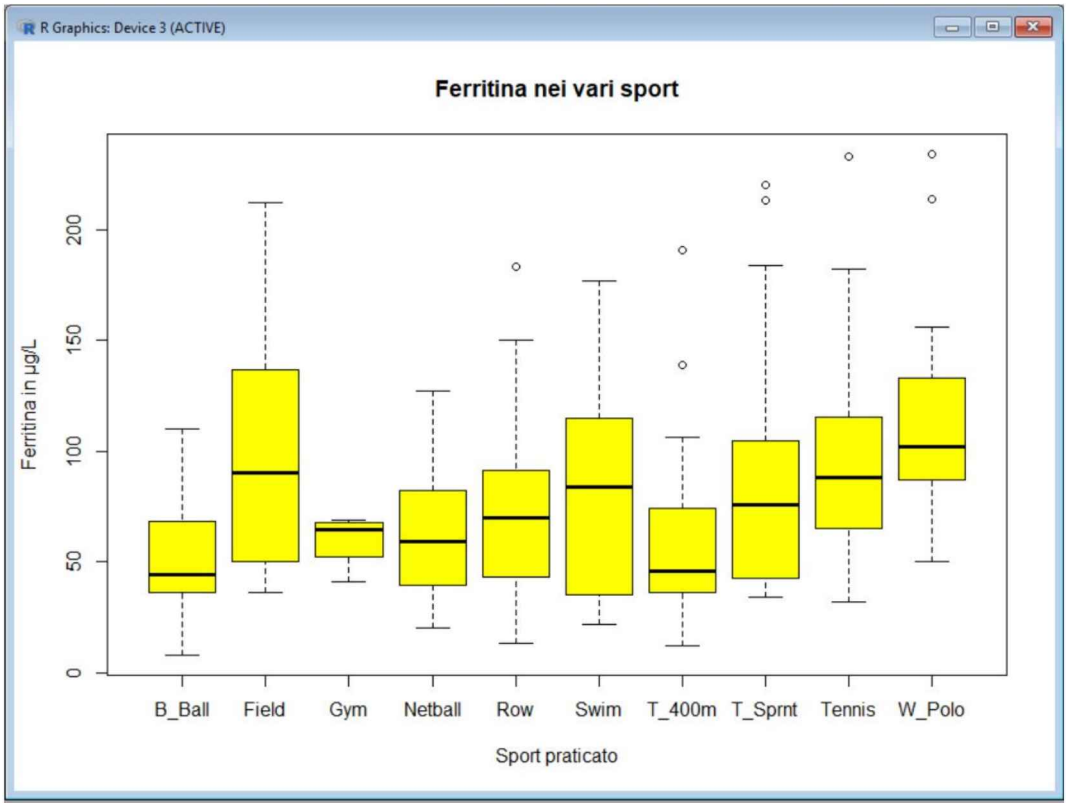



**Fig. 4.3.** Boxplot che illustrano la distribuzione dei valori della ferritina nei vari sport mediante una rappresentazione grafica non parametrica basata su mediana e quartili della distribuzione.

A questo punto se per i boxplot rappresentati nella **Fig. 4.3** volete vedere i valori numerici della mediana della ferritina per ciascuno sport praticato, potete farlo con questa semplice riga di codice:

```
by(ais$ferr, ais$sport, median) # valori della mediana della ferritina per sport
```

Da notare che se avessimo all'inizio dei due script (.a e .b) messo la riga
**attach(ais)**
avremmo potuto impiegare dappertutto nelle funzioni direttamente il nome della variabile (**rcc**, **wcc**, **hc**, eccetera) senza dovere specificare ogni volta *nomedelsetdidati$nomedellavariabile*.

Abbiamo visto negli script precedenti (3/5.a) come impiegare la funzione **sapply()** per calcolare con gli argomenti **mydata, mean** la media di tutte le variabili, con gli argomenti **mydata, sd** la deviazione standard di tutte le variabili, e così via. Ma è utile ricordare che le statistiche parametriche possono anche essere calcolate una per una, separatamente per ciascuna delle variabili (solo quelle distribuite in modo gaussiano), impiegando le funzioni **mean()**, **sd()** e le altre funzioni riportate qui di seguito per la variabile emoglobina

```
mean(mydata$hg) # media dell'emoglobina
sd(mydata$hg) # deviazione standard
var(mydata$hg) # varianza
min(mydata$hg) # valore minimo
```

```
max(mydata$hg) # valore massimo
range(mydata$hg) # range
#
```

di cui si riportano i risultati:

```
> mean(mydata$hg) # media dell'emoglobina
[1] 14.56634
> sd(mydata$hg) # deviazione standard
[1] 1.362451
> var(mydata$hg) # varianza
[1] 1.856274
> min(mydata$hg) # valore minimo
[1] 11.6
> max(mydata$hg) # valore massimo
[1] 19.2
> range(mydata$hg) # range
[1] 11.6 19.2
```

Si lascia come esercizio l'adattamento del codice alle altre variabili del set di dati **ais**.

**Nota bene**: effettuate una pulizia dell'area di lavoro per evitare interferenze con la sessione successiva[79].

Potete farlo anche con questo script, che consiste in una sola riga di comando:

```
q(save="no") # esce da R senza salvare l'area di lavoro che al rientro in R risulterà ripulita
```

---

[79] Vedere: **2.7. Pulizia dell'aera di lavoro e uscita dal programma**. Al bisogno cancellate i due file denominati `.Rdata` e `.rhistory` che si trovano nella cartella `Documenti`.

## 4.3. Confronto tra due campioni

Nel set di dati **ais**[80]**,** già impiegato nel capitolo precedente per illustrare le statistiche elementari, sono contenuti dati rilevati ad atleti di sesso femminile e di sesso maschile. Sorge così inevitabilmente l'idea di metterli a confronto con domande del tipo: quale è il peso medio delle donne? Quale è il peso medio degli uomini? E le due medie sono simili o sono sufficientemente diverse?

Il confronto tra due campioni ha lo scopo di stabilire se ci troviamo in presenza di medie (la tradizionale misura di posizione, parametrica) o mediane (la misura di posizione alternativa, non parametrica) simili o differenti, e può essere effettuato sia nel caso di **campioni indipendenti** sia nel caso di campioni non indipendenti (**dati appaiati**, come tipicamente avviene quando una misura viene effettuata sullo stesso soggetto/oggetto prima e dopo uno specifico trattamento). Accanto alla versione tradizionale parametrica (per il confronto tra medie) rappresentata dal **test t di Student**, esistono gli equivalenti non parametrici (per il confronto tra mediane), che devono essere impiegati quando i dati non sono distribuiti in modo gaussiano (qui vedremo il **test di Wilcoxon** per campioni indipendenti e il **test di Kruskal-Wallis**). Quindi anche in questo caso è necessario effettuare una analisi preliminare dei dati per decidere quale sia il test appropriato da impiegare.

### 4.3.1. Test parametrici e non parametrici per campioni indipendenti

Come prima cosa è necessario caricare il pacchetto **DAAG** che include l'oggetto/set di dati **ais**:

```
# CONFRONTO TRA DUE CAMPIONI (PER CAMPIONI INDIPENDENTI)                    1/5
#
library(DAAG) # carica il pacchetto DAAG inclusi i dati
str(ais) # mostra la struttura di ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
```

Subito dopo viene caricato il pacchetto **nortest** che ci consente di effettuare i **test di normalità**. Scegliamo uno dei più classici e collaudati, il test di Lilliefors (Kolmogorov-Smirnov) e lo applichiamo al **peso corporeo** (variabile `wt`) e alla **percentuale di grasso corporeo** (variabile `pcBfat`) di uomini (`m`) e donne (`f`):

```
# test di normalità                                                         2/5
#
library(nortest) # carica il pacchetto
lillie.test(ais$wt[ais$sex == "f"]) # test di Lilliefors (Kolmogorov-Smirnov) su peso corporeo donne
lillie.test(ais$wt[ais$sex == "m"]) # idem su peso corporeo uomini
lillie.test(ais$pcBfat[ais$sex == "f"]) # test di Lilliefors (Kolmogorov-Smirnov) su percentuale di grasso
# corporeo donne
lillie.test(ais$pcBfat[ais$sex == "m"]) # idem su percentuale di grasso corporeo uomini
#
```

Questo è il risultato dei test di normalità:

[80] Vedere: **A3. Il set di dati ais**.

```
> lillie.test(ais$wt[ais$sex == "f"]) # test di normalità di Lilliefors
(Kolmogorov-Smirnov) su peso corporeo donne

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  ais$wt[ais$sex == "f"]
D = 0.05469, p-value = 0.6532

> lillie.test(ais$wt[ais$sex == "m"]) # idem su peso corporeo uomini

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  ais$wt[ais$sex == "m"]
D = 0.054681, p-value = 0.6384

> lillie.test(ais$pcBfat[ais$sex == "f"]) # test di normalità di
Lilliefors (Kolmogorov-Smirnov) su percentuale di grasso corporeo donne

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  ais$pcBfat[ais$sex == "f"]
D = 0.066987, p-value = 0.3291

> lillie.test(ais$pcBfat[ais$sex == "m"]) # idem su percentuale di grasso
corporeo uomini

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  ais$pcBfat[ais$sex == "m"]
D = 0.17702, p-value = 2.254e-08
```

Nel caso del peso corporeo il test di Lilliefors conferma, sia per donne sia per uomini, che i dati non si discostano significativamente da una distribuzione gaussiana (`p = 0.6532` e `p = 0.6384` rispettivamente). Nel caso della percentuale di grasso corporeo invece ci troviamo di fronte a dati che, negli uomini, non sono distribuiti in modo gaussiano (`p = 2.254e-08`).

Se dobbiamo confrontare due campioni impiegando un test parametrico, è necessario che in entrambi i dati siano distribuiti in modo gaussiano. Pertanto applicheremo al peso corporeo una statistica parametrica, il test t di Student, e alla percentuale di grasso corporeo le statistiche non parametriche con il test di Wilcoxon per campioni indipendenti e il test di Kruskal-Wallis.

Questo è il codice per eseguire il test t di Student per il confronto tra medie di due campioni indipendenti, dove i due campioni sono ottenuti da **R** separando i valori della variabile peso corporeo (`wt`) in base al sesso mediante l'argomento **wt~sex**:

```
# test parametrico (confronto tra medie) per il peso corporeo                3/5
#
var.test(wt~sex) # controllo dell'omogeneità delle varianze
t.test(wt~sex, var.equal = TRUE) # test t di Student per varianze omogenee
#
```

Questi sono i risultati:

```
> var.test(wt~sex) # controllo dell'omogeneità delle varianze

        F test to compare two variances

data:  wt by sex
F = 0.77423, num df = 99, denom df = 101, p-value = 0.2029
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.5221795 1.1488587
sample estimates:
ratio of variances
         0.7742343

> t.test(wt~sex, var.equal = TRUE) # test t di Student per varianze
omogenee

        Two Sample t-test

data:  wt by sex
t = -9.2272, df = 200, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -18.42589 -11.93717
sample estimates:
mean in group f mean in group m
       67.34200        82.52353
```

Il test F (test di Fisher per il confronto tra le due varianze, quella nelle donne e quella negli uomini) indica che le varianze non sono significativamente diverse (la probabilità di osservare per caso una differenza tra le medie pari a quella effettivamente osservata è `p = 0.2029`) e anche l'ipotesi alternativa conferma questo. Pertanto si procede con il test t di Student per varianze omogenee (impostando l'argomento **var.equal = TRUE**).

La media del peso corporeo nelle donne è `67.342`, negli uomini è `82.52353` e le due medie sono significativamente diverse (la probabilità di osservare per caso una tale differenza è inferiore a 0.00000000000000022 (`p < 2.2e-16`)[81].

Soggiacente al test t di Student è l'ipotesi che le due medie siano uguali ovvero che la differenza tra le due medie sia zero (ipotesi $H_0$ o ipotesi nulla). Ma il test eseguito con **R** verifica anche l'ipotesi alternativa, cioè che la differenza tra le due medie non sia uguale a 0 (zero). L'intervallo di confidenza calcolato per questa ipotesi va da `-18.42589` a `-11.93717` e indica che la differenza tra le due medie è significativamente diversa da 0 (non lo sarebbe se l'intervallo di confidenza includesse lo 0). Le due soluzioni si confermano l'una con l'altra e ci consentono di affermare che le donne pesano significativamente (mediamente) meno degli uomini.

Questo è il codice per eseguire i test non parametrici per il confronto tra mediane di due campioni indipendenti, dove i due campioni sono ottenuti da **R** separando i valori della variabile percentuale di grasso corporeo (`pcBfat`) in base al sesso mediante l'argomento **wt~sex**:

[81] Viene riportato questo valore quando **R** non è più in grado di approssimare numericamente il risultato.

```
# test non parametrici (confronto tra mediane) per percentuale di grasso corporeo 4/5
#
wilcox.test(pcBfat~sex) # test di Wilcoxon per campioni indipendenti
#
kruskal.test(pcBfat~sex) # test di Kruskal-Wallis
#
median(ais$pcBfat[ais$sex == "f"]) # mediana della percentuale di grasso corporeo per sesso = f
median(ais$pcBfat[ais$sex == "m"]) # mediana della percentuale di grasso corporeo per sesso = m
#
```

Diversamente dalla funzione **t.test()**, che aggiunge al termine dei calcoli il valore della media dei due campioni, le funzioni **wilcox.test()** e **kruskal.test()** non forniscono la mediana, per cui sono aggiunte due righe di codice per calcolarla sia nelle donne sia negli uomini. Questi sono i risultati:

```
> wilcox.test(pcBfat~sex) # test di Wilcoxon per campioni indipendenti

        Wilcoxon rank sum test with continuity correction

data:  pcBfat by sex
W = 9417.5, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0

> #
> kruskal.test(pcBfat~sex) # test di Kruskal-Wallis

        Kruskal-Wallis rank sum test

data:  pcBfat by sex
Kruskal-Wallis chi-squared = 108.03, df = 1, p-value < 2.2e-16

> #
> median(ais$pcBfat[ais$sex == "f"]) # mediana della percentuale di
grasso corporeo per sesso = f
[1] 17.94
> median(ais$pcBfat[ais$sex == "m"]) # mediana della percentuale di
grasso corporeo per sesso = m
[1] 8.625
```

La mediana della percentuale di grasso corporeo nelle donne è `17.94`, negli uomini è `8.625` ed entrambi i test confermano che le due mediane sono significativamente diverse (la probabilità di osservare per caso una tale differenza è per entrambi i test inferiore a 0.00000000000000022 (`p < 2.2e-16`)[82].

Un ausilio ulteriore nell'interpretazione dei risultati ci viene ancora una volta dalla grafica, e in particolare dai boxplot:

[82] Di nuovo viene riportato questo valore in quanto **R** non è più in grado di approssimare numericamente il risultato.

```
# analisi grafica non parametrica delle distribuzioni                    5/5
#
windows() # apre e inizializza una nuova finestra grafica
# la riga seguente traccia i boxplot per sesso per il peso corporeo
boxplot(wt~sex, data=ais, main="Peso corporeo in relazione al sesso", xlab="Sesso", ylab="Peso corporeo
in kg", notch=TRUE, col="green")
#
windows() # apre e inizializza una nuova finestra grafica
# la riga seguente traccia i boxplot per sesso per la percentuale di grasso corporeo
boxplot(pcBfat~sex, data=ais, main="Percentuale di grasso corporeo in relazione al sesso", xlab="Sesso",
ylab="Grasso corporeo in %", notch=TRUE, col="green")
#
```

In questo caso i boxplot per sesso del peso corporeo e della percentuale di grasso corporeo sono tracciati con una incisura (**notch=TRUE**) che rappresenta i limiti di confidenza al 95% della mediana (**Fig. 4.4**). Sia per il peso corporeo sia per la percentuale di grasso corporeo le incisure di boxplot di donne e uomini non si sovrappongono, pertanto le corrispondenti mediane sono significativamente diverse.

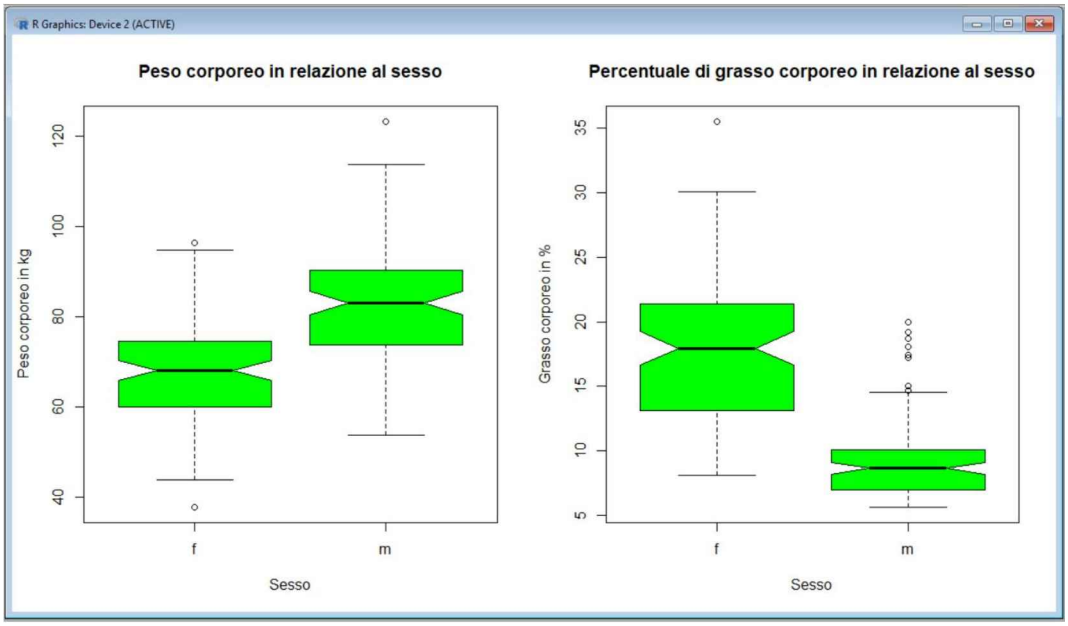



**Fig. 4.4** Boxplot con le incisure (notch) che rappresentano i limiti di confidenza al 95% della mediana. Incisure che non si sovrappongono indicano una differenza significativa tra le mediane.

Si rimanda al paragrafo che tratta dei grafici a scatola con i baffi nel capitolo **5. Rappresentazione grafica dei dati** per i chiarimenti su questo tipo di sintesi grafica dei risultati.

**Nota bene**: le finestre si sovrappongono l'una all'altra, iconizzarle per esaminare i grafici uno ad uno.

### 4.3.2. Test parametrici e non parametrici per dati appaiati

Il VEMS (Volume Espiratorio Massimo nel primo secondo), impiegato nella diagnostica della capacità respiratoria, è il volume di aria espirata nel corso del primo secondo di una espirazione massima forzata e indica il grado di pervietà delle grandi vie aeree. Viene anche denominato FEV1 dall'acronimo inglese di Forced Expiratory Volume in the 1st second. I seguenti dati, ricavati da Campbell[83], riportano i valori di VEMS misurati in 5 soggetti asmatici prima (t0) e dopo (t1) l'assunzione di un broncodilatatore.

| t0 | t1 | Differenza |
|---|---|---|
| 1.5 | 1.7 | -0.2 |
| 1.7 | 1.9 | -0.2 |
| 2.1 | 2.2 | -0.1 |
| 1.6 | 1.9 | -0.3 |
| 2.4 | 2.4 | 0 |

Fate il `download` del file `FEV1.csv` quindi copiatelo in `C:\Rdati\`.

In alternativa potete anche copiare le sei righe riportate qui sotto, salvarle in `C:\Rdati\` in un file di testo denominato `FEV1.csv` (attenzione all'estensione al momento del salvataggio del file).

```
t0;t1;Differenza
1.5;1.7;-0.2
1.7;1.9;-0.2
2.1;2.2;-0.1
1.6;1.9;-0.3
2.4;2.4;0
```

Quindi eseguite questo script:

```
# CONFRONTO TRA DUE CAMPIONI (PER DATI APPAIATI)
#
mydata <- read.table("c:/Rdati/FEV1.csv", header=TRUE, sep=";") # importa i dati
attach(mydata) # consente di impiegare direttamente i nomi delle variabili
mydata # mostra i dati
#
library(nortest) # carica il pacchetto
lillie.test(Differenza) # test di Lilliefors per la normalità sulla Differenza (t0 - t1)
#
t.test(t0, t1, paired=TRUE) # test t di Student per dati appaiati
#
wilcox.test(t0, t1, paired=TRUE, exact=FALSE) # test di Wilcoxon  per dati appaiati (test  non parametrico)
#
```

La scelta tra test parametrico (test t di Student) e non parametrico (test di Wilcoxon o Wilcoxon Signed Rank Test) può essere fatta mediante analisi della distribuzione dei dati.

---

[83] Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 142.

```
> lillie.test(mydata$Differenza) # test di Lilliefors per la normalità
sulla Differenza (t0 - 1)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  mydata$Differenza
D = 0.23714, p-value = 0.4686
```

Questa viene effettuata sulla variabile `Differenza (t0 - t1)` presa con il segno mediante il test di Lilliefors (Kolmogorov-Smirnov), che conferma una distribuzione gaussiana (`p = 0.4686`).

Vengono quindi impiegati i risultati del test t di Student per dati appaiati che con un valore di `p = 0.03492` indica significatività nella differenza media tra prima e dopo la somministrazione.

```
> t.test(Prima, Dopo, paired=TRUE) # test t di Student per dati appaiati

        Paired t-test

data:  Prima and Dopo
t = -3.1379, df = 4, p-value = 0.03492
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.30157148 -0.01842852
sample estimates:
mean of the differences
                  -0.16

> #
> wilcox.test(t0, t1, paired=TRUE, exact=FALSE) # test di Wilcoxon  per
dati appaiati (test non parametrico)

        Wilcoxon signed rank test with continuity correction

data:  Prima and Dopo
V = 0, p-value = 0.09751
alternative hypothesis: true location shift is not equal to 0
```

Da notare che assumendo come soglia di significatività il classico `p = 0.05` il test di Wilcoxon con un valore di `p = 0.09751` indica una differenza non significativa: un risultato opposto a quello del test t di Student.

**Nota bene**: effettuate una pulizia dell'area di lavoro per evitare interferenze con la sessione successiva[84].

---

[84] Vedere: **2.7. Pulizia dell'aera di lavoro e uscita dal programma**. Al bisogno cancellate i due file denominati `.Rdata` e `.rhistory` che si trovano nella cartella `Documenti`.

## 4.4. Regressione lineare

Il termine "regressione" è stato applicato al calcolo della retta che meglio approssima l'andamento di una serie di punti in seguito agli studi di Francis Galton[85].

Tradizionalmente il calcolo della regressione lineare viene accompagnato dal calcolo del coefficiente di correlazione **r** ovvero dei suoi equivalenti non parametrici. Sul fatto che "correlazione" non deve essere inteso come "causazione", e cioè che la presenza di una correlazione tra due variabili non implichi un rapporto di causalità tra l'una e l'altra sono stati versati giustamente fiumi di inchiostro[86].

I dati che ora impiegheremo sono contenuti nel set di dati **galton** del pacchetto **psych**.

Se non l'avete ancora fatto scaricate il pacchetto **psychTools** dal **CRAN** selezionando `Pacchetti >> Installa pacchetti...` come indicato al paragrafo **2.6. Pacchetti aggiuntivi di statistica e grafica**. Potete accedere alla documentazione del pacchetto **psych** selezionando `Aiuto >> Guida Html >> Packages >> psych`.

**Nota bene**: prima di proseguire verificate di avere scaricato dal **CRAN** anche i pacchetti **Hmisc** e **car**, e il pacchetto **gvlma**.

Nell'oggetto/set di dati **galton** del pacchetto **psychTools** sono contenuti 928 coppie di rilevazioni dell'altezza (espressa in pollici) nei genitori (variabile `parent`) e nei rispettivi figli (variabile `child`) raccolte da Galton nel suo famoso studio.

Dopo avere installato il pacchetto **DAAG** basterà eseguire in **R** questa semplice istruzione per caricare il pacchetto

```
library(psychTools) # carica il pacchetto psych che include il set di dati galton
```

con la quale l'oggetto/set di dati **galton** sarà immediatamente disponibile. Da notare che se nella `Console di R` digitate

```
data()
```

si apre una finestra nella quale sono elencati tutti i set di dati caricati con il pacchetto e che sempre mediante la stessa funzione **data()** possono essere caricati set di dati indipendenti dai pacchetti[87].

### 4.4.1. Coefficiente di correlazione

Con questo prima parte dello script viene caricato il set di dati **galton** e sono calcolati il coefficiente di correlazione classico **r** di **Pearson** (che è un test parametrico) e due coefficienti di correlazione non parametrici, il **coefficiente di correlazione per ranghi ρ (rho) di Spearman** e il **coefficiente di correlazione τ (tau) di Kendall**:

[85] Vedere: **A4. Francis Galton e la regressione**.
[86] Vedere: **A5. Correlazione e causazione**.
[87] Per la documentazione della funzione **data()** digitate **help(data)** nella `Console di R`.

```
# COEFFICIENTE DI CORRELAZIONE LINEARE                                    1/2
#
library(psychTools) # carica il pacchetto che include il set di dati galton
str(galton) # struttura dei dati
#
# calcola i coefficienti di correlazione secondo pearson (il classico r), spearman, kendall
#
cor(galton, use="complete.obs", method="pearson") # r di Pearson
cor(galton, use="complete.obs", method="spearman") # rho di Spearman
cor(galton, use="complete.obs", method="kendall") # tau di Kendall
#
```

Questo è il risultato:

```
> # COEFFICIENTE DI CORRELAZIONE LINEARE
> #
> library(psych) # carica il pacchetto che include il set di dati galton
> str(galton) # struttura dei dati
'data.frame':   928 obs. of  2 variables:
 $ parent: num  70.5 68.5 65.5 64.5 64 67.5 67.5 67.5 66.5 66.5 ...
 $ child : num  61.7 61.7 61.7 61.7 61.7 62.2 62.2 62.2 62.2 62.2 ...
> #
> # calcola i coefficienti di correlazione secondo pearson (il classico
r), spearman, kendall
> #
> cor(galton, use="complete.obs", method="pearson") # r di Pearson
          parent     child
parent 1.0000000 0.4587624
child  0.4587624 1.0000000
> cor(galton, use="complete.obs", method="spearman") # rho di Spearman
          parent     child
parent 1.0000000 0.4251345
child  0.4251345 1.0000000
> cor(galton, use="complete.obs", method="kendall") # tau di Kendall
          parent     child
parent 1.0000000 0.3315756
child  0.3315756 1.0000000
> #
```

L'argomento **use="complete.obs"** comporta l'eliminazione automatica dei casi nei quali manchi l'uno o l'altro dato (qui non servirebbe, ma viene riportato per completezza).

Con un pacchetto alternativo (**Hmisc**) è possibile calcolare i livelli di significatività del coefficiente di correlazione, ma solo per il test di Pearson e per il test di Kendall:

```
# calcola i coefficienti di correlazione con i livelli di significatività   2/2
#
library(Hmisc) # carica il pacchetto
rcorr(as.matrix(galton), type="pearson") # type può essere solo "pearson" (il classico r) o "spearman"
rcorr(as.matrix(galton), type="spearman")
#
```

La probabilità di osservare per caso i valori di **r** e di **rho** ottenuti è molto bassa, e viene addirittura riportata come **0** (zero).

### 4.4.2. Regressione lineare semplice parametrica

Nel caso della regressione lineare semplice (o regressione lineare classica) si assume la **x** (in ascisse) come variabile indipendente e la **y** (in ordinate) come variabile dipendente e si calcolano l'intercetta **a** e il coefficiente angolare **b** della "migliore" retta di regressione di equazione

$$y = a + b \cdot x$$

che approssima la distribuzione dei dati sperimentali.

In **R** il calcolo viene effettuato mediante la funzione **lm()** che sta per "linear model"[88] (come vedremo tra poco questa funzione può essere estesa anche a più variabili indipendenti, per il calcolo della regressione lineare multipla) con questa prima parte dello script:

```
# REGRESSIONE LINEARE SEMPLICE  PARAMETRICA  y = a + b · x                    1/3
#
library(psychTools) # carica il pacchetto che include il set di dati galton
str(galton) # struttura dei dati
#
x <- galton$parent # in ascisse altezza dei genitori
y <- galton$child # in ordinate altezza dei figli
reglin <- lm(y ~ x) # calcola intercetta (a) e coefficiente angolare (b)
coefficients(reglin) # mostra i coefficienti dell'equazione child = a + b · parent
confint(reglin, level=0.95) # calcola intervalli di confidenza dell'intercetta e del coefficiente angolare
#
```

L'assegnazione **<-** all'oggetto **x**, all'oggetto **y** e all'oggetto **reglin** rispettivamente dei valori di altezza dei genitori (**x <- galton$parent**) e dei figli (**y <- galton$child**) e dei risultati della regressione lineare (**reglin <- lm(y ~ x)** è stata effettuata per creare un codice più descrittivo e mnemonico. In realtà si possono ottenere gli stessi risultati con un codice molto più conciso, ma meno leggibile e meno intuitivo ad una prima lettura:

```
# REGRESSIONE LINEARE SEMPLICE PARAMETRICA
#
library(psychTools) # carica il pacchetto che include il set di dati galton
str(galton) # struttura dei dati
#
coefficients(lm(galton$child ~ galton$parent)) # mostra i coefficienti dell'equazione child = a + b · parent
confint(lm(galton$child ~ galton$parent), level=0.95) # calcola intervalli di confidenza dell'intercetta e del
# coefficiente angolare
#
```

Uno dei punti principali della regressione lineare è rappresentato dall'analisi dell'adeguatezza del modello ai dati. In altre parole dall'analisi di se ed eventualmente quanto la relazione tra i dati si presta ad essere spiegata mediante l'equazione di una retta. **R** fornisce per questa analisi una serie di grandezze e dei test

[88] Digitate **help(lm)** nella `Console di R` per la documentazione della funzione **lm()**.

specifici eseguiti con questa parte dello script:

```
# grandezze e test per la valutazione della adeguatezza della regressione lineare 2/3
#
summary(reglin) # mostra un riepilogo dei risultati
fitted(reglin) # ricalcola i valori mediante l'equazione della retta di regressione
residuals(reglin) # calcola le differenze residue tra valore osservato e valore calcolato
anova(reglin) # analisi della varianza per le differenze spiegate dalle x
vcov(reglin) # matrice di covarianza dell'intercetta e del coefficiente angolare
influence(reglin) # influenza dei singoli dati sulla regressione
#
t.test(residuals(reglin)) #  verifica che la media degli errori non sia significativamente diversa da zero
#
shapiro.test(residuals(reglin)) #  verifica la normalità della distribuzione degli errori
#
library(gvlma) # carica il pacchetto
summary(gvlma(reglin)) #  test globale per l'assunto di linearità
#
library(car) # carica il pacchetto
outlierTest(reglin) # valore p di Bonferonni per la presenza di dati aberranti (outliers)
#
```

Mediante **summary(reglin)** si verifica l'ipotesi che intercetta **a** e coefficiente angolare **b** siano significativamente diversi da 0 (zero), ovvero che tra la variabile dipendente **y** e la variabile indipendente **x** esista una relazione lineare dovuta a fattori sistematici e non casuali.

```
> summary(reglin)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-7.8050 -1.3661  0.0487  1.6339  5.9264

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 23.94153    2.81088   8.517   <2e-16 ***
x            0.64629    0.04114  15.711   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.239 on 926 degrees of freedom
Multiple R-squared:  0.2105,    Adjusted R-squared:  0.2096
F-statistic: 246.8 on 1 and 926 DF,  p-value: < 2.2e-16
```

Intercetta **a** (23.94153) e coefficiente angolare **b** (0.64629) risultano significativamente diversi da 0 (zero). Tuttavia va notato che il coefficiente di determinazione $R^2$ è uguale a 0.21 (penultima riga dei risultati, ed è sostanzialmente identico sia nella forma normale sia nella forma aggiustata). $R^2$, che nel caso di correlazione bivariata è semplicemente il quadrato del coefficiente di correlazione **r** di Pearson 0.4587624 calcolato in precedenza[89], indica la frazione di variabilità spiegata dal modello di regressione: e il fatto che la regressione lineare spieghi il 21% della variabilità osservata, restando inspiegato quindi il

[89] In effetti **r** era uguale a 0.4587624 che elevato al quadrato da appunto 0.21.

restante 79% della variabilità, potrebbe anche essere considerato come inadeguato.

Per le grandezze fornite dalle funzioni **fitted(reglin)**, **residuals(reglin)**, **anova(reglin)**, **vcov(reglin)**, **influence(reglin)** si rimanda alla documentazione di **R** ai testi di statistica.

Il test t di Student conferma che la media degli errori non è significativamente diversa da zero, condizione necessaria ma non sufficiente per accettare la regressione lineare:

```
> t.test(residuals(reglin)) #  verifica che la media degli errori non sia
significativamente diversa da zero

        One Sample t-test

data:  residuals(reglin)
t = -3.2132e-14, df = 927, p-value = 1
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.1441363  0.1441363
sample estimates:
    mean of x
-2.359884e-15
```

Tuttavia un test di normalità rileva che gli errori non sono distribuiti in modo gaussiano:

```
> shapiro.test(residuals(reglin)) #  verifica la normalità della
distribuzione degli errori

        Shapiro-Wilk normality test

data:  residuals(reglin)
W = 0.99275, p-value = 0.0001697
```

E un ulteriore test che verifica gli assunti che è necessario rispettare perché la regressione lineare possa essere considerata adeguata, conferma la presenza di vari problemi nell'applicazione del modello a questi dati:

```
> library(gvlma)
> summary(gvlma(reglin)) #  test globale per l'assunto di linearità

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-7.8050 -1.3661  0.0487  1.6339  5.9264

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 23.94153    2.81088   8.517   <2e-16 ***
x            0.64629    0.04114  15.711   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.239 on 926 degrees of freedom
Multiple R-squared:  0.2105,    Adjusted R-squared:  0.2096
```

```
F-statistic: 246.8 on 1 and 926 DF,  p-value: < 2.2e-16


ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = reglin)

                     Value   p-value                   Decision
Global Stat         25.489 4.011e-05 Assumptions NOT satisfied!
Skewness             8.979 2.731e-03 Assumptions NOT satisfied!
Kurtosis             1.965 1.610e-01    Assumptions acceptable.
Link Function        4.632 3.138e-02 Assumptions NOT satisfied!
Heteroscedasticity   9.913 1.641e-03 Assumptions NOT satisfied!
```

Come si vede **R** è dotato di molti strumenti per la diagnosi dei risultati della regressione lineare. Per la documentazione delle funzioni illustrate si rimanda alla documentazione di **R**. Una ottima illustrazione delle tecniche di regressione con **R** è quella presentata da Ricci[90]. Una breve trattazione del razionale alla base della regressione lineare è riportata in appendice[91].

Con questa terza parte dello script

```
# approfondimenti grafici sulla regressione lineare                                    3/3
#
library(car) # carica il pacchetto
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
newx = seq(min(x),max(x),by = 0.05) # valori della x per i quali calcolare l'intervallo di confidenza
conf_interval <- predict(reglin, newdata=data.frame(x=newx), interval="confidence", level = 0.95) #
# calcola gli intervalli di confidenza
plot(x, y, xlab="Altezza dei padri", ylab="Altezza dei figli", main="Regressione lineare y = a + b·x") #
# grafico dei dati
abline(reglin, col="lightblue") # retta di regressione
lines(newx, conf_interval[,2], col="blue", lty=2) # limite di confidenza inferiore
lines(newx, conf_interval[,3], col="blue", lty=2) # limite di confidenza superiore
#
plot(y, residuals(reglin), xlab="Altezza dei figli", ylab="Altezza osservata - calcolata", main="Analisi delle
differenza residue") # grafico delle differenza tra altezza osservata e altezza calcolata nei figli
#
influencePlot(lm(galton$child ~ galton$parent), fill=FALSE, xlab="t-quantili (il diametro dei cerchi",
sub="è proporzionale alla distanza di Cook)", ylab="Residui studentizzati", main="Influenza dei dati") #
grafico
# dell'influenza dei dati sulle conclusioni
#
qqPlot(lm(galton$child ~ galton$parent), xlab="t-quantili", ylab="Residui studentizzati", main="Quantili
vs. residui") # mostra il grafico dei quantili per i residui studentizzati
#
```

[90] Vito Ricci. *Principali tecniche di regressione con **R***. URL consultato il 07/10/2018: https://goo.gl/1jcrm9
[91] Vedere: **A6. La regressione lineare: assunti e modelli**.

realizziamo infine una rappresentazione grafica con l'obiettivo di “toccare con mano” i dati di Galton. I risultati di questa elaborazione grafica a supporto della precedente mole di dati statistici sono riportati nella **Fig. 4.5**.

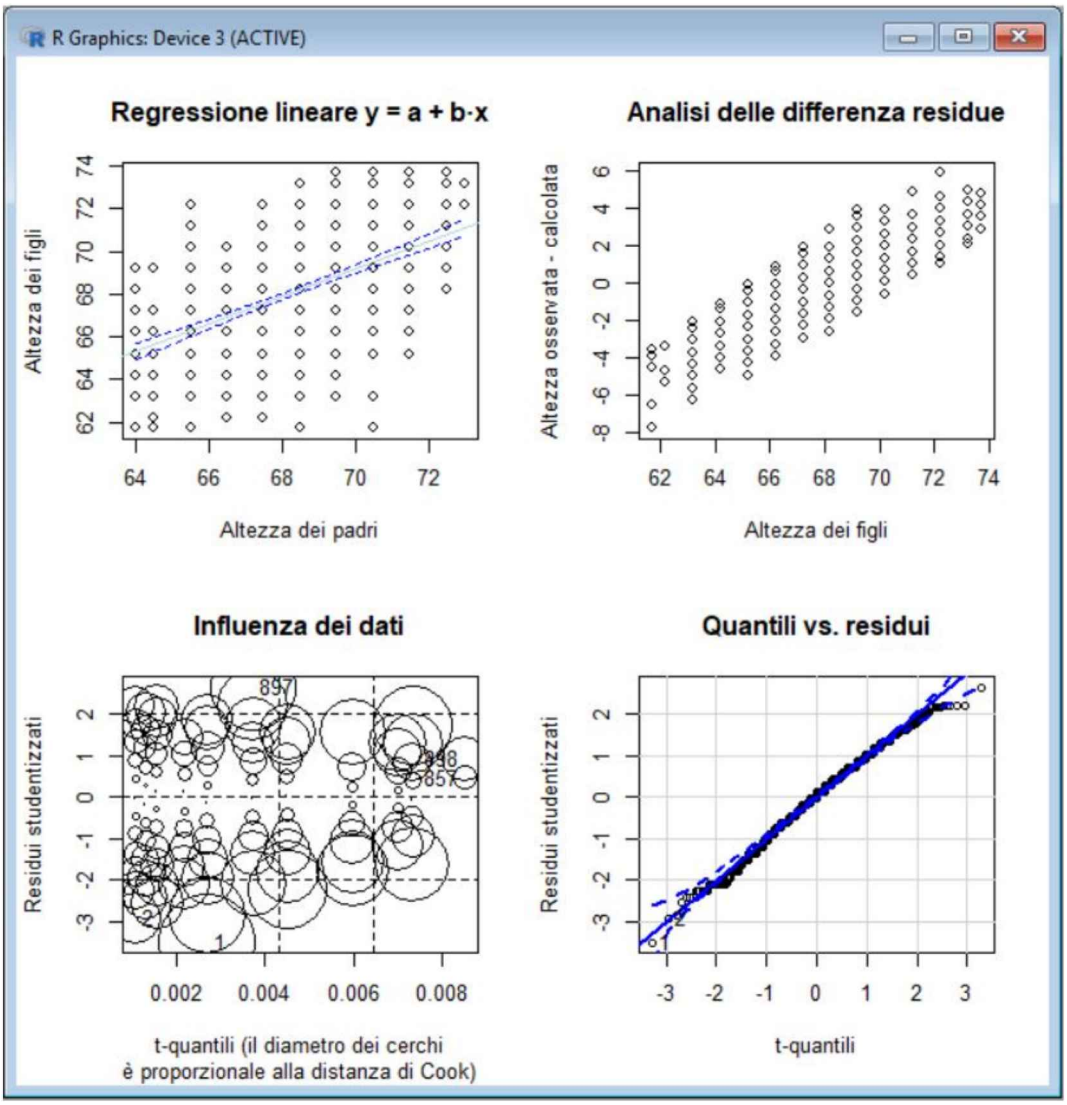



**Fig. 4.5** Mediante alcune funzioni fornite da **R** è possibile integrare l'analisi statistica con rappresentazioni grafiche che consentono di ottenere una ulteriore introspezione nei dati.

Nel grafico in alto a sinistra sono riportati i dati di Galton (altezza dei padri in ascisse e altezza dei figli in ordinate), la retta di regressione calcolata e i sui limiti di confidenza.

Nel grafico in alto a destra sono riportati in ascisse l'altezza dei figli e in ordinate la differenza, espressa con il segno, tra l'altezza osservata in un soggetto e l'altezza ricalcolata nello stesso soggetto impiegando l'equazione della retta di regressione.

Da notare che **R** nell'elaborare i grafici ha fornito anche, con gli ultimi due, dei risultati numerici:

```
> influencePlot(lm(galton$child ~ galton$parent), xlab="t-quantili (il
diametro dei cerchi", sub="è proporzionale alla distanza di Cook)",
ylab="Residui studentizzati", main="Influenza dei dati") # grafico
dell'influenza dei dati sulle conclusioni
       StudRes         Hat       CookD
1   -3.5126706 0.002699826 0.016499436
2   -2.9226403 0.001090010 0.004622768
857  0.4839886 0.008511029 0.001006222
897  2.6611087 0.003740530 0.013207269
898  0.9327550 0.008511029 0.003734739
> #
> qqPlot(lm(galton$child ~ galton$parent), xlab="t-quantili",
ylab="Residui studentizzati", main="Quantili vs. residui") # mostra il
grafico dei quantili per i residui studentizzati
[1] 1 2
```

I numeri `1, 2, 857, 897, 898` forniti dalla funzione **influencePlot()** stanno a indicare i casi che influenzano in modo importante la regressione. Si tratta di casi dei quali sarebbe importante controllare la validità, o che potrebbero essere situati in intervalli di valori per i quali potrebbe essere opportuno acquisire più dati. Anche la funzione **qqPlot()** fornisce, oltre al grafico, l'ndicazione di due punti da controllare, che sono punti `1, 2` già indicati dalla funzione precedente.

Quindi anche nel caso della regressione lineare le funzioni grafiche di **R** consentono di effettuare un'analisi dei dati complementare all'analisi numerica statistica tradizionale.

Le implicazioni degli assunti adottati nella costruzione della relazione di funzione che lega la **y** alla **x**, le conseguenze che questi assunti determinano nelle conclusioni tratte dalla regressione lineare, e il calcolo della regressione lineare come **componente principale standardizzata**, sono discussi a parte[92].

### 4.4.3. Regressione lineare semplice non parametrica

Anche per la regressione lineare esistono alternative non parametriche.

Quelle qui presentate sono incluse in due nuovi pacchetti, **mblm** e **quantreg**, che devono quindi essere scaricati dal **CRAN**. Trovate come al solito la documentazione dei pacchetti e in particolare il loro manuale di riferimento sul repository della documentazione[93].

Essendo `n` il numero delle coppie di dati/punti esistono `n · (n − 1) / 2` modi di connettere due punti qualsiasi con una retta, cioè esistono `n · (n − 1) / 2` coefficienti angolari. Nel caso del set di dati **galton** questo corrisponde a 215 064 (cioè quindicimila e duecentosessantaquattro) coefficienti angolari. Il coefficiente angolare `b` della retta di regressione è la mediana di questi valori. L'intercetta viene calcolato come mediana degli `n` valori possibili di `a` calcolati per ciascun punto mediante il coefficiente angolare trovato. Questi sono concettualmente i principi alla base dei tre modelli di regressione lineare non parametrica che sono impiegati nello script che segue:

```
# REGRESSIONE LINEARE SEMPLICE NON PARAMETRICA
#
```

[92] Vedere: **A6. La regressione lineare: assunti e modelli**.
[93] *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

```
library(psychTools) # carica il pacchetto che include il set di dati galton
#
library(mblm) # carica il pacchetto
regnon = mblm(child ~ parent, data=galton, repeated=FALSE) # metodo di Theil-Sen
summary(regnon) # presenta i risultati
#
regnon = mblm(child ~ parent, data=galton, repeated=TRUE) # metodo di Siegel
summary(regnon) # presenta i risultati
#
library(quantreg) # carica il pacchetto
regnon = rq(child ~ parent, data=galton) # regressione quantilica
summary(regnon) # presenta i risultati
#
```

**Nota bene**: i tempi di esecuzione dello script sono dell'ordine dei 10 minuti.

Questi sono i risultati:

```
> # REGRESSIONE LINEARE SEMPLICE NON PARAMETRICA
> #
> library(psych) # carica il pacchetto e il set di dati galton
> #
> library(mblm) # carica il pacchetto
> regnon = mblm(child ~ parent, data=galton, repeated=FALSE) # metodo di
Theil-Sen
> summary(regnon)

Call:
mblm(formula = child ~ parent, dataframe = galton, repeated = FALSE)

Residuals:
   Min     1Q Median     3Q    Max
-7.833 -1.333  0.000  1.667  6.000

Coefficients:
            Estimate     MAD   V value Pr(>|V|)
(Intercept)  22.5333  2.4710 4.311e+05   <2e-16 ***
parent        0.6667  1.4826 3.540e+10   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.239 on 926 degrees of freedom

> #
> regnon = mblm(child ~ parent, data=galton, repeated=TRUE) # metodo di
Siegel
> summary(regnon)

Call:
mblm(formula = child ~ parent, dataframe = galton, repeated = TRUE)

Residuals:
    Min      1Q  Median      3Q     Max
-8.8333 -2.3333 -1.0000  0.6667  5.0000
```

```
Coefficients:
            Estimate     MAD V value Pr(>|V|)
(Intercept)  23.5333 36.8179  345374   <2e-16 ***
parent        0.6667  0.4942  297880   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.446 on 926 degrees of freedom

> #
> library(quantreg)
Carico il pacchetto richiesto: SparseM

Attaching package: 'SparseM'

The following object is masked from 'package:base':

    backsolve

> regnon = rq(child ~ parent, data=galton) # regressione quantilica
> summary(regnon)

Call: rq(formula = child ~ parent, data = galton)

tau: [1] 0.5

Coefficients:
            coefficients lower bd upper bd
(Intercept) 22.53333     10.06973 32.61374
parent       0.66667      0.53689  0.90082
Warning message:
In rq.fit.br(x, y, tau = tau, ci = TRUE, ...) : Solution may be nonunique
```

Con una intercetta `a` rispettivamente di `22.5`, di `23.5` e di `22.5` e un coefficiente angolare `b` di `0.67` identico in tutti e tre i casi, i metodi non parametrici forniscono quindi gli stessi risultati. Che rappresentano inoltre una conferma dei risultati della regressione lineare semplice (parametrica) vista in precedenza che aveva fornito (arrotondati) per `a` un valore di `23.9` e per `b` un valore di `0.64`.

## 4.5. Analisi multivariata

Nonostante a rigore il termine di analisi multivariata sia da applicare allo studio di due o più variabili, avendo già trattato con la regressione lineare il caso dell'analisi bivariata, vediamo ora esempi di casi nei quali le variabili sono tre o più. E consideriamo per semplicità solamente il caso della regressione lineare multipla, cioè della regressione lineare a più di due variabili, con i relativi coefficienti di correlazione, e il caso dell'analisi dei gruppi, forse meglio nota come cluster analysis.

### 4.5.1. Coefficienti di correlazione

Nell'analisi multivariata i coefficienti di correlazione sono tanti quante sono le coppie di variabili. In **R** è che non è necessario specificare quante sono le coppie di variabili. Nell'esempio che segue viene impiegato il set di dati **ais** del pacchetto **DAAG**, già visto in precedenza nel calcolo delle statistiche elementari e i cui dati sono riportati in appendice[94].

Questa è la parte iniziale dello script. Dopo avere caricato il pacchetto **DAAG** con la funzione **str()** viene mostrata la struttura dell'oggetto/set di dati **ais** originale, che contiene 13 variabili, 11 numeriche e 2 non numeriche. Dato che siamo interessati alle sole variabili numeriche, queste, che sono comprese nelle colonne da 1 a 11 del `dataframe` (o `dataset`) **ais**, le salviamo in un nuovo oggetto denominato **mydata** che è quello sul quale lavoreremo.

```
# COEFFICIENTE DI CORRELAZIONE LINEARE                                    1/4
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
str(ais) # struttura di ais
mydata <- ais[c(1,2,3,4,5,6,7,8,9,10,11)] # salva sole le colonne con le variabili numeriche in mydata
str(mydata) # struttura di mydata
#
```

Infine per conferma dell'operazione di selezione delle colonne vediamo la struttura del nuovo oggetto con **str(mydata)**:

```
> str(mydata)
'data.frame':   202 obs. of  11 variables:
 $ rcc    : num  3.96 4.41 4.14 4.11 4.45 4.1 4.31 4.42 4.3 4.51 ...
 $ wcc    : num  7.5 8.3 5 5.3 6.8 4.4 5.3 5.7 8.9 4.4 ...
 $ hc     : num  37.5 38.2 36.4 37.3 41.5 37.4 39.6 39.9 41.1 41.6 ...
 $ hg     : num  12.3 12.7 11.6 12.6 14 12.5 12.8 13.2 13.5 12.7 ...
 $ ferr   : num  60 68 21 69 29 42 73 44 41 44 ...
 $ bmi    : num  20.6 20.7 21.9 21.9 19 ...
 $ ssf    : num  109.1 102.8 104.6 126.4 80.3 ...
 $ pcBfat: num  19.8 21.3 19.9 23.7 17.6 ...
 $ lbm    : num  63.3 58.5 55.4 57.2 53.2 ...
 $ ht     : num  196 190 178 185 185 ...
 $ wt     : num  78.9 74.4 69.1 74.9 64.6 63.7 75.2 62.3 66.5 62.9 ...
> #
```

La successiva parte dello script provvede ed effettuare il calcolo dei coefficienti di correlazione tra tutte le

[94] Vedere: **A3. Il set di dati ais**

coppie di variabili con un metodo parametrico (il classico **r** di **Pearson**) e con due metodi non parametrici (il **coefficiente di correlazione per ranghi ρ (rho) di Spearman** e il **coefficiente di correlazione τ (tau) di Kendall**):

```
# method può essere pearson (il classico r), spearman, kendall                2/4
#
round(cor(mydata, use="complete.obs", method="pearson"), digits=3) # r di Pearson
round(cor(mydata, use="complete.obs", method="spearman"), digits=3) # rho di Spearman
round(cor(mydata, use="complete.obs", method="kendall"), digits=3) # tau di Kendall
#
```

Fin qui è tutto identico a quanto abbiamo visto nel caso della regressione lineare semplice, tranne per il fatto che, per una miglior leggibilità dei risultati, arrotondiamo a tre cifre i decimali riportati con la funzione **round()** e specificando come argomento **digits=3**[95]:

```
> # calcola i coefficienti di correlazione, method può essere pearson (il classico r),
spearman, kendall
> round(cor(mydata, use="complete.obs", method="pearson"), digits=3)
          rcc   wcc     hc     hg   ferr   bmi    ssf pcBfat    lbm     ht    wt
rcc     1.000 0.147  0.925  0.889  0.251 0.299 -0.403 -0.494  0.551  0.359 0.404
wcc     0.147 1.000  0.153  0.135  0.132 0.177  0.137  0.108  0.103  0.077 0.156
hc      0.925 0.153  1.000  0.951  0.258 0.321 -0.449 -0.532  0.583  0.371 0.424
hg      0.889 0.135  0.951  1.000  0.308 0.383 -0.435 -0.532  0.611  0.352 0.455
ferr    0.251 0.132  0.258  0.308  1.000 0.303 -0.108 -0.183  0.318  0.123 0.274
bmi     0.299 0.177  0.321  0.383  0.303 1.000  0.321  0.188  0.714  0.337 0.846
ssf    -0.403 0.137 -0.449 -0.435 -0.108 0.321  1.000  0.963 -0.208 -0.071 0.154
pcBfat -0.494 0.108 -0.532 -0.532 -0.183 0.188  0.963  1.000 -0.362 -0.188 0.000
lbm     0.551 0.103  0.583  0.611  0.318 0.714 -0.208 -0.362  1.000  0.802 0.931
ht      0.359 0.077  0.371  0.352  0.123 0.337 -0.071 -0.188  0.802  1.000 0.781
wt      0.404 0.156  0.424  0.455  0.274 0.846  0.154  0.000  0.931  0.781 1.000
> round(cor(mydata, use="complete.obs", method="spearman"), digits=3)
          rcc   wcc     hc     hg   ferr   bmi    ssf pcBfat    lbm     ht     wt
rcc     1.000 0.168  0.914  0.887  0.254 0.287 -0.396 -0.500  0.592  0.401  0.421
wcc     0.168 1.000  0.165  0.137  0.049 0.230  0.152  0.144  0.108  0.041  0.173
hc      0.914 0.165  1.000  0.951  0.251 0.320 -0.438 -0.539  0.635  0.425  0.451
hg      0.887 0.137  0.951  1.000  0.310 0.370 -0.430 -0.543  0.667  0.415  0.486
ferr    0.254 0.049  0.251  0.310  1.000 0.296 -0.109 -0.193  0.339  0.171  0.299
bmi     0.287 0.230  0.320  0.370  0.296 1.000  0.294  0.149  0.703  0.354  0.840
ssf    -0.396 0.152 -0.438 -0.430 -0.109 0.294  1.000  0.955 -0.222 -0.099  0.132
pcBfat -0.500 0.144 -0.539 -0.543 -0.193 0.149  0.955  1.000 -0.406 -0.250 -0.048
lbm     0.592 0.108  0.635  0.667  0.339 0.703 -0.222 -0.406  1.000  0.801  0.914
ht      0.401 0.041  0.425  0.415  0.171 0.354 -0.099 -0.250  0.801  1.000  0.778
wt      0.421 0.173  0.451  0.486  0.299 0.840  0.132 -0.048  0.914  0.778  1.000
> round(cor(mydata, use="complete.obs", method="kendall"), digits=3)
          rcc   wcc     hc     hg   ferr   bmi    ssf pcBfat    lbm     ht     wt
rcc     1.000 0.110  0.750  0.711  0.173 0.190 -0.263 -0.329  0.407  0.271  0.281
wcc     0.110 1.000  0.112  0.095  0.035 0.153  0.104  0.100  0.077  0.029  0.116
hc      0.750 0.112  1.000  0.825  0.167 0.216 -0.291 -0.354  0.438  0.285  0.302
hg      0.711 0.095  0.825  1.000  0.216 0.251 -0.291 -0.364  0.461  0.277  0.330
ferr    0.173 0.035  0.167  0.216  1.000 0.200 -0.072 -0.128  0.225  0.111  0.202
bmi     0.190 0.153  0.216  0.251  0.200 1.000  0.197  0.110  0.518  0.239  0.652
ssf    -0.263 0.104 -0.291 -0.291 -0.072 0.197  1.000  0.823 -0.132 -0.068  0.095
pcBfat -0.329 0.100 -0.354 -0.364 -0.128 0.110  0.823  1.000 -0.240 -0.162 -0.008
lbm     0.407 0.077  0.438  0.461  0.225 0.518 -0.132 -0.240  1.000  0.607  0.772
ht      0.271 0.029  0.285  0.277  0.111 0.239 -0.068 -0.162  0.607  1.000  0.589
wt      0.281 0.116  0.302  0.330  0.202 0.652  0.095 -0.008  0.772  0.589  1.000
```

Le correlazioni più significative sono così evidenziate (i metodi non parametrici come al solito tendono a fornire valori di significatività inferiori a quello fornito dal metodo parametrico).

[95] Per maggiore chiarezza potrebbe essere utile confrontare in codice qui riportato con quello impiegato nel caso della regressione lineare semplice.

Il calcolo dei coefficienti di correlazione con i livelli di significatività, che riporta la probabilità `p` di osservare per caso il valore calcolato, è identico a quello effettuato nel caso della regressione lineare semplice

```
# calcola i coefficienti di correlazione con i livelli di significatività                3/4
#
library(Hmisc) # carica il pacchetto
rcorr(as.matrix(mydata), type="pearson") # type può essere solo “pearson” (il classico r) o “spearman”
rcorr(as.matrix(mydata), type="spearman")
#
```

e conferma i valori evidenziati alla pagina precedente, mentre un ausilio ulteriore nella loro interpretazione ci viene ancora una volta dalla rappresentazione grafica dei dati (**Fig. 4.6**) che può essere effettuata per tutte le variabili del set di dati **ais**, dopo avere caricato il pacchetto **car** con **library(car)** e dopo avere aperto una finestra grafica con **windows()**, mediante una sola riga di codice:

```
# approfondimenti grafici sulla correlazione lineare                                     4/4
#
library(car) # carica il pacchetto
windows() # apre e inizializza una nuova finestra grafica
scatterplotMatrix(~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt, regLine = list(method=lm, lty=1,
lwd=2, col="red"), smooth=FALSE, diagonal=list(method="density", bw="nrd0", adjust=1,
kernel="gaussian", na.rm=TRUE), col = "black", main="Matrice di dispersione", data=ais) # traccia il
# diagramma che incrocia tutte le variabili
#
```

Per tutti i necessari chiarimenti sulla funzione **scattarplotMatrix()** si rimanda alla parte che tratta dei grafici di dispersione nel capitolo **5. Rappresentazione grafica dei dati**. Qui interessa solamente evidenziare che con l'argomento **~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt** viene richiesta la rappresentazione di tutte le variabili. E che in vari casi, che sono poi quelli nei quali il coefficiente di correlazione risulta elevato, la grafica conferma che potrebbe essere ragionevole applicare un modello di regressione lineare.

### 4.5.2. Regressione lineare multipla

Nella regressione lineare multipla la variabile dipendente **y** viene fatta dipendere da due o più variabili indipendenti **x** (indicate come $x_1$, $x_2$, $x_3$ … $x_n$) secondo l'equazione

$$y = a + b_1 \cdot x_1 + b_2 \cdot x_2 + b_3 \cdot x_3 + ..... + b_n \cdot x_n$$

che è l'estensione a uno spazio n-dimensionale della retta di regressione bivariata costruita sul piano cartesiano che abbiamo appena visto. Il risultato è rappresentato quindi da una intercetta **a** e da tanti coefficienti angolari **b** (indicati come $b_1$, $b_2$ $b_3$ ..... $b_n$) quante sono le variabili indipendenti che contribuiscono a determinare il valore della variabile dipendente.

Mettere in relazione tra di loro molte variabili contemporaneamente non è saggio in quant porta a risultati di difficile comprensione. Nel caso dei dati in questione abbiamo alcuni **dati ematologici** (rcc, wbc, hc, hg, ferr) e alcuni **dati biometrici** (bmi, ssf, pcBfat, lbm, ht, wt) dell'atleta ed è all'interno di questi due gruppi di dati che compaiono le relazioni lineari più interessanti (**Fig. 4.6**).

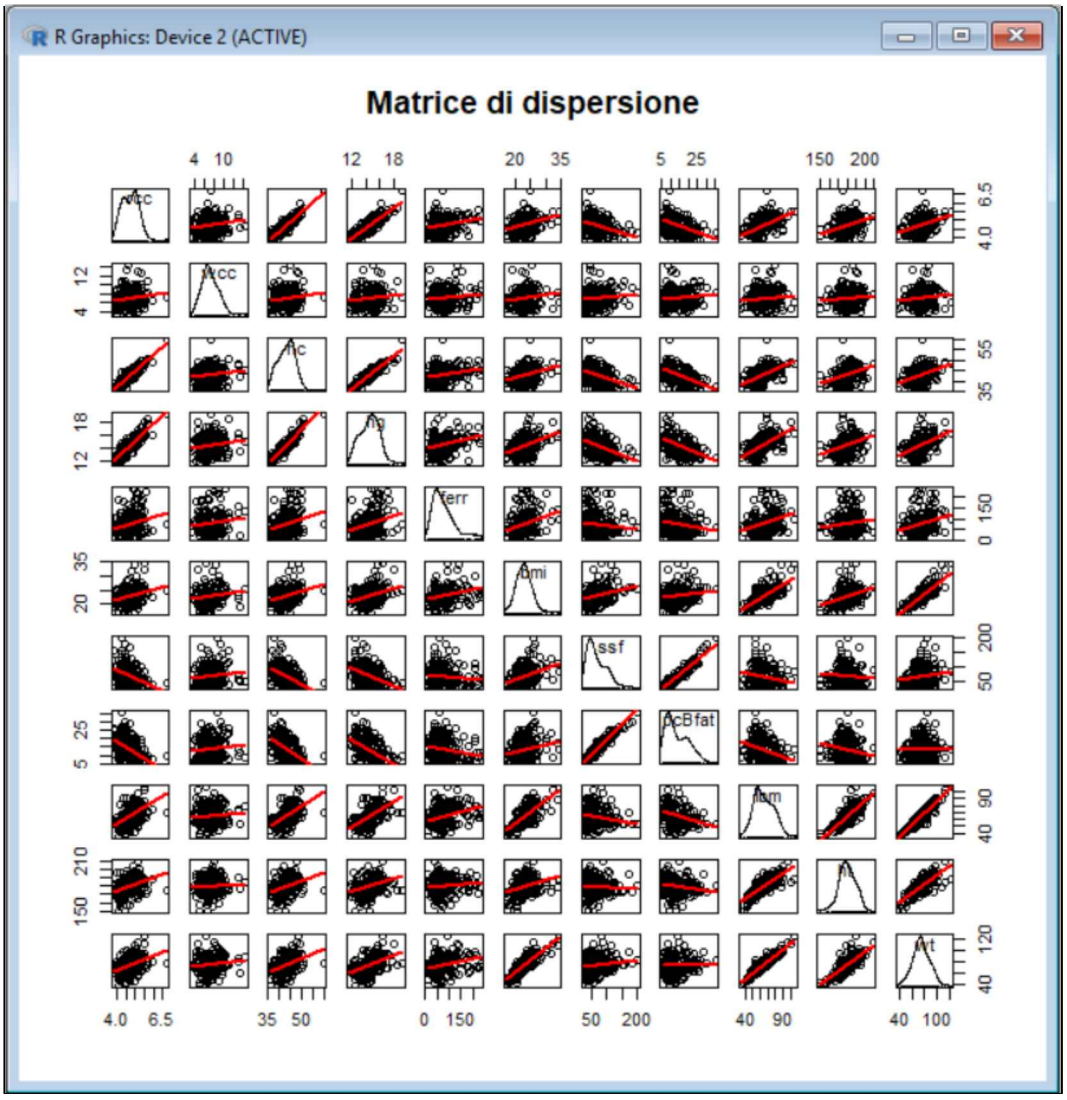



**Fig. 4.6** Matrice di dispersione dei dati ematologici e biometrici inclusi del set di dati **ais**.

Limitiamoci quindi ai **dati biometrici**. E per fornire un esempio di come calcolare la regressione lineare multipla con **R**, e di come questa sia in grado di fornire una stima ragionevolmente accurata di una grandezza, impieghiamo l'indice di massa corporea (BMI) che viene calcolato come rapporto tra il peso espresso in kg e il quadrato dell'altezza espressa in metri come variabile dipendente[96].

Nel set di dati **ais** disponiamo per tutti i 202 atleti i valori di BMI (variabile `bmi`), di peso (variabile `wt`) e di altezza (variabile `ht`). Possiamo quindi calcolare con **R** l'equazione della retta di regressione multipla

$$y = a + b_1 \cdot x_1 + b_2 \cdot x_2$$

ovvero, sostituendo alla **y** e alle **x** i nomi della variabili impiegate

[96] Vedere: **A7. Indice di massa corporea (BMI)**.

$$bmi = a + b_1 \cdot wt + b_2 \cdot ht$$

Il codice è molto semplice:

```
# REGRESSIONE LINEARE MULTIPLA y = a + b1·x1 + b2·x2 + b3·x3 + ..... + bn·xn          1/4
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
mydata <- ais[c(1,2,3,4,5,6,7,8,9,10,11)] # salva sole le colonne con le variabili numeriche in mydata
#
reglin <- lm(bmi~wt+ht, data=mydata) # calcola intercetta (a) e coefficienti angolari (b1) e (b2)
#
coefficients(reglin) # coefficienti dell'equazione bmi = a + b1·wt + b2·ht
confint(reglin, level=0.95) # intervalli di confidenza dell'intercetta e dei coefficienti angolari
#
```

Di fatto viene impiegata la stessa funzione **lm()** impiegata per la regressione lineare semplice a due variabili, semplicemente specificando questa volta due variabili indipendenti, per l'appunto peso e altezza. Questi i risultati:

```
> # REGRESSIONE LINEARE MULTIPLA y = a + b1·x1 + b2·x2 + b3·x3 + ..... +
bn·xn
> #
> reglin <- lm(bmi~wt+ht, data=mydata) # calcola intercetta (a) e
coefficienti angolari (b1) e (b2)
> coefficients(reglin) # coefficienti dell'equazione bmi = a + b1·wt +
b2·ht
(Intercept)          wt          ht
 43.8549825   0.3071626  -0.2439631
> confint(reglin, level=0.95) # intervalli di confidenza dell'intercetta
e dei coefficienti angolari
                2.5 %     97.5 %
(Intercept) 42.6586929 45.0512722
wt           0.3013181  0.3130071
ht          -0.2523237 -0.2356025
```

Quindi in base ai dati forniti nel set di dati **ais** l'equazione della retta di regressione lineare multipla assume la forma

$$bmi = 43.8549825 + 0.3071626 \cdot wt - 0.2439631 \cdot ht$$

Ora non resta che effettuare la prova del nove, che in questo caso è semplice. Prendiamo alcuni valori di peso e di altezza, e calcoliamo il BMI mediante la formula canonica

$$BMI = peso / altezza^2$$

dove l'altezza è espressa in centimetri come avviene nel set di dati **ais** (il risultato viene poi diviso per 10.000). Quindi riportiamo questi risultati nella tabella seguente (terza colonna) accanto ai valori di BMI ricalcolati mediante l'equazione della retta di regressione lineare multipla appena ricavata.

| Peso (kg) | Altezza (cm) | BMI formula | BMI regressione |
|---|---|---|---|
| 75 | 178 | 23,7 | 23,5 |
| 56 | 169 | 19,6 | 19,8 |
| 90 | 182 | 27,2 | 27,1 |
| 110 | 175 | 35,9 | 34,9 |
| 65 | 161 | 25,1 | 24,5 |
| 58 | 176 | 18,7 | 18,7 |
| 71 | 154 | 29,9 | 28,1 |
| 45 | 156 | 18,5 | 19,6 |

Come si vede le differenze sono contenute, confermando così la bontà del calcolo approssimato di questi dati mediante la regressione lineare multipla.

Altre funzioni per la diagnostica dell'adeguatezza della regressione che possono essere impiegate:

**summary(reglin)** # mostra un riepilogo dei risultati 2/4
**fitted(reglin)** # ricalcola i valori mediante l'equazione della retta di regressione
**residuals(reglin)** # calcola le differenze residue tra valore osservato e valore calcolato
**anova(reglin)** # analisi della varianza per le differenze spiegate dalle x
**vcov(reglin)** # matrice di covarianza dell'intercetta e del coefficiente angolare
#

La funzione **summary(reglin)** fornisce per $R^2$ (il coefficiente di determinazione che misura la frazione di variabilità spiegata dal modello di regressione) un confortante valore di 0.98 :

```
> summary(reglin) # mostra un riepilogo dei risultati

Call:
lm(formula = bmi ~ wt + ht, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0896 -0.1126  0.1188  0.1824  0.9908

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 43.854983   0.606651   72.29   <2e-16 ***
wt           0.307163   0.002964  103.64   <2e-16 ***
ht          -0.243963   0.004240  -57.54   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3655 on 199 degrees of freedom
Multiple R-squared:  0.9839,    Adjusted R-squared:  0.9837
F-statistic:  6072 on 2 and 199 DF,  p-value: < 2.2e-16
```

Per le grandezze fornite dalle funzioni **fitted(reglin)**, **residuals(reglin)**, **anova(reglin)**, **vcov(reglin)**, **influence(reglin)** si rimanda ancora una volta alla documentazione di **R** ai testi di statistica.

Possono essere eseguiti gli stessi test per la linearità e per l'identificazione dei dati aberranti già impiegati nel caso della regressione lineare semplice:

```
library(gvlma) # carica la libreria                                    3/4
summary(gvlma(reglin)) #  test globale per l'assunto di linearità
#
library(car) # carica la libreria
outlierTest(reglin) # valore p di Bonferonni per la presenza di dati aberranti (outliers)
#
```

Si può infine effettuare una diagnostica della regressione valutando graficamente l'importanza relativa di ciascuna variabile indipendente nel determinare il valore della variabile dipendente, tracciando i grafici che consentono di identificare i dati che influiscono particolarmente sulle conclusioni (leverage plot) ed effettuando una valutazione della linearità mediante il grafico di Ceres e il grafico dei quantili:

```
library(relaimpo) # carica la libreria                                 4/4
windows() # apre e inizializza una nuova finestra grafica
plot(calc.relimp(reglin, type = "lmg", rank=TRUE, diff=TRUE, rela=TRUE), main="Importanza relativa delle
variabili indipendenti") #  grafico dell'importanza relativa di ciascuna variabile indipendente
#
booteval.relimp(boot.relimp(reglin, b = 1000, type = "lmg", rank=TRUE, diff=TRUE, rela=TRUE)) # ripete
# il calcolo aggiungendo gli intervalli di confidenza dei valori mediante bootstrap
windows() # apre e inizializza una nuova finestra grafica
plot(booteval.relimp(boot.relimp(reglin, b = 1000, type = "lmg", rank=TRUE, diff=TRUE,
rela=TRUE)),sort=TRUE, main="Importanza relativa delle variabili indipendenti") # traccia il nuovo
# grafico
#
library(car) # carica la libreria
windows() # apre e inizializza una nuova finestra grafica
leveragePlots(reglin, ask=FALSE) # grafico dell'influenza dei dati sulle conclusioni (leverage plot)
windows() # apre e inizializza una nuova finestra grafica
ceresPlots(reglin, ask=FALSE) # grafico per la valutazione della linearità (grafico di Ceres)
windows() # apre e inizializza una nuova finestra grafica
qqPlot(reglin, main="Grafico dei quantili per i residui") # grafico dei quantili per i residui studentizzati
#
```

Questo rappresentazioni grafiche sono riportate per illustrare alcune delle potenzialità di **R** nell'analisi di dati multivariati, ma non sono ulteriormente discusse in quanto vanno al di la dei limiti di questa guida[97].

**Nota bene**: le finestre si sovrappongono l'una all'altra, iconizzarle per esaminare i grafici uno ad uno.

### 4.5.3. Analisi dei gruppi (cluster)

Il concetto che sta alla base dell'analisi dei gruppi o cluster analysis è semplice: raggruppare oggetti omogenei in insiemi (cluster) partendo dagli oggetti più simili e aggiungendo progressivamente gli oggetti più dissimili.

[97] Per i dettagli di queste rappresentazioni grafiche si rimanda ai manuali di riferimento dei relativi pacchetti **gvlma**, **relaimpo** e **car** su: *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

All'inizio del processo di classificazione ad ogni oggetto corrisponde un cluster (e viceversa). In questo stadio tutti gli oggetti sono considerati dissimili tra di loro. Al passaggio successivo i due oggetti più simili sono raggruppati nello stesso cluster. Il numero dei cluster risulta quindi pari al numero di oggetti diminuito di 1. Il procedimento viene ripetuto ciclicamente, fino ad ottenere (all'ultimo passaggio) un unico cluster. Stabilire a quale livello di aggregazione degli oggetti fermarsi, e quindi quali conclusioni trarre, dipende in larga parte dal giudizio di merito dell'utilizzatore. Per questo la cluster analysis riveste un ruolo centrale, in statistica, nella analisi esplorativa dei dati.

Fate il `download` del file `bmi.csv` quindi copiatelo in `C:\Rdati\`.

In alternativa potete anche copiare il testo riportato qui sotto e salvarlo in `C:\Rdati\` in un file di testo denominato `bmi.csv` (attenzione all'estensione al momento del salvataggio del file). Si tratta dei valori di BMI rilevati a livello europeo e pubblicati dall'Istat[98].

```
Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7
Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2
```

Si effettua un'analisi esplorativa dei dati mediante il metodo classico di clusterizzazione (**Fig. 4.7**).

I codice **R** necessario per generare questa rappresentazione è molto compatto ed è riportaato alla pagina seguente.

---

[98] Vedere: **A6. BMI nei Paesi europei**.

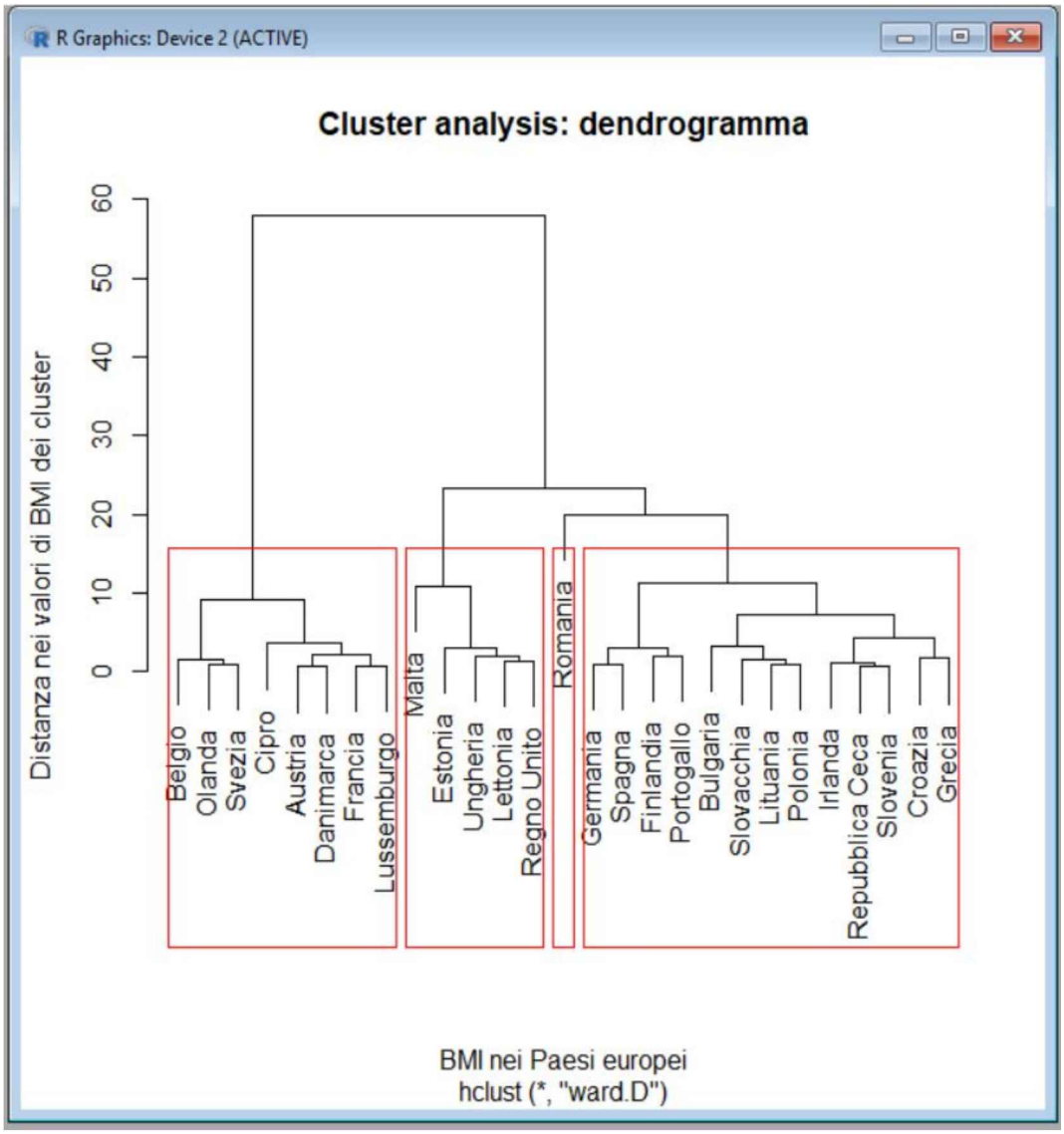



**Fig. 4.7** Cluster analysis dei valori di BMI rilevati nel 2015 nei Paesi europei.

```
# ANALISI DEI GRUPPI (CLUSTER ANALYSIS)
#
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";", row.names="Nazione") # con la prima
# riga sono importati i dati
mydata # mostra i dati
#
# effettua il clustering gerarchico con il metodo di Ward
#
mat <- hclust(dist(mydata, method = "euclidean"), method="ward.D") # matrice delle distanze euclidee
plot(mat, main="Cluster analysis: dendrogramma", xlab="BMI nei Paesi europei", ylab="Distanza nei
valori di BMI dei cluster") # traccia il dendrogramma
groups <- cutree(mat, k=4) # divide in 4 gruppi/cluster principali, valore k da cambiare al bisogno
rect.hclust(mat, k=4, border="red") # evidenzia i 4 gruppi/cluster
#
```

Con la funzione **dist()** viene calcolata la matrice delle distanze. Qui viene impiegato come metodo di calcolo il metodo di euclideo (**method = "euclidean"**), ma è possibile il calcolo anche con metodi alternativi[99].

| PAESI | INDICE MASSA CORPOREA | | | |
|---|---|---|---|---|
| | Sottopeso | Normale | Sovrappeso | Obeso |
| Belgio | 2,7 | 48,0 | 35,3 | 14,0 |
| Olanda | 1,6 | 49,0 | 36,0 | 13,3 |
| Svezia | 1,8 | 48,3 | 35,9 | 14,0 |
| Cipro | 3,9 | 47,8 | 33,8 | 14,5 |
| Austria | 2,4 | 49,6 | 33,3 | 14,7 |
| Danimarca | 2,2 | 50,0 | 32,9 | 14,9 |
| Francia | 3,2 | 49,6 | 31,9 | 15,3 |
| Lussemburgo | 2,8 | 49,3 | 32,4 | 15,6 |
| Malta | 2,0 | 37,0 | 35,0 | 26,0 |
| Estonia | 2,2 | 43,9 | 33,5 | 20,4 |
| Ungheria | 2,9 | 41,9 | 34,0 | 21,2 |
| Lettonia | 1,7 | 41,8 | 35,2 | 21,3 |
| Regno Unito | 2,1 | 42,2 | 35,6 | 20,1 |
| Romania | 1,3 | 42,9 | 46,4 | 9,4 |
| Germania | 1,8 | 46,1 | 35,2 | 16,9 |
| Spagna | 2,2 | 45,4 | 35,7 | 16,7 |
| Finlandia | 1,2 | 44,1 | 36,4 | 18,3 |
| Portogallo | 1,8 | 44,6 | 36,9 | 16,6 |
| Bulgaria | 2,2 | 43,8 | 39,2 | 14,8 |
| Slovacchia | 2,1 | 43,6 | 38,0 | 16,3 |
| Lituania | 1,9 | 42,5 | 38,3 | 17,3 |
| Polonia | 2,4 | 42,9 | 37,5 | 17,2 |
| Irlanda | 1,9 | 42,3 | 37,0 | 18,7 |
| Repubblica Ceca | 1,1 | 42,1 | 37,6 | 19,3 |
| Slovenia | 1,6 | 41,8 | 37,4 | 19,2 |
| Croazia | 1,9 | 40,7 | 38,7 | 18,7 |
| Grecia | 1,9 | 41,3 | 39,4 | 17,3 |

**Fig. 4.8** L'ordinamento della tabella contenente i dati analizzati consente di confermare empiricamente la validità dei criteri e dei risultati ottenuti mediante la cluster analysis.

Con la funzione **hclust()** viene realizzata, sulla base della matrice delle distanze, l'aggregazione gerarchica dei casi. Il metodo qui impiegato è quello di Ward (**method="ward.D"**) ma cambiando il valore di questo argomento è possibile impiegare anche metodi alternativi[100].

Infine l'argomento **k=4** (che può essere cambiato al bisogno) della funzione **cutree()** consente di definire il numero dei gruppi da evidenziare nel dendrogramma.

Per confermare la validità della clusterizzazione i dati originali forniti dall'Istat sono stati importati in un foglio elettronico e ordinati e raggruppati nei quattro cluster generati da **R** nella **Fig. 4.8** nella quale è possibile notare che:
→ gli individui sottopeso sono distribuiti in modo omogeneo in un ambito di valori piuttosto ristretto e

[99] Digitate **help(dist)** nella Console di R per vedere gli altri valori possibili per l'argomento **method** della funzione **dist()**.
[100] Digitate **help(hclust)** nella Console di R per vedere gli altri valori possibili per l'argomento **method** della funzione **hclust()**.

quindi tendono ad influenzare poco la suddivisione in cluster dei Paesi;
→ gli individui con peso normale superano sempre il 40% con la sola eccezione di Malta;
→ solo in Romania la percentuale di sovrappeso supera il 40%, con un netto salto numerico rispetto agli altri Paesi;
→ in tutti i Paesi la percentuale di obesi è a due cifre, tranne che in Romania, che di nuovo si discosta dagli altri Paesi con un netto salto numerico;
→ nei Paesi in colore giallo la percentuale di obesi è sempre inferiore al 16%;
→ nei paesi in colore azzurro la percentuale di obesi è sempre superiore al 16% (con una sola eccezione) e inferiore al 20%;
→ nei paesi in colore verde la percentuale di obesi supera sempre il 20%.

Da questa analisi si ricava una sostanziale conferma, anche se empirica, dei criteri di clusterizzazione.

## 4.6. Teorema di Bayes

Si consideri uno schema come il seguente (**Fig. 4.9**) nel quale **C** rappresenta una delle tante cause osservabili, **E** rappresenta uno dei tanti effetti osservabili, e le frecce dirette dalla causa all'effetto rappresentano i rapporti di causa/effetto.

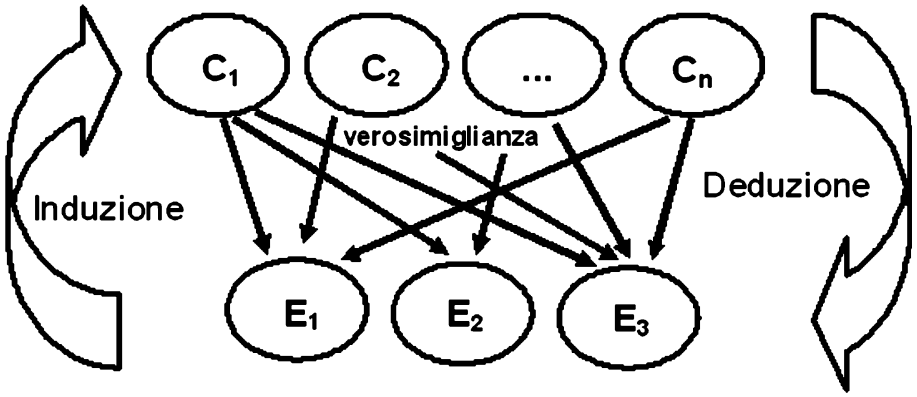



**Fig. 4.9** Il teorema di Bayes consente, a partire dagli effetti osservati, di calcolare la verosimiglianza delle cause.

Il "problema classico" della probabilità[101] può essere risolto applicando la deduzione logica. Conoscendo la causa, possiamo da questa dedurre gli effetti corrispondenti. Così sapendo che un dado ha sei facce, quindi conoscendo la causa, possiamo da questa dedurre la probabilità che ha una specifica faccia di comparire al prossimo lancio del dado (l'effetto). La soluzione del cosiddetto "problema inverso", ovvero l'induzione logica, è più difficile, in quanto ad un effetto possono corrispondere più cause. Il teorema di Bayes[102] consente, a partire dagli effetti osservati, di calcolare la verosimiglianza delle cause, espressa ovviamente nell'unico modo possibile e cioè in termini di probabilità.

Il teorema di Bayes trova applicazione nel campo della diagnosi medica[103,104,105]. La quale prevede che data la conoscenza delle malattie (cause) e dei rispettivi sintomi e segni (effetti), il medico debba risalire dai sintomi e dai segni presenti in uno specifico paziente (effetti) alla malattia (causa) che li ha determinati. Compito sovente non facile dato che gli effetti possono essere attribuiti a differenti cause.

Un nuovo test per la diagnosi della malattia X è stato provato su 1586 pazienti[106]. Di 744 pazienti che

[101] Vedi: **A2. Definizioni della probabilità**.
[102] Reverend Thomas Bayes. *An assay toward solving a problem in the doctrine of chance*. Philo. Trans. Roy. Soc., vol. 53, 370-418, 1763. URL consultato il 28/11/2018: https://doi.org/10.1098/rstl.1763.0053
[103] Galen RS, Gambino RS. *Oltre il concetto di normalità: il valore predittivo e l'efficienza delle diagnosi mediche*. Piccin Editore, 1980, ISBN 88-212-0770-6.
[104] Federspil G. *Logica clinica. I principe del metodo in medicina*. The McGraw-Hill Companies, Milano, 2004, ISBN 88-386-2984-6.
[105] Motterlini M, Crupi V. *Decisioni mediche*. Raffaello Cortina Editore, Milano, 2005, ISBN 88-7078-632-3.
[106] L'esempio è tratto da Scott IA, Greenberg PB, Poole PJ. *Cautionary tales in the clinical interpretation of studies of*

avevano la malattia, 670 sono risultati positivi al test. Di 842 pazienti che non avevano la malattia, 640 sono risultati negativi al test. Il teorema di Bayes consente di rispondere alla domanda: quale probabilità ha di essere malato un soggetto nel quale i test è risultato positivo?

Possiamo riportate i dati forniti in una tabella e completarla calcolando i valori mancanti:

| | Malattia + | Malattia - | Totale |
|---|---|---|---|
| | | | |
| Test + | 670 | 202 | 872 |
| Test - | 74 | 640 | 714 |
| | | | |
| Totale | 744 | 842 | 1.586 |

Dai dati forniti si ricava che erano[107]:
→ 670 i **veri positivi** (VP) cioè i soggetti **malati con il test positivo**;
→ 202 i **falsi positivi** (FP) cioè i soggetti **sani con il test positivo**;
→ 74 i **falsi negativi** (FN) cioè i soggetti **malati con il test negativo**;
→ 640 i **veri negativi** (VN) cioè i soggetti **sani con il test negativo**.

Questo è quindi lo schema dei dati in ingresso necessari per il pacchetto **epiR**, che deve essere preventivamente scaricato dal **CRAN**, corrispondenti ai dati della tabella precedente:

| | Malattia + | Malattia - |
|---|---|---|
| | | |
| Test + | VP | FP |
| Test - | FN | VN |

Questo è lo script:

```
# TEOREMA DI BAYES
#
library(epiR) # carica il pacchetto
data <- as.table(matrix(c(670,202,74,640), nrow = 2, byrow = TRUE)) # i dati sono immessi direttamente
# dalla Console di R
test <- epi.tests(data, conf.level = 0.95) # sono calcolate le statistiche
summary(test) # sono mostrate le statistiche
#
```

I dati sono caricati manualmente dal codice **R** con questa sequenza:
→ la funzione **c()** ha come argomento i dati (**670,202,74,640**) e genera il vettore che li contiene;
→ la funzione **matrix()** li inserisce in una matrice di due righe (**nrow = 2**) e dato l'argomento **byrow = TRUE** li inserisce per riga, quindi da sinistra a destra e dall'alto in basso;
→ la funzione **as.table()** genera la tabella definitiva.

Il test viene eseguito mediante la funzione **epi.test()** che prevede come argomenti solamente:
→ i dati in ingresso (**data**);

---

*diagnostic tests*. Internal Medicine Journal 38 (2008) 120–129.
[107] Vedi : **A8. Teorema di Bayes e diagnosi medica**.

→ l'indicazione del livello di confidenza (**conf.level = 0.95**) con il quale calcolare i limiti inferiore e superiore dell'intervallo di confidenza di ciascuna delle statistiche.

Le grandezze calcolate con **epiR** sono quindi riportate ciascuna con il rispettivo intervallo di confidenza al 95%:

```
> # TEOREMA DI BAYES
> #
> library(epiR) # carica il pacchetto
> data <- as.table(matrix(c(670,202,74,640), nrow = 2, byrow = TRUE)) # i
dati sono immessi direttamente dalla Console di R
> test <- epi.tests(data, conf.level = 0.95) # sono calcolate le
statistiche
> summary(test) # sono mostrate le statistiche
                est      lower      upper
aprev     0.5498108  0.5249373  0.5744996
tprev     0.4691047  0.4443055  0.4940184
se        0.9005376  0.8767462  0.9210923
sp        0.7600950  0.7297765  0.7885803
diag.acc  0.8259773  0.8064049  0.8443346
diag.or  28.6861119 21.5181917 38.2417364
nnd       1.5137005  1.4091004  1.6487431
youden    0.6606326  0.6065226  0.7096726
ppv       0.7683486  0.7388926  0.7959784
npv       0.8963585  0.8716393  0.9177402
plr       3.7537262  3.3206884  4.2432346
nlr       0.1308552  0.1050643  0.1629771
```

Si riassumono qui di seguito le statistiche calcolate riportando (tra parentesi) il loro significato, le formule con cui sono calcolate e il risultato numerico ottenuto con (tra parentesi) il relativo intervallo di confidenza al 95%:

**aprev** (prevalenza apparente, soggetti con il test positivo)
(VP+FP)/(VP+FP+FN+VN) = 0.5498108 (0.5249373 - 0.5744996)

**tprev** (prevalenza reale, soggetti con la malattia)
(VP+FN)/(VP+FP+FN+VN) = 0.4691047 (0.4443055 - 0.4940184)

**se** (sensibilità, positività nei malati)
VP/(VP+FN) = 0.9005376 (0.8767462 - 0.9210923)

**sp** (specificità, negatività nei sani)
VN/(VN+FP) = 0.7600950 (0.7297765 - 0.7885803)

**diag.acc** (accuratezza diagnostica)
(VP+VN)/(VP+FP+FN+VN) = 0.8259773 (0.8064049 - 0.8443346)

**diag.or** (odd ratio)
rapporto LR+/LR- = 28.6861119 (21.5181917 - 38.2417364)

**nnd** (numero necessario per la diagnosi)

1/(sensibilità - (1 - specificità)) = 1.5137005 (1.4091004 - 1.6487431)

**youden** (indice di Youden)
sensibilità + specificità - 1 = 0.6606326 (0.6065226 - 0.7096726)

**ppv** (valore predittivo di un test positivo)
VP/(VP+FP) = 0.7683486 (0.7388926 - 0.7959784)

**npv** (valore predittivo di un test negativo)
VN/(VN+FN) = 0.8963585 (0.8716393 - 0.9177402)

**plr** (rapporto di verosimiglianza LR+ per un test positivo)[108]
(VP/(VP+FN))/(FP/(FP+VN)) = 3.7537262 (3.3206884 - 4.2432346)

**nlr** (rapporto di verosimiglianza LR- per un test negativo)
(FN/(VP+FN))/(VN/(FP+VN)) = 0.1308552 (0.1050643 - 0.1629771)

---

[108] Altman DG, Deeks JJ. Diagnostic tests 4: likelihood ratios. BMJ 2004;329;168-169. URL consultato il 29/10/2018: https://goo.gl/ahpbpN

# 5. R - rappresentazione grafica dei dati

*"La semplicità, cosa rarissima ai nostri tempi."*
*(Ovidio)*

*"Se non si è un genio, è bene mirare ad essere chiaro."*
*(Anthony Hope)*

---

La **tabulazione** di dati è per definizione propedeutica all'analisi statistica e all'analisi grafica. Tabulando i dati è possibile verificare la eventuale mancanza di qualcuno di essi, ovvero la presenza di dati aberranti (per la distinzione tra errori e sbagli vedere Baldini[109]).

Al pari della tabulazione dei dati, tipicamente sviluppata mediante un foglio elettronico, la **rappresentazione grafica** è un semplice ma importante strumento di analisi esplorativa dei dati: può anch'essa risultare utile per la identificazione dei dati aberranti e consente di effettuare una ulteriore valutazione preliminare dei risultati.

Rispetto alla tabulazione dei dati, che offre una visione analitica e numericamente completa dei singoli dati raccolti, la rappresentazione grafica comporta una perdita di dettaglio (il singolo valore numerico non è più caratterizzato con esattezza): tuttavia offre il vantaggio di una visione più concisa e sintetica e, fatto molto importante, tende a fare emergere l'informazione più rilevante. Per questo motivo la rappresentazione grafica risulta utile per caratterizzare il tipo di distribuzione assunto da una variabile, come pure il tipo di relazione che intercorre tra due variabili, e addirittura può fare emergere relazioni nascoste o impreviste tra variabili. Proprio per lo stesso fatto di offrire una visione più concisa e sintetica e di tendere a fare emergere l'informazione più rilevante, la rappresentazione grafica è molto usata come strumento per il riepilogo dei risultati. Per l'importanza della rappresentazione grafica come strumento per l'analisi esplorativa dei dati e per il riepilogo dei risultati vedere Bossi[110], Lantieri[111] e Campbell[112]. Un indicazione sull'impiego della rappresentazione grafica appropriata ij relazione alla scala impiegata per esprimere i dati è riportata in appendice[113].

Al bisogno le rappresentazioni grafiche qui riportate possono essere facilmente salvate sotto forma di file in tutti i più comuni formati grafici[114].

**Nota bene**: negli script che seguono le nuove finestre grafiche aperte si sovrappongono alle precedenti, è necessario spostarle o iconizzarle per visualizzare i grafici uno ad uno.

---

[109] Baldini M. Citato in (a cura di) Binanti L. *Sbagliando s'impara. Una rivalutazione dell'errore*. Armando, Roma, 2005, ISBN 88-8358-819-3, p. 77. Vedere anche: Baldini M. *L'errore nella scienza*. Biochim Clin, 1991;15:28-38.
[110] Bossi A, Cortinovis I, Duca P, Marubini E. *Introduzione alla statistica medica*. Roma: La Nuova Italia Scientifica, 1994:37-68.
[111] Lantieri PB, Risso D, Rovida S, Ravera G. *Statistica medica ed elementi di informatica*. Milano: McGraw-Hill, 1994:71-101.
[112] Campbell MJ, Machin D. *Medical statistics. A commonsense approach*. Chichester: John Wiley & Sons, 1993:44-59.
[113] Vedere: **A9. Scale nominali, scale ordinali e scale numeriche**.
[114] Vedere: **5.8. Salvare i grafici di R in un file**.

## 5.1. Grafici a torta

Per la rappresentazione grafica di dati qualitativi il metodo più generico, ma anche più limitato, in quanto adatto essenzialmente alla rappresentazione dei rapporti percentuali, è rappresentato dai diagrammi a torta (o pie-chart). Tuttavia a causa del fatto che l'occhio umano ha una scarsa capacità nella valutazione degli angoli i grafici a torta non sono raccomandati nella rappresentazione di dati scientifici[115].

In ogni caso anche in **R** sono disponibili strumenti grafici per la realizzazione dei grafici a torta.

Nell'esempio fornito da Marubini[116] il numero di individui prodotti per ciascuna delle varietà **AB**, **Ab**, **aB**, **ab** in un esperimento di ibridazione di specie vegetali era:

| **AB** | **Ab** | **aB** | **ab** |
|---|---|---|---|
| 72 | 29 | 36 | 12 |

In questa prima parte dello script viene generato prima un grafico a torta molto semplice, poi lo stesso grafico a torta viene riproposto con una legenda esterna e riportando le percentuali calcolate per ciascuno dei valori originali:

```
# GRAFICI A TORTA                                                    1/3
#
val <-  c(72, 29, 36, 12) # valori
eti <-  c("AB","Ab","aB","ab") # etichette
percent <- round(100*val/sum(val), 1) # percentuale per ciascun valore
#
windows() # apre e inizializza una nuova finestra grafica
pie(val, eti, main = "Individui prodotti per ciascuna varietà", col = rainbow(length(val))) # grafico a torta
# semplice
#
windows() # apre e inizializza una nuova finestra grafica
pie(val, labels = percent, main = "Individui prodotti per ciascuna varietà",col = rainbow(length(val))) #
# grafico a torta con percentuali e legenda esterna
legend("topright", eti, cex = 0.8, fill = rainbow(length(val))) # legenda
#
```

Con le prime tre righe di codice sono generati l'array **val** che contiene i valori corrispondenti alle quattro "fette" della torta, l'array **eti** che contiene le rispettive etichette/descrittori, quindi (terza riga) ognuno dei valori dell'array **val** viene diviso per la somma dei valori **sum(val)** e moltiplicato per 100 al fine di trasformare i valori originali in percentuale, e infine il risultato viene arrotondato a un decimale con la funzione **round(, 1)**. e salvato (**<-** ) nell'array **percent** .

Dopo avere aperto alla quarta riga una nuova finestra grafica (**windows()**), alla quinta riga la funzione **pie()**

[115] "*Percentages ... can also be expressed using a pie-chart, but since the human eye is very poor at comparing angles, we do not recommend these for display purposes*". Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 54.
[116] Vedere: **4.1. Test chi-quadrato ($\chi^2$)**

traccia un grafico a torta molto semplice impiegando i valori **val** e le etichette **eti**, mette un titolo (**main=""**) al grafico e lo colora cambiando colore a ciascuna "fetta" (**rainbow()**) impiegando tanti colori quante sono le fette (**lenght(val)**). Questo primo grafico non viene riportato.

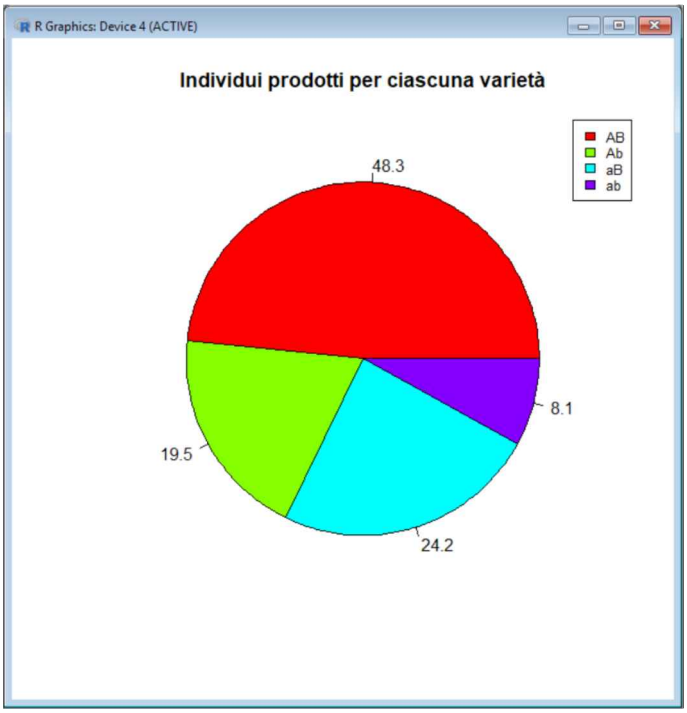



**Fig. 5.1** Grafico a torta che riporta le percentuali calcolate per ciascuno dei valori originali e una legenda esterna.

Nelle ultime tre righe di codice viene aperta una nuova finestra grafica (**windows()**)[117], viene tracciato di nuovo il grafico a torta questa volta riportando come etichette i valori percentuali (**labels = percent**) calcolati in precedenza, e infine viene riportata la legenda (**legend()**) in alto a destra (**"topright"**) con le etichette **eti**, i colori definiti dall'argomento **fill**, legenda che avrà le dimensioni definite mediante l'argomento **cex = 0.8**, che consente di rimpicciolirla o di allargarla (potete fare una rapida prova). Il risultato è riportato nella **Fig. 5.1**.

Oltre ai grafici a torta tradizionali è possibile generare grafici a torta 3D. Lo facciamo mediante l'impiego di un pacchetto aggiuntivo, scaricando il pacchetto **plotrix** dal **CRAN**, seguendo la procedura che a questo punto dovrebbe essere familiare. Lo script continua caricando il pacchetto (**library(plotrix)**), viene quindi aperta una nuova finestra grafica (**windows()**) e viene generato un grafico a torta 3D semplice (che non viene riportato):

```
# grafico a torta 3D                                                    2/3
#
library(plotrix) # carica il pacchetto
windows() # apre e inizializza una nuova finestra grafica
pie3D(val, labels = eti, explode = 0.1, main = "Individui prodotti per ciascuna varietà") # grafico
#
```

[117] Attenzione: in **R** ogni nuova finestra si sovrappone alla/e precedente/i.

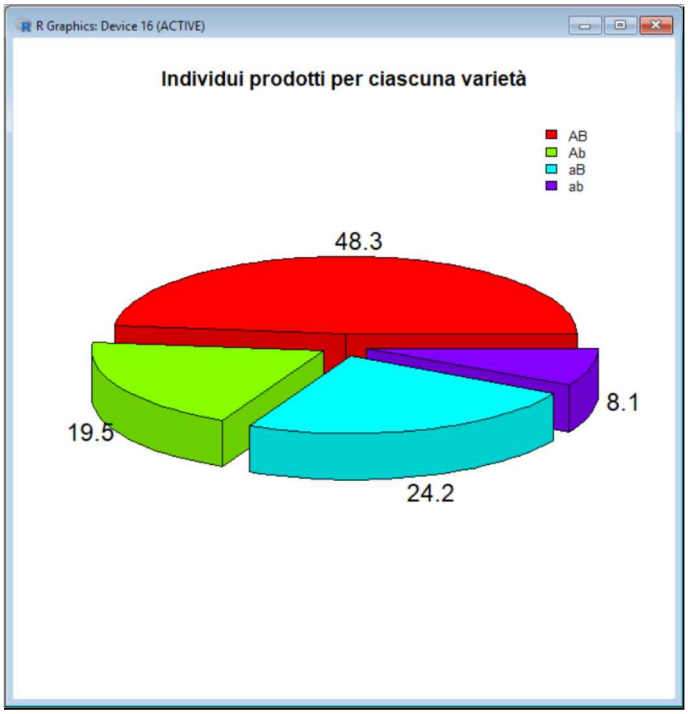



**Fig. 5.2** Grafico a torta 3D che riporta le percentuali calcolate per ciascuno dei valori originali e una legenda esterna.

Le novità rispetto al codice precedente sono rappresentate dalla funzione **pie3D()** in sostituzione della funzione **pie()** e dall'argomento **explode = 0.1** che consente di variare quanto le fette sono "esplose". Provate a cambiare il valore, è possibile scegliere non solo valori superiori ma anche valori inferiori a **0.1**.

Con il codice che segue, a completamento dello script sui grafici a torta, lo stesso grafico 3D viene riproposto completato con una legenda esterna e riportando le percentuali calcolate per ciascuno dei valori originali (**Fig. 5.2**). L'argomento **bty="n"** consente di eliminare il riquadro delle didascalie che era presente nella figura precedente.

```
# grafico a torta 3D con percentuali e legenda esterna                    3/3
#
windows() # apre e inizializza una nuova finestra grafica
pie3D(val, labels = percent, explode = 0.1, main = "Individui prodotti per ciascuna varietà", col =
rainbow(length(val))) # grafico
legend("topright", eti, cex = 0.8, fill = rainbow(length(val)), bty="n") # legenda
#
```

## 5.2. Grafici a barre

Riprendiamo i dati di Snedecor relativi ai decessi avvenuti in un gruppo di non fumatori e in un gruppo di fumatori di pipa[118]

| *Esito* | *Non fumatori* | *Fumatori di pipa* |
|---|---|---|
| Deceduti | 117 | 54 |
| Viventi | 950 | 348 |

e per un confronto grafico adeguato li trasformiamo in percentuale:

| *Esito* | *Non fumatori* | *Fumatori di pipa* |
|---|---|---|
| Deceduti | 11.0 | 13.4 |
| Viventi | 89.0 | 86.6 |

Vediamo ora uno script che permette di generare due differenti **grafici a barre** per questi ultimi dati:

```
# GRAFICI A BARRE AFFIANCATE
#
cells <- c(11.0,13.4,89.0,86.6) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Deceduti", "Viventi") # genera l'array rnames con i nomi delle righe
cnames <- c("Non_fumatori", "Fumatori_di_pipa") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
#
mydata # matrice dati
t(mydata) # matrice trasposta
#
windows() # apre e inizializza una nuova finestra grafica
barplot(mydata,beside=TRUE, legend=TRUE, ylim=c(0,150), col=c("darkblue","red"), ylab="Frequenze
osservate", xlab="", main="Grafico a barre matrice originale") # grafico a barre matrice originale
#
windows() # apre e inizializza una nuova finestra grafica
barplot(t(mydata),beside=TRUE, legend=TRUE, ylim=c(0,150), col=c("darkblue","red"), ylab="Frequenze
osservate", xlab="", main="Grafico a barre matrice trasposta") # grafico a barre matrice trasposta
#
```

Gli array **cells**, **rnames** e **cnames** generati con le prime tre righe di codice contengono rispettivamente le percentuali di casi osservati, i nomi delle righe e i nomi delle colonne. I tre array sono combinati nella matrice dati mediante la funzione **matrix()**. Questa ricava i dati dall'array **cells** e li inserisce in due righe (**nrow=2**) di cinque colonne (**ncol=5**) procedendo per riga (**byrow=TRUE**) cosicché i dati risulteranno inseriti da sinistra a destra e dall'alto in basso, e quindi i primi due valori dell'array **cells** (**11.0,13.4**) verranno collocati nella prima riga e i successivi due valori (**89.0,86.6**) verranno collocati nella seconda riga della matrice dati.

[118] Vedere: **4.1. Test chi-quadrato ($\chi^2$)**.

A questo punto per fare il punto della situazione sono mostrate sia la matrice dati **mydata** sia la sua matrice trasposta **t(mydata)**:

```
> mydata # matrice dati
          Non_fumatori Fumatori_di_pipa
Deceduti            11             13.4
Viventi             89             86.6
> t(mydata) # matrice trasposta
                 Deceduti Viventi
Non_fumatori         11.0    89.0
Fumatori_di_pipa     13.4    86.6
```

Viene quindi aperta una nuova finestra (**windows()**) e generato con la funzione **barplot()** un primo grafico a barre (**Fig. 5.3**) impiegando questi argomenti:
→ per i dati viene impiegata la matrice dati originale **mydata**;
→ l'argomento **beside=TRUE** specifica che le barre sono affiancate e non sovrapposte in pila;
→ con l'argomento **legend=TRUE** viene generata una legenda;
→ con l'argomento **ylim=c(0,150)** sono impostati il valore minimo e il valore massimo per l'asse delle y;
→ con l'argomento **col** è impostato il colore delle barre.

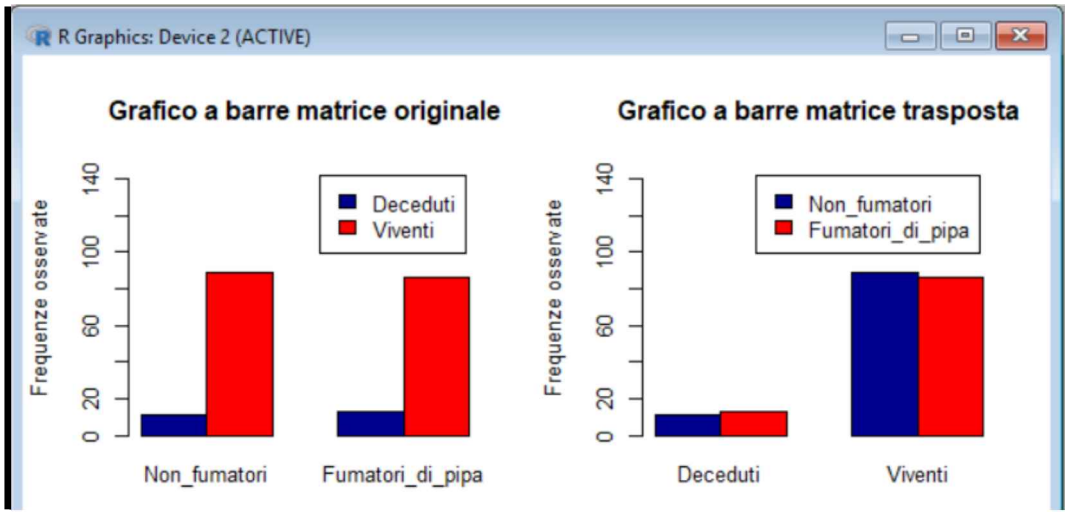



**Fig. 5.3** Grafici a barre per i dati di Snedecor sulla mortalità nei fumatori di pipa.

Il secondo grafico con gli stessi identici argomenti presenta i dati della matrice trasposta **t(mydata)**.

Riprendiamo ora i dati sottoposti in precedenza al test di McNemar. Nella sperimentazione 250 pazienti sofferenti di artrite erano sottoposti al trattamento con il farmaco A e al trattamento con il farmaco B. Era stato poi rilevato il grado di soddisfazione di ciascuno di essi rispetto all'uno e all'altro trattamento:

| *Esito* | *Soddisfatto da A* | *Non soddisfatto da A* |
|---|---|---|
| Soddisfatto da B | 150 | 20 |
| Non soddisfatto da B | 30 | 50 |

In totale 150 soggetti si erano mostrati soddisfatti di entrambi i trattamenti, 50 avevano espresso insoddisfazione per entrambi, mente altri 50 si sono mostrati insoddisfatti 20 dell'uno e 30 dell'altro. In questo caso è inutile trasformare i dati in percentuali, in quanto le proporzioni rimarrebbero immodificate.

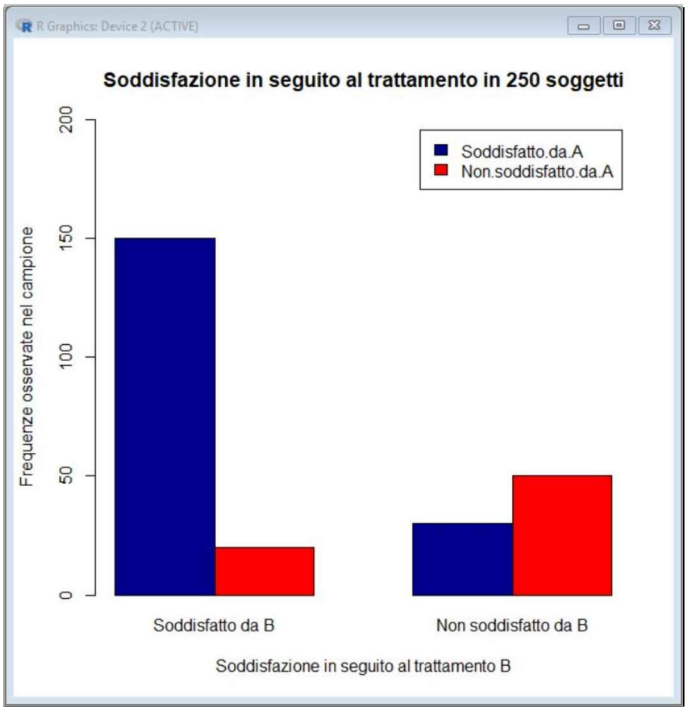



**Fig. 5.4.** I risultati del test di McNemar ilustrati con un grafico a barre affiancate.

Questo è lo script:

```
# GRAFICI A BARRE AFFIANCATE E A BARRE SOVRAPPOSTE
#
cells <- c(150,20,30,50) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Soddisfatto_da_B", "Non_soddisfatto_da_B") # genera l'array rnames con i nomi delle
# righe
cnames <- c("Soddisfatto_da_A", "Non_soddisfatto_da_A") # genera l'array cnames con i nomi delle
# colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
mydata # matrice dati
t(mydata) # matrice trasposta
#
windows() # apre e inizializza una nuova finestra grafica
barplot(t(mydata),beside=TRUE, legend=TRUE, ylim=c(0,200), col=c("darkblue","red"), ylab="Frequenze
osservate nel campione", xlab="Soddisfazione in seguito al trattamento B", main="Soddisfazione in
seguito al trattamento in 250 soggetti") # i risultati sono arricchiti con un grafico a barre
#
windows() # apre e inizializza una nuova finestra grafica
barplot(t(mydata),beside=FALSE,legend=TRUE, ylim=c(0,200), col=c("darkblue","red"), ylab="Frequenze
osservate nel campione", xlab="Soddisfazione in seguito al trattamento B", main="Soddisfazione in
seguito al trattamento in 250 soggetti") # i risultati sono arricchiti con un grafico a barre
#
```

Qui non compare nulla di rilevante rispetto allo script precedente, se non l'argomento **beside** che nel primo grafico (**Fig. 5.4**) viene posto **beside=TRUE** generando un garfico a barre affiancate mentre in questo secondo grafico (**Fig. 5.5**) viene posto **beside=FALSE** generando così ora un grafico a barre sovrapposte.

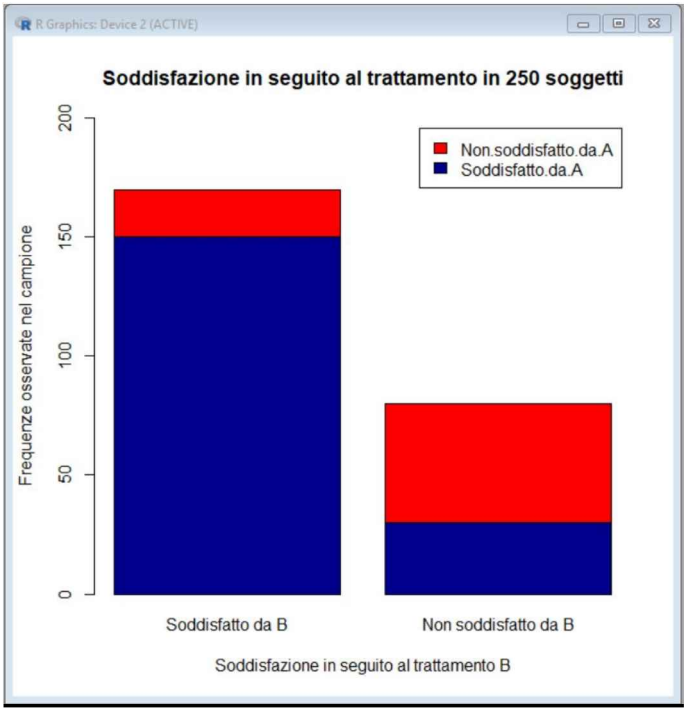



**Fig. 5.5.** I risultati del test di McNemar illustrati con un grafico a barre sovrapposte.

Utilizziamo infine i dati dell'esempio tratto da Marubini e relativo all'efficacia di cinque differenti vaccini influenzali già visti per il test chi-quadrato. Questi erano i risulti espressi come percentuale:

| *Esito* | *Vaccino_1* | *Vaccino_2* | *Vaccino_3* | *Vaccino_4* | *Vaccino_5* |
|---|---|---|---|---|---|
| Influenza_si | 18.1 | 21.0 | 9.1 | 18.3 | 20.0 |
| Influenza_no | 81.9 | 79.0 | 90.8 | 81.7 | 80.0 |

Nello script viene aperta una nuova finestra grafica (**windows()**) quindi per realizzare il grafico viene impiegata di nuovo la funzione **barplot()** con queste variazioni:
→ per i dati viene impiegata la matrice **mydata**;
→ l'argomento **beside=FALSE** specifica che le barre sono sovrapposte in pila;
→ con l'argomento **legend=TRUE** viene generata una legenda;
→ con l'argomento **col=c("yellow", "green")** viene fornito l'array contenente i colori da applicare;
→ con gli argomenti **density = 40, angle = 45** sono specificati la densità e l'angolo delle linee colorate che andranno a riempire le barre.

Questo sono lo script e la figura risultante (**Fig. 5.6**):

```
# GRAFICO A BARRE SOVRAPPOSTE
#
cells <- c(18.1,21,9.1,18.3,20,81.9,79,90.9,81.7,80) # genera l'array cells con le percentuali di casi
rnames <- c("Influenza_si", "Influenza_no") # genera l'array rnames con i nomi delle righe
cnames <- c("Vaccino_1", "Vaccino_2", "Vaccino_3", "Vaccino_4", "Vaccino_5") # genera l'array cnames
# con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=5, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la
# matrice dati
#
windows() # apre e inizializza una nuova finestra grafica
barplot(mydata, beside=FALSE, legend=TRUE, ylim=c(0,150), col=c("red", "green"), density = 40, angle =
45, ylab="Frequenze osservate", xlab="Vaccino infuenzale", main="Valutazione dell'efficacia di 5 vaccini
influenzali") # traccia il grafico a barre verticali
#
```

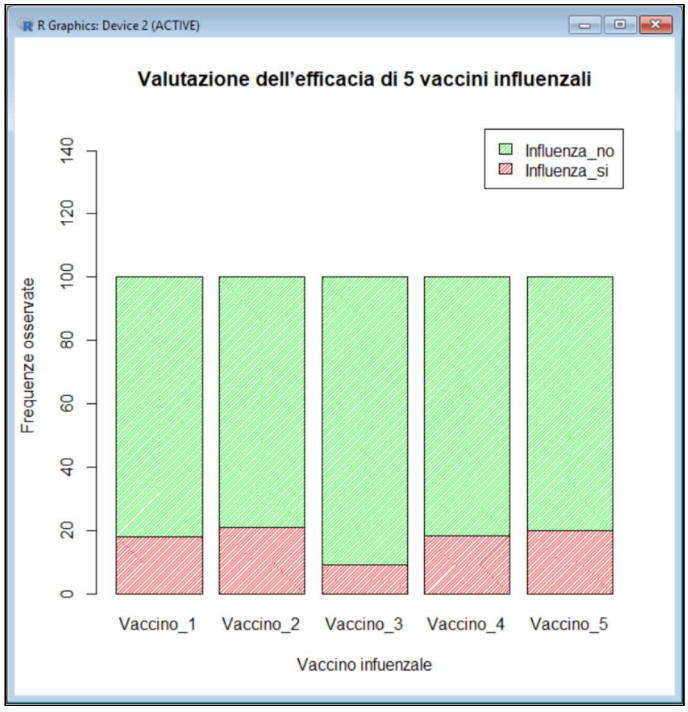



**Fig. 5.6** Il grafico a barre generato con la funzione **barplot()** con l'argomento **beside=FALSE**.

Finora le barre le abbiamo generate specificando per ciascuna di esse il numero dei casi. Ma per fortuna è possibile che **R** li conti per noi se impieghiamo la funzione **table()**.

In questo **script** generiamo due grafici a barre, il primo che presenta il numero di casi per sport praticato (grafico che non viene rappresentato) e il secondo che presenta il numero di casi per sport praticato suddividendo ciascun risultato per sesso (**Fig. 5.7**).

```
# GRAFICI A BARRE AFFIANCATE CONTANDO I CASI CON R
#
library(DAAG) # carica il pacchetto e il set di dati ais
#
# grafico a barre per sport praticato
#
tab <- table(ais$sport) # tabula i dati per sport
windows() # apre e inizializza una nuova finestra grafica
barplot(t(tab), beside=TRUE, legend=TRUE, xlab= "Sport praticato", ylab = "Numero degli atleti", main =
"Numero degli atleti per sport praticato", col = rainbow(length(tab))) # grafico a barre
#
# grafico a barre per sport praticato e per sesso
#
tab <- table(ais$sport, ais$sex) # tabula i dati per sport e per sesso
windows() # apre e inizializza una nuova finestra grafica
barplot(t(tab), beside=TRUE, legend=TRUE, xlab= "Sport praticato", ylab = "Numero degli atleti", main =
"Numero degli atleti per sport praticato e per sesso", col = c("green", "yellow")) # grafico a barre
#
```

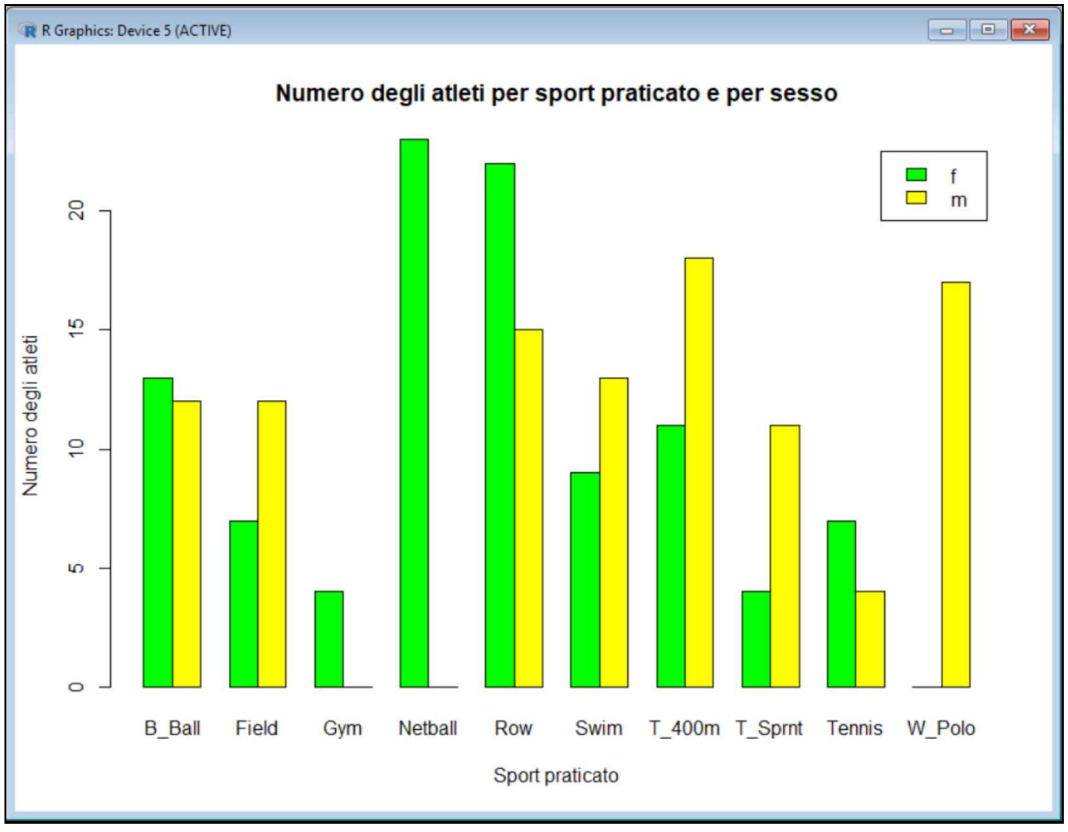



**Fig. 5.7** Grafico a barre del numero degli atleti per sport e per sesso

Se vi infastidisce il fatto che date le impostazioni di default le barre escono dai limiti dell'asse delle `y` potete ridefinirne i limiti aggiungendo come argomento un **ylim = c(0, 25)** e rieseguendo l'ultima parte dello script:

```
windows() # apre e inizializza una nuova finestra grafica
barplot(t(tab), beside=TRUE, legend=TRUE, ylim = c(0, 25), xlab= "Sport praticato", ylab = "Numero degli
atleti", main = "Numero degli atleti per sport praticato e per sesso", col = c("green", "yellow")) #
# grafico a barre
#
```

---

## 5.3. Istogrammi e kernel density plot

L'istogramma è lo strumento grafico fondamentale per la rappresentazione di una distribuzione univariata, o, più semplicemente, per la rappresentazione della distribuzione di una singola variabile. Ma non è il solo disponibile in **R**.

La stima di densità kernel è in motodo non parametrico di stima della densità della distribuzione di una variabile. I kernel density plot, cioè i grafici basati sulla stima kernel di densità, possono essere utilizzati in alternativa al tradizionale istogramma del quale rappresentano in qualche modo l'evoluzione. Invece di raccogliere le osservazioni in barre come negli istogrammi, lo stimatore di densità kernel colloca piccole "gobbe" (bump), determinate da un fattore **K** (kernel) e da un parametro **h** di smussamento (smoothing) detto ampiezza di banda (bandwidth), in corrispondenza di ogni osservazione. Lo stimatore somma quindi i bump risultanti generando una curva smussata. Una maggiore densità di punti comporta valori di più alti della stima.

Vediamo con questo script come rappresentare in **R** sia gli istogrammi sia i kernel density plot di una variabile numerica:

```
# ISTOGRAMMI E KERNEL DENSITY PLOT
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
hist(ais$rcc, xlim=c(3,7), main = "Istogramma semplice", xlab = "Eritrociti in 10^12/L", ylab =
"Frequenza") # traccia un istogramma semplice dei dati
#
hist(ais$rcc, xlim=c(3,7), breaks=30, col="red", main = "Istogramma colorato", xlab = "Eritrociti in
10^12/L", ylab = "Frequenza") # traccia un istogramma colorato
#
isto <-hist(ais$rcc, xlim=c(3,7), breaks=30, col="red", main = "Istogramma con curva gaussiana", xlab =
"Eritrociti in 10^12/L", ylab = "Frequenza") # traccia un istogramma colorato
xfit <-seq(min(ais$rcc),max(ais$rcc),length=40) # ricostruisce...
y <-dnorm(xfit,mean=mean(ais$rcc),sd=sd(ais$rcc)) # la corrispondente...
yfit <- y*diff(isto$mids[1:2])*length(ais$rcc) # curva gaussiana
lines(xfit, yfit, col="blue", lwd=2) # sovrappone all'istogramma la curva gaussiana
#
plot(density(ais$rcc), xlim=c(3,7), main = "Kernel density plot", xlab="Eritrociti in 10^12/L", ylab = "Stima
kernel di densità") # traccia il kernel density plot
polygon(density(ais$rcc), col="red", border="blue") # colora il kernel density plot
#
```

Viene impiegato nuovamente il set di dati **ais**, e sono tracciati tre differenti istogrammi e il kernel density plot per la variabile **rcc** che corrisponde alla concentrazione dei globuli rossi del sangue (eritrociti) espressa in migliaia di miliardi di globuli rossi per litro di sangue ($10^{12}$/L ) in che è numericamente identico ad esprimerla in milioni di globuli rossi per milionesimo di litro di sangue ($10^{6}$/µL) come talora viene riportata.

Per favorire il confronto tra le diverse rappresentazioni impiegate la finestra grafica aperta con la funzione **windows()** viene divisa in quattro quadranti (**par(mfrow=c(2,2))**), uno per grafico, che verranno riempiti nell'ordine da sinistra verso destra e dall'alto verso il basso.

Il primo dei quattro istogrammi (**Fig. 5.8**) è il più semplice che si possa fare, specifica come argomenti solamente i dati dei quali tracciare l'istogramma (**ais$rcc**), limite inferiore e superiore dell'asse delle x (**xlim=c(3,7)**), e questo solo per rendere la rappresentazione omogenea con gli altri istogrammi, più il titolo (**main = ""**) e le etichette dell'asse delle x (**xlab = ""**) e dell'asse delle y (**ylab = ""**).

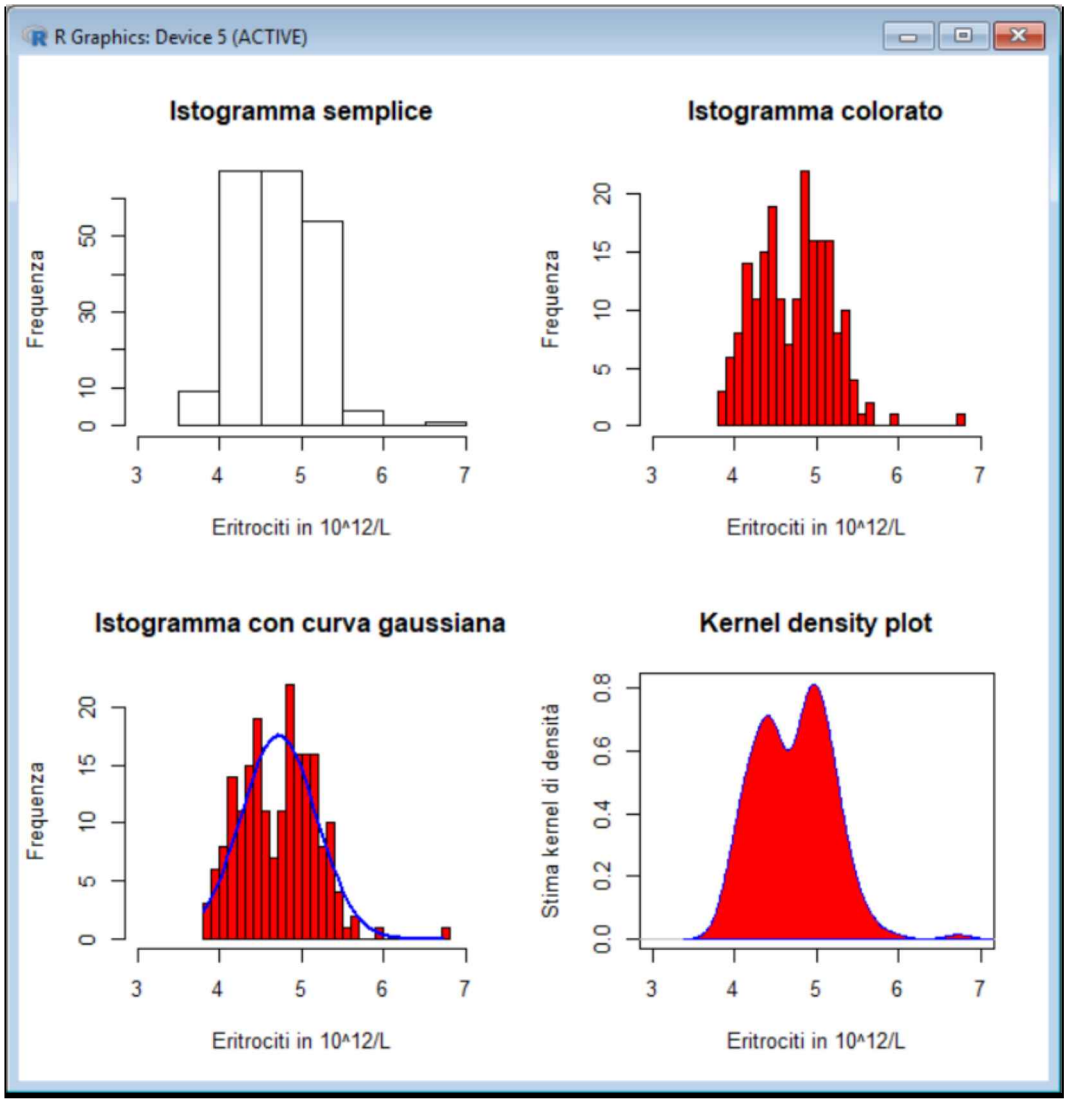



**Fig. 5.8** Istogrammi e kernel density plot della concentrazione degli eritrociti nel sangue.

Il secondo dei quattro istogrammi prevede in aggiunta solamente l'argomento che specifica il numero di classi (**breaks=30**), ovviamente adattabile secondo le esigenze, e l'argomento che specifica il colore dell'istogramma (**breaks=30, col="red"**).

Il terzo istogramma è identico al secondo, a parte il fatto che ad esso viene sovrapposta la curva gaussiana

teorica avente come media (**mean=mean(ais$rcc)**) e come deviazione standard (**sd=sd(ais$rcc)**) quelle della variabile in questione (**rcc**).

Il quarto grafico è il kernel density plot. Per tracciarlo viene impiegata la funzione **plot()** e l'argomento principale, i dati da rappresentare, non è più costituito come nel caso degli istogrammi dalla variabile **ais$rcc** bensì dalla sua densità kernel **density(ais$rcc)**. La successiva funzione **polygon()** colora il kernel density plot di colore rosso e ne traccia il profilo in colore blu.

L'istogramma e il kernel density plot sono un ottimo esempio di come la rappresentazione grafica, oltre che essere lo strumento principe per il riepilogo dei risultati, sia un importante strumento per l'analisi esplorativa dei dati.

Ora supponete di esservi accontentati, per l'analisi esplorativa dei valori degli eritrociti, di queste quattro righe di codice

```
library(nortest) # carica il pacchetto
mean(ais$rcc) # calcola la media degli eritrociti
sd(ais$rcc) # calcola la deviazione standard degli eritrociti
lillie.test(ais$rcc)  # esegue il test di normalità di Lilliefors (Kormogoorov-Smirnov)
```

e del primo degli istogrammi riportati sopra, l'istogramma semplice riportato in alto a sinistra.

Il risultato quattro righe di codice è il seguente:

```
> mean(ais$rcc) # calcola la media degli eritrociti
[1] 4.718614
> sd(ais$rcc) # calcola la deviazione standard degli eritrociti
[1] 0.4579764
> lillie.test(ais$rcc)  # esegue il test di normalità di Lilliefors
(Kormogoorov-Smirnov)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  ais$rcc
D = 0.071669, p-value = 0.01344
```

Il test di Lilliefors indica uno scostamento dalla normalità se se consideriamo come limite di significatività il tradizionale (ma anche abusato) valore del 5% (`p < 0.05`), ma se avessimo deciso di non accontentarci di questo, e considerare significativo un valore <u>almeno</u> inferiore all'1%, quindi un `p < 0.01`, avremmo potuto anche concludere che i valori di eritrociti sono distribuiti normalmente. Fatto avvalorato dall'istogramma in alto a sinistra che mostra una distribuzione appena asimmetrica sulla destra, che potrebbe giustificare appunto il `p-value = 0.01344` quindi compreso tra l'1% e il 5% del test di Lilliefors .

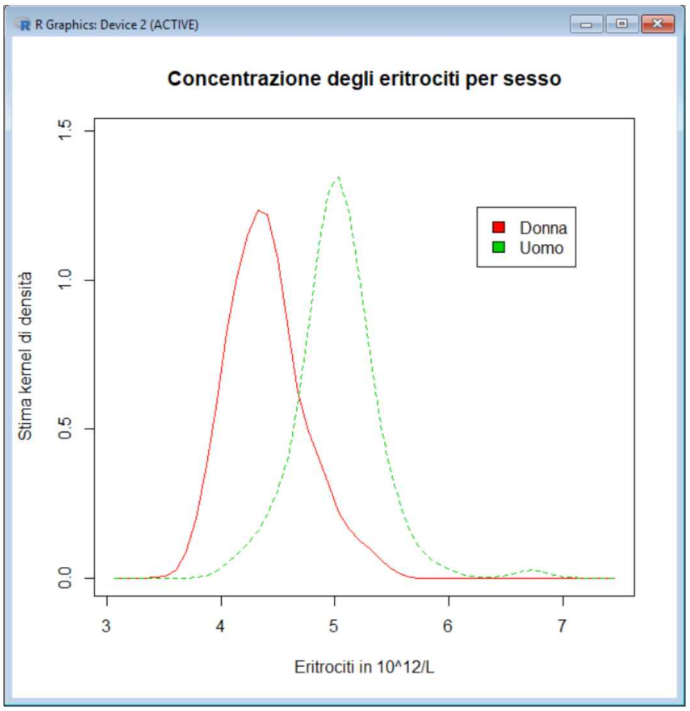
Ma approfondendo l'analisi grafica, aumentando il numero delle classi negli istogrammi e tracciando il kernel density plot, emerge una distribuzione chiaramente bimodale: i valori degli eritrociti potrebbero essere il risultato del sovrapporsi, all'interno dei dati rilevati sugli atleti australiani, di due popolazione differenti, con differente media.

In effetti è noto che la concentrazione degli eritrociti nel sangue nella donna è mediamente inferiore a quella presente nell'uomo. Possiamo allora dotarci di uno script che genera due kernel density plot separati per il sesso/fattore `m` e per il sesso/fattore `f` e vedere se si sovrappongono o no. La **Fig. 5.9** è generata

da questo nuovo script che prevede la libreria **sm**:

```
# KERNEL DENSITY PLOT SOVRAPPOSTI                                    1/2
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
#
# al termine fare click con il tasto sinistro del mouse nel punto in cui si desidera la legenda
library(sm) # carica il pacchetto
attach(ais) # funzione per impiegare direttamente i nomi delle variabili di ais
windows() # apre e inizializza una nuova finestra grafica
sesso.f <- factor(sex, levels= c("f","m"), labels = c("Donna", "Uomo")) # identifica i casi Uomo e Donna
sm.density.compare(rcc, sex, xlab="Eritrociti in 10^12/L", ylab="Stima kernel di densità") # traccia il
# kernel density plot
title(main="Concentrazione degli eritrociti per sesso") # aggiunge il titolo
colfill<-c(2:(2+length(levels(sesso.f)))) # prepara la legenda
legend(locator(1), levels(sesso.f), fill=colfill) # posiziona la legenda
#
```

**Fig. 5.9** Kernel density separati per sesso mostrano per la distribuzione degli eritrociti la presenza di due sottopopolazioni e confermano la distribuzione bimodale della variabile.




Il codice fa una cosa piuttosto interessante: traccia due kernel density plot indipendenti e sovrapposti per i soggetti di sesso maschile (m) e di sesso femminile (f) e rimane in attesa. A questo punto, posizionate il mouse dove volete che compaia la legenda, e fate click con il tasto sinistro del mouse per farla comparire.

Il codice riportato non consente di spostare la legenda. Se non si è soddisfatti della sua posizione, è necessario rieseguire l'intero script e fare nuovamente click con il tasto sinistro del mouse nel punto in cui si vuole posizionare la legenda.

I kernel density separati per sesso della **Fig. 5.9** mostrano per la concentrazione degli eritrociti nel siero la presenza di due sottopopolazioni che giustificano la distribuzione bimodale della variabile osservata negli istogrammi e nel kernel density plot della **Fig. 5.8**.

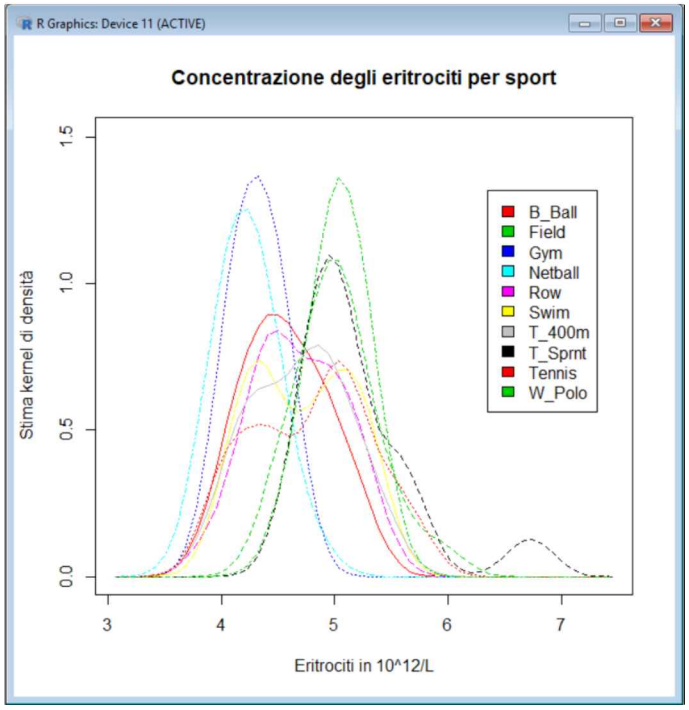



**Fig. 5.10** Kernel density plot separati per sport confermano l'eterogeneità dei dati ematologici del campione di atleti.

Possiamo effettuare un altro approfondimento grafico sugli eritrociti mediante kernel density plot per ciascuno degli sport (**Fig.5.10**):

```
windows() # apre e inizializza una nuova finestra grafica                    2/2
# al termine fare click con il tasto sinistro del mouse nel punto in cui si desidera la legenda
sprt <- c("B_Ball", "Field", "Gym", "Netball", "Row", "Swim", "T_400m", "T_Sprnt", "Tennis", "W_Polo")
sprt.f <- factor(sport, levels= sprt, labels = sprt) # identifica gli sport
sm.density.compare(rcc, sport, xlab="Eritrociti in 10^12/L", ylab="Stima kernel di densità") # traccia i
# kernel density plot
title(main="Concentrazione degli eritrociti per sport") # aggiunge il titolo
colfill<-c(2:(2+length(levels(sprt.f)))) # prepara la legenda
legend(locator(1), levels(sprt.f), fill=colfill) # posiziona la legenda
#
```

Di nuovo posizionate il mouse dove volete che compaia la legenda, e fate click con il tasto sinistro del mouse per farla comparire.

## 5.4. Grafici a scatola con i baffi

I **grafici a scatola con i baffi** o **box-and-whiskers plot**, denominazione in genere abbreviata in **boxplot**, sono generati in **R** mediante la funzione **boxplot()** che impiega i seguenti criteri di rappresentazione[119]:
→ le scatole (**box**) includono il 50% delle osservazioni;
→ il bordo inferiore delle scatole corrisponde al 25° percentile (o primo quartile / $Q_1$);
→ la linea interna alle scatole corrisponde alla mediana ovvero al 50° percentile (o secondo quartile / $Q_2$);
→ il bordo superiore delle scatole corrisponde al 75° percentile (o terzo quartile / $Q_3$);
→ i baffi (**whiskers**) corrispondono al valore minimo (baffo inferiore) e al valore massimo (baffo superiore) osservati;
→ la differenza interquartile[120] viene definita come IQR = $Q_3 - Q_1$ ovvero come differenza tra il valore corrispondente al terzo quartile ($Q_3$) e il valore corrispondente al primi quartile ($Q_1$);
→ i valori inferiori a $Q_2 - 1.5 \cdot IQR$ e i valori superiori a $Q_2 + 1.5 \cdot IQR$ sono considerati outliers;
→ i dati anomali o dati aberranti (outliers), ovvero dati singoli che si discostano eccessivamente dal comportamento degli altri dati, sono riportati come punti singoli separati.

I boxplot forniscono un'analisi non parametrica dei dati complementare a quella numerica. La applichiamo ora all'analisi dei valori di concentrazione degli eritrociti nel sangue per suddividerli in base ai fattori inclusi nel set di dati **ais**, e cioè al sesso dell'atleta e/o allo sport praticato.

Iniziamo con queste quattro righe di codice:

```
# GRAFICI A SCATOLA CON I BAFFI                                    1/4
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
attach(ais) # funzione per impiegare direttamente i nomi delle variabili di ais
#
windows() # apre e inizializza una nuova finestra grafica
boxplot(rcc~sport, horizontal=FALSE, data=ais, main="Eritrociti per sport praticato", xlab="Sport
praticato", ylab="Eritrociti in 10^12/L", notch=FALSE, col="yellow") # eritrociti per ciascuno sport
# praticato
#
```

Le prime tre righe si limitano a caricare il pacchetto che contiene il set di dati (**library(DAAG)**), a mostrarne la struttura (**str(ais)**) e ad aprire una finestra grafica (**windows()**).

Per tracciare il boxplot (**Fig. 5.11**) viene impiegata una sola riga di codice, la quarta. Il primo argomento **rcc~sport** indica che la rappresentazione degli eritrociti sotto forma di boxplot deve essere effettuata aggregando ( ) i valori della variabile eritrociti (**rcc**) nei sottoinsiemi corrispondenti ai valori (B_Ball, Field, Gym, Netball, Row, Swim, T_400m, T_Sprnt, Tennis, W_Polo) assunti dal fattore variabile sport (**sport**). Per tracciare boxplot orizzontali (grafico non riportato) è sufficiente aggiungere l'argomento **horizontal=TRUE** (e ovviamente scambiare tra di loro le etichette degli assi **xlab** e **ylab**).

Se si vanno a riesaminare i dati originali[121] si nota che negli sport Gym e Netball sono incluse solamente

[119] Digitate **help(boxplot)** nella `Console di R` per la documentazione della funzione **boxplot()**.
[120] O differenza o scarto o ampiezza o, in inglese, range, da cui IQR (Inter Quartile Range).
[121] Vedere: **A3. Il set di dati ais**.

donne. Quindi si procede a riesaminare i dati aggregando i valori per sesso (**rcc~sex**):

```
windows() # apre e inizializza una nuova finestra grafica                    2/4
boxplot(rcc~sex, data=ais, main="Eritrociti per sesso", xlab="Sesso", ylab="Eritrociti in 10^12/L",
notch=TRUE, col="green") # eritrociti per sesso
#
```

In questo caso, a parte il colore dei box che diventa **col="green"** la novità è che i boxplot della concentrazione degli eritrociti per sesso sono tracciati con una incisura (**notch=TRUE**) che rappresenta i limiti di confidenza al 95% della mediana (**Fig. 5.12**).

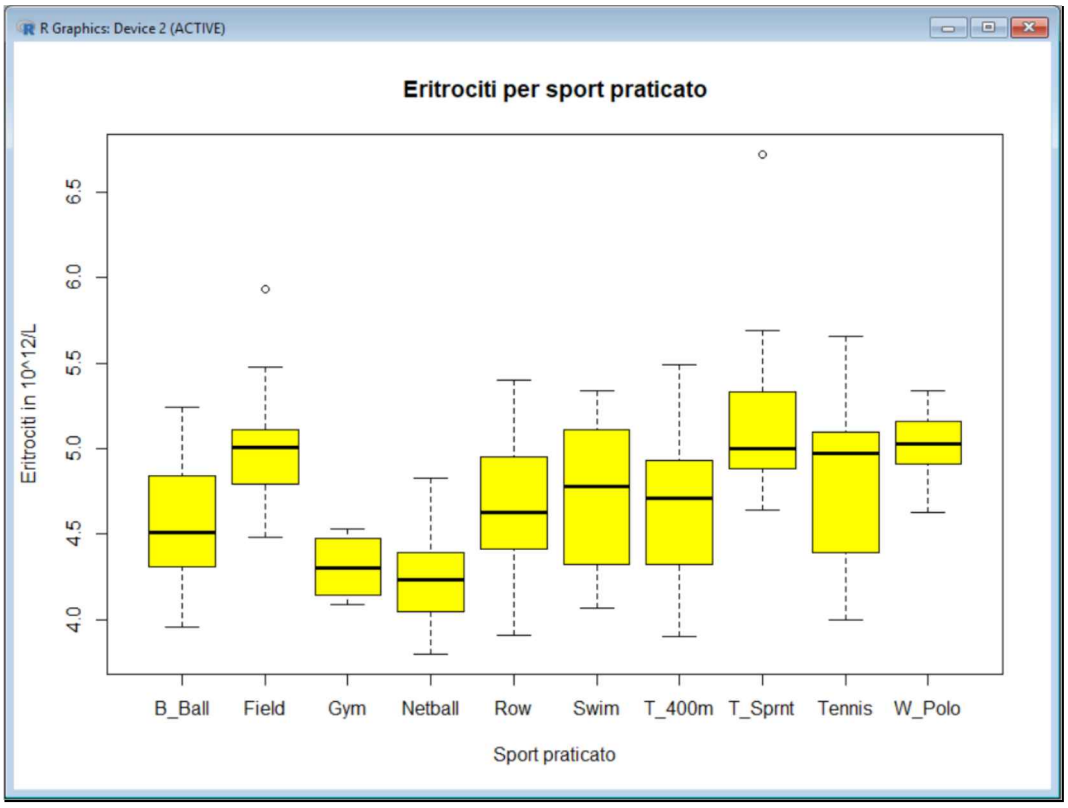



**Fig. 5.11** Boxplot della concentrazione degli eritrociti nel sangue nei vari sport praticati.

Questo corrisponde ad un test per la significatività della differenza tra le mediane. Se le incisure di due boxplot non si sovrappongono, come in questo caso, la conclusione è che le mediane delle due distribuzioni sono significativamente diverse.

Quindi il fatto interessante è che una rappresentazione grafica può essere impiegata non solo per effettuare un'analisi esplorativa dei dati, ma addirittura per effettuare un test statistico (un confronto tra mediane). E il test ci conferma che la mediana della concentrazione degli eritrociti nelle atlete è inferiore a quella dagli atleti (è all'incirca di 4.4 · 10^12/L contro 5.0 · 10^12/L) e che la differenza tra le mediane, all'incirca di 0.6 · 10^12/L, è statisticamente significativa.

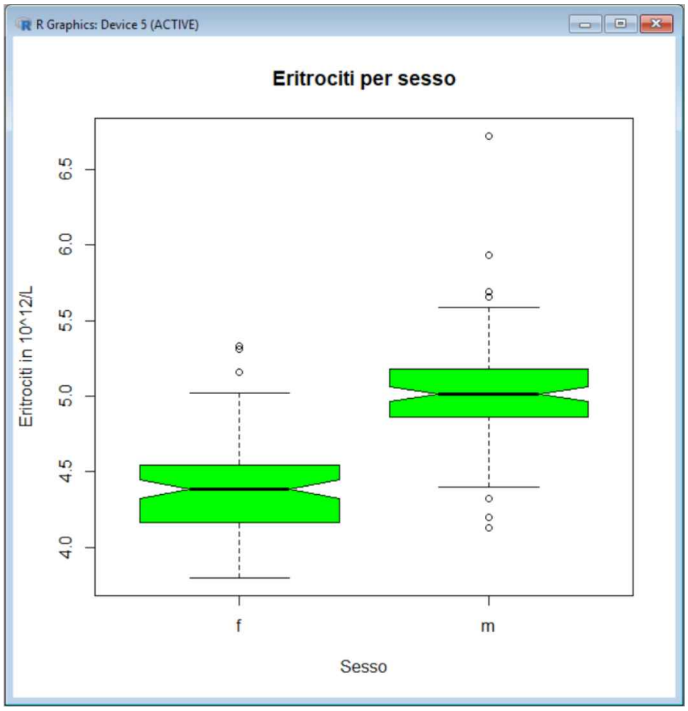



**Fig. 5.12** Boxplot della concentrazione degli eritrociti nel sangue distinta per sesso.

Quando nella funzione **boxplot()** si mette l'argomento **notch=TRUE** potrebbe comparire nella `Console di R` un messaggio che avverte che in alcuni casi le incisure sono uscite dai bordi della scatola e che suggerisce di valutare l'opportunità di sostituire l'argomento con **notch=FALSE**.

Questo vi accadrà ad esempio se eseguite il seguente (e ultimo) codice, che applica l'argomento **notch=TRUE** ai boxplot differenziati per sport:

```
windows() # apre e inizializza una nuova finestra grafica                    3/4
boxplot(rcc~sport, data=ais, main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in 10^12/L", notch=TRUE, col="green") # eritrociti per ciascuno sport praticato
#
```

In questo caso il grafico (**Fig. 5.13**) evidenzierà la comparsa del problema nel caso degli sport Field, Gym, T_sprnt e Tennis e viene riportato questo messaggio:

```
> boxplot(rcc~sport, data=ais, main="Eritrociti per sport praticato",
xlab="Sport praticato", ylab="Eritrociti in 10^12/L", notch=TRUE,
col="green") # eritrociti per ciascuno sport praticato
Warning message:
In bxp(list(stats = c(3.96, 4.31, 4.51, 4.84, 5.24, 4.48, 4.79,  :
  some notches went outside hinges ('box'): maybe set notch=FALSE
```

Il problema è determinato dal fatto che il numero delle osservazioni è troppo ridotto, e questo comporta nelle conclusioni un livello di incertezza che si estende al di la delle osservazioni reali.

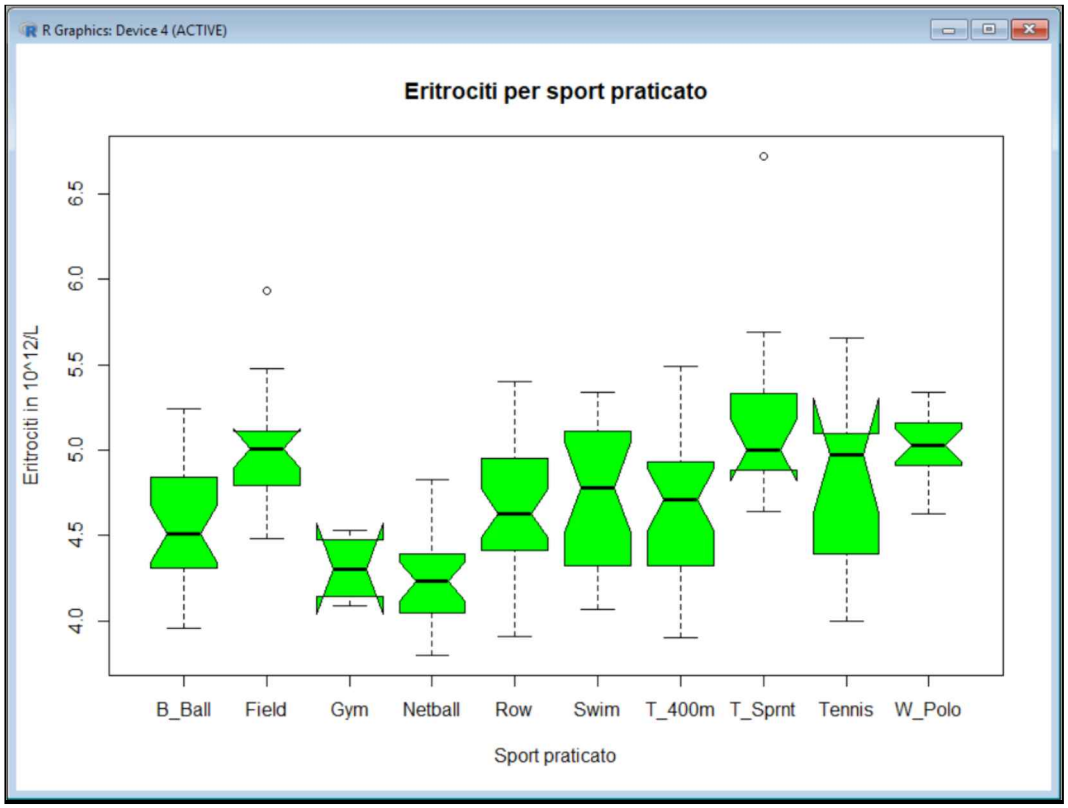



**Fig. 5.13** Boxplot della concentrazione degli eritrociti nel sangue distinta per sport con l'argomento **notch=TRUE** determinano una segnalazione di problemi per alcuni di essi.

In questi casi vi sono solamente due modi per superare il problema: o rinunciare a trarre delle conclusioni, mettendo l'argomento **notch = FALSE**, come del resto consigliato nella nota che appare in questi casi, e quindi rinunciare ad effettuare il test di significatività, o aumentare adeguatamente, sempre che sia possibile farlo, il numero delle osservazioni. Ora tracciamo i boxplot per sport e per sesso:

```
windows() # apre e inizializza una nuova finestra grafica                                    4/4
sesso.f <- factor(sex, levels= c("f","m"), labels = c("Donna", "Uomo")) # identifica i casi Uomo e Donna
boxplot(rcc~sex+sport, horizontal=FALSE, boxwex = 0.4, cex.axis = 0.8, las = 2, data=ais, main="Eritrociti
per sport e per sesso", xlab="", ylab="Eritrociti in 10^12/L", notch=FALSE, col=c("green", "yellow"),
legend=sesso.f) # eritrociti per ciascuno sport praticato
colfill<-c("green", "yellow") # prepara la legenda
legend(locator(1), levels(sesso.f), fill=colfill) # posiziona la legenda
#
```

I risultati sono riportati nella **Fig. 5.14**. Qui viene ripreso il codice che consente di posizionale la legenda: posizionate il mouse dove volete che compaia la legenda, e fate click con il tasto sinistro del mouse per farla comparire. Si rammenta che il codice non consente di muoverla. Se non si è soddisfatti della sua posizione, è necessario rieseguire l'intero script e fare nuovamente click con il tasto sinistro del mouse nel punto in cui si vuole posizionare la legenda.

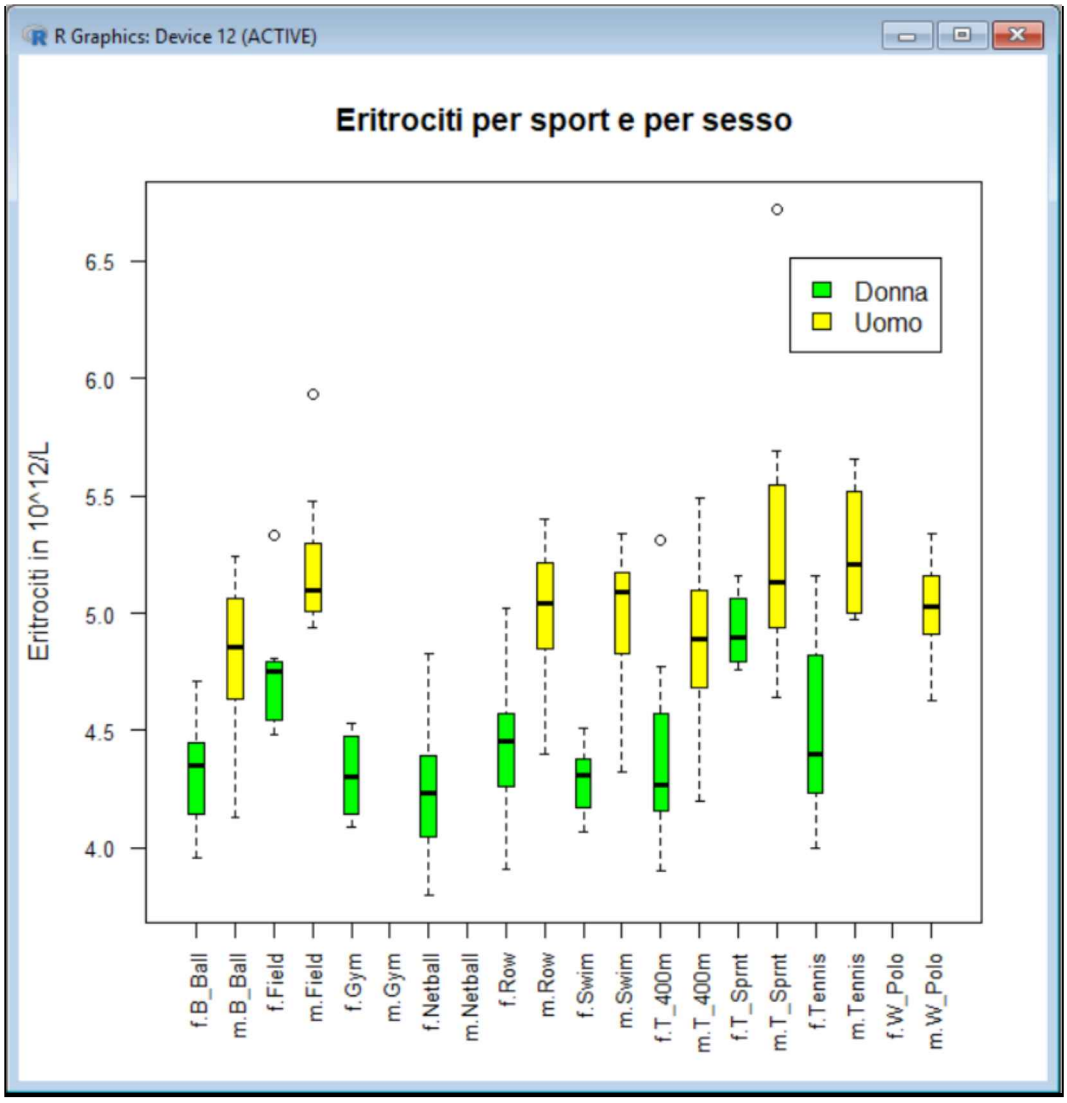



**Fig. 5.14** Boxplot della concentrazione degli eritrociti nel sangue per sport e per sesso.

Da notare che in questo **script** compaiono alcuni nuovi argomenti:
→ l'argomento **boxwex = 0.4** che gestisce la larghezza dei boxplot;
→ l'argomento **cex.axis = 0.8** che gestisce la dimensione dei caratteri;
→ l'argomento **las = 2** che ruota verticalmente le etichette.

## 5.5. Correlogrammi

Nonostante le critiche che gli possono essere (e giustamente) rivolte a cause sia dell'abuso nell'impiego che ne è stato fatto sia a causa della soggiacente implicita e subdola confusione tra correlazione e causazione legata la suo impiego, la correlazione lineare può avere un senso. Per questo in **R**, oltre al calcolo dei coefficienti di correlazione (parametrici e non parametrici) che abbiamo visto nella parte statistica, sono disponibili grafici di correlazione detti **correlogrammi**. Richiedono uno specifico pacchetto denominato **corrgram** che deve essere preventivamente scaricato dal **CRAN**.

In questo script vediamo la funzione **corrgram()** con la quale rappresenteremo tre diversi correlogrammi che forniscono informazioni sintetiche e diversamente utili sulla relazione tra le variabili contenuto nel set di dati **ais**. Questa è la prima parte dello script, con la quale viene generata la rappresentazione grafica illustrata nella **Fig. 5.15**:

```
# CORRELOGRAMMI                                                    1/3
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
library(corrgram)   # carica il pacchetto che genera i correlogrammi
#
windows() # apre e inizializza una nuova finestra grafica
corrgram(ais, cor.method = "pearson", order=FALSE, lower.panel=panel.shade, upper.panel=panel.pie,
text.panel=panel.txt, main="Correlogramma semplice") # correlogramma semplice
#
```

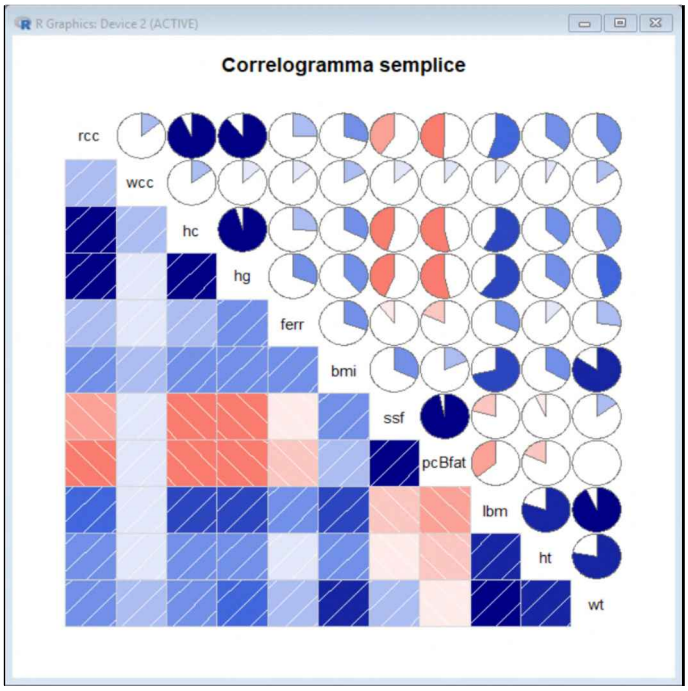



**Fig. 5.15** Correlogramma tra le variabili del set di dati **ais**, il valore **r** di Pearson viene rappresentato sotto forma di quantità di colore nelle torte (metà superiore destra) e di intensità di colore dei quadrati (metà inferiore sinistra). In blu le correlazioni positive in colore rosato quelle negative.

Nella prima parte dello script dopo avere caricato il pacchetto **DAAG** (**library(DAAG)**) viene mostrata la struttura del set di dati **ais** (**str(ais)**), quindi viene caricata il pacchetto necessario per rappresentare i correlogrammi (**library(corrgram)**) e viene aperta una nuova finestra grafica (**windows()**).

Nella funzione **corrgram()** compare come primo argomento il set di dati da analizzare (**ais**), come secondo argomento l'ordine nel quale presentare i risultati (**order=FALSE**), che lascia le variabili nello stesso ordine nel quale sono presenti nel set di dati originali. In alternativa l'ordine in cui presentare i risultati può essere espresso come **order="PCA"**, **order="OLO"**, **order="GW"**, **order="HC"**[122].

Per il metodo di correlazione viene impiegato l'argomento **cor.method = "pearson"**, corrispondente al classico **r**, ma possono essere in alternativa impiegati l'argomento **cor.method = "spearman"** o l'argomento **cor.method = "kendall"** che riportano i risultati del coefficiente **rho** di Spearman e del coefficiente **tau** di Kendall.

Il valore **r** di Pearson nella **Fig. 5.15** viene rappresentato sotto forma di quantità di colore nelle torte nella metà superiore destra del correlogramma, nelle quali una maggior quantità di colore corrisponde a un più elevato valore di **r**, e di intensità del colore dei quadrati nella metà inferiore sinistra. In blu sono riportate le correlazioni positive in colore rosato quelle negative. Questo tipo di rappresentazione dei correlogrammi è realizzato mettendo nella funzione **corrgram()** l'argomento **lower.panel=panel.shade** e l'argomento **upper.panel=panel.pie**.

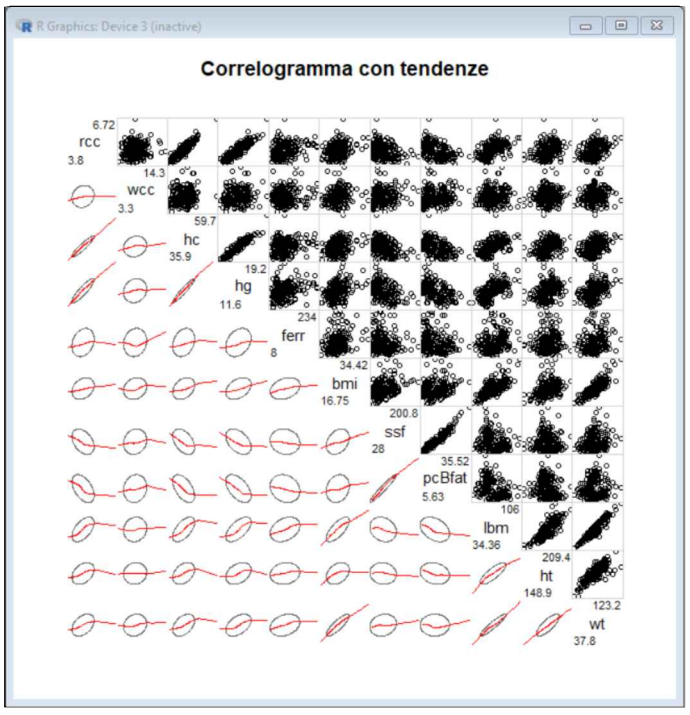



**Fig. 5.16** Correlogramma per le variabili del set di dati **ais**, in alto a destra le distribuzioni dei dati delle coppie di variabili, in basso a sinistra gli elissoidi che indicano la tendenza delle variabili a variare congiuntamente, nelle diagonale i nomi delle variabili con il valore minimo e il valore massimo osservati.

[122] Getting Things in Order: An Introduction to the R Package seriation. URL consultato il 27/10/2018: https://goo.gl/2LC8Wk

Il correlogramma della **Fig. 5.16** è realizzato con la seconda parte dello script mettendo nella funzione **corrgram()** l'argomento **lower.panel=panel.ellipse** e l'argomento **upper.panel=panel.pts**.

```
windows() # apre e inizializza una nuova finestra grafica                    2/3
corrgram(ais, cor.method = "pearson", order=FALSE, lower.panel=panel.ellipse, upper.panel=panel.pts,
text.panel=panel.txt, diag.panel=panel.minmax, main="Correlogramma con tendenze") #
# correlogramma con evidenza delle tendenze
#
```

In alto a destra è riportato per ciascuna delle possibili coppie di variabili il grafico xy con la distribuzione dei valori osservati. In basso a sinistra sono mostrati gli elissoidi che indicano la tendenza delle variabili a variare congiuntamente con in rosso la stima della curva che ne approssima la tendenza, elissoidi sempre più stretti e più allungati e curva sempre meno curva man mano che la relazione tra le variabili si avvicina a una retta. Nella diagonale compaiono infine i nomi delle variabili con il valore minimo e il valore massimo osservati.

Il correlogramma della **Fig. 5.17** è realizzato con questa terza parte dello script mettendo nella funzione **corrgram()** l'argomento **lower.panel=panel.pts**, l'argomento **upper.panel=panel.conf** e l'argomento **diag.panel=panel.density**:

```
windows() # apre e inizializza una nuova finestra grafica                    3/3
corrgram(ais, cor.method = "pearson", order=FALSE, lower.panel=panel.pts, upper.panel=panel.conf,
diag.panel=panel.density, main="Correlogramma con i coefficienti di correlazione") # correlogramma
# con i coefficienti di correlazione e i loro limiti di confidenza
#
```

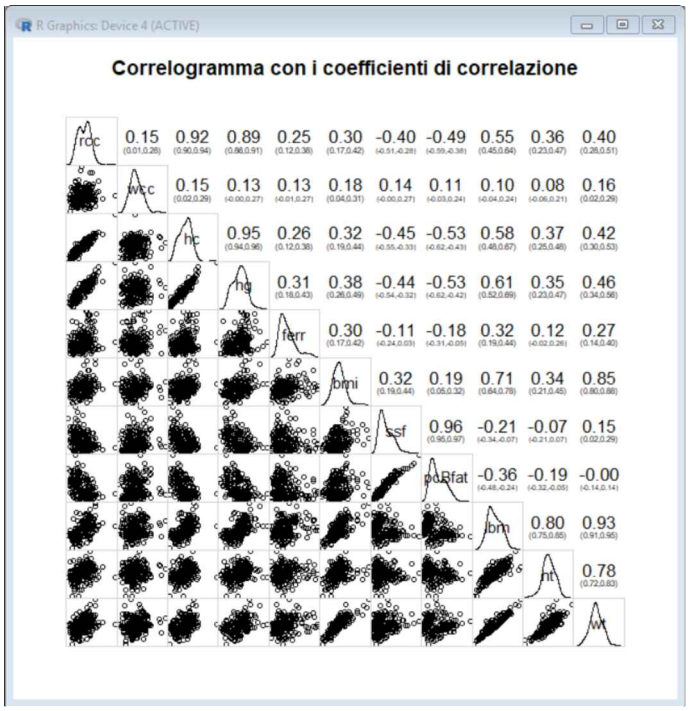



**Fig. 5.17** Correlogramma per le variabili del set di dati **ais**, in alto a destra i valori di **r** con i loro limiti di confidenza, in basso a sinistra le distribuzioni dei dati delle coppie di variabili, nelle diagonale i kernel density plot di ciascuna variabile.

In alto a destra è riportato per ciascuna delle possibili coppie di variabili i coefficiente di correlazione **r** di

Pearson con i sui limiti di confidenza al 95%. In basso sinistra è riportato per ciascuna delle possibili coppie di variabili il grafico xy con la distribuzione dei valori osservati. Nella diagonale sono riportati infine i kernel density plot di ciascuna variabile.

**Nota bene**: gli eventuali warnings che compaiono eseguendo questa terza parte dello script possono essere ignorati.

---

## 5.6. Grafici di dispersione

I grafici di dispersione (o scatterplot, scatter graph, scatter chart, scattergram, scatter diagram) o grafici xy o grafici cartesiani o diagrammi cartesiani, sono lo strumento base per mettere in relazione su un sistema di coordinate cartesiane i valori di due variabili numeriche e quindi per lo studio delle distribuzioni numeriche bivariate.

Prendiamo come esempio il set di dati **ais**, e in particolare i dati ematologici relativi agli eritrociti o globuli rossi, la cui funzione essenziale come noto è trasportare l'ossigeno dai polmoni a tutti gli altri organi e tessuti, e che mediamente:
→ hanno una concentrazione nel sangue all'incirca di $5 \cdot 10^{12}$/L (ovvero ve ne sono all'incirca cinquemila miliardi in un litro di sangue);
→ hanno un volume all'incirca di 90 fL (un femtolitro corrisponde a $10^{-15}$ litri ovvero a 0.0000000000000090 litri);
→ contengono ciascuno all'incirca 30 pg di emoglobina (un picogrammo corrisponde a $10^{-12}$ g ovvero a 0.0000000000030 grammi).

Da questi dati si ricava che la percentuale del volume del sangue occupata dagli eritrociti (detta valore ematòcrito) è mediamente all'incirca il 45% (la restante parte, esclusi i globuli bianchi e le piastrine, che occupano un volume irrilevante, è liquida ed è rappresentata dal plasma) e che la concentrazione dell'emoglobina nel sangue è mediamente all'incirca di 15 g/dL (grammi per decilitro di sangue). Ma non solo. Da questi dati si ricava anche che la concentrazione dell'emoglobina e il valore ematòcrito dipendono in modo lineare dalla concentrazione degli eritrociti. Possiamo quindi dare un senso allo script[123] che traccia i grafici di dispersione che includono queste grandezze:

```
# GRAFICI DI DISPERSIONE (SCATTERPLOT)
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
str(ais) # struttura di ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
plot(rcc, hg, xlab="Eritrociti (10^12/L)", ylab="Emoglobina (g/dL)", main="Grafico di dispersione") #
# grafico di dispersione semplice
#
plot(rcc, hg, pch = 2, col = "goldenrod", xlab="Eritrociti (10^12/L)", ylab="Emoglobina (g/dL)",
main="Cambia simbolo e colore") # cambia simbolo e colore
#
plot(rcc, hg, pch = 20, col = "black", abline(lm(hg ~ rcc), col="black", lty=1, lwd=1), xlab="Eritrociti
(10^12/L)", ylab="Emoglobina (g/dL)", main="Regressione lineare") # grafico con retta di regressione
#
library(nlshelper) # carica il pacchetto
plot_loess(loess(hg ~ rcc, data=ais), col = "black", band=FALSE, pch = 20, xlab="Eritrociti (10^12/L)",
ylab="Emoglobina (g/dL)", main="Regressione polinomiale") # grafico con regressione polinomiale
#
```

[123] Lo script richiede il pacchetto **nlshelper** che deve essere preventivamente scaricato dal **CRAN**.

Dopo avere come al solito caricato il pacchetto, che include il set di dati, e mostrata la sua struttura, la funzione **attach(ais)** consente, nelle funzioni successive, di impiegare direttamente i nomi delle variabili del set di dati.

Per semplificare la presentazione grafica viene prevista con **par(mfrow=c(2,2))** la suddivisione della finestra in quattro quadranti, uno per ciascuno dei grafici che verranno preparati (**Fig. 5.18**).

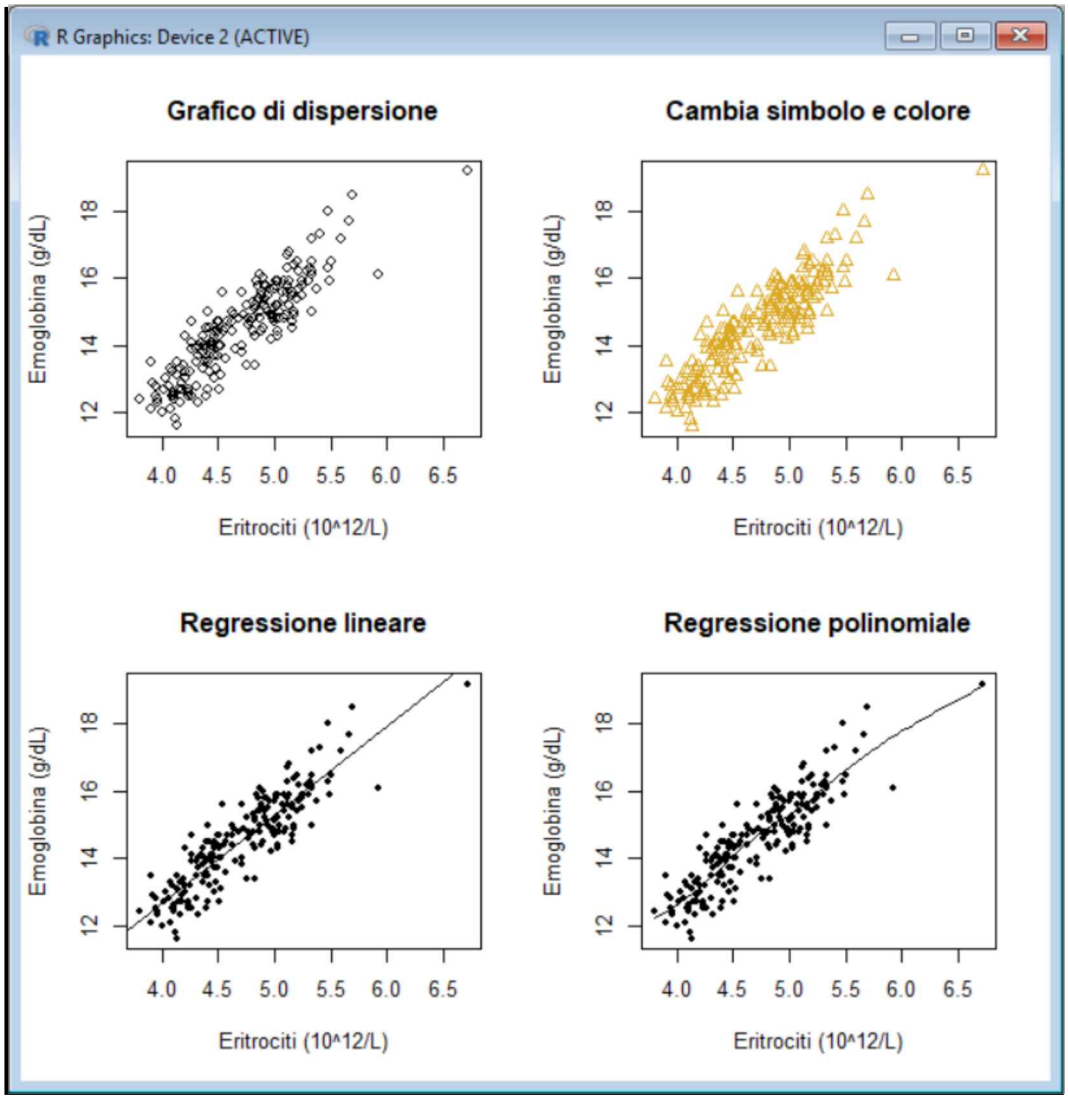



**Fig. 5.18** Grafici di dispersione per l'analisi della relazione tra emoglobina e globuli rossi.

Il primo grafico, a parte le etichette degli assi e il titolo, impiega gli argomenti di default della funzione **plot()**.

Nel secondo grafico con l'argomento **pch = 2** viene specificato per i dati l'impiego di un simbolo (triangolo) differente da quello di default (che è il cerchio)[124]. Inoltre con l'argomento **col = "goldenrod"** viene

[124] Per i simboli che è possibile impiegare per tracciare i punti vedere: **A10. Simboli dei punti e tipi di linea in R**.

specificato l'impiego di un colore diverso dal nero[125].

Nel terzo grafico viene riportata anche la retta di regressione **abline()** calcolata con la funzione **lm()** che a sua volta impiega l'argomento **hg ~ rcc** per specificare rispettivamente la variabile in ordinate e la variabile in ascisse. Inoltre per la retta sono specificati il colore (**col="black"**), il tipo di linea da impiegare (**lty=1**), uno dei sei possibili con i valori da 1 a 6[126], e lo spessore della linea (**lwd=1**) espresso come multiplo dello spessore di default (1 uguale spessore, 2 spessore doppio, eccetera).

Per realizzare il quarto grafico viene caricato il pacchetto **nlshelper** quindi la retta di regressione viene sostituita da un polinomio, al fine di verificare visivamente un eventuale scostamento dalla linearità, che se esiste è trascurabile e probabilmente è addirittura determinato da un singolo dato, quello di un soggetto con una concentrazione di eritrociti superiore a $6.5 \cdot 10^{12}$/L. Ma non si sa a quale caso appartenga il dato.

Questa osservazione fa sorgere un nuovo tipo di esigenza che viene risolta da **R** con uno strumento molto efficace per identificare i singoli dati in un grafico di dispersione, la funzione **identify()**, il cui utilizzo viene illustrato con questo script:

```
# IDENTIFICA I PUNTI IN UN GRAFICO DI DISPERSIONE
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
windows() # apre e inizializza una nuova finestra grafica
#
plot(rcc, hg, main="Identifica i punti in un grafico di dispersione", xlab="Eritrociti, 10^12/L ", ylab="Emoglobina, g/dL", pch=1) # traccia il grafico
#
# fare click sul punto da identificare, ripetere secondo necessità, premere <esc> per terminare
identify(rcc, hg, plot = TRUE, atpen = FALSE, offset = 0.5, tolerance = 0.25)
#
```

Nella **Fig. 5.19** è riportato il grafico ottenuto dopo avere fatto click con il tasto sinistro del mouse su due sospetti dati aberranti, che sono stati immediatamente identificati da **R** come i casi numero 161 e numero 166. Il processo può essere ripetuto più volte estendendolo a tutti i punti che si ritiene necessario identificare. Per uscire dal processo di identificazione è sufficiente premere <esc>[127].

Nello script la funzione **identify()** richiede come argomenti innanzitutto quale sia la variabile x in ascisse (**rcc**) e quale sia la variabile y in ordinate (**hg**). L'argomento **plot = TRUE** indica che l'identificativo del caso deve essere visualizzato vicino al punto sul quale si è fatto click con il mouse.

Con l'argomento **atpen = FALSE** l'identificativo del caso è posto lievemente discosto dal punto che lo rappresenta (provare a cambiare l'argomento ponendo **atpen = TRUE** per vedere la differenza), l'argomento **offset = 0.5** indica la distanza che l'identificativo del caso deve avere dal punto che lo rappresenta, e l'argomento **tolerance = 0.25** indica la tolleranza ammessa per questa distanza.

Di default viene emesso un suono ogniqualvolta si fa click su un punto che viene identificato, a meno che non venga specificato l'argomento **locatorBell = FALSE**.

---

[125] Per i colori che è possibile impiegare vedere: **A11. Tabella dei colori di R**.
[126] Per i tipi di linea che è possibile impiegare vedere: **A10. Simboli dei punti e tipi di linea di R**.
[127] Per gli argomenti impiegati dalla funzione **identify()** digitare **help(identify)** nella `Console di R`.

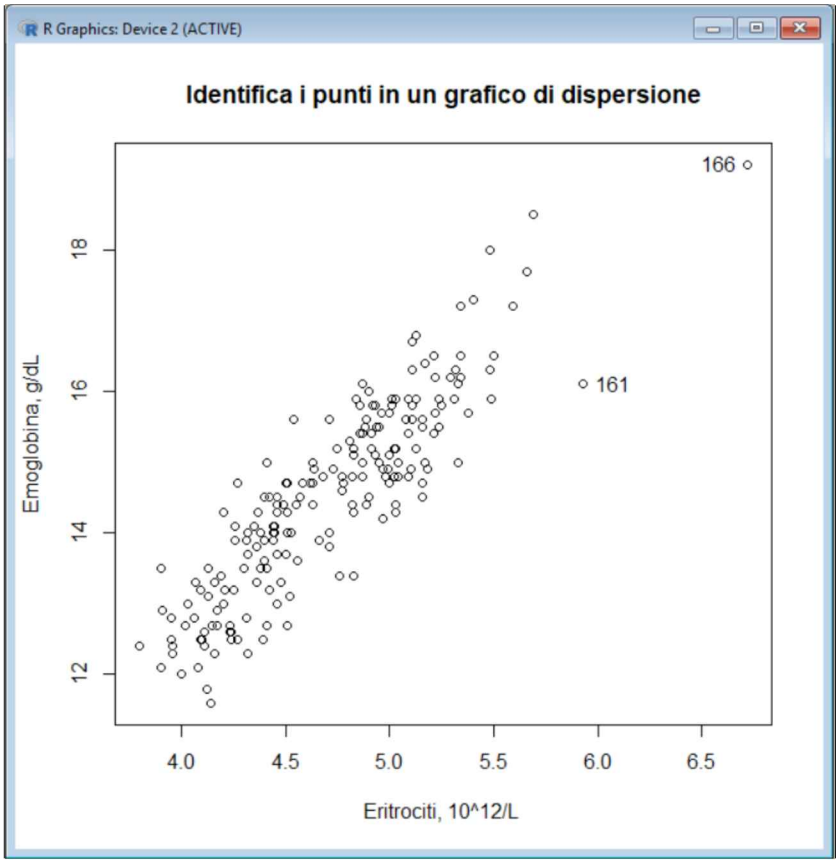



**Fig. 5.19** Facendo clik sui sospetti dati aberranti **R** riporta sul grafico di dispersione il loro identificativo, consentendo di effettuare su questi casi i necessari approfondimenti.

Aiuta ad evidenziare le relazioni tra le variabili un grafico di dispersione multiplo (**Fig. 5.20**) che riporta tutti i possibili grafici di dispersione singoli che risultano incrociando i dati ematologici degli atleti contenuti nelle cinque variabili **rcc** (eritrociti), **wcc** (leucociti), **hc** (ematòcrito), **hg** (emoglobina) e **ferr** (ferritina). Per ciascuna delle variabili viene inoltre riportato l'istogramma.

Questo è il codice per realizzare il grafico:

```
# GRAFICI DI DISPERSIONE MULTIPLI                                    1/2
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
library(car) # carica il pacchetto
windows() # apre e inizializza una nuova finestra grafica
scatterplotMatrix(~rcc+wcc+hc+hg+ferr, col="black", pch=20, regLine = list(method=lm, lty=1, lwd=2,
col="chartreuse3"), smooth=FALSE, diagonal=list(method ="histogram", breaks="FD"), main="Matrice di
dispersione con rette di regressione", data=ais) # grafico che incrocia i dati ematologici
#
```

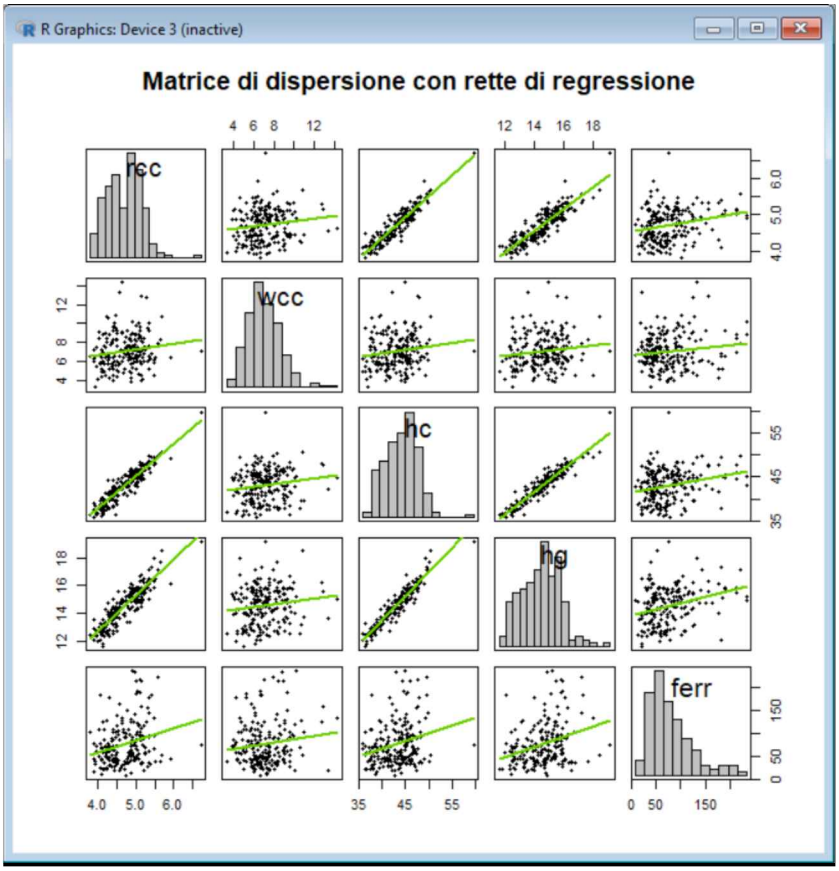



**Fig. 5.20** Matrice con i grafici di dispersione dei dati ematologici per tutte le combinazioni di variabili.

L'argomento **~rcc+wcc+hc+hg+ferr** definisce le variabili da rappresentare, e può essere facilmente riscritto per cambiare la rappresentazione a piacere. Ad esempio ponendo questo argomento uguale a **~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt** si possono rappresentare in uno stesso grafico tutte le variabili del set di dati **ais**.

Per la retta di regressione sono impiegati una linea continua (**lty=1**), uno spessore doppio (**lwd=2**) e un colore (**col="chartreuse3"**) diverso dal nero, mentre l'argomento **smooth=FALSE** fa si che non venga aggiunta ai grafici la rappresentazione della funzione non lineare prevista di default nel pacchetto **car**.

Infine l'argomento **diagonal=list()** indica la rappresentazione della variabile riportata nella diagonale, che qui è l'istogramma (**method ="histogram"**) con il numero di classi (**breaks="FD"**) calcolato mediante la regola di Freedman-Diaconis. L'istogramma può essere sostituito con altri tipi di rappresentazione[128].

Dopo avere scaricato dal **CRAN** il pacchetto **gclus** potete eseguire questa seconda parte dello script:

[128] Questo è l'elenco delle espressioni che è possibile impiegare in alternativa per questo argomento (provatele):
→ **diagonal=list(method="density", bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE)**
→ **diagonal=list(method="boxplot")**
→ **diagonal=list(method="qqplot")**
→ **diagonal=list(method="oned")**

```
# grafici di dispersione ordinati in base al valore di r                    2/2
#
library(gclus) # carica il pacchetto
windows() # apre e inizializza una nuova finestra grafica
mydata <- ais[c(1,2,3,4,5)] # tabella con i dati delle variabili delle colonne 1,2,3,4,5
mydata.r <- abs(cor(mydata)) # calcola la correlazione
mydata.col <- dmat.color(mydata.r) # applica i colori
mydata.o <- order.single(mydata.r) # ordina le variabili, le meglio correlate più vicine alla diagonale
cpairs(mydata, mydata.o, panel.colors=mydata.col, gap=.5, main="Variabili meglio correlate vicine alla
diagonale") # mostra il grafico
#
```

Viene realizzato di nuovo un grafico di dispersione multiplo (**Fig. 5.21**) che incrocia ancora i dati ematologici degli atleti contenuti nelle cinque variabili **rcc** (eritrociti), **wcc** (leucociti), **hc** (ematòcrito), **hg** (emoglobina) e **ferr** (ferritina) ma con due novità.

La prima è che le variabili ora sono individuate non mediante il loro nome bensì indicando il set di dati che le contiene e il numero della colonna corrispondente a ciascuna variabile con l'espressione **ais[c(1,2,3,4,5)]**.

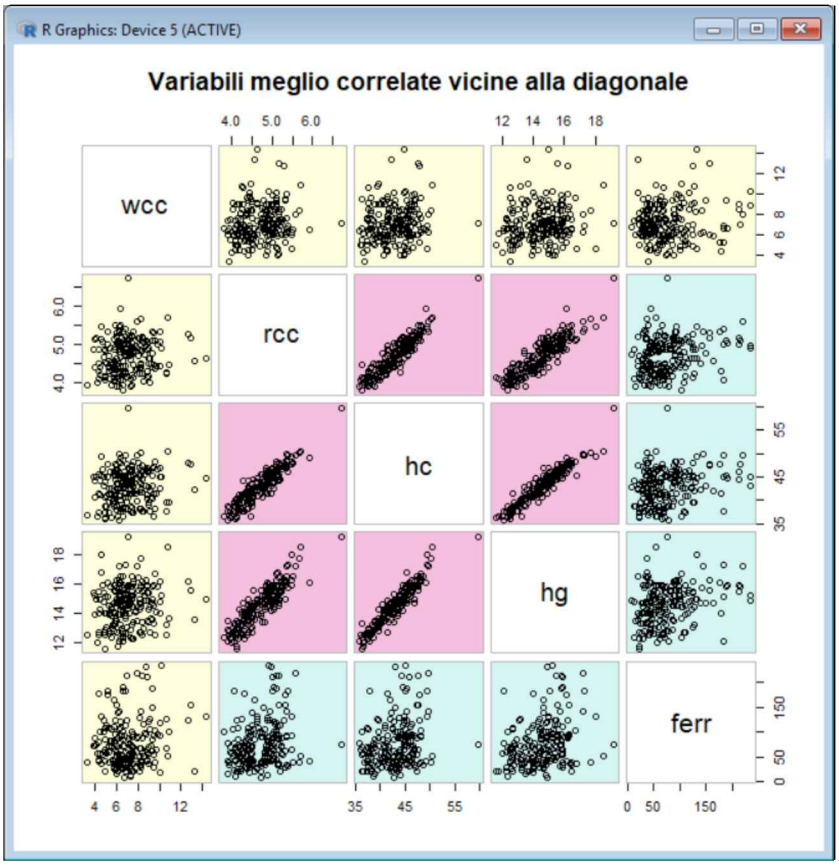



**Fig. 5.21** Matrice con i grafici di dispersione ordinati in base al valore del coefficiente di correlazione **r** ed evidenziati in colore.

La seconda novità è che impiegando la funzione **cpairs()**[129] i singoli grafici di dispersione vengono ordinati in base al valore del coefficiente di correlazione **r** di Pearson, in modo che le variabili meglio correlate siano rappresentate più vicine alla diagonale ed evidenziate in colore, aumentando la chiarezza della rappresentazione.

---

[129] Digitate **help(cpairs)** nella Console di R per la documentazione della funzione **cpairs()** ovvero consultate il manuale di riferimento del pacchetto **gclus** su: *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

## 5.7. Grafici 3D

Per realizzare con **R** grafici che rappresentano tre variabili in uno spazio a tre dimensioni sono necessarie le funzioni contenute in tre pacchetti, rispettivamente **scatterplot3d**, **rgl** e **Rcmdr** che è necessario scaricare dal **CRAN**. Dopo avere scaricato il pacchetto **Rcmdr** si consiglia di digitare nella `Console di R`

```
library(Rcmdr)
```

in quanto eseguendo questa funzione per la prima volta potrebbero essere scaricati ulteriori pacchetti aggiuntivi necessari per creare l'ambiente grafico.

Trovate come al solito il manuale di riferimento dei pacchetti, con la documentazione completa delle relative funzioni e dei relativi argomenti, sul repository della documentazione[130].

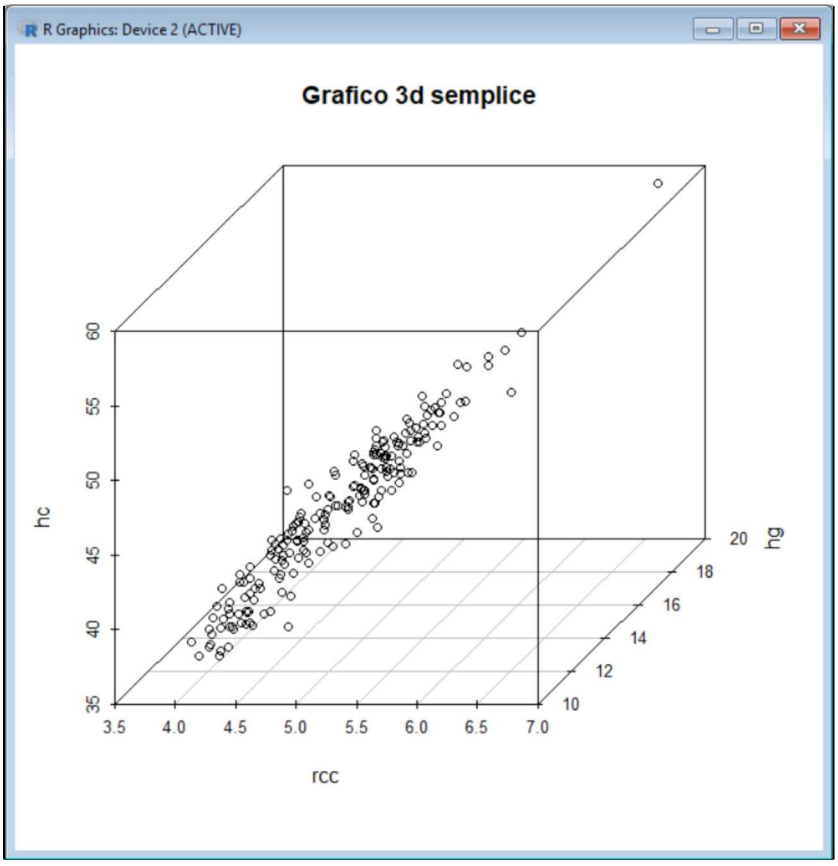



**Fig. 5.22** Grafico 3D delle distribuzioni di eritrociti (**x**), emoglobina (**y**) ed ematòcrito (**z**).

Come dati impieghiamo ancora una volta i valori degli eritrociti (**rcc**), dell'emoglobina (**hg**) e dell'ematòcrito (**hc**) degli atleti australiani contenuti nel set di dati **ais**. Le prime tre righe di codice dello script servono per caricare i dati (**library(DAAG)**), visualizzarne la struttura (**str(ais)**) e per impiegare nelle funzioni successive

[130] *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

direttamente i nomi delle variabili (**attach(ais)**).

Il primo grafico 3D (**Fig. 5.22**) è molto semplice da realizzare, in quanto richiede solamente di caricare il pacchetto con il comando **library(scatterplot3d)** e di eseguire la funzione **scatterplot3D()** specificando come argomento le variabili da rappresentare (**rcc, hg, hc**) e (facoltativamente) un titolo. Da notare che l'ordine nel quale sono inserite la variabili corrisponde all'ordine degli assi **x**, **y** e **z**.

```
# SCATTERPLOT 3D                                                    1/3
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
str(ais) # struttura di ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
# grafico tridimensionale (3D) semplice
#
library(scatterplot3d) # carica il pacchetto
scatterplot3d(rcc, hg, hc, main="Grafico 3d semplice") # grafico 3D
#
```

Chiudete la finestra grafica e con questa seconda parte dello script realizzate un grafico 3D che può essere ruotato (**Fig. 5.23**):

```
# spinning 3D scatterplot                                           2/3
#
library(rgl) # carica il pacchetto
plot3d(rcc, hg, hc, type="p", col="red", size=3) # grafico 3D
#
```

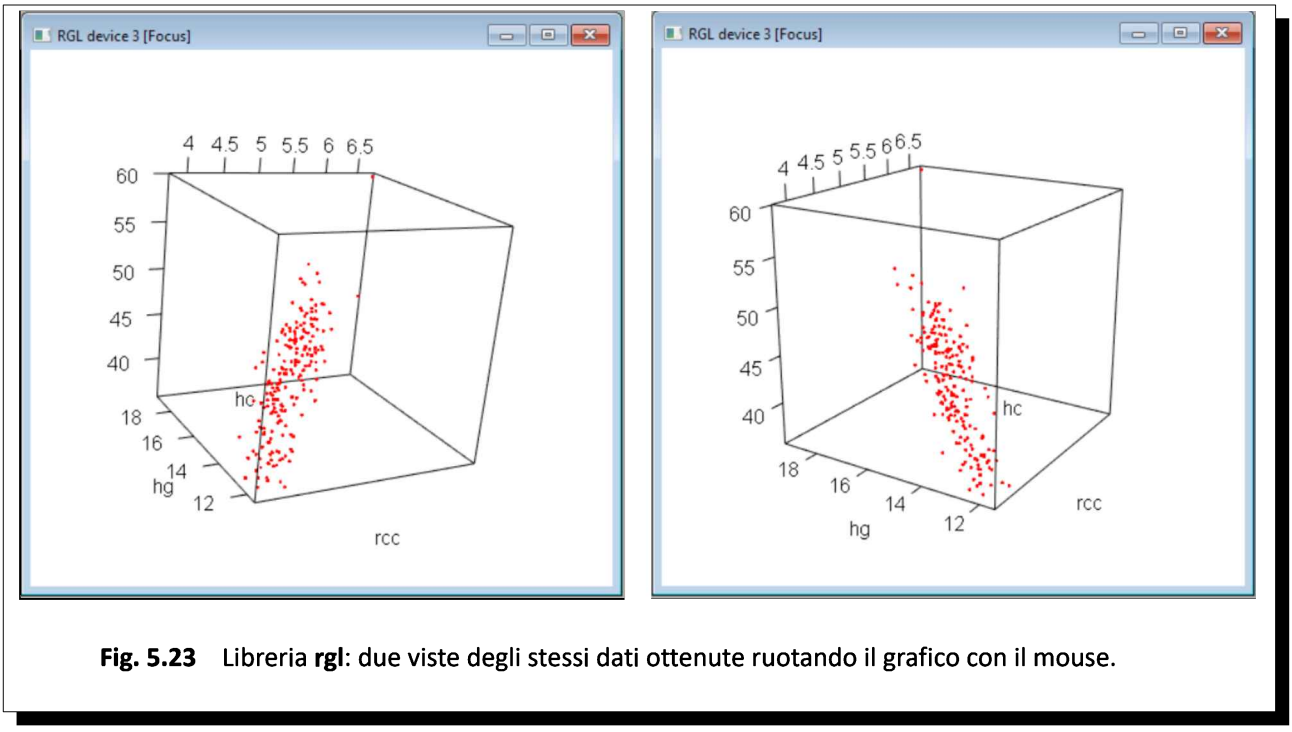



**Fig. 5.23** Libreria **rgl**: due viste degli stessi dati ottenute ruotando il grafico con il mouse.

Se “afferrate” il grafico 3D risultante facendo click su di esso e tenendo premuto il tasto sinistro del mouse senza rilasciarlo, potete ruotare il grafico a vostro piacimento spostando il mouse. In questo modo sono state realizzate le due immagini della **Fig. 5.23** che, affiancate, forniscono una vista interessante sui dati.

Chiudete la finestra grafica prima di eseguire questa terza e ultima parte della script nel quale viene impiegato il pacchetto **Rcmdr** per realizzare un altro tipo di grafico 3D che può essere ruotato (**Fig. 5.24**):

```
# spinning 3D scatterplot                                    3/3
#
library(Rcmdr) # carica il pacchetto
scatter3d(rcc, hg, hc) # grafico 3D
#
```

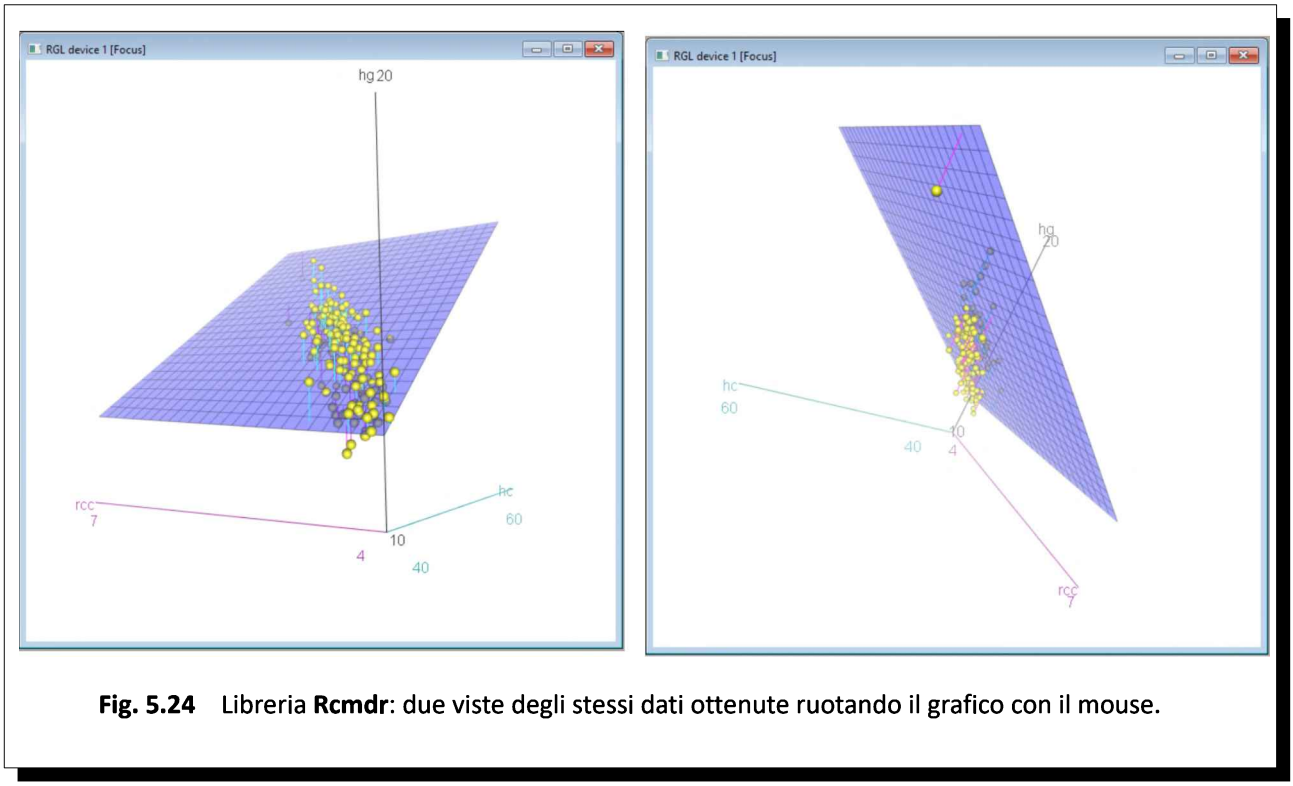



**Fig. 5.24** Libreria **Rcmdr**: due viste degli stessi dati ottenute ruotando il grafico con il mouse.

Anche in questo caso se “afferrate” il grafico 3D risultante, facendo click con il tasto sinistro del mouse e tenendolo premuto senza rilasciarlo, potete ruotarlo a vostro piacimento spostando il mouse. In questo modo sono state realizzate le due immagini della **Fig. 5.24** che, affiancate, forniscono di nuovo una vista interessante sui dati.

Se ritenete che sia più utile salvare le tre parti dello script come script separati e autonomi ricordatevi che nella seconda e nella terza parte dovete aggiungere, anteponendole al codice riportato, sia la riga di codice che carica il pacchetto con il set di dati **ais** (**library(DAAG)** # ….) sia quella che consente di impiegare direttamente i nomi delle variabili nelle funzioni che tracciano i grafici (**attach(ais)** # ….) .

## 5.8. Salvare i grafici di R in un file

Una rappresentazione grafica è importante non solo visualizzarla, ma anche salvarla sotto forma di file per archiviarla o inserirla in una pubblicazione, in un post o in un sito web.

In questo script lo stesso grafico viene salvato sotto forma di file in formati grafici differenti. L'esempio è quello già impiegato nella presentazione dei grafici a scatola con i baffi (o boxplot)[131] e riguarda la distribuzione della concentrazione degli eritrociti per sport praticato ricavata mediante la funzione **boxplot()** dal set di dati **ais**.

```
# SALVA I GRAFICI SU DISCO SOTTO FORMA DI FILE                                    1/2
#
# salva grafici come immagini in formato raster (immagini bitmap)
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
#
# salva in formato .bmp (Windows bitmap)
#
bmp("c:/Rdati/boxplot.bmp", units = "px", width = 1400, height = 1000, pointsize = 24, bg = "white") #
# predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, data=ais, main="Eritrociti per sport praticato", xlab="Sport
praticato", ylab="Eritrociti in 10^12/L", notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
# salva in formato .jpeg (Joint Photographic Experts Group)
#
jpeg("c:/Rdati/boxplot.jpg", width = 1400, height = 1000, pointsize = 24, bg = "white") # predispone
# nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, data=ais, main="Eritrociti per sport praticato", xlab="Sport
praticato", ylab="Eritrociti in 10^12/L", notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
# salva in formato .png (Portable Network Graphics)
#
png("c:/Rdati/boxplot.png", width = 1400, height = 1000, pointsize = 24, bg = "white") # predispone
# nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, data=ais, main="Eritrociti per sport praticato", xlab="Sport
praticato", ylab="Eritrociti in 10^12/L", notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
```

Per i formati grafici `.bmp` (*Windows bitmap*), .jpeg (*Joint Photographic Experts Group*) e `.png` (*Portable Network Graphics*) gli argomenti **width=** e **height=** specificano larghezza e altezza dell'immagine in pixel. Questa unità di misura può essere cambiata con l'argomento **units = "px"** che può essere espresso in alternativa come **"in"** (pollici), **"cm"** (centimetri) o **"mm"** (millimetri).

[131] Vedere: **5.4. Grafici a scatola con i baffi**.

L'argomento **pointsize = 24** specifica la dimensione dei caratteri, mentre l'argomento **bg = "white"** specifica il colore dello sfondo.

```
# salva grafici come immagini in formato vettoriale                    2/2
#
# salva in formato .pdf (Portable Document Format)
#
pdf("c:/Rdati/boxplot.pdf", width = 10, height = 7) # predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, data=ais, main="Eritrociti per sport praticato", xlab="Sport
praticato", ylab="Eritrociti in 10^12/L", notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
# salva in formato .ps (postscript)
#
postscript("c:/Rdati/boxplot.ps", width = 10, height = 7) # predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, data=ais, main="Eritrociti per sport praticato", xlab="Sport
praticato", ylab="Eritrociti in 10^12/L", notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
```

I formati `.pdf` (*Portable Document Format*) e `.ps` (*Postscript*) che non sono formati raster ma sono formati vettoriali[132] prevedono le dimensioni solamente in pollici.

La funzione **dev(off)** salva il file e chiude il dispositivo che ha generato la stampa, e a questo punto il dispositivo al quale sono inviati i grafici torna ad essere la finestra che si apre al bisogno sul monitor/schermo del PC o notebook.

---

[132] Una illustrazione concisa ma documentata della differenza la potete trovare alla voce *Grafica vettoriale* della Wikipedia. URL consultato il 9/11/2018: https://goo.gl/bPMrdo

# Appendici

# A1. Codifica dei caratteri ASCII, ANSI, Unicode e UTF

**ASCII** è l'acronimo di **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange ed è il primo standard introdotto nella codifica dei caratteri di scrittura[133]. Il codice **ASCII** è definito nella norma ISO/IEC 646 *Information technology - ISO 7-bit coded character set for information interchange*[134]. Nel codice **ASCII** i caratteri sono codificati con 7 bit cosa che permette di codificare $2^7 = 128$ caratteri numerati da 0 a 127.

I caratteri da 0 a 31 sono caratteri di controllo e non sono stampabili. Il carattere 32 corrisponde a uno spazio. I caratteri da 33 a 126 sono caratteri stampabili. Il carattere 127 corrisponde a `Delete` (equivalente a `Backspace`).

Questo è il **set di caratteri ASCII standard** o **set di caratteri ASCII di base**:

| Codice | Simbolo | Codice | Carattere | Codice | Carattere | Codice | Carattere |
|---|---|---|---|---|---|---|---|
| 0 | NUL | 32 | | 64 | @ | 96 | ` |
| 1 | SOH | 33 | ! | 65 | A | 97 | a |
| 2 | STX | 34 | " | 66 | B | 98 | b |
| 3 | ETX | 35 | # | 67 | C | 99 | c |
| 4 | EOT | 36 | $ | 68 | D | 100 | d |
| 5 | ENQ | 37 | % | 69 | E | 101 | e |
| 6 | ACK | 38 | & | 70 | F | 102 | f |
| 7 | BEL | 39 | ' | 71 | G | 103 | g |
| 8 | BS | 40 | ( | 72 | H | 104 | h |
| 9 | TAB | 41 | ) | 73 | I | 105 | i |
| 10 | LF | 42 | * | 74 | J | 106 | j |
| 11 | VT | 43 | + | 75 | K | 107 | k |
| 12 | FF | 44 | , | 76 | L | 108 | l |
| 13 | CR | 45 | - | 77 | M | 109 | m |
| 14 | SO | 46 | . | 78 | N | 110 | n |
| 15 | SI | 47 | / | 79 | O | 111 | o |
| 16 | DLE | 48 | 0 | 80 | P | 112 | p |
| 17 | DC1 | 49 | 1 | 81 | Q | 113 | q |
| 18 | DC2 | 50 | 2 | 82 | R | 114 | r |
| 19 | DC3 | 51 | 3 | 83 | S | 115 | s |
| 20 | DC4 | 52 | 4 | 84 | T | 116 | t |
| 21 | NAK | 53 | 5 | 85 | U | 117 | u |
| 22 | SYN | 54 | 6 | 86 | V | 118 | v |
| 23 | ETB | 55 | 7 | 87 | W | 119 | w |
| 24 | CAN | 56 | 8 | 88 | X | 120 | x |
| 25 | EM | 57 | 9 | 89 | Y | 121 | y |
| 26 | SUB | 58 | : | 90 | Z | 122 | z |
| 27 | ESC | 59 | ; | 91 | [ | 123 | { |
| 28 | FS | 60 | < | 92 | \ | 124 | \| |
| 29 | GS | 61 | = | 93 | ] | 125 | } |
| 30 | RS | 62 | > | 94 | ^ | 126 | ~ |
| 31 | US | 63 | ? | 95 | _ | 127 | |

[133] *ASCII*. Fonte Wikipedia. URL consultato il 18/10/2018: https://goo.gl/4amvuf
[134] *ISO/IEC 646*. Fonte Wikipedia. URL consultato il 18/10/2018: https://goo.gl/hAuERi

Gli altri set di caratteri che vediamo ora includono il set di caratteri **ASCII** stampabili come set di base e lo estendono con l'obiettivo di includere tutti i caratteri di tutte le lingue.

**ANSI** è l'acronimo di **A**merican **N**ational **S**tandard **I**nstitute. Nel set di caratteri **ANSI** i caratteri sono codificati con 8 bit cosa che permette di codificare $2^8 = 256$ caratteri numerati da 0 a 255. Il set di caratteri **ANSI** include il set di caratteri **ASCII di base** (caratteri numerati da 0 a 127) più un set di caratteri denominati **ASCII esteso** (caratteri numerati da 128 a 255). Purtroppo per i caratteri da 128 a 255 non si è stabilito subito un accordo e questo ha determinato l'insorgenza di ambiguità ad esempio tra il set di caratteri Windows 1252 (che viene denominato impropriamente **ANSI** nel `Blocco note` di Windows e che è ampiamente utilizzato) e il set di caratteri ISO-8859-1 anch'esso a 8 bit.

Data la vitale importanza delle codifica univoca dei caratteri è nato **Unicode**, un consorzio no-profit privato al quale partecipano tutte le principali aziende di informatica, con lo scopo di assegnare ad ogni carattere impiegato per la scrittura un numero univoco indipendente dalla lingua, dalla piattaforma informatica e dal software impiegati[135].

Lo standard **Unicode** è denominato **U**nicode **T**ransformation **F**ormat (**UTF**). Dal 1991 il Consorzio Unicode ha sviluppato questo in parallelo con la norma ISO/IEC 10646 che definisce l'**U**niversal **C**oded **C**haracter **S**et (**UCS**). Entrambi hanno lo stesso repertorio di caratteri, con gli stessi numeri. Si rimanda pertanto a **Unicode** come fonte ufficiale per la codifica dei caratteri che includono anche i tradizionali e storici:
→ **codice ASCII di base** (Basic Latin (ASCII))[136]
→ **codice ASCII esteso** (Latin-1 Supplement)[137]
oltre ovviamente agli altri set di caratteri che sono nati per coprire tutte le lingue del mondo[138].

Attualmente il sistema di codifica più diffuso è l'**UTF-8**: dai dati che vengono riportati sembra che più di metà delle pagine web siano codificate con questo standard.

---

[135] *The Unicode Consortium*. URL consultato il 18/10/2018: https://goo.gl/MwjR84
[136] *Unicode 11.0 Character Code Charts. Basic Latin (ASCII)*. URL consultato il 18/10/2018: https://goo.gl/GJHqfW
[137] *Unicode 11.0 Character Code Charts. Latin-1 Supplement*. URL consultato il 18/10/2018: https://goo.gl/cfhgLd
[138] *Unicode Consortium. Unicode 11.0 Character Code Charts*. URL consultato il 18/10/2018: https://goo.gl/rsFsgv

# A2. Definizioni della probabilità

Diceva Bruno de Finetti: *"Ci fosse fuori di noi, per voi e per me, ci fosse una signora probabilità mia e una signora probabilità vostra, dico per se stesse, e uguali, immutabili. Non c'è..."*[139].

E proprio per il fatto di essere un concetto intuitivamente facile, ma difficile da formalizzare, sono oggi ben quattro le definizioni di probabilità.

La **definizione classica** (di Laplace) dice che:

"*La probabilità è il rapporto fra il numero di eventi favorevoli e il numero di eventi possibili, essendo questi ultimi tutti equiprobabili*" ovvero

$$P(A) = \frac{nA}{n}$$

Da notare che la definizione classica di probabilità contiene un vizio logico, in quanto la probabilità viene utilizzata per definire sé stessa.

La **definizione frequentista** (di Von Mises) dice che:

"*La probabilità di un evento è il rapporto fra il numero di esperimenti in cui esso si è verificato e il numero totale di esperimenti eseguiti nelle stesse condizioni, essendo tale numero opportunamente grande*"

$$P(A) = \lim_{n \to \infty} \frac{nA}{n}$$

La **definizione soggettivista** (di De Finetti) dice che:

*"...la probabilità che qualcuno attribuisce alla verità - o al verificarsi - di un certo evento (fatto singolo univocamente descritto e precisato) altro non è che la misura del grado di fiducia nel suo verificarsi"*

La **definizione assiomatica** (di Kolmogorov) dice che :

*"La probabilità è un numero compreso tra 0 (evento impossibile) e 1 (evento certo) che soddisfa i*

[139] De Finetti B. *L'invenzione della verità*. Raffaello Cortina Editore, Milano, 2006, ISBN 88-6030-060-6, p. 35.

*tre assiomi di Kolmogorov*[140]"

L'impostazione assiomatica della probabilità venne proposta da Andrey Nikolaevich Kolmogorov nel 1933 in *Grundbegriffe der Wahrscheinlichkeitsrechnung* (*Concetti fondamentali del calcolo delle probabilità*), sviluppando la ricerca che era ormai cristallizzata sul dibattito fra quanti consideravano la probabilità come limiti di frequenze relative (impostazione frequentista) e quanti cercavano un fondamento logico della stessa. La sua impostazione assiomatica si mostrava adeguata a prescindere dall'adesione a una o all'altra scuola di pensiero.

Un esempio dovuto a De Finetti consente di illustrare la differenza tra le prime tre definizioni. Immaginiamo una partita di calcio per la quale gli eventi possibili sono:
- la vittoria della squadra di casa;
- la vittoria della squadra ospite;
- il pareggio.

Secondo la teoria classica esiste 1 probabilità su 3 che avvenga la vittoria della squadra di casa.

Secondo la teoria frequentista ci si può dotare di un almanacco, controllare tutte le partite precedenti e calcolare la frequenza di un evento.

Secondo la teoria soggettiva, ci si può documentare sullo stato di forma dei calciatori, sul terreno di gioco e così via fino ad emettere un giudizio di probabilità (soggettiva).

---

[140] La probabilità assiomatica è una funzione d'insieme **P** definita sullo spazio degli eventi **S**, ovvero è una legge in grado di assegnare ad ogni evento **E** appartenente ad **S** un numero che soddisfa i tre assiomi di Kolmogorov:
1) la probabilità **P**(**E**) di un evento **E** è un numero reale non negativo;
2) la probabilità **P**(**U**) dell'evento certo è 1;
3) la probabilità di un evento complesso costituito dal verificarsi dell'evento elementare **A** o dell'evento elementare **B**, mutuamente incompatibili, è la somma delle probabilità di **A** e di **B**: **P**(**A** o **B**) = **P**(**A**) + **P**(**B**)"

# A3. Il set di dati ais

Nel set di dati **ais**, che viene distribuito con il pacchetto **DAAG** sono contenuti **dati rilevati su 202 atleti australiani** così espressi[141]:
→ [campo senza nome] = un numero progressivo da 1 a 202 identificativo del caso[142]
→ rcc = eritrociti (globuli rossi) espressi in $10^{12}$/L o in $10^{6}$/µL (le due espressioni sono numericamente equivalenti)
→ wbc = leucociti (globuli bianchi) espressi in $10^{9}$/L o in $10^{3}$/µL (le due espressioni sono numericamente equivalenti)
→ hc = ematòcrito espresso in %
→ hg = emoglobina espressa in g/dL
→ ferr = ferritina espressa in µg/L
→ bmi = indice di massa corporea espresso in kg/m$^2$
→ ssf = somma dello spessore delle pliche cutanee di tricipite e bicipite (nell'avambraccio), sottoscapolare e sovrailiaca espressa in mm
→ pcBfat = percentuale di grasso corporeo espresso in %
→ lbm = massa magra espressa in kg
→ ht = altezza espressa in cm
→ wt = peso espresso in kg
→ sex = sesso (m,f)
→ sport = sport praticato ( B_Ball, Field, Gym, Netball, Row, Swim, T_400m, T_Sprnt, Tennis, W_Polo)

Lo scopo è mettere in relazione sesso e sport praticato con alcuni dati ematologici (rcc, wbc, hc, hg, ferr) e con alcuni dati biometrici (bmi, ssf, pcBfat, lbm, ht, wt) dell'atleta.

Questo script esporta i dati da **R** verso un programma esterno (foglio elettronico o database) in formato `.csv` nel file `C:\Rdati\esporta_ais.csv` impiegando come separatore di campo il punto e virgola (;) e come separatore dei decimali la virgola (,):

```
# ESPORTA DA R IN UN FILE .csv IL SET DI DATI ais DEL PACCHETTO DAAG
# impiega come separatore di campo il punto e virgola (;) come separatore delle cifre decimali la virgola (,)
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
write.table(ais, file="C:/Rdati/esporta_ais.csv", quote=FALSE, sep=";", dec=",", na="", col.names=TRUE, row.names=FALSE) # esporta i dati, notare / invece di \ su windows
#
```

Nella tabella che segue sono riportati i dati originali forniti con il pacchetto **DAAG**.

| | rcc | wcc | hc | hg | ferr | bmi | ssf | pcBfat | lbm | ht | wt | sex | sport |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.96 | 7.5 | 37.5 | 12.3 | 60 | 20.56 | 109.1 | 19.75 | 63.32 | 195.9 | 78.9 | f | B_Ball |
| 2 | 4.41 | 8.3 | 38.2 | 12.7 | 68 | 20.67 | 102.8 | 21.3 | 58.55 | 189.7 | 74.4 | f | B_Ball |
| 3 | 4.14 | 5 | 36.4 | 11.6 | 21 | 21.86 | 104.6 | 19.88 | 55.36 | 177.8 | 69.1 | f | B_Ball |
| 4 | 4.11 | 5.3 | 37.3 | 12.6 | 69 | 21.88 | 126.4 | 23.66 | 57.18 | 185 | 74.9 | f | B_Ball |
| 5 | 4.45 | 6.8 | 41.5 | 14 | 29 | 18.96 | 80.3 | 17.64 | 53.2 | 184.6 | 64.6 | f | B_Ball |

[141] Questa precisazione si rende necessaria in quanto la documentazione ufficiale del set di dati **ais** contiene alcuni errori nelle unità di misura, ad esempio per la concentrazione dell'emoglobina viene riportato "*hemaglobin concentration, in g per decaliter*", dove oltre a un primo refuso ("*hemaglobin*" in luogo di "*hemoglobin*" o "*haemoglobin*") compare "*decaliter*" cioè decalitro (dieci litri) invece di "*deciliter*" cioè decilitro (un decimo di litro).
[142] In assenza di un identificativo del caso **R** lo crea automaticamente assegnando un numero progressivo identificativo univoco che viene riportato in un campo senza nome.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 4.1 | 4.4 | 37.4 | 12.5 | 42 | 21.04 | 75.2 | 15.58 | 53.77 | 174 | 63.7 | f | B_Ball |
| 7 | 4.31 | 5.3 | 39.6 | 12.8 | 73 | 21.69 | 87.2 | 19.99 | 60.17 | 186.2 | 75.2 | f | B_Ball |
| 8 | 4.42 | 5.7 | 39.9 | 13.2 | 44 | 20.62 | 97.9 | 22.43 | 48.33 | 173.8 | 62.3 | f | B_Ball |
| 9 | 4.3 | 8.9 | 41.1 | 13.5 | 41 | 22.64 | 75.1 | 17.95 | 54.57 | 171.4 | 66.5 | f | B_Ball |
| 10 | 4.51 | 4.4 | 41.6 | 12.7 | 44 | 19.44 | 65.1 | 15.07 | 53.42 | 179.9 | 62.9 | f | B_Ball |
| 11 | 4.71 | 5.3 | 41.4 | 14 | 38 | 25.75 | 171.1 | 28.83 | 68.53 | 193.4 | 96.3 | f | B_Ball |
| 12 | 4.62 | 7.3 | 43.8 | 14.7 | 26 | 21.2 | 76.8 | 18.08 | 61.85 | 188.7 | 75.5 | f | B_Ball |
| 13 | 4.35 | 7.8 | 41.4 | 14.1 | 30 | 22.03 | 117.8 | 23.3 | 48.32 | 169.1 | 63 | f | B_Ball |
| 14 | 4.26 | 6.2 | 41 | 13.9 | 48 | 25.44 | 90.2 | 17.71 | 66.24 | 177.9 | 80.5 | f | Row |
| 15 | 4.63 | 6 | 43.7 | 14.7 | 30 | 22.63 | 97.2 | 18.77 | 57.92 | 177.5 | 71.3 | f | Row |
| 16 | 4.36 | 5.8 | 40.3 | 13.3 | 29 | 21.86 | 99.9 | 19.83 | 56.52 | 179.6 | 70.5 | f | Row |
| 17 | 3.91 | 7.3 | 37.6 | 12.9 | 43 | 22.27 | 125.9 | 25.16 | 54.78 | 181.3 | 73.2 | f | Row |
| 18 | 4.51 | 8.3 | 43.7 | 14.7 | 34 | 21.27 | 69.9 | 18.04 | 56.31 | 179.7 | 68.7 | f | Row |
| 19 | 4.37 | 8.1 | 41.8 | 14.3 | 53 | 23.47 | 98 | 21.79 | 62.96 | 185.2 | 80.5 | f | Row |
| 20 | 4.9 | 6.9 | 44 | 14.5 | 59 | 23.19 | 96.8 | 22.25 | 56.68 | 177.3 | 72.9 | f | Row |
| 21 | 4.46 | 5.7 | 39.2 | 13 | 43 | 23.17 | 80.3 | 16.25 | 62.39 | 179.3 | 74.5 | f | Row |
| 22 | 3.95 | 3.3 | 36.9 | 12.5 | 40 | 24.54 | 74.9 | 16.38 | 63.05 | 175.3 | 75.4 | f | Row |
| 23 | 4.46 | 9.5 | 41.5 | 14.5 | 92 | 22.96 | 83 | 19.35 | 56.05 | 174 | 69.5 | f | Row |
| 24 | 5.02 | 6.4 | 44.8 | 15.2 | 48 | 19.76 | 91 | 19.2 | 53.65 | 183.3 | 66.4 | f | Row |
| 25 | 4.26 | 5.8 | 41.2 | 14.1 | 77 | 23.36 | 76.2 | 17.89 | 65.45 | 184.7 | 79.7 | f | Row |
| 26 | 4.46 | 5.6 | 41.1 | 14.3 | 71 | 22.67 | 52.6 | 12.2 | 64.62 | 180.2 | 73.6 | f | Row |
| 27 | 4.16 | 5.8 | 39.8 | 13.3 | 37 | 24.24 | 111.1 | 23.7 | 60.05 | 180.2 | 78.7 | f | Row |
| 28 | 4.49 | 7.6 | 41.8 | 14.4 | 71 | 24.21 | 110.7 | 24.69 | 56.48 | 176 | 75 | f | Row |
| 29 | 4.21 | 7.5 | 38.4 | 13.2 | 73 | 20.46 | 74.7 | 16.58 | 41.54 | 156 | 49.8 | f | Row |
| 30 | 4.57 | 6.6 | 42.8 | 14.5 | 85 | 20.81 | 113.5 | 21.47 | 52.78 | 179.7 | 67.2 | f | Row |
| 31 | 4.87 | 6.4 | 44.8 | 15 | 64 | 20.17 | 99.8 | 20.12 | 52.72 | 180.9 | 66 | f | Row |
| 32 | 4.44 | 10.1 | 42.7 | 14 | 19 | 23.06 | 80.3 | 17.51 | 61.29 | 179.5 | 74.3 | f | Row |
| 33 | 4.45 | 6.6 | 42.6 | 14.1 | 39 | 24.4 | 109.5 | 23.7 | 59.59 | 178.9 | 78.1 | f | Row |
| 34 | 4.41 | 5.9 | 41.1 | 13.5 | 41 | 23.97 | 123.6 | 22.39 | 61.7 | 182.1 | 79.5 | f | Row |
| 35 | 4.87 | 7.3 | 44.1 | 14.8 | 13 | 22.62 | 91.2 | 20.43 | 62.46 | 186.3 | 78.5 | f | Row |
| 36 | 4.56 | 13.3 | 42.2 | 13.6 | 20 | 19.16 | 49 | 11.29 | 53.14 | 176.8 | 59.9 | f | Netball |
| 37 | 4.15 | 6 | 38 | 12.7 | 59 | 21.15 | 110.2 | 25.26 | 47.09 | 172.6 | 63 | f | Netball |
| 38 | 4.16 | 7.6 | 37.5 | 12.3 | 22 | 21.4 | 89 | 19.39 | 53.44 | 176 | 66.3 | f | Netball |
| 39 | 4.32 | 6.4 | 37.7 | 12.3 | 30 | 21.03 | 98.3 | 19.63 | 48.78 | 169.9 | 60.7 | f | Netball |
| 40 | 4.06 | 5.8 | 38.7 | 12.8 | 78 | 21.77 | 122.1 | 23.11 | 56.05 | 183 | 72.9 | f | Netball |
| 41 | 4.12 | 6.1 | 36.6 | 11.8 | 21 | 21.38 | 90.4 | 16.86 | 56.45 | 178.2 | 67.9 | f | Netball |
| 42 | 4.17 | 5 | 37.4 | 12.7 | 109 | 21.47 | 106.9 | 21.32 | 53.11 | 177.3 | 67.5 | f | Netball |
| 43 | 3.8 | 6.6 | 36.5 | 12.4 | 102 | 24.45 | 156.6 | 26.57 | 54.41 | 174.1 | 74.1 | f | Netball |
| 44 | 3.96 | 5.5 | 36.3 | 12.4 | 71 | 22.63 | 101.1 | 17.93 | 55.97 | 173.6 | 68.2 | f | Netball |
| 45 | 4.44 | 9.7 | 41.4 | 14.1 | 64 | 22.8 | 126.4 | 24.97 | 51.62 | 173.7 | 68.8 | f | Netball |
| 46 | 4.27 | 10.6 | 37.7 | 12.5 | 68 | 23.58 | 114 | 22.62 | 58.27 | 178.7 | 75.3 | f | Netball |
| 47 | 3.9 | 6.3 | 35.9 | 12.1 | 78 | 20.06 | 70 | 15.01 | 57.28 | 183.3 | 67.4 | f | Netball |
| 48 | 4.02 | 9.1 | 37.7 | 12.7 | 107 | 23.01 | 77 | 18.14 | 57.3 | 174.4 | 70 | f | Netball |
| 49 | 4.39 | 9.6 | 38.3 | 12.5 | 39 | 24.64 | 148.9 | 26.78 | 54.18 | 173.3 | 74 | f | Netball |
| 50 | 4.52 | 5.1 | 38.8 | 13.1 | 58 | 18.26 | 80.1 | 17.22 | 42.96 | 168.6 | 51.9 | f | Netball |
| 51 | 4.25 | 10.7 | 39.5 | 13.2 | 127 | 24.47 | 156.6 | 26.5 | 54.46 | 174 | 74.1 | f | Netball |
| 52 | 4.46 | 10.9 | 39.7 | 13.7 | 102 | 23.99 | 115.9 | 23.01 | 57.2 | 176 | 74.3 | f | Netball |
| 53 | 4.4 | 9.3 | 40.4 | 13.6 | 86 | 26.24 | 181.7 | 30.1 | 54.38 | 172.2 | 77.8 | f | Netball |
| 54 | 4.83 | 8.4 | 41.8 | 13.4 | 40 | 20.04 | 71.6 | 13.93 | 57.58 | 182.7 | 66.9 | f | Netball |
| 55 | 4.23 | 6.9 | 38.3 | 12.6 | 50 | 25.72 | 143.5 | 26.65 | 61.46 | 180.5 | 83.8 | f | Netball |
| 56 | 4.24 | 8.4 | 37.6 | 12.5 | 58 | 25.64 | 200.8 | 35.52 | 53.46 | 179.8 | 82.9 | f | Netball |
| 57 | 3.95 | 6.6 | 38.4 | 12.8 | 33 | 19.87 | 68.9 | 15.59 | 54.11 | 179.6 | 64.1 | f | Netball |
| 58 | 4.03 | 8.5 | 37.7 | 13 | 51 | 23.35 | 103.6 | 19.61 | 55.35 | 171.7 | 68.8 | f | Netball |
| 59 | 4.36 | 5.5 | 41.4 | 13.8 | 82 | 22.42 | 71.3 | 14.52 | 55.39 | 170 | 64.8 | f | Swim |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 60 | 4.07 | 5.9 | 39.5 | 13.3 | 25 | 20.42 | 54.6 | 11.47 | 52.23 | 170 | 59 | f | Swim |
| 61 | 4.17 | 4.9 | 38.9 | 12.9 | 86 | 22.13 | 88.2 | 17.71 | 59.33 | 180.5 | 72.1 | f | Swim |
| 62 | 4.23 | 8.1 | 38.2 | 12.7 | 22 | 25.17 | 95.4 | 18.48 | 61.63 | 173.3 | 75.6 | f | Swim |
| 63 | 4.46 | 8.3 | 42.2 | 14.4 | 30 | 23.72 | 47.5 | 11.22 | 63.39 | 173.5 | 71.4 | f | Swim |
| 64 | 4.38 | 5.8 | 42 | 14 | 27 | 21.28 | 55.6 | 13.61 | 60.22 | 181 | 69.7 | f | Swim |
| 65 | 4.31 | 5.3 | 41.1 | 13.9 | 60 | 20.87 | 62.9 | 12.78 | 55.73 | 175 | 63.9 | f | Swim |
| 66 | 4.51 | 5.1 | 40.9 | 14 | 115 | 19 | 52.5 | 11.85 | 48.57 | 170.3 | 55.1 | f | Swim |
| 67 | 4.13 | 7 | 39.7 | 13.1 | 124 | 22.04 | 62.6 | 13.35 | 51.99 | 165 | 60 | f | Swim |
| 68 | 4.48 | 9.5 | 36.5 | 13.3 | 54 | 20.12 | 49.9 | 11.77 | 51.17 | 169.8 | 58 | f | Field |
| 69 | 5.31 | 9.5 | 47.1 | 15.9 | 29 | 21.35 | 57.9 | 11.07 | 57.54 | 174.1 | 64.7 | f | T_400m |
| 70 | 4.58 | 5.8 | 42.1 | 14.7 | 164 | 28.57 | 109.6 | 21.3 | 68.86 | 175 | 87.5 | f | Field |
| 71 | 4.81 | 6.8 | 42.7 | 15.3 | 50 | 26.95 | 98.5 | 20.1 | 63.04 | 171.1 | 78.9 | f | Field |
| 72 | 4.51 | 9 | 39.7 | 14.3 | 36 | 28.13 | 136.3 | 24.88 | 63.03 | 172.7 | 83.9 | f | Field |
| 73 | 4.77 | 7.1 | 40.6 | 14.6 | 40 | 26.85 | 103.6 | 19.26 | 66.85 | 175.6 | 82.8 | f | Field |
| 74 | 5.33 | 9.3 | 47 | 15 | 62 | 25.27 | 102.8 | 19.51 | 59.89 | 171.6 | 74.4 | f | Field |
| 75 | 4.75 | 7.5 | 43.8 | 15.2 | 90 | 31.93 | 131.9 | 23.01 | 72.98 | 172.3 | 94.8 | f | Field |
| 76 | 4.11 | 7.3 | 38.7 | 12.4 | 12 | 16.75 | 33.8 | 8.07 | 45.23 | 171.4 | 49.2 | f | T_400m |
| 77 | 4.76 | 7.6 | 42.9 | 13.4 | 36 | 19.54 | 43.5 | 11.05 | 55.06 | 178 | 61.9 | f | T_Sprnt |
| 78 | 4.27 | 6.9 | 44.1 | 14.7 | 45 | 20.42 | 46.2 | 12.39 | 46.96 | 162 | 53.6 | f | T_400m |
| 79 | 4.44 | 6.1 | 42.6 | 13.9 | 43 | 22.76 | 73.9 | 15.95 | 53.54 | 167.3 | 63.7 | f | T_400m |
| 80 | 4.2 | 6.5 | 39.1 | 13 | 51 | 20.12 | 36.8 | 9.91 | 47.57 | 162 | 52.8 | f | T_400m |
| 81 | 4.71 | 6.9 | 43.5 | 13.8 | 22 | 22.35 | 67 | 16.2 | 54.63 | 170.8 | 65.2 | f | T_400m |
| 82 | 4.09 | 6.4 | 40.1 | 13.2 | 44 | 19.16 | 41.1 | 9.02 | 46.31 | 163 | 50.9 | f | T_400m |
| 83 | 4.24 | 6.6 | 38.2 | 12.6 | 26 | 20.77 | 59.4 | 14.26 | 49.13 | 166.1 | 57.3 | f | T_400m |
| 84 | 3.9 | 6 | 38.9 | 13.5 | 16 | 19.37 | 48.4 | 10.48 | 53.71 | 176 | 60 | f | T_400m |
| 85 | 4.82 | 7.6 | 43.2 | 14.4 | 58 | 22.37 | 50 | 11.64 | 53.11 | 163.9 | 60.1 | f | T_Sprnt |
| 86 | 4.32 | 6.8 | 40.6 | 13.7 | 46 | 17.54 | 54.6 | 12.16 | 46.12 | 173 | 52.5 | f | T_400m |
| 87 | 4.77 | 7.2 | 43.3 | 14.8 | 43 | 19.06 | 42.3 | 10.53 | 53.41 | 177 | 59.7 | f | T_400m |
| 88 | 5.16 | 8.2 | 45.3 | 14.7 | 34 | 20.3 | 46.1 | 10.15 | 51.48 | 168 | 57.3 | f | T_Sprnt |
| 89 | 4.97 | 7.8 | 44.7 | 14.2 | 41 | 20.15 | 46.3 | 10.74 | 53.2 | 172 | 59.6 | f | T_Sprnt |
| 90 | 4 | 4.2 | 36.6 | 12 | 57 | 25.36 | 109 | 20.86 | 56.58 | 167.9 | 71.5 | f | Tennis |
| 91 | 4.4 | 4 | 40.8 | 13.9 | 73 | 22.12 | 98.1 | 19.64 | 56.01 | 177.5 | 69.7 | f | Tennis |
| 92 | 4.38 | 7.9 | 39.8 | 13.5 | 88 | 21.25 | 80.6 | 17.07 | 46.52 | 162.5 | 56.1 | f | Tennis |
| 93 | 4.08 | 6.6 | 37.8 | 12.1 | 182 | 20.53 | 68.3 | 15.31 | 51.75 | 172.5 | 61.1 | f | Tennis |
| 94 | 4.98 | 6.4 | 44.8 | 14.8 | 80 | 17.06 | 47.6 | 11.07 | 42.15 | 166.7 | 47.4 | f | Tennis |
| 95 | 5.16 | 7.2 | 44.3 | 14.5 | 88 | 18.29 | 61.9 | 12.92 | 48.76 | 175 | 56 | f | Tennis |
| 96 | 4.66 | 6.4 | 40.9 | 13.9 | 109 | 18.37 | 38.2 | 8.45 | 41.93 | 157.9 | 45.8 | f | Tennis |
| 97 | 4.19 | 9 | 39 | 13.4 | 69 | 18.93 | 43.5 | 10.16 | 42.95 | 158.9 | 47.8 | f | Gym |
| 98 | 4.53 | 5 | 40.7 | 14 | 41 | 17.79 | 56.8 | 12.55 | 38.3 | 156.9 | 43.8 | f | Gym |
| 99 | 4.09 | 4.9 | 36 | 12.5 | 66 | 17.05 | 41.6 | 9.1 | 34.36 | 148.9 | 37.8 | f | Gym |
| 100 | 4.42 | 6.4 | 42.8 | 14.5 | 63 | 20.31 | 58.9 | 13.46 | 39.03 | 149 | 45.1 | f | Gym |
| 101 | 5.13 | 7.1 | 46.8 | 15.9 | 34 | 22.46 | 44.5 | 8.47 | 61 | 172.7 | 67 | m | Swim |
| 102 | 4.83 | 7.6 | 45.2 | 15.2 | 97 | 23.88 | 41.8 | 7.68 | 69 | 176.5 | 74.4 | m | Swim |
| 103 | 5.09 | 4.7 | 46.6 | 15.9 | 55 | 23.68 | 33.7 | 6.16 | 74 | 183 | 79.3 | m | Swim |
| 104 | 5.17 | 4.1 | 44.9 | 15 | 76 | 23.15 | 50.9 | 8.56 | 80 | 194.4 | 87.5 | m | Swim |
| 105 | 5.11 | 6.7 | 46.1 | 15.6 | 93 | 22.32 | 40.5 | 6.86 | 78 | 193.4 | 83.5 | m | Swim |
| 106 | 5.03 | 7.1 | 45.1 | 15.2 | 46 | 24.02 | 51.2 | 9.4 | 71 | 180.2 | 78 | m | Swim |
| 107 | 5.32 | 6 | 47.5 | 16.3 | 155 | 23.29 | 54.4 | 9.17 | 71 | 183 | 78 | m | Swim |
| 108 | 4.75 | 8.6 | 45.5 | 15.2 | 99 | 25.11 | 52.3 | 8.54 | 78 | 184 | 85 | m | Swim |
| 109 | 5.34 | 6.6 | 48.6 | 16.5 | 35 | 22.81 | 57 | 9.2 | 77 | 192.7 | 84.7 | m | Swim |
| 110 | 4.87 | 4.8 | 44.9 | 15.4 | 124 | 26.25 | 65.3 | 11.72 | 81 | 187.2 | 92 | m | Swim |
| 111 | 5.33 | 5.2 | 47.8 | 16.1 | 176 | 21.38 | 52 | 8.44 | 66 | 183.9 | 72.3 | m | Swim |
| 112 | 4.81 | 6.2 | 45.2 | 15.3 | 107 | 22.52 | 42.7 | 7.19 | 77 | 192 | 83 | m | Swim |
| 113 | 4.32 | 4.3 | 41.6 | 14 | 177 | 26.73 | 35.2 | 6.46 | 91 | 190.4 | 96.9 | m | Swim |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 114 | 4.87 | 8.2 | 43.8 | 15 | 130 | 23.57 | 49.2 | 9 | 78 | 190.7 | 85.7 | m | Row |
| 115 | 5.04 | 7.1 | 44 | 14.8 | 64 | 25.84 | 61.8 | 12.61 | 75 | 181.8 | 85.4 | m | Row |
| 116 | 4.4 | 5.3 | 42.5 | 14.5 | 109 | 24.06 | 46.5 | 9.03 | 78 | 188.3 | 85.3 | m | Row |
| 117 | 4.95 | 5.9 | 45.4 | 15.5 | 125 | 23.85 | 34.8 | 6.96 | 87 | 198 | 93.5 | m | Row |
| 118 | 4.78 | 9.3 | 43 | 14.7 | 150 | 25.09 | 60.2 | 10.05 | 78 | 186 | 86.8 | m | Row |
| 119 | 5.21 | 6.8 | 44.5 | 15.4 | 115 | 23.84 | 48.1 | 9.56 | 79 | 192 | 87.9 | m | Row |
| 120 | 5.22 | 8.4 | 47.5 | 16.2 | 89 | 25.31 | 44.5 | 9.36 | 79 | 185.6 | 87.2 | m | Row |
| 121 | 5.18 | 6.5 | 45.4 | 14.9 | 93 | 19.69 | 54 | 10.81 | 48 | 165.3 | 53.8 | m | Row |
| 122 | 5.4 | 6.8 | 49.5 | 17.3 | 183 | 26.07 | 44.7 | 8.61 | 82 | 185.6 | 89.8 | m | Row |
| 123 | 4.92 | 5.4 | 46.2 | 15.8 | 84 | 25.5 | 64.9 | 9.53 | 82 | 189 | 91.1 | m | Row |
| 124 | 5.24 | 7.5 | 46.5 | 15.5 | 70 | 23.69 | 43.8 | 7.42 | 82 | 193.4 | 88.6 | m | Row |
| 125 | 5.09 | 10.1 | 44.9 | 14.8 | 118 | 26.79 | 58.3 | 9.79 | 83 | 185.6 | 92.3 | m | Row |
| 126 | 4.83 | 5 | 43.8 | 15.1 | 61 | 25.61 | 52.8 | 8.97 | 88 | 194.6 | 97 | m | Row |
| 127 | 5.22 | 6 | 46.6 | 15.7 | 72 | 25.06 | 43.1 | 7.49 | 83 | 189 | 89.5 | m | Row |
| 128 | 4.71 | 8 | 45.5 | 15.6 | 91 | 24.93 | 78 | 11.95 | 78 | 188.1 | 88.2 | m | Row |
| 129 | 5.24 | 7.2 | 46.6 | 15.9 | 58 | 22.96 | 40.8 | 7.35 | 85 | 200.4 | 92.2 | m | B_Ball |
| 130 | 4.54 | 5.9 | 44.4 | 15.6 | 97 | 20.69 | 41.5 | 7.16 | 73 | 195.3 | 78.9 | m | B_Ball |
| 131 | 5.13 | 5.8 | 46.1 | 15.9 | 110 | 23.97 | 50.9 | 8.77 | 82 | 194.1 | 90.3 | m | B_Ball |
| 132 | 5 | 6.7 | 45.3 | 15.7 | 72 | 24.64 | 49.6 | 9.56 | 79 | 187.9 | 87 | m | B_Ball |
| 133 | 5.17 | 8 | 47.9 | 16.4 | 36 | 25.93 | 88.9 | 14.53 | 97 | 209.4 | 113.7 | m | B_Ball |
| 134 | 4.89 | 7.5 | 41.6 | 14.4 | 53 | 23.69 | 48.3 | 8.51 | 90 | 203.4 | 98 | m | B_Ball |
| 135 | 4.5 | 9.2 | 40.7 | 13.7 | 72 | 25.38 | 61.8 | 10.64 | 90 | 198.7 | 100.2 | m | B_Ball |
| 136 | 4.84 | 8.3 | 46.3 | 15.9 | 39 | 22.68 | 43 | 7.06 | 74 | 187.1 | 79.4 | m | B_Ball |
| 137 | 4.13 | 8.9 | 40.3 | 13.5 | 61 | 23.36 | 61.1 | 8.87 | 82 | 196.6 | 90.3 | m | B_Ball |
| 138 | 4.87 | 7.4 | 43.5 | 15 | 49 | 22.44 | 43.8 | 7.88 | 72 | 186.1 | 77.7 | m | B_Ball |
| 139 | 4.82 | 6.4 | 44.3 | 14.8 | 35 | 22.57 | 54.2 | 9.2 | 76 | 192.8 | 83.9 | m | B_Ball |
| 140 | 4.73 | 6.7 | 42.8 | 14.9 | 8 | 19.81 | 41.8 | 7.19 | 70 | 195.2 | 75.5 | m | B_Ball |
| 141 | 4.55 | 5.6 | 42.6 | 14.4 | 106 | 21.19 | 34.1 | 6.06 | 57 | 169.1 | 60.6 | m | T_400m |
| 142 | 4.71 | 7.2 | 43.6 | 14 | 32 | 20.39 | 30.5 | 5.63 | 67 | 186.6 | 71 | m | T_400m |
| 143 | 4.93 | 7.3 | 46.2 | 15.1 | 41 | 21.12 | 34 | 6.59 | 67 | 184.4 | 71.8 | m | T_400m |
| 144 | 5.21 | 7.5 | 47.5 | 16.5 | 20 | 21.89 | 46.7 | 9.5 | 70 | 187.3 | 76.8 | m | T_400m |
| 145 | 5.09 | 8.9 | 46.3 | 15.4 | 44 | 29.97 | 71.1 | 13.97 | 88 | 185.1 | 102.7 | m | Field |
| 146 | 5.11 | 9.6 | 48.2 | 16.7 | 103 | 27.39 | 65.9 | 11.66 | 83 | 185.5 | 94.2 | m | Field |
| 147 | 4.94 | 6.3 | 45.7 | 15.5 | 50 | 23.11 | 34.3 | 6.43 | 74 | 184.9 | 79 | m | Field |
| 148 | 4.87 | 6.3 | 45.8 | 16.1 | 41 | 21.75 | 34.6 | 6.99 | 62 | 175 | 66.6 | m | T_400m |
| 149 | 4.41 | 4.5 | 44.2 | 15 | 101 | 20.89 | 31.8 | 6 | 67 | 185.4 | 71.8 | m | T_400m |
| 150 | 4.86 | 3.9 | 44.9 | 15.4 | 73 | 22.83 | 34.5 | 6.56 | 70 | 181 | 74.8 | m | T_400m |
| 151 | 4.91 | 9 | 46.3 | 15.4 | 56 | 22.02 | 31 | 6.03 | 64 | 176 | 68.2 | m | T_400m |
| 152 | 4.93 | 7.3 | 45.2 | 15.8 | 74 | 20.07 | 32.6 | 6.33 | 58 | 176.2 | 62.3 | m | T_400m |
| 153 | 4.2 | 4.5 | 41.2 | 14.3 | 58 | 20.15 | 31.5 | 6.82 | 57 | 174 | 61 | m | T_400m |
| 154 | 5.1 | 6.1 | 45.3 | 14.9 | 87 | 21.24 | 32.6 | 6.2 | 73 | 191 | 77.5 | m | T_400m |
| 155 | 4.5 | 6.1 | 42.2 | 14.7 | 139 | 19.63 | 31 | 5.93 | 54 | 171 | 57.4 | m | T_400m |
| 156 | 4.89 | 5.8 | 45.5 | 15.6 | 82 | 23.58 | 28 | 5.8 | 67 | 174 | 71.4 | m | T_Sprnt |
| 157 | 5.13 | 4 | 44.1 | 15.2 | 87 | 21.65 | 33.7 | 6.56 | 66 | 180.2 | 70.3 | m | T_Sprnt |
| 158 | 4.88 | 4.3 | 45.6 | 15.5 | 80 | 25.17 | 30.3 | 6.76 | 75 | 178.5 | 80.2 | m | T_Sprnt |
| 159 | 5 | 8.2 | 46.8 | 14.7 | 67 | 23.25 | 38 | 7.22 | 78 | 190.3 | 84.2 | m | Field |
| 160 | 5.48 | 4.6 | 49.4 | 18 | 132 | 32.52 | 55.7 | 8.51 | 102 | 185 | 111.3 | m | Field |
| 161 | 5.93 | 6.4 | 49.1 | 16.1 | 43 | 22.59 | 37.5 | 7.72 | 74 | 189 | 80.7 | m | Field |
| 162 | 5.01 | 8.9 | 46 | 15.9 | 212 | 30.18 | 112.5 | 19.94 | 78 | 180.1 | 97.9 | m | Field |
| 163 | 5.48 | 6.2 | 48.2 | 16.3 | 94 | 34.42 | 82.7 | 13.91 | 106 | 189.2 | 123.2 | m | Field |
| 164 | 5.16 | 8.4 | 44.4 | 15.5 | 213 | 21.86 | 29.7 | 6.1 | 68 | 182.6 | 72.9 | m | T_Sprnt |
| 165 | 4.64 | 9 | 42.9 | 14.9 | 122 | 23.99 | 38.9 | 7.52 | 77 | 186 | 83 | m | T_Sprnt |
| 166 | 6.72 | 7.1 | 59.7 | 19.2 | 76 | 24.81 | 44.8 | 9.56 | 69 | 174.9 | 75.9 | m | T_Sprnt |
| 167 | 4.83 | 6.6 | 43.8 | 14.3 | 53 | 21.68 | 30.9 | 6.06 | 66 | 180.6 | 70.7 | m | T_400m |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 168 | 5.34 | 7.6 | 48.3 | 16.2 | 91 | 21.04 | 44 | 7.35 | 62 | 178.6 | 67.1 | m | T_400m |
| 169 | 5.13 | 4.6 | 45.3 | 16.8 | 36 | 23.12 | 37.5 | 6 | 65 | 173 | 69.2 | m | T_400m |
| 170 | 4.68 | 4.8 | 43 | 14.8 | 101 | 20.76 | 37.6 | 6.92 | 62 | 179.7 | 67.1 | m | T_400m |
| 171 | 5 | 5.2 | 45.1 | 15.1 | 184 | 23.13 | 31.7 | 6.33 | 66 | 174.6 | 70.5 | m | T_Sprnt |
| 172 | 4.99 | 7.2 | 41.4 | 14.9 | 44 | 22.35 | 36.6 | 5.9 | 67 | 178 | 70.8 | m | T_Sprnt |
| 173 | 5.49 | 5.9 | 47.7 | 15.9 | 66 | 22.28 | 48 | 8.84 | 65 | 178.5 | 71 | m | T_400m |
| 174 | 5.59 | 7.9 | 49.7 | 17.2 | 220 | 23.55 | 41.9 | 8.94 | 63 | 171.3 | 69.1 | m | T_Sprnt |
| 175 | 5.03 | 6.6 | 44.7 | 15.9 | 191 | 19.85 | 30.9 | 6.53 | 59 | 178 | 62.9 | m | T_400m |
| 176 | 5.5 | 6.4 | 48.1 | 16.5 | 40 | 26.51 | 52.8 | 9.4 | 86 | 189.1 | 94.8 | m | T_Sprnt |
| 177 | 5.11 | 9.3 | 45.4 | 15.8 | 189 | 24.78 | 43.2 | 8.18 | 87 | 195.4 | 94.6 | m | Field |
| 178 | 4.96 | 8.3 | 45.3 | 15.7 | 141 | 33.73 | 113.5 | 17.41 | 89 | 179.1 | 108.2 | m | Field |
| 179 | 5.01 | 8.9 | 46 | 15.9 | 212 | 30.18 | 96.9 | 18.08 | 80 | 180.1 | 97.9 | m | Field |
| 180 | 5.11 | 8.7 | 46.5 | 16.3 | 97 | 23.31 | 49.3 | 9.86 | 68 | 179.6 | 75.2 | m | Field |
| 181 | 5.69 | 10.8 | 50.5 | 18.5 | 53 | 24.51 | 42.3 | 7.29 | 69 | 174.7 | 74.8 | m | T_Sprnt |
| 182 | 4.63 | 9.1 | 42.1 | 14.4 | 126 | 25.37 | 96.3 | 18.72 | 77 | 192.7 | 94.2 | m | W_Polo |
| 183 | 4.91 | 10.2 | 45 | 15.2 | 234 | 23.67 | 56.5 | 10.12 | 68 | 179.3 | 76.1 | m | W_Polo |
| 184 | 4.95 | 7.5 | 44.5 | 15 | 50 | 24.28 | 105.7 | 19.17 | 77 | 197.5 | 94.7 | m | W_Polo |
| 185 | 5.34 | 10 | 46.8 | 16.2 | 94 | 25.82 | 100.7 | 17.24 | 71 | 182.7 | 86.2 | m | W_Polo |
| 186 | 5.16 | 12.9 | 47.6 | 15.6 | 156 | 21.93 | 56.8 | 9.89 | 72 | 190.5 | 79.6 | m | W_Polo |
| 187 | 5.29 | 12.7 | 48 | 16.2 | 124 | 23.38 | 75.9 | 13.06 | 74 | 191 | 85.3 | m | W_Polo |
| 188 | 5.02 | 6.1 | 43.6 | 14.8 | 87 | 23.07 | 52.8 | 8.84 | 68 | 179.6 | 74.4 | m | W_Polo |
| 189 | 5.01 | 9.8 | 46.5 | 15.8 | 97 | 25.21 | 47.8 | 8.87 | 85 | 192.6 | 93.5 | m | W_Polo |
| 190 | 5.03 | 7.5 | 43.6 | 14.4 | 102 | 23.25 | 76 | 14.69 | 75 | 194.1 | 87.6 | m | W_Polo |
| 191 | 5.25 | 7.4 | 47.3 | 15.8 | 55 | 22.93 | 61.2 | 8.64 | 78 | 193 | 85.4 | m | W_Polo |
| 192 | 5.08 | 8.5 | 46.3 | 15.6 | 117 | 26.86 | 75.6 | 14.98 | 86 | 193.9 | 101 | m | W_Polo |
| 193 | 5.04 | 6 | 45.9 | 15 | 52 | 21.26 | 43.3 | 7.82 | 69 | 187.7 | 74.9 | m | W_Polo |
| 194 | 4.63 | 14.3 | 44.8 | 15 | 133 | 25.43 | 49.5 | 8.97 | 79 | 185.3 | 87.3 | m | W_Polo |
| 195 | 5.11 | 7 | 47.7 | 15.8 | 214 | 24.54 | 70 | 11.63 | 80 | 191.5 | 90 | m | W_Polo |
| 196 | 5.34 | 6.2 | 49.8 | 17.2 | 143 | 27.79 | 75.7 | 13.49 | 82 | 184.6 | 94.7 | m | W_Polo |
| 197 | 4.86 | 8.9 | 46.9 | 15.8 | 65 | 23.58 | 57.7 | 10.25 | 68 | 179.9 | 76.3 | m | W_Polo |
| 198 | 4.9 | 7.6 | 45.6 | 16 | 90 | 27.56 | 67.2 | 11.79 | 82 | 183.9 | 93.2 | m | W_Polo |
| 199 | 5.66 | 8.3 | 50.2 | 17.7 | 38 | 23.76 | 56.5 | 10.05 | 72 | 183.5 | 80 | m | Tennis |
| 200 | 5.03 | 6.4 | 42.7 | 14.3 | 122 | 22.01 | 47.6 | 8.51 | 68 | 183.1 | 73.8 | m | Tennis |
| 201 | 4.97 | 8.8 | 43 | 14.9 | 233 | 22.34 | 60.4 | 11.5 | 63 | 178.4 | 71.1 | m | Tennis |
| 202 | 5.38 | 6.3 | 46 | 15.7 | 32 | 21.07 | 34.9 | 6.26 | 72 | 190.8 | 76.7 | m | Tennis |

# A4. Francis Galton e la regressione

L'espressione "regressione" è stato associato all'equazione della retta in seguito agli studi di Francis Galton[143]. Galton era cugino di Charles Darwin e dopo la lettura de "*L'origine delle specie*" si era convinto che la preminenza delle persone nei vari campi fosse dovuta essenzialmente a fattori genetici. Si era quindi dedicato alla ricerca delle leggi della genetica. E l'osservazione di fenomeni "di regressione" lo portò a concepire l'idea della necessità di opporvisi, applicando quelli che riteneva essere i principi di una "buona" genetica ("eugenetica").

In un suo famoso lavoro[144] Galton così si esprime:

"*... This memoir contains the data upon which the remarks on the Law of Regression were founded ... My object is to place beyond doubt the existence of a simple and far-reaching law that governs the hereditary transmission of, I believe, every one of those simple qualities which all possess, though in unequal degrees.*"

"*It is some years since I made an extensive series of experiments on the produce of seeds of different size but of the same species. They yielded results that seemed very noteworthy, and I used them as the basis of a lecture before the Royal Institution on February 9th, 1877. It appeared from these experiments that the offspring did not tend to resemble their parent seeds in size, but to be always more mediocre than they - to be smaller than the parents, if the parents were large; to be larger than the parents, if the parents were very small.*"

"*It was anthropological evidence that I desired, caring only for the seeds as means of throwing light on heredity in man ... inducing me to make an offer of prizes for Family Records, which was largely responded to, and furnished me last year with what I wanted ... An analysis of the Records fully confirms and goes far beyond the conclusions I obtained from the seeds. It gives the numerical value of* ***the regression towards mediocrity in the case of human stature***[145] *... My data consisted of the heights of 930 adult children and of their respective parentages, 205 in number...*"

A pagina 254 del lavoro Galton cita "*... a total of 928 children ...*" che corrisponde al numero delle coppie di valori riportati in **R** nel set di dati **galton**.

---

[143] *Francis Galton*. MacTutor History of Mathematics archive. School of Mathematics and Statistics. University of St Andrews, Scotland. URL consultato il 21/10/2018: https://goo.gl/BVMXGn

[144] Francis Galton. *Regression Towards Mediocrity in Hereditary Stature*. The Journal of the Anthropological Institute of Great Britain and Ireland, 1886, Vol. 15, pp. 246-263. URL consultato il 18/10/2018: https://goo.gl/VWvBau

[145] Il grassetto è mio.

# A5. Correlazione e causazione

Per ridurre la componente di incertezza nelle nostre decisioni, oltre che utilizzare metodi di ragionamento "razionali", è necessario acquisire dati e informazioni "utili". Oggi si sente spesso dire che ci si deve basare sull'evidenza. E i numeri sono per definizione portatori di "evidenza".

Ma cosa è l'evidenza? Vediamo un esempio di evidenza statistica impiegando variabili rilevate negli anni dal 1998 al 2002 dall'Istituto Nazionale di Statistica e dall'Istituto Superiore di Sanità.

La prima variabile è: *Procedimenti civili di primo grado pendenti a fine anno*.
Da "*ISTAT, Annuario statistico italiano 2004, pag. 136. Tavola 6.1 - Movimento dei procedimenti civili per grado di giudizio e ufficio giudiziario - Anni 1998-2002*".

La seconda variabile è: *Casi [di AIDS] diagnosticati*.
Da "*Notiziario dell'Istituto Superiore di Sanità, Supplemento 1-2007, Aggiornamento dei casi di AIDS notificati in Italia e delle nuove diagnosi di infezione da HIV, pag. 4. Tabella 1 - Distribuzione annuale dei casi di AIDS, dei casi corretti per ritardo di notifica, dei decessi e del tasso di letalità*".

I valori sono riportati in questa tabella

| *Anno* | *Procedimenti civili ... pendenti a fine anno (x)* | *Casi [di AIDS] diagnosticati (y)* |
|---|---|---|
| 1998 | 10376 | 2441 |
| 1999 | 9159 | 2135 |
| 2000 | 8290 | 1948 |
| 2001 | 7924 | 1812 |
| 2002 | 6872 | 1756 |

e i dati sono analizzati con questo **script** con il quale ci calcoliamo il coefficiente di correlazione **r** e la regressione lineare ponendo i procedimenti civili … pendenti a fine anno in ascisse e i casi [di AIDS] diagnosticati in ordinate:

```
#  r di Pearson ed equazione della retta y = a + b·x
#
x <- c(10376,9159,8290,7924,6872) # valori in ascisse
y <- c(2441,2135,1948,1812,1756) # valori in ordinate
mydata <- data.frame(x,y) # per la funzione cor()
mymatrix <- as.matrix(mydata) # per la funzione rcorr()
#
coefficients(lm(y ~ x)) # intercetta (a) e coefficiente angolare (b)
#
cor(mydata, method="pearson") # coefficiente di correlazione r
library(Hmisc) # carica il pacchetto
rcorr(mymatrix, type="pearson") # significatività di r
#
```

Questi sono i risultati:

```
> #  r di Pearson ed equazione della retta y = a + b·x
> #
> x <- c(10376,9159,8290,7924,6872) # valori in ascisse
> y <- c(2441,2135,1948,1812,1756) # valori in ordinate
> mydata <- data.frame(x,y) # per la funzione cor()
> mymatrix <- as.matrix(mydata) # per la funzione rcorr()
> #
> coefficients(lm(y ~ x)) # intercetta (a) e coefficiente angolare (b)
(Intercept)           x
270.6801141   0.2050304
> #
> cor(mydata, method="pearson") # coefficiente di correlazione r
          x         y
x 1.0000000 0.9748914
y 0.9748914 1.0000000
> library(Hmisc) # carica il pacchetto
Carico il pacchetto richiesto: lattice
Carico il pacchetto richiesto: survival
Carico il pacchetto richiesto: Formula
Carico il pacchetto richiesto: ggplot2

Attaching package: 'Hmisc'

The following objects are masked from 'package:base':

    format.pval, units

> rcorr(mymatrix, type="pearson") # significatività di r
     x    y
x 1.00 0.97
y 0.97 1.00

n= 5


P
  x      y
x        0.0048
y 0.0048
```

L'equazione della retta di regressione è `y = 270.6801141 + 0.2050304 · x` e il coefficiente di correlazione r è uguale a `0.9748914`. La probabilità di osservare per caso questo valore è `p = 0.0048` cioè è dello 0.48%. Anche se assumiamo un livello di significatività dell'1%, molto più prudente del classico 5%, possiamo concludere che tra le due variabili esiste una correlazione significativa.

Le fonti sono documentate e autorevoli. Ed esiste una "evidenza" suffragata dai numeri che le due variabili variano congiuntamente. Tuttavia l'evidenza dei numeri è in contrasto con la ragionevolezza: perché non abbiamo un modello che faccia un'ipotesi ragionevole su come all'aumentare dell'efficienza nell'amministrazione della giustizia civile la gente si ammali meno di AIDS. I due fatti sono "evidentemente" slegati, anche se dal punto di vista statistico "evidentemente" correlati.

Possiamo anche arricchire l'analisi statistica tracciando il grafico (**Fig. A5.1**) con questo script:

```
#
windows() # apre e inizializza una nuova finestra grafica
plot(x, y, xlim = c(6000,11000), ylim = c(1500,2500), xlab="Procedimenti civili ... prendenti a fine anno",
ylab="Casi [di AIDS] ... diagnosticati", main="Regressione x variabile indipendente") # grafico dei dati
abline(coefficients(lm(y ~ x)), col="lightblue") # retta di regressione x variabile indipendente
#
```

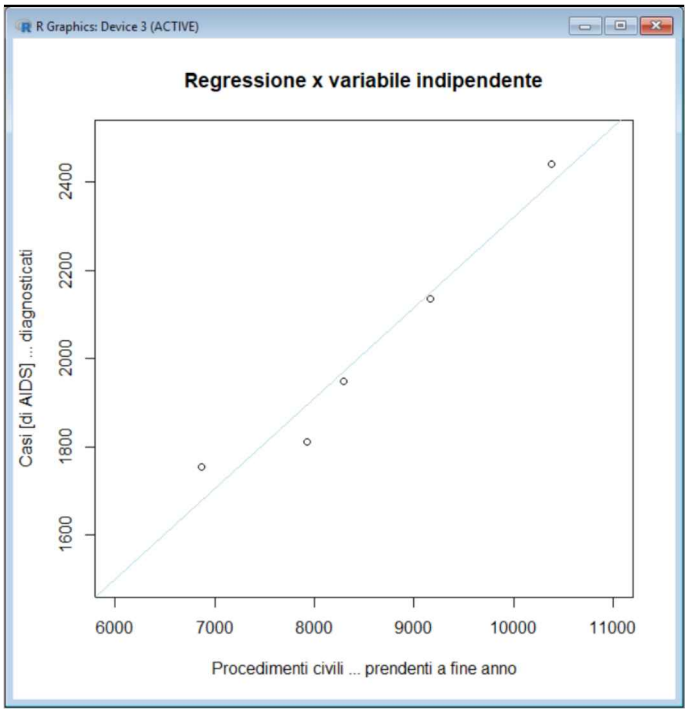



**Fig. A5.1** Due fatti tra loro scollegati possono apparire statisticamente correlati.

L'assunzione implicita di un coefficiente di correlazione significativo come riprova di un rapporto di causa-effetto è una delle trappole più subdole e più ricorrenti nella statistica. Richiami in tal senso si trovano tra l'altro in Marubini[146] e in Campbell[147], e a tutt'oggi esiste almeno un sito web interamente dedicato ad illustrare esempi grotteschi su questo tema[148].

---

[146] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp.89-90.
[147] "*Correlation is not causation*". In: Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 103.
[148] *Spurious correlations*. URL consultato il 20/10/2018: https://goo.gl/8ZJHNC

# A6. La regressione lineare: assunti e modelli

La **regressione lineare standard** viene trattata estesamente in tutti i testi di statica, e a questi si rimanda per gli assunti che sono posti alla base del modello (vedere la bibliografia).

Tuttavia esiste un aspetto relativo al calcolo della regressione lineare che viene spesso sottovalutato, ed è il fatto che fornisce risultati che dipendono da quale delle due variabili in esame sia considerata la variabile indipendente e quale sia considerata la variabile dipendente.

Si considerino i seguenti dati (simulati):

| Variabile 1 | Variabile 2 |
|---|---|
| 9 | 9 |
| 10 | 10 |
| 11 | 11 |
| 9 | 10 |
| 10 | 9 |
| 10 | 11 |
| 11 | 10 |

Non avendo alcuna ragione per assumere l'una o l'altra delle due variabili come la variabile indipendente:
→ calcoliamo l'equazione della retta di **regressione x variabile indipendente** assumendo come variabile indipendente la **Variabile 1**;
→ ricalcoliamo l'equazione assumendo questa volta come variabile indipendente la **Variabile 2**;
→ ricaviamo da quest'ultima l'equazione della retta di **regressione y variabile indipendente**;
→ infine calcoliamo l'equazione della retta di regressione con un metodo che non impiega nessuna delle due come variabile indipendente, ma media le due soluzioni (**componente principale standardizzata**).

```
# MODELLI DI REGRESSIONE PARAMETRICA retta y = a + b·x con dati simulati                    1/4
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
var_1 <- c(9,10,11,9,10,10,11) # variabile 1
var_2 <- c(9,10,11,10,9,11,10) # variabile 2
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 1", ylab="Variabile 2", main="Regressione x
variabile indipendente", cex.main = 0.9) # grafico dei dati
regpar <- lm(y ~ x) # regressione lineare standard
a_yx <- regpar$coefficients[1] # intercetta a
b_yx <- regpar$coefficients[2] # coefficiente angolare b
abline(a_yx, b_yx, col="lightblue") # retta di regressione x variabile indipendente
c(a_yx, b_yx) # mostra a e b
#
```

Il risultato di questa prima parte dello script è il primo in alto a sinistra dei grafici della **Fig. A6.1**, nella quale sul grafico di dispersione dei dati vediamo tracciata la retta di **regressione x variabile indipendente**:

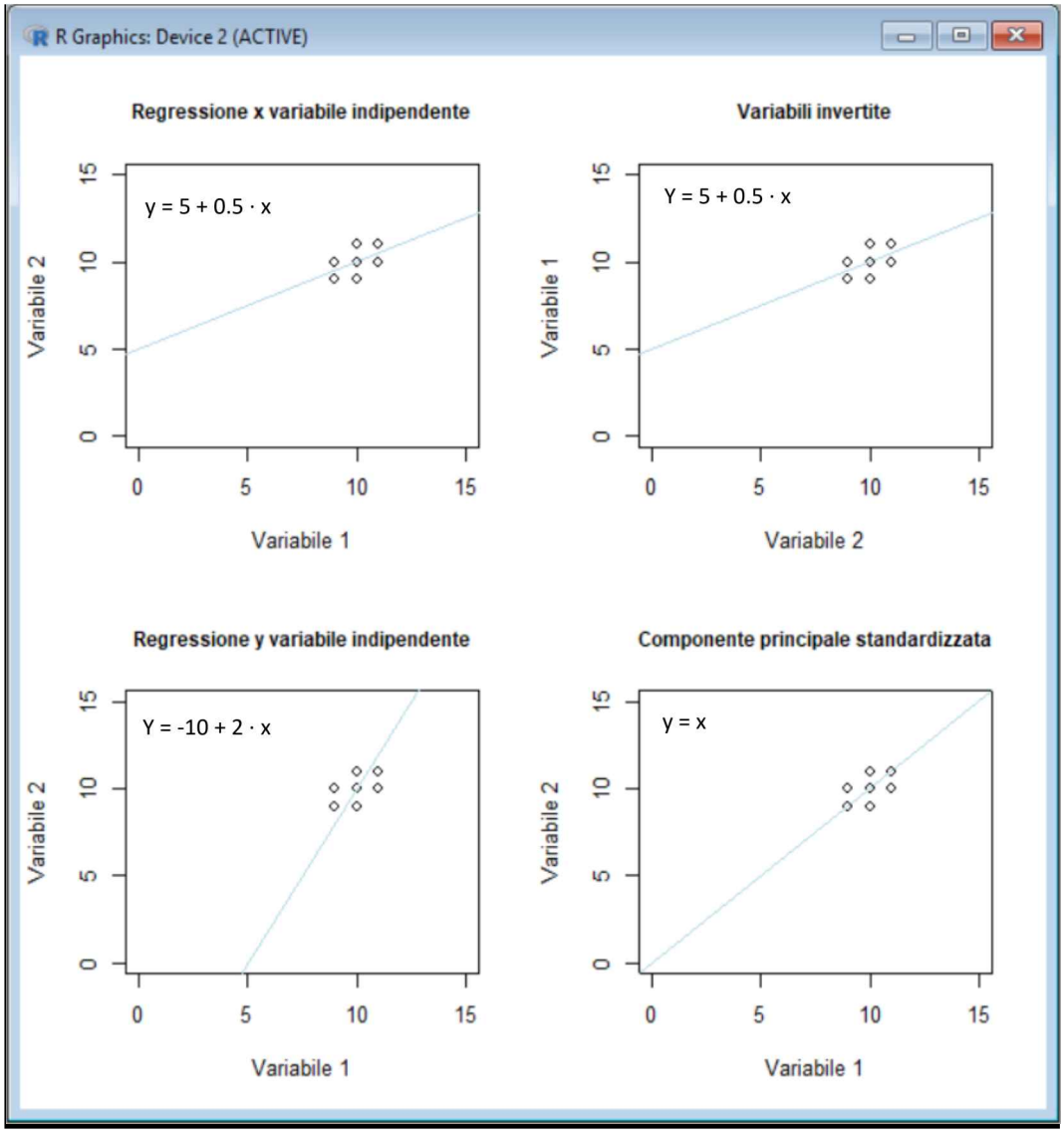



**Fig. A6.1** Tre modelli di regressione lineare applicati a dati simulati.

A questo punto scambiamo tra di loro le due variabili e ricalcoliamo l'equazione della retta di regressione con questa seconda parte dello script:

```
x <- var_2 # variabile 2 in ascisse                                    2/4
y <- var_1 # variabile 1 in ordinate
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 2", ylab="Variabile 1", main="Variabili invertite",
cex.main = 0.9) # grafico dei dati
regpar_1 <- lm(y ~ x) # regressione lineare standard
a_yx_1 <- regpar_1$coefficients[1] # intercetta a
```

```
b_yx_1 <- regpar_1$coefficients[2] # coefficiente angolare b
abline(a_yx_1, b_yx_1, col="lightblue") # retta di regressione con variabili scambiate
c(a_yx_1, b_yx_1) # mostra a e b
#
```

Ora la **Variabile 2** è riportata in ascisse e la **Variabile 1** è riportata in ordinate (**Fig. A6.1**, grafico in alto a destra). L'equazione della retta di regressione è la stessa (questo è atteso visto che **Variabile 1** e **Variabile 2** comprendono identici valori). Per confrontare questo grafico e questa retta di regressione con quelli del primo grafico (**regressione x variabile indipendente**) dobbiamo capovolgere il grafico orizzontalmente, quindi ruotarlo a destra di 90°, riportando così la **Variabile 2** (la variabile indipendente) in ordinate e riportando la **Variabile 1** in ascisse, cosa che facciamo con questa terza parte dello **script**:

```
x <- var_1 # variabile 1 in ascisse                                    3/4
y <- var_2 # variabile 2 in ordinate
a_xy <- - a_yx_1 / b_yx_1 # intercetta a
b_xy <- 1 / b_yx_1 # coefficiente angolare b
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 1", ylab="Variabile 2", main="Regressione y
variabile indipendente", cex.main = 0.9) # grafico dei dati
abline(a_xy, b_xy, col="lightblue") # retta di regressione y variabile indipendente
c(a_xy, b_xy) # mostra a e b
#
```

Abbiamo così ottenuta la **regressione y variabile indipendente**, illustrata nella **Fig. A6.1** in basso a sinistra.

Infine con l'ultima parte dello **script** calcoliamo l'equazione della retta di regressione espressa come **componente principale standardizzata** (**Fig. A6.1** in basso a destra):

```
x <- var_1 # variabile 1 in ascisse                                    4/4
y <- var_2 # variabile 2 in ordinate
b_cps <- sqrt(b_yx*b_xy) # coefficiente angolare b
a_cps <- mean(y) - (b_cps * (mean(x))) # intercetta a
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 1", ylab="Variabile 2", main="Componente
principale standardizzata", cex.main = 0.9) # grafico dei dati
abline(a_cps, b_cps, col="lightblue") # retta di regressione componente principale standardizzata
c(a_cps, b_cps) # mostra a e b
#
```

La **componente principale standardizzata** con la salomonica conclusione che $y = x$ fornisce una soluzione intermedia che appare logica e più più accettabile rispetto all'una e all'altra regressione lineare standard.

Nel caso del set di dati **galton** di **R**, visto in precedenza, e ricavato dagli studi di Francis Galton che abbiamo citato, ci troviamo proprio in una situazione di questo genere. Nell'analizzare mediante la regressione lineare le altezze dei genitori e le altezze dei figli non vi sono ragioni per assumere le prime come variabile indipendente. Ma non vi sono neppure ragioni per assumere le altezze dei figli come variabile indipendente.

```
# MODELLI DI REGRESSIONE PARAMETRICA retta y = a + b·x set di dati galton
#
library(psychTools) # carica il pacchetto psych incluso il set di dati galton
#
```

```
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
var_1 <- galton$parent # variabile 1
var_2 <- galton$child # variabile 2
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli",
main="Regressione x variabile indipendente", cex = 0.5, cex.main = 0.9) # grafico dei dati
regpar <- lm(y ~ x) # regressione lineare standard
a_yx <- regpar$coefficients[1] # intercetta a
b_yx <- regpar$coefficients[2] # coefficiente angolare b
abline(a_yx, b_yx, col="lightblue") # retta di regressione x variabile indipendente
c(a_yx, b_yx) # visualizza a e b
#
x <- var_2 # variabile 2 in ascisse
y <- var_1 # variabile 1 in ordinate
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei figli", ylab="Altezza dei padri", main="Variabili
invertite", cex = 0.5, cex.main = 0.9) # grafico dei dati
regpar_1 <- lm(y ~ x) # regressione lineare standard
a_yx_1 <- regpar_1$coefficients[1] # intercetta a
b_yx_1 <- regpar_1$coefficients[2] # coefficiente angolare b
abline(a_yx_1, b_yx_1, col="lightblue") # retta di regressione con variabili scambiate
c(a_yx_1, b_yx_1) # visualizza a e b
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
a_xy <- - a_yx_1 / b_yx_1 # intercetta a
b_xy <- 1 / b_yx_1 # coefficiente angolare b
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli",
main="Regressione y variabile indipendente", cex = 0.5, cex.main = 0.9) # grafico dei dati
abline(a_xy, b_xy, col="lightblue") # retta di regressione y variabile indipendente
c(a_xy, b_xy) # visualizza a e b
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
b_cps <- sqrt(b_yx * b_xy) # coefficiente angolare b
a_cps <- mean(y) - (b_cps * (mean(x))) # intercetta a
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli",
main="Componente principale standardizzata", cex = 0.5, cex.main = 0.9) # grafico dei dati
abline(a_cps, b_cps, col="lightblue") # retta di regressione componente principale standardizzata
c(a_cps, b_cps) # visualizza a e b
#
```

Possiamo allora effettuare un'analisi di questi dati provando a calcolare per essi oltre alla **regressione x variabile indipendente**, la **regressione y variabile indipendente** e la **componente principale standardizzata** con lo **script** riportato qui sopra, e visualizzarle (**Fig. A6.2**).

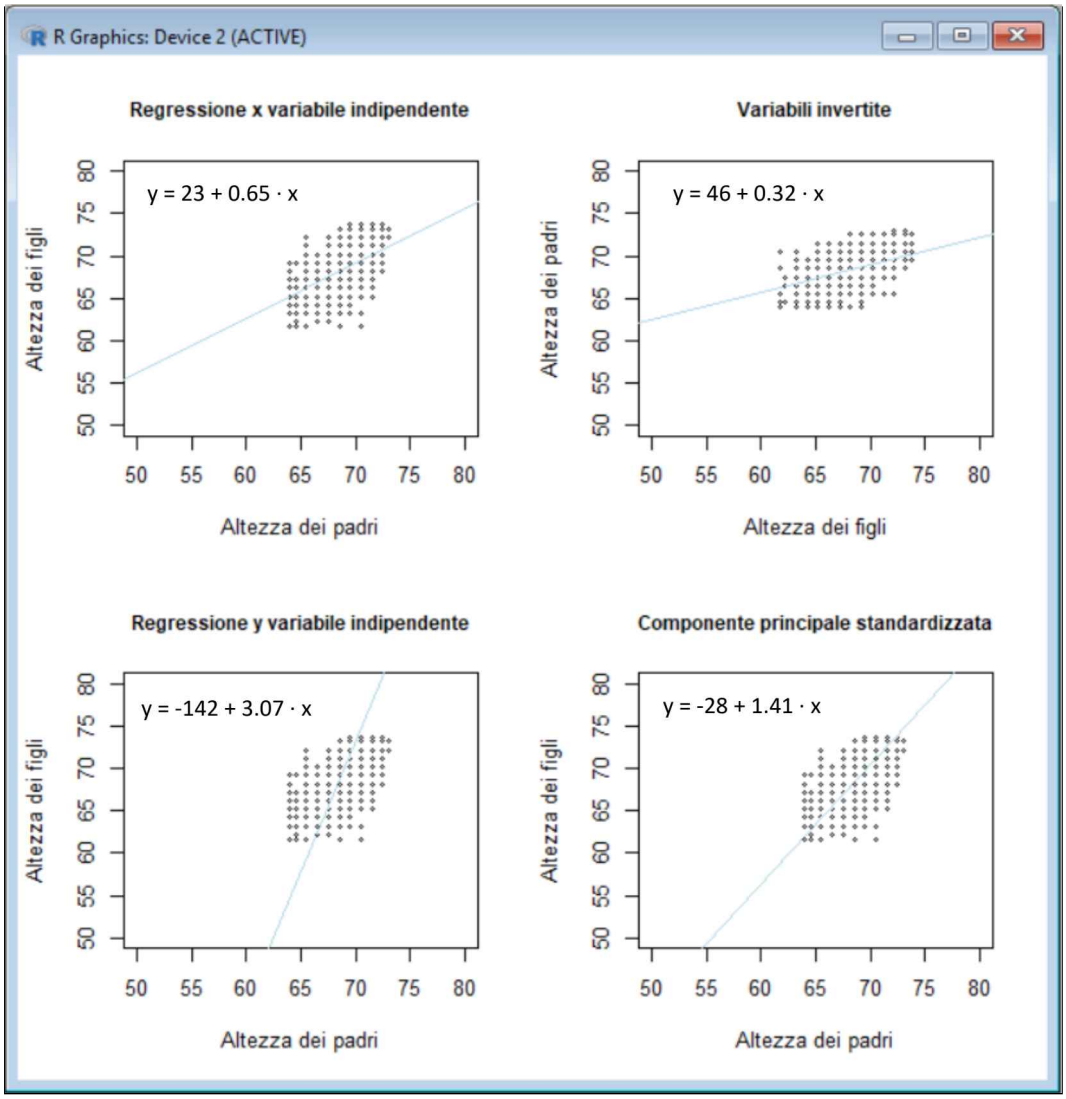



**Fig. A6.2** Tre modelli di regressione lineare parametrica applicati al set di dati **galton**.

Come si vede chiaramente assumendo come variabile indipendente i padri o i figli i risultati cambiano. Il problema, generato a scopo didattico nei dati simulati analizzati con il primo script e riportati nella **Fig. A6.1**, e che si presenta spontaneamente nei dati di Francis Galton riportati nella **Fig. A6.2**, nasce dal fatto che l'equazione della retta di regressione è determinata dall'interazione di due fattori:
→ gli **assunti** alla base del modello di regressione lineare impiegato;
→ l'**informazione** che il modello di regressione lineare può ricavare dai dati elaborati.

Più i dati sono allineati su una retta:
→ più consistente è l'informazione in essi contenuta;
→ più **r** si avvicina a 1;
→ meno rilevante è l'influenza degli assunti alla base del modello di regressione sulle conclusioni. Tanto che nel caso limite di dati perfettamente allineati su una retta (situazione dalla quale si allontanano molto i dati dei due esempi riportati), **r** è uguale a 1 e tutti i modelli di regressione forniscono lo stesso identico risultato, indipendentemente dagli assunti alla base del modello di regressione.

Meno i dati sono allineati su una retta:
→ meno consistente è l'informazione in essi contenuta;
→ meno r è vicino a 1 (più r si avvicina a 0);
→ più rilevante è l'influenza degli assunti alla base del modello di regressione sulle conclusioni. Tanto che nel caso limite di dati perfettamente distribuiti in una palla rotonda (situazione alla quale si avvicinano abbastanza i dati dei due esempi), **r** è uguale a 0 (zero) e i modelli di regressione forniscono risultati divergenti al massimo grado, risultati che oramai prescindono dalla informazione contenuta nei dati (che è nulla: manca l'informazione necessaria per decidere come far passare una retta da punti così distribuiti) e riflettono oramai esclusivamente gli assunti alla base del modello di regressione.

I dati di Galton presentano un ambito di valori ristretto. Il coefficiente di correlazione **r** calcolato in precedenza è uguale a `0.4587624` e il coefficiente di determinazione $\mathbf{R}^2$ di 0.21 indica che solamente il 21% della variabilità osservata viene spiegato dalla retta. Ci troviamo di fronte a un caso tipico di dati che non forniscono una informazione adeguata al tracciamento di una retta dalla quale trarre conclusioni "affidabili". Non potendo inoltre assumere né le altezze dei padri né le altezze dei figli come variabile indipendente, è forse opportuno accettare le conclusioni tratte dal calcolo della (retta di regressione espressa come) **componente principale standardizzata**, che delle tre regressioni sembra la meglio allineata con l'asse maggiore dell'ellissoide che include i dati (**Fig. A6.2** grafico in basso a destra).

Qui di seguito sono specificati i dettagli che portano al calcolo dell'intercetta `a` e del coefficiente angolare `b` delle tre diverse rette di regressione.

Nella **regressione lineare x variabile indipendente p**er adattare la retta ai dati sperimentali viene impiegato il metodo dei minimi quadrati, una tecnica di approssimazione ben nota, che consente di minimizzare la somma dei quadrati delle differenze che residuano fra i punti sperimentali e la retta. Il modello matematico impiegato presuppone si basa sui seguenti assunti:
→ la `x`, cioè la variabile indipendente, è misurata senza errore;
→ l'errore con cui si misura la `y` è distribuito normalmente e con una varianza che non cambia al variare del valore della x.

Sia allora `n` il numero dei punti aventi coordinate note $(x_i, y_i)$, e siano $\bar{x}$ la media dei valori delle $x_i$ e $\bar{y}$ la media dei valori delle $y_i$: il coefficiente angolare $b_{yx}$ e l'intercetta $a_{yx}$ dell'equazione della retta di regressione x variabile indipendente

$$y = a_{yx} + b_{yx} \cdot x$$

che meglio approssima (in termini di minimi quadrati) i dati vengono calcolati rispettivamente come

$$b_{yx} = \Sigma(x_i - \bar{x}) \cdot (y_i - \bar{y}) \; / \; \Sigma(x_i - \bar{x})^2$$

$$a_{yx} = \bar{y} - b_{yx} \cdot \bar{x}$$

Nella **regressione lineare y variabile indipendente** il modello matematico impiegato si basa sui seguenti assunti:
→ la `y`, cioè la variabile indipendente, è misurata senza errore
→ l'errore con cui si misura la `x` è distribuito normalmente e con una varianza che non cambia al variare del valore della `y`.

Si noti che per effettuare i calcoli la `y` (variabile indipendente) viene posta in ascisse e la `x` (variabile dipendente) viene posta in ordinate e successivamente, per sovrapporre dati e retta a quelli della regressione x variabile indipendente, le coordinate `x` e `y` sono scambiate tra di loro.

Sia allora $n$ il numero dei punti aventi coordinate note $(x_i, y_i)$, e siano $\bar{x}$ la media dei valori delle $x_i$ e $\bar{y}$ la media dei valori delle $y_i$: il coefficiente angolare $b_{xy}$ e l'intercetta $a_{xy}$ dell'equazione della retta di regressione y variabile indipendente

$$x = a_{xy} + b_{xy} \cdot y$$

che meglio approssima (in termini di minimi quadrati) i dati vengono calcolati rispettivamente come

$$b_{xy} = \Sigma (x_i - \bar{x}) \cdot (y_i - \bar{y}) / \Sigma(y_i - \bar{y})^2$$

$$a_{xy} = \bar{x} - b_{xy} \cdot \bar{y}$$

Per riportare i dati sullo stesso sistema di coordinate cartesiane utilizzato per la regressione x variabile indipendente, si esplicita l'equazione della retta di regressione $y$ variabile indipendente

$$x = a_{xy} + b_{xy} \cdot y$$

rispetto alla $y$ ottenendo

$$x - a_{xy} = b_{xy} \cdot y$$

e quindi, dividendo entrambi i membri per $b_{xy}$

$$y = - a_{xy} / b_{xy} + 1 / b_{xy} \cdot x$$

Quindi l'intercetta $a$ e il coefficiente angolare $b$ che consentono di rappresentare la regressione y variabile indipendente sullo stesso sistema di coordinate cartesiane della regressione x variabile indipendente saranno rispettivamente uguali a

$$a = - a_{xy} / b_{xy}$$
$$b = 1 / b_{xy}$$

Nella **componente principale standardizzata** il modello matematico impiegato presuppone tanto la x quanto la y siano affette da un errore di misura equivalente.

Sia allora $n$ il numero dei punti aventi coordinate note $(x_i, y_i)$, siano $\bar{x}$ la media dei valori delle $x_i$ e $\bar{y}$ la media dei valori delle $y_i$, sia $b_{yx}$ il coefficiente angolare dell'equazione della retta di regressione x variabile indipendente, e sia $b_{xy}$ il coefficiente angolare dell'equazione della retta di regressione y variabile indipendente. Il coefficiente angolare $b_{cps}$ dell'equazione della retta di regressione calcolata come componente principale standardizzata è allora uguale a

$$b_{cps} = \sqrt{} (b_{yx} \cdot b_{xy})$$

cioè alla media geometrica tra il coefficiente angolare $b_{yx}$ della regressione x variabile indipendente e il coefficiente angolare $b_{xy}$ della regressione y variabile indipendente, cioè

$$b_{xy} = \sqrt{}(\Sigma(y_i - \bar{y})^2 / \Sigma(x_i - \bar{x})^2)$$

mentre l'intercetta $a_{cps}$ dell'equazione della retta di regressione calcolata come componente principale standardizzata è uguale a

$$a_{cps} = \bar{y} - b_{cps} \cdot \bar{x}$$

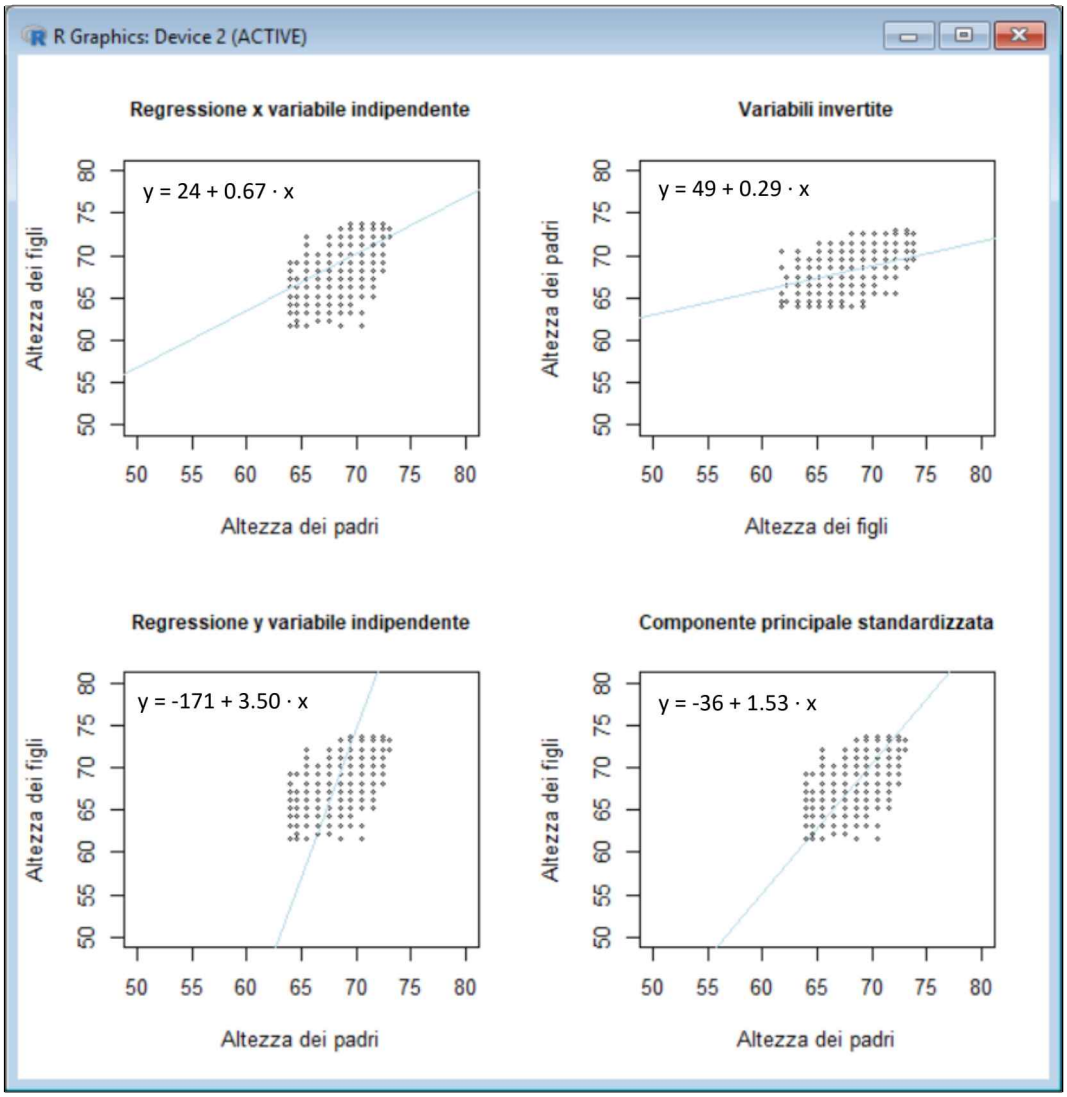



**Fig. A6.3** La regressione lineare non parametrica applicati al set di dati **galton**.

I risultati non cambiano ricalcolando le equazioni delle retta con un metodo non parametrico (**Fig. A6.3**) con questo **script** (per eseguirlo è necessario scaricare dal **CRAN** il pacchetto **mblm**.):

```
# REGRESSIONE NON PARAMETRICA retta y = a + b·x set di dati galton
#
library(psychTools) # carica il pacchetto psych incluso il set di dati galton
library(mblm) # carica il pacchetto per la regressione lineare non parametrica
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
var_1 <- galton$parent # variabile 1
var_2 <- galton$child # variabile 2
#
```

```
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli", main="Regressione x variabile indipendente", cex = 0.5, cex.main = 0.9) # grafico dei dati
regnon = mblm(y ~ x, repeated=TRUE) # metodo di Siegel
a_yx <- regnon$coefficients[1] # intercetta a
b_yx <- regnon$coefficients[2] # coefficiente angolare b
abline(a_yx, b_yx, col="lightblue") # retta di regressione x variabile indipendente
c(a_yx, b_yx) # visualizza a e b
#
x <- var_2 # variabile 2 in ascisse
y <- var_1 # variabile 1 in ordinate
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei figli", ylab="Altezza dei padri", main="Variabili invertite", cex = 0.5, cex.main = 0.9) # grafico dei dati
regnon_1 = mblm(y ~ x, repeated=TRUE) # metodo di Siegel
a_yx_1 <- regnon_1$coefficients[1] # intercetta a
b_yx_1 <- regnon_1$coefficients[2] # coefficiente angolare b
abline(a_yx_1, b_yx_1, col="lightblue") # retta di regressione con variabili scambiate
c(a_yx_1, b_yx_1) # visualizza a e b
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
a_xy <- - a_yx_1 / b_yx_1 # intercetta a
b_xy <- 1 / b_yx_1 # coefficiente angolare b
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli", main="Regressione y variabile indipendente", cex = 0.5, cex.main = 0.9) # grafico dei dati
abline(a_xy, b_xy, col="lightblue") # retta di regressione y variabile indipendente
c(a_xy, b_xy) # visualizza a e b
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
b_cps <- sqrt(b_yx * b_xy) # coefficiente angolare b
a_cps <- median(y) - (b_cps * (median(x))) # intercetta a
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli", main="Componente principale standardizzata", cex = 0.5, cex.main = 0.9) # grafico dei dati
abline(a_cps, b_cps, col="lightblue") # retta di regressione componente principale standardizzata
c(a_cps, b_cps) # visualizza a e b
#
```

Dei due modelli di regressione non parametrica che sono inclusi nel pacchetto **mblm** viene qui impiegata la regressione non parametrica secondo Siegel, già vista al termine del paragrafo sulla regressione lineare. Ma può essere impiegato in alternativa il modello di Theil-Sen mettendo l'argomento **repeated=FALSE** nella funzione **mblm()**.

Le differenze con lo script precedente che calcolava la regressione lineare parametrica sono limitate a:
→ **library(mblm)** che carica il pacchetto per la regressione lineare non parametrica;
→ **mblm(y ~ x, repeated=TRUE)** per effettuare il calcolo della regressione non parametrica secondo Siegel;
→ **regpar** che diventa ovunque **regnon**;
→ **mean()** che diventa **median()** nella quart'ultima riga di codice.

I risultati della regressione non parametrica confermano quelli della regressione parametrica e le relative conclusioni. Il tutto fa dubitare del fatto che in questi dati si possa trovare una “evidenza” di quel fenomeno

di “regressione” dei caratteri ereditati che Francis Galton aveva portato a supporto della necessità dell'eugenetica e che ha lasciato traccia nella denominazione della tecnica statistica.

# A7. Indice di massa corporea (BMI)

L'indice di massa corporea (BMI[149]) viene calcolato come rapporto tra il peso espresso in kg e il quadrato dell'altezza espressa in metri, ed è pertanto espresso in kg /$m^2$. I valori di BMI sono classificati dal punto di vista medico come segue:

| Situazione peso | Min (kg) | Max (kg) |
|---|---|---|
| Obesità di III classe (gravissima) | > 40,00 | |
| Obesità di II classe (grave) | 35,01 | 40 |
| Obesità di I classe (moderata) | 30,01 | 35 |
| Sovrappeso | 25,01 | 30 |
| Regolare | 18,51 | 25 |
| Leggermente sottopeso | 17,51 | 18,5 |
| Sottopeso | 16,01 | 17,5 |
| Grave magrezza (inedia) | < 16,01 | |

Questi sono i risultati dell'indagine **EHIS 2015** come riportati dall'Istat[150].

**Tavola 6.1 - Persone di 18 anni e più per Indice di Massa Corporea (in classi), per paese europeo. Anno 2015** (a)

*(per 100 persone con le stesse caratteristiche)*

| PAESI | INDICE MASSA CORPOREA | | | | Totale |
|---|---|---|---|---|---|
| | Sottopeso | Normale | Sovrappeso | Obeso | |
| **Italia** | **3,3** | **51,9** | **34,1** | **10,8** | **100** |
| **Unione europea (28 paesi)** | **2,3** | **46,1** | **35,7** | **15,9** | **100** |
| Austria | 2,4 | 49,6 | 33,3 | 14,7 | 100 |
| Belgio | 2,7 | 48,0 | 35,3 | 14,0 | 100 |
| Bulgaria | 2,2 | 43,8 | 39,2 | 14,8 | 100 |
| Cipro | 3,9 | 47,8 | 33,8 | 14,5 | 100 |
| Croazia | 1,9 | 40,7 | 38,7 | 18,7 | 100 |
| Danimarca | 2,2 | 50,0 | 32,9 | 14,9 | 100 |
| Estonia | 2,2 | 43,9 | 33,5 | 20,4 | 100 |
| Finlandia | 1,2 | 44,1 | 36,4 | 18,3 | 100 |
| Francia | 3,2 | 49,6 | 31,9 | 15,3 | 100 |
| Germania | 1,8 | 46,1 | 35,2 | 16,9 | 100 |
| Grecia | 1,9 | 41,3 | 39,4 | 17,3 | 100 |
| Irlanda | 1,9 | 42,3 | 37,0 | 18,7 | 100 |
| Lettonia | 1,7 | 41,8 | 35,2 | 21,3 | 100 |
| Lituania | 1,9 | 42,5 | 38,3 | 17,3 | 100 |
| Lussemburgo | 2,8 | 49,3 | 32,4 | 15,6 | 100 |
| Malta | 2,0 | 37,0 | 35,0 | 26,0 | 100 |
| Olanda | 1,6 | 49,0 | 36,0 | 13,3 | 100 |
| Polonia | 2,4 | 42,9 | 37,5 | 17,2 | 100 |
| Potogallo | 1,8 | 44,6 | 36,9 | 16,6 | 100 |
| Regno Unito | 2,1 | 42,2 | 35,6 | 20,1 | 100 |
| Repubblica Ceca | 1,1 | 42,1 | 37,6 | 19,3 | 100 |
| Romania | 1,3 | 42,9 | 46,4 | 9,4 | 100 |
| Slovacchia | 2,1 | 43,6 | 38,0 | 16,3 | 100 |
| Slovenia | 1,6 | 41,8 | 37,4 | 19,2 | 100 |
| Spagna | 2,2 | 45,4 | 35,7 | 16,7 | 100 |
| Svezia | 1,8 | 48,3 | 35,9 | 14,0 | 100 |
| Ungheria | 2,9 | 41,9 | 34,0 | 21,2 | 100 |

*Fonte:* Indagine "European Health Interview Survey" - Anno 2015; Eurostat database http://ec.europa.eu/eurostat/data/database.

[149] Anche in Italia viene usualmente impiegato l'acronimo **BMI** derivato dall'inglese **B**ody **M**ass **I**ndex.

[150] *Prevenzione e stili di vita in Italia e nell'Unione Europea. Indagine EHIS 2015. Tavole*. Tavola 6.1 bmi in europa. URL consultato il 18/10/2018: https://goo.gl/ZxGi9U

# A8. Teorema di Bayes e diagnosi medica

Per affrontare il tema sono necessarie alcune definizioni. Innanzitutto la **probabilità marginale** che si verifichi l'evento A è

$$P(A)$$

Dal punto di vista numerico la probabilità di un evento è un numero positivo compreso tra 0 (evento che non accade mai) e 1 (evento certo) ovvero

$$0 \leq P(A) \leq 1$$

*Corollario*: la probabilità che non si verifichi l'evento A è

$$P(nonA) = 1 - P(A)$$

La **probabilità condizionata**

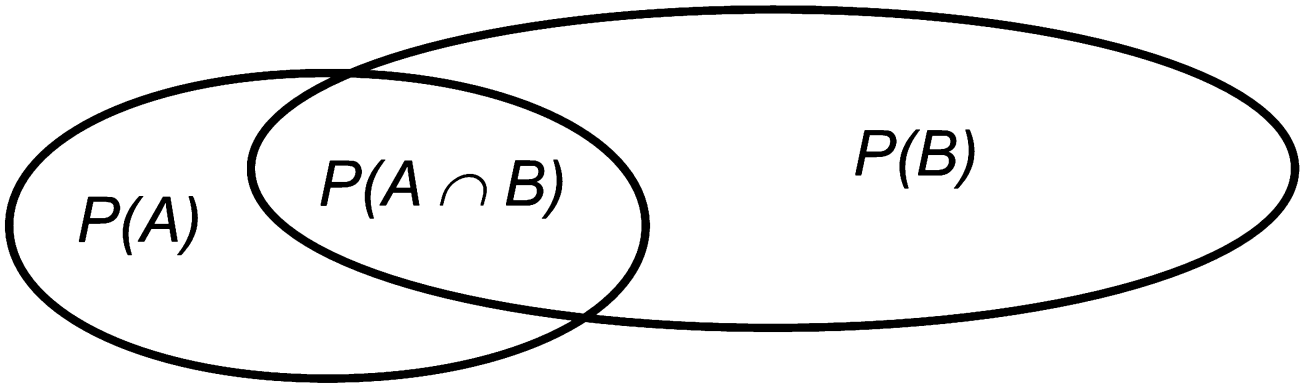
$$P(A|B)$$

è la probabilità di un evento A condizionata_ad un evento B, ovvero è la **probabilità che si verifichi A a condizione che si sia verificato B** e si legge "*la probabilità di A, dato B*" ovvero "*la probabilità di A condizionata a B*"

La **probabilità congiunta**

$$P(A \cap B)$$

è la probabilità di due eventi congiunti, ovvero è la **probabilità che si verifichino sia A sia B**, e si legge "*la probabilità congiunta di A e B*" ovvero "la *probabilità di A e B*". La probabilità congiunta può essere meglio compresa facendo riferimento al diagramma seguente.




La relazione fra probabilità condizionata e probabilità congiunta è la seguente:

$$P(A \cap B) = P(A|B) \cdot P(B) \qquad \text{(i)}$$

ma deve essere anche

$$P(A \cap B) = P(B|A) \cdot P(A) \qquad \text{(ii)}$$

(perché, intuitivamente, dobbiamo avere lo stesso risultato sia partendo da A sia partendo da B) e combinando la (i) con la (ii) si ottiene il **teorema delle probabilità composte**

$$P(A \cap B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

dal quale si ricava il **teorema di Bayes**[151]:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Se poniamo *A = la causa* e *B = l'effetto*, il teorema di Bayes suona così

$$P(\text{della causa}|\text{dato l'effetto}) = \frac{P(\text{dell'effetto}|\text{data la causa}) \times P(\text{della causa})}{P(\text{dell'effetto})}$$

Se consideriamo il problema specifico della diagnosi medica, nel caso di una malattia (la causa) che può essere presente (soggetti malati, M+) o assente (soggetti sani, M-), e di un segno (effetto) determinato da questa malattia, come è tipicamente il risultato di un test per la sua diagnosi, che può essere positivo (T+) o negativo (T-), il teorema di Bayes può essere riscritto come

$$P(M+|T+) = \frac{P(T+|M+) \cdot P(M+)}{P(T+)} \qquad \text{(iii)}$$

Il teorema di Bayes consente di calcolare il **valore predittivo di un test positivo** *P(M+|T+)* ovvero la probabilità di essere malato per un soggetto che presenta un test positivo, partendo dalla probabilità condizionata

| | Malattia + | Malattia - |
|---|---|---|
| Test + | *P(T+\|M+)* | *P(T+\|M-)* |
| Test - | *P(T-\|M+)* | *P(T-\|M-)* |

che moltiplicando le probabilità della prima colonna (Malattia +) per la probabilità di presenza della malattia nella popolazione *P(M+)*, cioè per la prevalenza della malattia, e moltiplicando le probabilità della seconda colonna (Malattia -) per la probabilità di assenza di malattia della malattia nella popolazione *P(M-)*, essendo *P(M-) = 1 - P(M+)*, diventa

| | Malattia + | Malattia - |
|---|---|---|
| Test + | *P(T+\|M+) · P(M+)* | *P(T+\|M-) · P(M-)* |
| Test - | *P(T-\|M+) · P(M+)* | *P(T-\|M-) · P(M-)* |

[151] Reverend Thomas Bayes. *An assay toward solving a problem in the doctrine of chance*. Philo. Trans. Roy. Soc., vol. 53, 370-418, 1763. URL consultato il 27/10/2018: https://royalsocietypublishing.org/doi/10.1098/rstl.1763.0053

cosa che consente di esprimere il teorema di Bayes (iii) in questa forma

$$P(M+|T+) = \frac{P(T+|M+) \cdot P(M+)}{P(T+|M+) \cdot P(M+) + P(T+|M-) \cdot P(M-)} \quad \text{(iv)}$$

Le grandezze in gioco nel calcolo del valore predittivo di un test positivo $P(M+|T+)$ sono[152]:
→ la **sensibilità** $P(T+|M+)$ del test diagnostico, che è la probabilità che il test sia **positivo in un soggetto malato**;
→ la **specificità** $P(T-|M-)$ del test diagnostico che è la probabilità che il test sia **negativo in un soggetto sano**;
→ la **prevalenza** della malattia $P(M+)$ cioè la probabilità di incontrare un soggetto malato nella popolazione generale.

In genere tuttavia, proprio a causa del fatto che le statistiche sono effettuate contando i casi, cosicché per conoscere ad esempio la prevalenza dell'influenza nel mese di dicembre si conta il numero dei malati di influenza in questo mese, risulta più semplice e intuitivo impiegare le probabilità per ricavare il numero di casi osservati per ciascuna delle celle della tabella, riportandoli in questo modo

| | Malattia + | Malattia - |
|---|---|---|
| Test + | *VP* | *FP* |
| Test - | *FN* | *VN* |

e definendoli come[153]:
→ **veri positivi** (*VP*), i soggetti **malati con il test positivo**;
→ **falsi positivi** (*FP*), i soggetti **sani con il test positivo**;
→ **falsi negativi** (*FN*), i soggetti **malati con il test negativo**;
→ **veri negativi** (*VN*), i soggetti **sani con il test negativo**.

A partire dal numero di casi osservati *VP*, *FP*, *FN*, *VN* possono essere calcolate in modo semplice, oltre al valore predittivo del test positivo[154], anche le numerose altre grandezze di interesse epidemiologico[155] riportate dal pacchetto **epiR** che abbiamo visto nella parte statistica.

---

[152] Altman DG, Bland JM. Statistics Notes: Diagnostic tests 1: sensitivity and specificity. BMJ 1994;308;1552. URL consultato il 29/10/2018: https://goo.gl/x7Txjz
[153] Gerhardt W, Keller H. Evaluation of test data from clinical studies. Scand J Clin Lab Invest, 46, 1986 (supplement 181).
[154] Altman DG, Bland JM. Statistics Notes: Diagnostic tests 2: predictive values. BMJ 1994;309;102. URL consultato il 29/10/2018: https://goo.gl/xffgB9
[155] Una ottima introduzione si trova nel *Quaderno di Epidemiologia veterinaria* del Prof. Ezio Bottarelli. URL consultato il 29/10/2018: https://goo.gl/ZvrZCq

# A9. Scale nominali, scale ordinali e scale numeriche[156]

La relazione tra il tipo di variabile (qualitativa / quantitativa), il processo mediante il quale viene ottenuto il dato (classificazione / conteggio / misura) e la scala numerica nella quale il dato viene espresso è qui riportato in estrema sintesi:

| **Variabile** | **Dato ottenuto mediante un processo di...** | **Espressione del dato** | **Scala nella quale il dato viene espresso** |
|---|---|---|---|
| Qualitativa | Classificazione qualitativa | Espressione verbale (maschio / femmina) | Scala nominale |
| | Classificazione semiquantitativa | Attributo ordinabile (neonato < bambino < adulto) | Scala ordinale |
| Quantitativa | Conteggio | Numero naturale (1, 2, 3, ...) | Scala numerica discreta |
| | Misura | Numero razionale (1.435, 123.9, 84.327, ...) | Scala numerica continua |

Il tipo più semplice di scala è rappresentato dalla **scala nominale**. Corrisponde ai dati della natura più elementare, quella di dati qualitativi che non possiedono criteri di ordinamento. Ne sono un esempio la classificazione maschio/femmina e la classificazione dei gruppi sanguigni O / A / B / AB. La misura più intuitiva e più utilizzata, nel caso della scala nominale, è costituita dalla percentuale (o dalla proporzione o frazione). In questo caso la rappresentazione più appropriata è il **grafico a torta** (pie-chart). Quando invece le grandezze da rappresentare sono indipendenti, e non costituiscono la parte di un tutto, è consigliabile utilizzare un **grafico a barre** (barplot) con le barre staccate e disposte orizzontalmente.

Nel caso di dati qualitativi che possiedono un criterio di ordinamento si ha a che fare con una **scala ordinale**. Questo avviene per esempio nella classificazione in neonati, bambini e adulti, nella quale è appropriato applicare un ordinamento per età (neonato < bambino < adulto). Le classificazioni per ranghi ne sono un altro esempio (nelle classificazioni per ranghi il valore osservato viene trasformato nel corrispondente rango, cioè nel numero della posizione che il dato occupa nella lista ordinata dei dati). Per le scale ordinali è consigliabile utilizzare un **grafico a barre** (barplot) con le barre staccate per indicare la discontinuità.

Una **scala numerica discret**a è tipicamente quella relativa a dati numerici ottenuti mediante operazioni di conteggio i cui dati sono riportati sotto forma di **numeri naturali** (1, 2, 3, ….). Si prenda ad esempio il conteggio degli eritrociti (globuli rossi) nel sangue. La differenza minima tra due conteggi teoricamente misurabile è rappresentata da un globulo rosso: i dati sono numerici, ma tra l'uno e il successivo (o il precedente) vi è un salto, essendo esclusi i valori intermedi. Un altro esempio di scala numerica discreta può essere costituito dal numero di controlli medici cui si sottopone una donna durante la gravidanza. Per le scale numeriche discrete è opportuno utilizzare un **grafico a barre** (barplot) o, nel caso in cui si debbano rappresentare contemporaneamente due variabili, un **grafico di dispersione** (o **xy** o **cartesiano**) precisando che i valori compresi tra due interi non hanno un corrispettivo reale.

La **scala numerica continua** è tipicamente quella impiegata per rappresentare dati numerici ottenuti mediante procedimenti di misura come quelli chimici e quelli fisici i cui dati sono riportati sotto forma di **numeri razionali** (1.435, 123.9, 84.327, ...). Esempi ne sono la misura della concentrazione del colesterolo

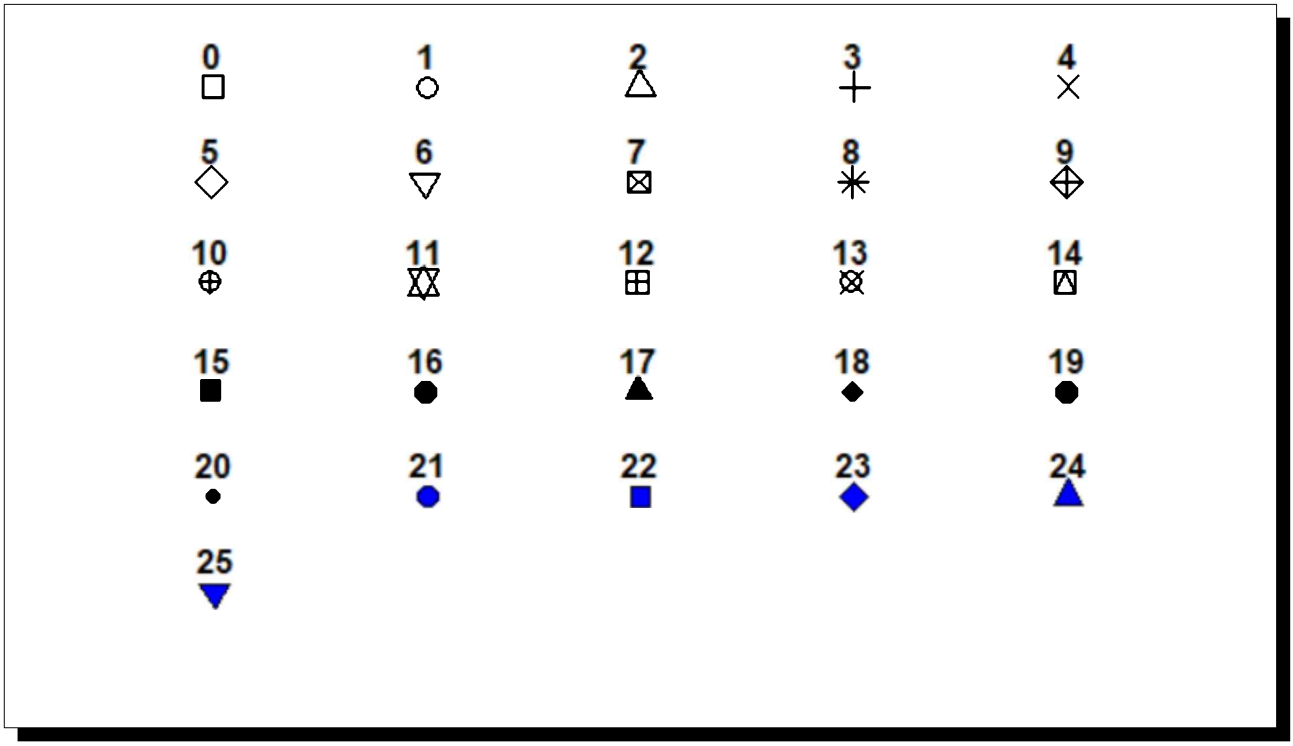
[156] Bossi A. *Guida a una corretta rappresentazione grafica dei risultati scientifici*. Aggiornamento del Medico 1990;14:XXX-XXXIX.

nel sangue e la misura dei valori di temperatura. L'intervallo tra due valori è riducibile a piacere, anche se per motivi pratici, legati al potere di risoluzione degli strumenti di misura impiegati, nell'uso ordinario ci si accontenta di un numero limitato di cifre significative dopo la virgola. Per questa scala è appropriato di volta in volta un **istogramma**, un grafico a linee spezzate, come un **poligono di frequenza**, o un **grafico di dispersione** (o **xy** o **cartesiano**) nel caso in cui si debbano rappresentare contemporaneamente due variabili.

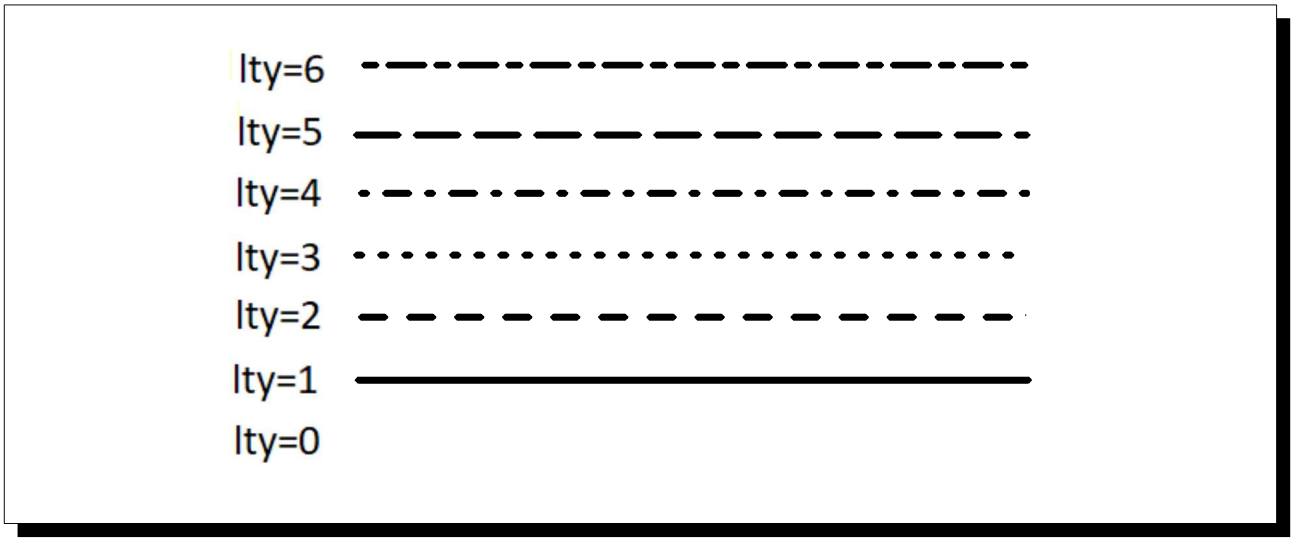
---

# A10. Simboli dei punti e tipi di linea di R

Nei grafici di dispersione (scatterplot) con l'argomento **pch=***num* ove *num* è un numero compreso tra 0 e 25 i punti possono essere rappresentati mediante uno dei **simboli** qui riportati:

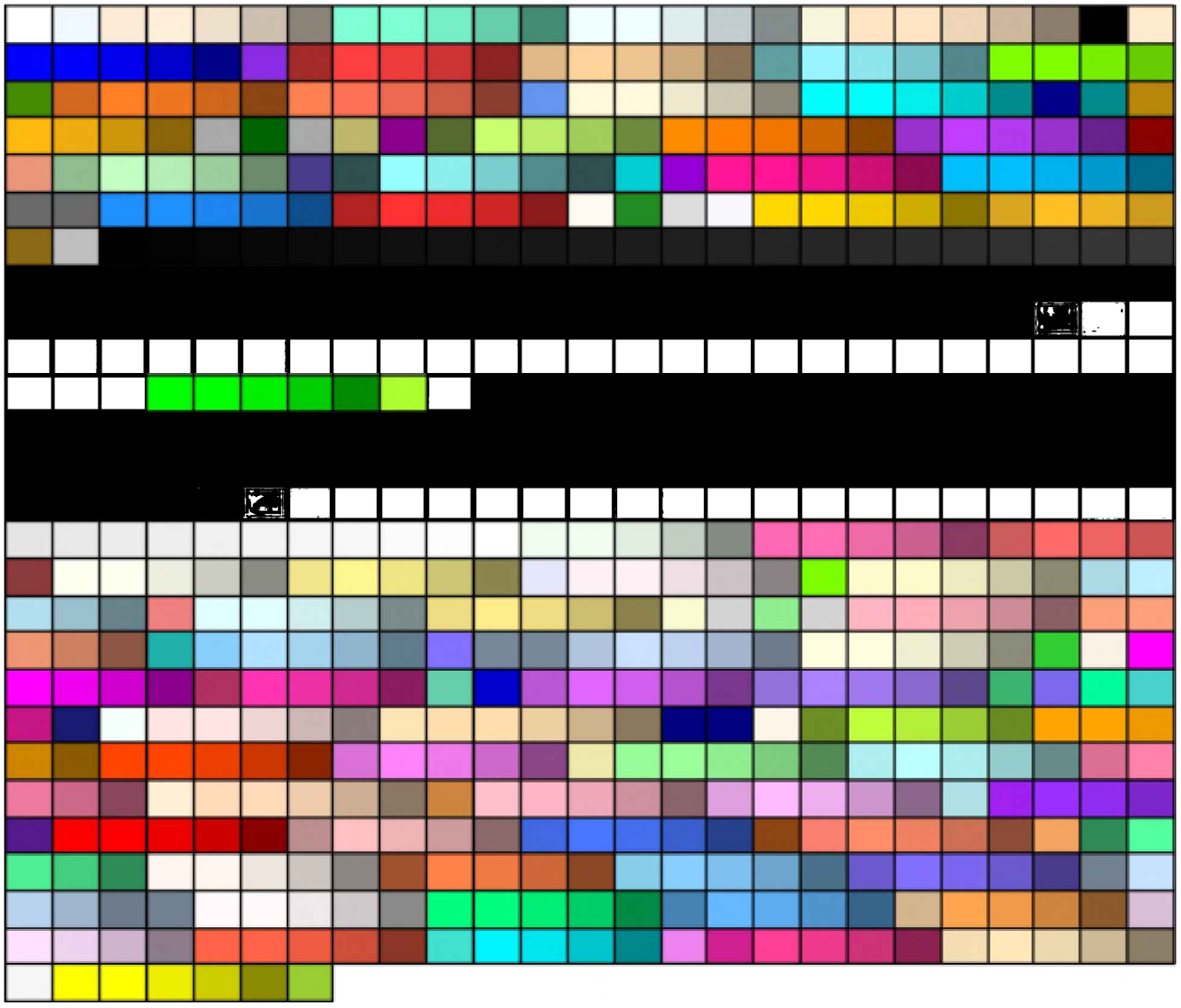



Se in un grafico di dispersione volete sovrapporre ai punti per esempio la retta di regressione (ma anche una curva) potete con l'argomento **lty=***num* dove *num* è un numero questa volta compreso tra 0 e 6 cambiare il **tipo di linea** e impiegare uno dei **tratteggi** qui riportati:



# A11. Tabella dei colori di R

In **R** sono disponibili i 657 colori qui rappresentati



i cui nomi vengono visualizzati nella `Console di R` digitando:

```
colors()
```

Tuttavia per avere di ogni colore contemporaneamente il nome, il colore visualizzato e il codice esadecimale che può essere impiegato in alternativa al nome, è necessaria una tabella di corrispondenza che può essere generata con uno scrip non banale, come quello che segue[157].

Lo script[158] salva la tabella dei colori di **R** nel file `C:\Rdati\Rcolors.pdf` che può essere conservato

[157] Il codice dello script è stato adattato dal codice di Earl F. Glynn pubbicato sul sito dello *Stowers Institute for Medical Research*. URL consultato il 27/10/2018: https://goo.gl/EpYzad

[158] Lo script può essere copiato dal testo della guida ma può anche essere scaricato per intero da questo link. URL consultato il 29/10/2018: https://goo.gl/6fRz9N

come documentazione aggiuntiva:

```
# TABELLA DEI COLORI DI R
#
# adattato dal codice di Earl F. Glynn tratto da http://research.stowers.org/mcm/efg/R/Color/Chart/
#
pdf("c:/Rdati/Rcolors.pdf", width=6, height=10)
oldparameters <- par(mar=c(1,1,2,1), mfrow=c(2,1))
#
stopifnot(length(colors()) == 657)
SetTextContrastColor <- function(color)
{  ifelse( mean(col2rgb(color)) > 127, "black", "white")}
TextContrastColor <- unlist( lapply(colors(), SetTextContrastColor) )
#
# tabella dei colori di R ordinati per numero
#
colCount <- 25  # number per row
rowCount <- 27
plot( c(1,colCount), c(0,rowCount), type="n", ylab="", xlab="", axes=FALSE, ylim=c(rowCount,0))
title("Tabella dei colori di R")
for (j in 0:(rowCount-1))
{base <- j*colCount
remaining <- length(colors()) - base
RowSize <- ifelse(remaining < colCount, remaining, colCount)
rect((1:RowSize)-0.5,j-0.5, (1:RowSize)+0.5,j+0.5, border="black", col=colors()[base + (1:RowSize)])
text((1:RowSize), j, paste(base + (1:RowSize)), cex=0.7, col=TextContrastColor[base + (1:RowSize)])
}
#
# colori di R ordinati per tonalità e saturazione
#
RGBColors <- col2rgb(colors()[1:length(colors())])
HSVColors <- rgb2hsv( RGBColors[1,], RGBColors[2,], RGBColors[3,], maxColorValue=255)
HueOrder  <- order( HSVColors[1,], HSVColors[2,], HSVColors[3,] )
plot(0, type="n", ylab="", xlab="", axes=FALSE, ylim=c(rowCount,0), xlim=c(1,colCount))
title("Colori di R ordinati per tonalità e saturazione")
for (j in 0:(rowCount-1))
{for (i in 1:colCount)
{k <- j*colCount + i
if (k <= length(colors()))
{rect(i-0.5,j-0.5, i+0.5,j+0.5, border="black", col=colors()[ HueOrder[k] ])
text(i,j, paste(HueOrder[k]), cex=0.7, col=TextContrastColor[ HueOrder[k] ])}}}
#
# colori di R con nomi e codice esadecimale del colore
#
GetColorHexAndDecimal <- function(color)
{c <- col2rgb(color)
sprintf("#%02X%02X%02X   %3d %3d %3d", c[1],c[2],c[3], c[1], c[2], c[3])}
par(oldparameters)
oldparameters <- par(mar=c(1,1,1,1))
index <- paste(1:length(colors()))
HexAndDec <- unlist( lapply(colors(), GetColorHexAndDecimal) )
PerColumn <- 50
```

```
PerPage   <- 2*PerColumn
for (page in 1: (trunc( (length(colors()) + (PerPage-1)) / PerPage) )  )
{plot(0, type="n", ylab="", xlab="",
axes=FALSE, ylim=c(PerColumn,0), xlim=c(0,1))
title("Colori di R con nome e codice esadecimale")
mtext(paste("pag.", page, "/ 7"), SOUTH<-1, adj=1, line=-1)
base <- PerPage*(page-1)
remaining <- length(colors()) - base
ColumnSize <- ifelse(remaining < PerColumn, remaining, PerColumn)
rect(0.00, 0:(ColumnSize-1), 0.49, 1:ColumnSize, border="black", col=colors()[(base+1):
(base+ColumnSize)])
text(0.045, 0.45+(0:(ColumnSize-1)), adj=1, index[(base+1):(base+ColumnSize)], cex=0.6,
col=TextContrastColor[(base+1):(base+ColumnSize)])
text(0.06, 0.45+(0:(ColumnSize-1)), adj=0, colors()[(base+1):(base+ColumnSize)], cex=0.6,
col=TextContrastColor[(base+1):(base+ColumnSize)])
save <- par(family="mono")  # use mono-spaced font with number columns
text(0.25, 0.45+(0:(ColumnSize-1)), adj=0, HexAndDec[(base+1):(base+ColumnSize)], cex=0.6,
col=TextContrastColor[(base+1):(base+ColumnSize)])
par(save)
if (remaining > PerColumn)
{remaining <- remaining - PerColumn
ColumnSize <- ifelse(remaining < PerColumn, remaining, PerColumn)
rect(0.51, 0:(ColumnSize-1), 1.00, 1:ColumnSize, border="black", col=colors()[(base+PerColumn+1):
(base+PerColumn+ColumnSize)])
text(0.545, 0.45+(0:(ColumnSize-1)), adj=1, index[(base+PerColumn+1):(base+PerColumn+ColumnSize)],
cex=0.6, col=TextContrastColor[(base+PerColumn+1):(base+PerColumn+ColumnSize)])
text(0.56, 0.45+(0:(ColumnSize-1)), adj=0, colors()[(base+PerColumn+1):(base+PerColumn+ColumnSize)],
cex=0.6, col=TextContrastColor[(base+PerColumn+1):(base+PerColumn+ColumnSize)])
save <- par(family="mono")
text(0.75, 0.45+(0:(ColumnSize-1)), adj=0, HexAndDec[(base+PerColumn+1):
(base+PerColumn+ColumnSize)], cex=0.6, col=TextContrastColor[(base+PerColumn+1):
(base+PerColumn+ColumnSize)])
par(save)}}
par(oldparameters)
dev.off()
#
```

---

# Testi, pubblicazioni e risorse web consultati

## Parte generale


Altman DG, Bland JM. *Statistics Notes: Diagnostic tests 1: sensitivity and specificity*. BMJ 1994;308;1552. URL consultato il 29/10/2018: https://goo.gl/x7Txjz

Altman DG, Bland JM. *Statistics Notes: Diagnostic tests 2: predictive values*. BMJ 1994;309;102. URL consultato il 29/10/2018: https://goo.gl/xffgB9

Altman DG, Deeks JJ. *Diagnostic tests 4: likelihood ratios*. BMJ 2004;329;168-169. URL consultato il 29/10/2018: https://goo.gl/ahpbpN

Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, Milano, 1979.

Baldini M. *L'errore nella scienza*. Biochim Clin, 1991;15:28-38.

Baldini M. Citato in (a cura di) Binanti L. *Sbagliando s'impara. Una rivalutazione dell'errore*. Armando, Roma, 2005, ISBN 88-8358-819-3, p. 77.

Bayes T. *An assay toward solving a problem in the doctrine of chance*. Philo. Trans. Roy. Soc., vol. 53, 370-418, 1763. URL consultato il 28/11/2018: https://doi.org/10.1098/rstl.1763.0053

Best EWR et al. *A Canadian Study on Smoking and Health (Final Report)*. Dept. Natl. Health and Welfare, Canada, 1966. Citato in: Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6, p. 124.

Boniolo G, Vidali P. *Introduzione alla filosofia della scienza*. Pearson Paravia Bruno Mondadori, 2003, ISBN 88-424-9549-2.

Bossi A. *Guida a una corretta rappresentazione grafica dei risultati scientifici*. Aggiornamento del Medico 1990;14:XXX-XXXIX.

Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8.

Bottarelli E. *Quaderno di Epidemiologia veterinaria*. URL consultato il 29/10/2018: https://goo.gl/ZvrZCq

Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9.

Cavalli-Sforza L. *Analisi statistica per medici e biologi*. Boringhieri, Torino, 1965, CL[159] 74-7940-9.


[159] CL è l'abbreviazione di Codice Libro, un sistema di codifica simile all'ISBN, e in uso in Italia negli anni '70 e '80.

Colton T. *Statistica in medicina*. Piccin Editore, Padova, 1991, ISBN 88-299-0095-8.

D'Agostini G. *Probabilità e incertezze di misura*. URL consultato il 29/10/2018: https://goo.gl/HQGL4i

De Finetti B. *L'invenzione della verit*à. Raffaello Cortina Editore, Milano, 2006, ISBN 88-6030-060-6, p. 35.

Federspil G. *Logica clinica. I principe del metodo in medicina*. The McGraw-Hill Companies, Milano, 2004, ISBN 88-386-2984-6.

Galen RS, Gambino RS. *Oltre il concetto di normalità: il valore predittivo e l'efficienza delle diagnosi mediche*. Piccin Editore, 1980, ISBN 88-212-0770-6.

Gerhardt W, Keller H. *Evaluation of test data from clinical studies*. Scand J Clin Lab Invest, 46, 1986 (supplement 181).

Glantz SA. *Statistica per discipline bio-mediche*. McGraw-Hill Libri Italia, Milano, 1988, ISBN 88-386-0618-8.

Ingelfinger JA, Mosteller F, Thibodeau LA, Ware JH. *Biostatistica in medicina*. Raffaello Cortina Editore, MIlano, 1986, ISBN 88-7078-065-1.

Lantieri PB, Risso D, Rovida S, Ravera G. *Statistica medica ed elementi di informatica*. McGraw-Hill Libri Italia, Milano, 1994, ISBN 88-386-2614-6.

Marchetti GM. *Introduzione all'analisi statistica dei dati multivariati*. URL consultato il 28/11/2018: https://goo.gl/KW6Nmj

Maritz JS. *Distribution-free statistical methods*. Chapman and Hall, New York, 1984, ISBN 0-412-25410-7.

Motterlini M, Crupi V. *Decisioni mediche*. Raffaello Cortina Editore, Milano, 2005, ISBN 88-7078-632-3.

Scott IA, Greenberg PB, Poole PJ. *Cautionary tales in the clinical interpretation of studies of diagnostic tests*. Internal Medicine Journal, 2008, 38, 120-129.

Siegel S, Castellan NJ. *Statistica non parametrica*. McGraw-Hill Libri Italia, Milano, 1992, ISBN 88-386-0620-X.

Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6.

Soliani L. *Manuale di statistica per la ricerca e la professione. Statistica univariata e bivariata parametrica e non parametrica per le discipline ambientali e biologiche*. URL consultato il 29/10/2018: https://goo.gl/a9jk9Z

Spiegel MR. *Probabilità e statistica*. Gruppo Editoriale Fabbri-Bompiani, Sonzogno, Etas, Milano, 1979.

Spiegel MR. *Statistic*a. Gruppo Editoriale Fabbri-Bompiani, Sonzogno, Etas, Milano, 1976.

Wonnacott TH, Wonnacott RJ. *Introduzione alla statistica*. Franco Angeli Editore, Milano, 1980, ISBN 88-204-0323-4.

## R - documentazione di base


Chi Yau. *R-Tutorial*. URL consultato il 30/10/2018: https://goo.gl/d3hSTF

Hahsler M, Hornik K, Buchta C. *Getting Things in Order: An Introduction to the R Package seriation*. URL consultato il 29/10/2018: https://goo.gl/1rdhuk

Mineo AM. *Una Guida all'utilizzo dell'Ambiente Statistico R*. URL consultato il 29/10/2018: https://goo.gl/fvCxAY

Muggeo VMR, Ferrara G. *Il linguaggio R: concetti introduttivi ed esempi*. URL consultato il 29/10/2018: https://goo.gl/oJmPyX

Venables WN, Smith DM and the R Core Team. *An Introduction to R.* URL consultato il 17/10/2018: https://goo.gl/PnjsBW

- - - -

*Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

*Contributed Documentation*. URL consultato il 16/10/2018: https://goo.gl/sXq5Uc

*R Data Import/Export*. URL consultato il 17/10/2018: https://goo.gl/8mzH4L

*R: A Language and Environment for Statistical Computing, Reference Index*. URL consultato il 17/10/2018: https://goo.gl/kes91X

*Revolutions*. URL consultato il 30/10/2018: https://goo.gl/rFVUi9

*The R Manuals*. URL consultato il 16/10/2018: https://goo.gl/CG7dzh

*The R Project for Statistical Computing*. URL consultato il 07/08/2018: https://goo.gl/MW66w1

*xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files*. URL consultato il 12/08/2018: https://goo.gl/DJCHTx


## R - analisi statistica dei dati


Dell'Omodarme M. *Esercitazioni di statistica biomedica. Alcune note su R*. URL consultato il 29/10/2018: https://goo.gl/EXKkcK

Frascati R. *Formulario di statistica con R*. URL consultato il 29/10/2018: https://goo.gl/Pfd6rX

Kobacoff RI. *Quick-R for SAS/SPSS/Stata Users*. URL consultato il 28/11/2018: https://goo.gl/GhAEAh

Krijnen WP. *Applied Statistics for Bioinformatics using R*. URL consultato il 30/10/2018: https://goo.gl/mJKqWF

Maindonald JH. *Using R for Data Analysis and Graphics - Introduction, Code and Commentary*. URL consultato il 30/10/2018: https://goo.gl/7BVzoe

Mangiafico SS. *An R Companion for the Handbook of Biological Statistics*. URL consultato il 31/10/2018: https://goo.gl/7VNgbq

Mangiafico SS. *Summary and Analysis of Extension Program Evaluation in R*. URL consultato il 31/10/2018: https://goo.gl/WNMqQB

*R-bloggers*. URL consultato il 31/10/2018: https://goo.gl/PvuEJu

R-bloggers. *About a Curious Feature and Interpretation of Linear Regressions*. URL consultato il 31/10/2018: https://goo.gl/jA7X5i

Ricci V. *Analisi delle serie storiche con R*. URL consultato il 30/10/2018: https://goo.gl/R5zaXJ

Ricci V. *Principali tecniche di regressione con R*. URL consultato il 30/10/2018: https://goo.gl/1jcrm9

Ricci V. *Rappresentazione analitica delle distribuzioni statistiche con R*. URL consultato il 30/10/2018: https://goo.gl/LxcmRX

Rousseeuw PJ, Croux C. *Alternatives to the Median Absolute Deviation*. Journal of the American Statistical Association 88 (424), 1273-1283, 1993. URL consultato il 30/08/2018: https://goo.gl/4Rh53b

Verzani J. *simpleR - Using R for Introductory Statistics*. URL consultato il 30/10/2018: https://goo.gl/3Lm2tP

## R - rappresentazione grafica dei dati

Glynn EF. *R Color Chart*. URL consultato il 28/11/2018: https://bit.ly/2WJrJrR

Kobacoff RI. *Quick-R for SAS/SPSS/Stata Users*. URL consultato il 30/10/2018: https://goo.gl/GhAEAh

Maindonald JH. *Using R for Data Analysis and Graphics - Introduction, Code and Commentary*. URL consultato il 30/10/2018: https://goo.gl/7BVzoe

Osamu Ogasawara. *R Graphical Manual*. URL consultato il 30/10/2018: https://goo.gl/UHU7CZ

# Fonti delle frasi citate

*“Se ascolto dimentico, se vedo ricordo, se faccio capisco.”*
*(Proverbio)*
Questo aforisma viene nella maggior parte dei casi attribuito a Confucio (K'ung-fu-tzu, 551-479 a.C.). Tuttavia non se ne trova riscontro nelle sue opere, e probabilmente il detto è derivato da un adattamento piuttosto libero di un aforisma di Xunzi (340-245 a.C.), uno dei suoi tre primi principali discepoli[160].

*"Ma a cosa serve saper calcolare le radici quadrate?"*
*(Fabio Fazio)*
Gino & Michele, Matteo Molinari. *Anche le formiche nel loro piccolo s'incazzano*. Baldini&Castoldi, Milano, 1995, ISBN 88-8089-545-1, p. 154.

*"Quando non sai fare niente, bisogna essere almeno ambiziosi."*
*(Georges Wolinsky)*
Gino & Michele, Matteo Molinari. *Anche le formiche nel loro piccolo s'incazzano*. Baldini&Castoldi, Milano, 1995, ISBN 88-8089-545-1, p. 309.

*“È più facile modificare le esigenze in funzione del programma che viceversa.”*
*(Arthur Bloch)*
Arthur Bloch. *La legge di Murphy del 2000, e altre cose che possono andar male nel nuovo millennio*. Longanesi & C, Milano, 1999, ISBN 88.304-1692-4, p. 86.

*“La nuova versione di un programma si bloccherà non appena la vecchia è stata cancellata.”*
*(Arthur Bloch)*
Arthur Bloch. *La legge di Murphy del 2000, e altre cose che possono andar male nel nuovo millennio*. Longanesi & C, Milano, 1999, ISBN 88.304-1692-4, p. 80.

*“Non esiste vento favorevole per il marinaio che non sa dove andare.[161]”*
*(Lucio Anneo Seneca)*
Seneca. *Lettere a Lucilio*. A cura di C. Barone. Garzanti Libri, Milano, ISBN 978-88-11-37012-3, lettera 71, 3, p.380.

*“Il meteorologo non sbaglia mai. Se c’è l’80 % di probabilità di pioggia, e non piove, vuol dire che siamo nel 20 %.”*
*(Saul Barron)*
Darrel Huff. *Mentire con le statistiche*. Monti&Ambrosini, ISBN 978-88-89479-09-4, p. 180.

"*Sai ched’è la statistica? È ’na cosa / che serve pe’ fa’ un conto in generale / de la gente che nasce, che sta male, / che more, che va in carcere e che spósa. / Ma pe’ me la statistica curiosa / è dove c’entra la percentuale, / pe’ via che, lì, la media è sempre eguale / puro co’ la persona bisognosa. / Me spiego: da li conti che se fanno / secondo le statistiche d’adesso / risurta che te tocca un pollo all’anno: / e, se nun entra ne le spese tue, / t’entra ne la statistica lo stesso / perché c’è un antro che ne magna due.*"
*(Trilussa)*

---

[160] *What I hear I forget, what I see I remember, what I do I understand*. Posted by Kim Bennett on September 12, 2007. URL consultato il 07/08/2018: https://goo.gl/iJFBYL
[161] Traduzione un po' libera ma efficace dell'originale "*Ignoranti quem portum petat nullus suus ventus est*", traducibile più letteralmente in "*Non esiste vento favorevole per chi non sa in quale porto dirigersi*".

Trilussa. *Le Poesie*. A cura di Pietro Pancrazi. Note di Luigi Huetter. Arnoldo Mondadori Editore, Verona, 1951, p. 392.

*"Nei tempi antichi non c'erano le statistiche, perciò era necessario ripiegare sulle menzogne."*
*(Stephen Leacock)*
Darrel Huff. *Mentire con le statistiche*. Monti&Ambrosini editori, ISBN 978-88-89479-09-4, p. 183.

*"La semplicità, cosa rarissima ai nostri tempi.*[162]*"*
*(Ovidio)*
Ovidio. *L'arte di amare*. Newton Compton Editori, Roma,2014, ISBN 978-88-541-6631-8, 241-242 (p. 18).

*"Se non si è un genio, è bene mirare ad essere chiaro."*
*(Anthony Hope)*
F. Palazzi, S. Spaventa Filippi. *Il libro dei mille savi*. Ulrico Hoepli, Milano, 1955, p. 786.

---

[162] "... a*evo rarissima nostro simplicitas...*"